No.

In the Supreme Court of the United States

GOOGLE LLC, PETITIONER

v.

ORACLE AMERICA, INC.

ON PETITION FOR A WRIT OF CERTIORARI TO THE UNITED STATES COURT OF APPEALS FOR THE FEDERAL CIRCUIT

PETITION FOR A WRIT OF CERTIORARI

THOMAS C. GOLDSTEIN
GOLDSTEIN & RUSSELL, P.C.
5225 Wisconsin Avenue, N.W., Suite 404
Washington, DC 20015

ROBERT A. VAN NEST
CHRISTA M. ANDERSON
EUGENE M. PAIGE
REID P. MULLEN
KEKER, VAN NEST & PETERS LLP
633 Battery Street
San Francisco, CA 94111

BRUCE W. BABER
MARISA C. MALECK
KING & SPALDING LLP
1180 Peachtree Street, N.E.
Atlanta, GA 30309

KANNON K. SHANMUGAM
Counsel of Record
CHARLES L. MCCLOUD
MENG JIA YANG
WILLIAMS & CONNOLLY LLP
725 Twelfth Street, N.W.
Washington, DC 20005
(202) 434-5000
kshanmugam@wc.com

MICHAEL S. KWUN
KWUN BHANSALI LAZARUS LLP
555 Montgomery Street, Suite 750
San Francisco, CA 94111

QUESTIONS PRESENTED

The Copyright Act provides that, while "original works of authorship" are generally eligible for copyright protection, 17 U.S.C. 102(a), "[i]n no case does copyright protection for an original work of authorship extend to any idea, procedure, process, system, method of operation, concept, principle, or discovery, regardless of the form in which it is described, explained, illustrated, or embodied in such work," 17 U.S.C. 102(b). The Act also makes clear that "the fair use of a copyrighted work * * * is not an infringement of copyright." 17 U.S.C. 107.

As is relevant here, software interfaces are lines of computer code that allow developers to operate prewritten libraries of code used to perform particular tasks. Since the earliest days of software development, developers have used interfaces to access essential tools for building new computer programs. Contravening that longstanding practice, the Federal Circuit in this case held both that a software interface is copyrightable and that petitioner's use of a software interface in a new computer program cannot constitute fair use as a matter of law.

The questions presented are:

1. Whether copyright protection extends to a software interface.

2. Whether, as the jury found, petitioner's use of a software interface in the context of creating a new computer program constitutes fair use.

CORPORATE DISCLOSURE STATEMENT

Petitioner Google LLC is an indirect subsidiary of Alphabet Inc., a publicly held company. Alphabet Inc. has no parent corporation, and no publicly held company owns 10% or more of its stock.

TABLE OF CONTENTS

TABLE OF AUTHORITIES

Cases:

Page

Cases—continued:

Page

Cases—continued:

Statutes:

Miscellaneous:

Page

Miscellaneous—continued:

In the Supreme Court of the United States

No.

GOOGLE LLC, PETITIONER

v.

ORACLE AMERICA, INC.

ON PETITION FOR A WRIT OF CERTIORARI TO THE UNITED STATES COURT OF APPEALS FOR THE FEDERAL CIRCUIT

PETITION FOR A WRIT OF CERTIORARI

Google LLC respectfully petitions for a writ of certiorari to review the judgment of the United States Court of Appeals for the Federal Circuit in this case.

OPINIONS BELOW

The opinion of the court of appeals regarding fair use (App., *infra*, 1a-55a) is reported at 886 F.3d 1179. The district court's orders denying respondent's motions for judgment as a matter of law (App., *infra*, 92a-120a) and for a new trial (App., *infra*, 56a-91a) are unreported.

The earlier opinion of the court of appeals regarding copyrightability (App., *infra*, 121a-192a) is reported at 750 F.3d 1339. The district court's order granting petitioner's motion for judgment as a matter of law (App., *infra*, 212a-272a) is reported at 872 F. Supp. 2d 974.

JURISDICTION

The judgment of the court of appeals was entered on March 27, 2018. A petition for rehearing was denied on August 28, 2018 (App., *infra*, 299a-300a). On October 23, 2018, the Chief Justice extended the time within which to file a petition for a writ of certiorari to and including January 25, 2019. The jurisdiction of this Court is invoked under 28 U.S.C. 1254(1).

STATUTORY PROVISIONS INVOLVED

The relevant provisions of the Copyright Act of 1976, Pub. L. No. 94-553, 90 Stat. 2541, are reproduced in the appendix to this petition (App., *infra*, 285a-299a).

STATEMENT

This case has been aptly described as the "copyright lawsuit of the decade." Anandashankar Mazumdar, *Oracle Victory Stirs Uncertainties in Software Copyright*, Bloomberg Law (May 10, 2018) <tinyurl.com/suitdecade>. As it comes to this Court, the case presents two exceptionally important questions concerning the copyrightability and fair use of software interfaces—lines of computer code that are necessary to allow developers to operate prewritten libraries of code used to perform particular tasks.

The first question is whether copyright protection extends to a software interface. This Court granted certiorari to resolve a closely related question in *Lotus Development Corp.* v. *Borland International, Inc.*, 516 U.S. 233 (1996) (per curiam), but the Court deadlocked 4-4 on that question after oral argument. The second question is whether, as the jury found, petitioner's use of a software interface in the context of creating a new computer program constitutes fair use. The lower courts are badly in

need of guidance on how to apply the fair-use doctrine in the context of computer code.

This case involves a high-profile dispute between two leading technology companies, petitioner Google and respondent Oracle. Sun Microsystems originally developed the Java platform, which includes the free Java programming language. The software interfaces at issue are part of the Java language's application programming interface (API). Sun encouraged developers to learn the Java language by touting the ability to use software interfaces—also known as "the Java API declarations"—to access preexisting libraries of code used to perform particular tasks. The interfaces thereby facilitated development of programs in the Java language.

Google used some of the Java API declarations to build Android, a revolutionary platform for modern mobile devices such as smartphones and tablets. Google incorporated those declarations to allow developers to write applications for Android using the Java language. Sun originally applauded Google for using the Java language. But after Oracle acquired Sun, it sued Google for copyright infringement.

After years of litigation, the Federal Circuit (which had jurisdiction because of Oracle's initial assertion of now-dismissed patent claims) has twice reversed judgments in Google's favor. It first held that the Java API declarations are copyrightable and then overturned a jury's verdict that Google's use of the declarations constituted fair use. The Federal Circuit thereby deepened the acknowledged conflicts among the courts of appeals concerning the application of the Copyright Act and the merger doctrine in the context of computer software.

Google has never disputed that some forms of computer code are entitled to copyright protection. But the Federal Circuit's widely criticized opinions—in an area in

which that court has no specialized expertise—go much further, throwing a devastating one-two punch at the software industry. If allowed to stand, the Federal Circuit's approach will upend the longstanding expectation of software developers that they are free to use existing software interfaces to build new computer programs. Developers who have invested in learning free and open programming languages such as Java will be unable to use those skills to create programs for new platforms—a result that will undermine both competition and innovation. Because this case is an optimal vehicle for addressing the exceptionally important questions presented, the petition for a writ of certiorari should be granted.

A. Background

1. Modern smartphones are "such a pervasive and insistent part of daily life that the proverbial visitor from Mars might conclude they were an important feature of human anatomy." *Riley* v. *California*, 134 S. Ct. 2473, 2484 (2014). Given the ubiquity of smartphones today, it is easy to forget the challenges that developers initially faced in building the operating systems that allow modern smartphones to perform their myriad functions. Among other things, developers had to account for smaller processors, limited memory and battery life, and the need to support mobile communications and interactive applications. C.A. App. 21958.

In 2008, Google overcame those challenges and released Android, an open-source platform designed to enable mobile devices such as smartphones and tablets. The Android platform took over three years to build, and Google had almost 100 engineers working on the project. C.A. App. 21858, 21861-21862. In the decade since its release, Android has become one of the most widely used

mobile operating systems, with billions of users worldwide.

Google built Android using the Java programming language. Sun Microsystems released the Java language in the 1990s and made it free and open for all to use without a license. Sun's motives for doing so were not entirely altruistic: Sun believed that the resulting proliferation of Java developers would drive sales of Sun hardware and other services. Java has become one of the world's most popular programming languages. App., *infra*, 216a.

Java 2 Standard Edition (Java SE) is a platform used to write and run programs for desktop and server computers. Java SE includes the Java language, which in turn contains the Java application programming interface (API). C.A. App. 51447-51448. The Java API provides access to prewritten "methods." In Java, as in many other programming languages, methods are used to program specific, commonly performed tasks. Each method consists of two parts: a method header and a method body. The method header is also known as a "declaration" or "declaring code," because it labels (or "declares") the method, typically by reference to what the method will do. The declaration also includes information about where the method is located in the Java API libraries. The method body, also known as the "implementing code," is the underlying code that actually performs the task stated in the declaration. App., *infra*, 126a.

The relationship between the declaration and the implementing code is analogous to the interaction between a keyboard and a word-processing program. Just as a typist writes "a" by pressing a particular key, causing the word-processing program to display that letter, a developer triggers a particular function by using the relevant declaration to run the corresponding implementing code. By allowing developers easily to access the libraries of

prewritten code in a standard manner, the Java API facilitates the creation of programs in the Java language across different platforms, much as the now-standard QWERTY keyboard layout facilitates the creation of documents by enabling more efficient typing regardless of the specific word-processing program being used.

Once the corresponding declaration is used for a particular method, the developer does not need to worry about or even understand the specifics of the method's implementing code. Instead, all a developer needs to do to invoke a method is to use a shorthand command derived from the method's declaration. App., *infra*, 4a-5a, 126a-127a. By using the shorthand commands, a developer can create complex software without having to write new implementing code for every routine task. *Id.* at 4a.

3. In 2005, Google and Sun began discussing a partnership that would have allowed Google to adapt the entire Java SE platform for smartphones. Google and Sun conducted negotiations but were unable to reach an agreement. In the absence of such an agreement, Google used the freely available Java language (and its declarations) to develop its own libraries of methods that enabled developers to build smartphone applications for use on Android devices. App., *infra*, 106a-107a, 117a, 218a-219a.

At the same time, Google understood that developers would want to use their existing Java language skills to create Android applications, including their knowledge of familiar declarations and shorthand commands to trigger common operations. For those commands to work on the Android platform, Google had to replicate the syntax and structure of the Java API declarations *exactly*; any change to those declarations would have prevented developers from reusing the same commands, thereby forcing them to learn new commands for each routine task. Google accordingly used the same declarations for certain

methods in 37 Java API libraries that were determined by Google to be "key to mobile devices." App., *infra*, 219a. For every one of those methods, however, Google wrote its own implementing code, tailoring the code to accommodate the unique challenges of the smartphone environment. *Id.* at 218a-219a.

Because Google independently wrote the implementing code that formed the body of each method, using only certain declarations, only 3% of the code was the same across the 37 disputed Java API libraries and the corresponding Android libraries. App., *infra*, 220a. In total, that overlapping code represented less than 0.1% of the over 15 million relevant lines of code in Android.

B. Procedural History

1. Sun was aware that Google was developing Android using the Java language, including the API declarations, but never objected or mentioned its Java copyrights to Google. C.A. App. 50363, 51692-51693. To the contrary, Sun initially celebrated the launch of Android. Its chief executive officer publicly offered "heartfelt congratulations" to Google, stating that Google had "strapped another set of rockets to the [Java] community's momentum." *Id.* at 55325.

In 2010, however, Oracle acquired Sun. A few months after the acquisition, Oracle sued Google in the United States District Court for the Northern District of California, alleging seven counts of patent infringement. Oracle eventually withdrew five of its patent-infringement claims, and the jury found against Oracle on the two remaining claims.

Oracle's complaint also asserted a single claim of copyright infringement. Although the Java language was free and open, Oracle claimed that Google's use of the

Java API declarations infringed Oracle's copyrights. Oracle asserted that Google had impermissibly copied the declarations and also the "structure, sequence, and organization" of the Java API; Oracle premised the latter claim on the theory that the declarations "embod[ied] the structure" of the Java API by specifying the name and location of each method. App., *infra*, 140a.

After a two-week trial, the jury considered the copyright-infringement claims but was ultimately unable to reach a verdict, hanging on Google's fair-use defense. The district court then granted Google's motion for judgment as a matter of law on the copyright claims. App., *infra*, 212a-272a. The district court held that the Java API declarations were not copyrightable because they constituted a "method of operation" under 17 U.S.C. 102(b). App., *infra*, 262a-263a, 265a. The court further held that the declarations were not copyrightable under the merger doctrine, which provides that, "when there is only one (or only a few) ways to express something, then no one can claim ownership of such expression by copyright." *Id.* at 261a. The court also denied Oracle's motion for judgment as a matter of law on Google's fair-use defense. *Id.* at 211a.

2. The Federal Circuit reversed and remanded. App., *infra*, 121a-192a.

Recognizing a three-way circuit conflict on the copyrightability question, the Federal Circuit first reasoned that the merger doctrine was "irrelevant" to copyrightability and was in any event not satisfied here, because Sun could have written the declarations in more than one way. App., *infra*, 142a-143a, 148a, 150a-151a. The Federal Circuit then reasoned that Section 102(b) "does not extinguish the protection accorded a particular expression of an idea merely because that expression is embodied in a method of operation." *Id.* at 161a (internal quota-

tion marks and citation omitted). In the court's view, Section 102(b) served only to codify the "idea/expression dichotomy"—that is, the principle that "[c]opyright protection extends only to the expression of an idea—not to the underlying idea itself." *Id.* at 137a. The Federal Circuit remanded for a new trial on Google's fair-use defense, concluding that the record did not "contain[] sufficient factual findings upon which [the court] could base a de novo assessment." *Id.* at 184a.

3. Google petitioned for a writ of certiorari, and this Court called for the Solicitor General's views. The government acknowledged that Google's petition raised "substantial and important" concerns about the effects of enforcing Oracle's copyrights on software development, including lock-in effects and restrictions on interoperability. 14-410 U.S. Br. 10, 17. But the government recommended against certiorari, citing the case's then-interlocutory posture and noting that its concerns could be addressed on remand through the fair-use defense. See *id.* at 10, 22. This Court denied review. 135 S. Ct. 2887 (2015).

4. On remand, after another two-week trial featuring dozens of witnesses and hundreds of exhibits, the jury found that Google had engaged in fair use. The district court denied Oracle's motions for judgment as a matter of law and for a new trial. App., *infra*, 56a-120a.

5. The Federal Circuit again reversed and remanded. Having concluded in the first appeal that the fair-use defense should be decided by a jury because the panel could not resolve the underlying factual issues, the same panel reversed course and held that Google had not engaged in fair use as a matter of law. App., *infra*, 1a-55a.

The non-exclusive factors relevant to determining fair use include (1) the purpose and character of the use; (2) the nature of the copyrighted work; (3) the amount and

substantiality of the portion used in relation to the copyrighted work as a whole; and (4) the effect on the potential market for or value of the copyrighted work. See 17 U.S.C. 107. In holding that Google had not engaged in fair use, the Federal Circuit focused primarily on the first and fourth factors. App., *infra*, 25a-53a.

As to the first factor, the Federal Circuit determined that the commercial nature of Google's use of the declarations weighed against a finding of fair use. App., *infra*, 25a-28a. In considering whether Google's use was transformative, the Federal Circuit asserted that the declarations served the same function in Android as in the Java platform, and it concluded on that basis that the declarations themselves had not been transformed—even though Google used the declarations to create an entirely new smartphone platform and developed new implementing code tailored to the smartphone environment. *Id.* at 28a-37a.

As to the fourth factor, the Federal Circuit found that Java SE had been used in early mobile phones, which meant that "Android competed directly with [Java] in the market for mobile devices." App., *infra*, 50a. And even if Java SE had not been so used, the Federal Circuit would still have concluded that there was market harm by considering how "Google's copying affected potential markets Oracle *might* enter or derivative works it *might* create or license others to create." *Id.* at 51a (emphases added).

Weighing the four enumerated factors together, and without considering other relevant evidence as this Court has required, the Federal Circuit held that Google did not engage in fair use as a matter of law. App., *infra*, 53a-54a. Having overturned the jury's verdict, the court remanded for a trial on damages. *Id.* at 54a-55a.

6. After calling for a response, the Federal Circuit denied Google's petition for rehearing. App., *infra*, 283a-284a.

REASONS FOR GRANTING THE PETITION

The questions presented in this case are of critical importance to the computer software industry, one of the principal drivers of the nation's economy. Because new software builds on components of existing software, innovation in this field largely depends on how copyright law treats software interfaces, the essential building blocks of software development. The Federal Circuit has upended the computer industry's longstanding expectation that developers are free to use software interfaces to build new computer programs. In the opinions under review, the Federal Circuit first deemed software interfaces to be copyrightable, then held that petitioner's reuse of such interfaces could not be fair use as a matter of law because the interfaces performed the same function in the new software.

The Federal Circuit has deepened an existing circuit conflict over the copyrightability of software interfaces. Other courts of appeals have concluded that similar interfaces are not copyrightable under both the plain language of Section 102(b) of the Copyright Act and the merger doctrine. And as to fair use, the Federal Circuit misapplied the doctrine and rendered it essentially impossible for the reuse of software interfaces to qualify as fair use. The Court should review and correct the Federal Circuit's distortion of copyright law in an area crucial to technological innovation.

A. This Court Should Grant Review To Decide Whether Copyright Protection Extends To A Software Interface

The Federal Circuit's first opinion deepens acknowledged conflicts among the courts of appeals regarding the

proper interpretation of Section 102(b) of the Copyright Act and the application of the merger doctrine.

1. Under Section 102(b), copyright protection does not extend to "any idea, procedure, process, system, method of operation, concept, principle, or discovery, regardless of the form in which it is described, explained, illustrated, or embodied in such [original] work." 17 U.S.C. 102(b). More than two decades ago, the Court granted certiorari to consider whether that provision "bars protection for [a] menu command hierarchy despite its expressive characteristics, because it assists users in communicating with a computer program in order to perform useful operations." Br. at i, *Lotus Development Corp.* v. *Borland International, Inc.*, 516 U.S. 233 (1996) (No. 94-2003). The Court deadlocked 4-4 on that question after oral argument, and the division among the courts of appeals has only grown with the intervening rise of the modern software industry. Given its obvious importance and its close relation to the question left unresolved in *Lotus*, the question presented here cries out for the Court's review.

a. The First and Sixth Circuits have held that Section 102(b) precludes copyright protection for all methods of operation, including those embodied in computer software interfaces.

The First Circuit's decision in *Lotus* concerned the menu command hierarchy in Lotus 1-2-3, a then-ubiquitous spreadsheet program. See 49 F.3d 807, 809 (1995). The First Circuit acknowledged that "the Lotus developers made some expressive choices" in creating the hierarchy, but it nevertheless held that the hierarchy constituted a "method[] of operation" and was thus excluded from copyright protection under Section 102(b). *Id.* at 816. That was true regardless of whether the developers "could have designed the Lotus menu command hierarchy

differently." *Ibid.* Because the "menu command hierarchy provides the means by which users control and operate" the spreadsheet program, the hierarchy constituted a method of operation. *Id.* at 815.

The Sixth Circuit adopted a similar rule in *Lexmark International, Inc.* v. *Static Control Components, Inc.*, 387 F.3d 522 (2004). The court reasoned that, "even if a work is in some sense 'original' under § 102(a), it still may not be copyrightable because [of] § 102(b)." *Id.* at 534. The Sixth Circuit reaffirmed that understanding in *ATC Distribution Group, Inc.* v. *Whatever It Takes Transmissions & Parts, Inc.*, 402 F.3d 700 (2005). There, it explained that, although methods of operation may be "[o]riginal and creative," Section 102(b) excludes them from copyright protection because they are "the idea itself" rather than the "expression of the idea." *Id.* at 707.

b. For its part, the Third Circuit has taken the diametrically opposite position, holding that Section 102(b) was "not intended to enlarge or contract the scope of copyright protection" but rather to codify the "somewhat metaphysical" dichotomy between idea and expression. *Apple Computer, Inc.* v. *Franklin Computer Corp.*, 714 F.2d 1240, 1252, 1253 (1983), cert. denied, 464 U.S. 1033 (1984). In the Third Circuit's view, a "method of operation" embodied in a software interface is copyrightable as long as it could have been written differently and still serve the same high-level purpose, such as "to aid in the business operations of a dental laboratory." *Whelan Associates, Inc.* v. *Jaslow Dental Laboratory, Inc.*, 797 F.2d 1222, 1238 (1986), cert. denied, 479 U.S. 1031 (1987).

c. The Second Circuit has adopted still another approach: the so-called "abstraction/filtration/comparison" test. See *Computer Associates International, Inc.* v. *Altai, Inc.*, 982 F.2d 693, 706 (1992). Under that test, a court

should first "dissect the allegedly copied program's structure and isolate each level of abstraction contained within it." *Id.* at 707. The court should then "filter[] * * * protectable expression from non-protectable material." *Ibid.* Finally, after isolating the "golden nugget" of "protectable expression," the court should inquire whether "the defendant copied any aspect of this protected expression." *Id.* at 710. The Second Circuit's test has since been adopted by the Fifth and Tenth Circuits (with the Tenth Circuit expressly rejecting the First Circuit's approach in *Lotus*). See *Engineering Dynamics, Inc.* v. *Structural Software, Inc.*, 26 F.3d 1335, 1342 (5th Cir. 1994); *Engineering Dynamics, Inc.* v. *Structural Software, Inc.*, 46 F.3d 408, 409 (5th Cir. 1995) (supplemental opinion); *Mitel, Inc.* v. *Iqtel, Inc.*, 124 F.3d 1366, 1371-1372 (10th Cir. 1997).

d. In this case, the Federal Circuit purported to apply the Second Circuit's abstraction/filtration/comparison test. At the same time, however, the Federal Circuit also relied on the Third Circuit's more categorical test, which the Second Circuit's test was intended to replace. App., *infra*, 142a-143a, 161a-162a; see *Computer Associates*, 982 F.2d at 705-706.

In short, the courts of appeals are deeply divided on the appropriate standard for determining the circumstances under which a software interface is copyrightable under Section 102(b). At a minimum, the Federal Circuit's standard directly conflicts with the standard adopted by the First and Sixth Circuits. The Court should grant review to resolve the conflict among the courts of appeals on this exceptionally important issue.

2. In addition to deepening a circuit conflict regarding the proper interpretation of Section 102(b), the Federal Circuit took sides in a related circuit conflict concern-

ing the merger doctrine. That doctrine follows from Section 102(b)'s exclusion of ideas from the scope of copyright protection; it provides that, where an idea is incapable of being expressed in more than one way, the idea and expression "merge" and become unprotectable. 4 Melville B. Nimmer & David Nimmer, *Nimmer on Copyright* §13.03[B][3] (2015) (Nimmer).

The Federal Circuit concluded that the merger doctrine does not restrict copyright protection for computer code necessary for interoperability as long as the original author could have written the code in more than one way. App., *infra*, 150a-151a. In so concluding, the Federal Circuit aligned itself with the Third Circuit, which has stated that, "once the plaintiff creates a copyrightable work, a defendant's desire 'to achieve total compatibility * * * is a commercial and competitive objective which does not enter into the * * * issue of whether particular ideas and expressions have merged.'" *Id.* at 171a (quoting *Apple Computer*, 714 F.2d at 1253).

The position of the Third and Federal Circuits on the role of interoperability in the merger doctrine is directly contrary to that of the Sixth Circuit, which has found that "[p]rogram code that is strictly necessary to achieve current compatibility presents a merger problem, almost by definition, and is thus excluded from the scope of any copyright." *Lexmark*, 387 F.3d at 536 (internal quotation marks omitted).

The Federal Circuit further concluded that the merger doctrine plays no role in the copyrightability analysis and is instead merely an affirmative defense to *infringement* once copyrightability has been established. App., *infra*, 144a-145a. That position accords with holdings of two circuits, but it cannot be reconciled with the holdings of two others. See *Kregos* v. *Associated Press*, 937 F.2d 700, 705 (2d Cir. 1991), cert. denied, 510 U.S. 1112 (1992);

Ets-Hokin v. *Skyy Spirits, Inc.*, 225 F.3d 1068, 1082 (9th Cir. 2000); but see *Lexmark*, 387 F.3d at 535; *Veeck* v. *Southern Building Code Congress International, Inc.*, 293 F.3d 791, 801-802 (5th Cir. 2002) (en banc), cert. denied, 539 U.S. 969 (2003). That conflict is important in its own right, and it further supports this Court's review here.

3. The Federal Circuit erred in holding that the Java API declarations were copyrightable.

a. To begin with, the declarations constitute uncopyrightable methods of operation. Section 102(b) provides that "in no case does copyright protection for an original work of authorship extend to any * * * method of operation * * * regardless of the form in which it is described, explained, illustrated, or embodied in such work." 17 U.S.C. 102(b).

Ignoring the plain text of the statute, the Federal Circuit held that "components of a program that can be characterized as a 'method of operation' may nevertheless be copyrightable." App., *infra*, 161a. According to that court, "Section 102(a) and 102(b) are to be considered collectively so that certain expressions are subject to greater scrutiny." *Id.* at 141a-142a.

The Federal Circuit's approach is untenable, and its bottom line is incorrect. The Java API declarations simply tell developers how to access the prewritten methods to perform tasks carried out by the implementing code. App., *infra*, 4a-5a, 126a-127a. In that respect, the declarations are analogous to a set of rules developers are trained to follow when writing programs in the Java language. If the rules were changed, the prewritten methods would not work. For that reason, the declarations are necessarily part of the method of operating the libraries of prewritten code. See, *e.g.*, *Lotus*, 49 F.3d at 817-818.

The conclusion that the declarations are uncopyrightable is not affected by the fact that other aspects of the Java API libraries, like the implementing code, may be copyrightable. Quite to the contrary, "[t]he mere fact that a work is copyrighted does not mean that every element of the work may be protected." *Feist Publications, Inc.* v. *Rural Telephone Service Co.*, 499 U.S. 340, 348 (1991); see Pamela Samuelson, *Why Copyright Law Excludes Systems and Processes From the Scope of Its Protection*, 85 Tex. L. Rev. 1921, 1921 (2007). In the specific context of computer programs, the legislative history of Section 102(b) shows that Congress intended to "make clear that the expression adopted by the programmer is the copyrightable element in a computer program," while "the actual processes or methods embodied in the program are not within the scope of the copyright law." H.R. Rep. No. 1476, 94th Cong., 2d Sess. 56-57 (1976); S. Rep. No. 473, 94th Cong., 1st Sess. 54 (1975).

Consistent with Congress's expectation, courts have held that "aspects" of a computer program that constitute "functional requirements for compatibility" with other programs are "not protected by copyright" under Section 102(b). *Sega Enterprises Ltd.* v. *Accolade, Inc.*, 977 F.2d 1510, 1522 (9th Cir. 1992); see *Sony Computer Entertainment, Inc.* v. *Connectix Corp.*, 203 F.3d 596, 603 (9th Cir.), cert. denied, 531 U.S. 871 (2000). The Federal Circuit erred when it failed to draw a similar distinction here.

b. In holding that the declarations are copyrightable, the Federal Circuit also misapplied the merger doctrine.

That doctrine originated in this Court's seminal decision in *Baker* v. *Selden*, 101 U.S. (11 Otto) 99 (1880). The plaintiff had developed an accounting system and wrote a book explaining it. See *id.* at 100. His book included "certain forms or blanks, consisting of ruled lines, and headings, illustrating the system and showing how it is to be

used and carried out in practice." *Ibid.* The plaintiff contended that the forms were part of the book and therefore copyrightable. See *id.* at 101.

The Court rejected the plaintiff's argument. "The copyright of a work," the Court explained, "cannot give to the author an exclusive right to the methods of operation which he propounds, or to the diagrams which he employs to explain them." 101 U.S. at 103. A monopoly over those methods and diagrams could be secured only by patent law, not copyright, and in the absence of a patent, "any person may practise and use the art itself." *Id.* at 104. In short, "where the [useful] art [a work] teaches cannot be used without employing the methods and diagrams used to illustrate the book, or such as are similar to them, such methods and diagrams are to be considered as necessary incidents to the art, and given therewith to the public." *Id.* at 103.

In the context of computer programs, the merger doctrine "means that when specific instructions, even though previously copyrighted, are the only and essential means of accomplishing a given task, their later use by another will not amount to an infringement." National Commission on New Technological Uses of Copyrighted Works, *Final Report of the National Commission on New Technological Uses of Copyrighted Works*, 3 Computer L.J. 53, 74 (1981). But the Federal Circuit took a completely different approach. In its view, the merger doctrine was inapplicable because "alternative expressions [we]re available" for the ideas embodied in the declarations. App., *infra*, 151a. Under that approach, however, the merger doctrine would be a nullity in the software context: it is difficult to see how any idea embodied in computer code could ever merge with its expression, because there will always be an alternative way of naming and stating the rules for

a specific function (such as determining the larger of two numbers).

Baker illustrates why that approach cannot be correct. It made no difference to the Court's analysis whether the defendant could have performed accounting generally without the plaintiff's forms, or whether the defendant could have developed his own, analogous accounting method. The critical point was that, having created an accounting system, the plaintiff disclosed it to the public. And because the plaintiff's forms were necessary for the public to use that precise system, they could not be copyrighted.

Here, as the district court found, using the Java API declarations was the only way to allow independent developers to rely on their preexisting knowledge of the Java language when creating new programs. App., *infra*, 103a-105a, 263a. If Google had not replicated the declarations exactly, developers' code that is "clearly [their] own work product" and was written using the industry-standard Java shorthand commands would not have run on Android. *Lotus*, 49 F.3d at 818. The developers would have been locked into the Java platform (which Oracle controlled) and would have been unable to reuse their own code, or their knowledge of familiar interfaces and commands, on the Android platform (or any other).

The First Circuit's decision in *Lotus* strongly supports the foregoing analysis. The spreadsheet program at issue allowed users to write customized programs, or "macros," that enabled them to execute a series of commands automatically by typing a single pre-programmed keystroke. See *Lotus*, 49 F.3d at 809. The defendant enabled prospective customers who had created their own macros in Lotus 1-2-3 to switch to its competing spreadsheet program without learning new commands or rewriting their macros. See *id.* at 810.

In *Lotus*, the First Circuit rejected the notion that Lotus could compel a user to "rewrite his or her macro using [another] program's menu command hierarchy." 49 F.3d at 818. As the court recognized, "forcing the user to cause the computer to perform the same operation in a different way ignores Congress's direction in § 102(b) that 'methods of operation' are not copyrightable." *Ibid.*; see *id.* at 819-820 (Boudin, J., concurring). So too here: the Federal Circuit's decision to extend copyright protection to the Java API declarations effectively grants Oracle a patent-like monopoly over the Java language.

In rejecting that reasoning, the Federal Circuit relied on the premise that Google could have given different names to Android's methods and libraries. App., *infra*, 153a-154a. Because Oracle had already selected names for Java's methods and libraries, however, the declarations could be written only in one way to permit Java-fluent developers to use the familiar shorthand commands. In any event, as the district court noted, it is well settled that such names and short phrases are not copyrightable. *Id.* at 264a.

The Federal Circuit compounded the error in its merger analysis by focusing exclusively on the choices available to Sun, Oracle's predecessor, at the time it created Java. With respect to the merger inquiry under Section 102(b), the question is "not whether *any* alternatives theoretically exist"; instead, it is "whether other options practically exist under the circumstances" and are "feasible within real-world constraints." *Lexmark*, 387 F.3d at 536.

The Federal Circuit should have taken account of the expressive choices available to *Google* when it created Android. Only then could it properly evaluate Google's claim that duplicating aspects of the Java API declarations was necessary to allow developers to create applications for

the Android platform using the free Java language, with the same functionality they were taught by Sun to expect of that language. See Pamela Samuelson, *Questioning Copyrights in Standards*, 48 B.C. L. Rev. 193, 215 (2007); *Lotus*, 49 F.3d at 819-820 (Boudin, J., concurring). For that additional reason, the Federal Circuit's merger analysis was deeply flawed, and its holding that the Java API declarations were copyrightable warrants further review.

B. This Court Should Grant Review To Decide Whether, As The Jury Found, Petitioner's Use Of A Software Interface In The Context Of Creating A New Computer Program Constitutes Fair Use

Even assuming that the Java API declarations were copyrightable, Google engaged in fair use when it used some of those declarations. In holding that Google's use was not fair as a matter of law, the Federal Circuit misapplied the precedents of this Court and others on the fair-use doctrine. In particular, the Federal Circuit failed to account for the functional nature of software interfaces; took an unduly constrained view of transformative use; and rendered the market-harm factor an essentially circular inquiry.

The Federal Circuit not only misapplied the fair-use doctrine; relying on its own (unsupported) findings, it also overturned a jury verdict along the way. In the earlier appeal in this case, the Federal Circuit remanded the fair-use issue to the jury out of "due respect for the limit of [its] appellate function." App., *infra*, 182a. After the retrial, the jury issued a verdict in Google's favor on fair use. At that point, the Federal Circuit did an about-face, taking the highly unusual step of setting aside the jury's verdict and deciding fair use in Oracle's favor as a matter of law. *Id.* at 53a-54a.

Because the jury returned a general verdict on fair use, the Federal Circuit correctly stated that it "must assume that the jury resolved all factual issues relating to the historical facts in favor of the verdict." App., *infra*, 23a. But the Federal Circuit said one thing and did another: it reconsidered for itself a number of factual issues presented to the jury and resolved those issues in support of the conclusion that Google's use was unfair as a matter of law. See p. 28, *infra*. To permit that approach would condone an unprecedented degree of appellate second-guessing of factual determinations in fair-use cases. This Court's intervention is urgently warranted to rectify the Federal Circuit's profoundly flawed approach.

1. The fair-use doctrine has long been a cornerstone of copyright law. That exception to copyright infringement, now codified in 17 U.S.C. 107, grew out of the recognition that new works "must necessarily borrow[] and use much which was well known and used before." *Campbell* v. *Acuff-Rose Music, Inc.*, 510 U.S. 569, 575 (1994) (citation omitted). As a result, "[f]rom the infancy of copyright protection, some opportunity for fair use of copyrighted materials has been thought necessary to fulfill copyright's very purpose." *Ibid.* To that end, the fair-use doctrine seeks to balance the "need simultaneously to protect copyrighted material and to allow others to build upon it." *Ibid.*

Four non-exclusive factors guide the fair-use analysis: (1) the purpose and character of the use; (2) the nature of the copyrighted work; (3) the substantiality of the portion used in relation to the copyrighted work as a whole; and (4) the effect on the potential market for or value of the copyrighted work. 17 U.S.C. 107. Those factors are to be "weighed together[] in light of the purposes of copyright." *Campbell*, 510 U.S. at 578. The ultimate aim in applying those factors is to "avoid rigid application of the copyright

statute" when "it would stifle the very creativity which that law is designed to foster." *Stewart* v. *Abend*, 495 U.S. 207, 236 (1990).

2. In its opinion on fair use, the Federal Circuit engaged in precisely the kind of rigid application that this Court warned against. It made at least three critical errors, one that infected the entire analysis and two that pertained to the fair-use factors that drove its reasoning—the purpose and character of the use and the effect on the actual or potential market.

a. As a matter of overall approach, the Federal Circuit failed to adapt the fair-use doctrine to the functional nature of software interfaces. Each software interface is designed to trigger the performance of a specific operation, such as finding the larger of two numbers. In considering the second fair-use factor, the nature of the copyrighted work, the Federal Circuit acknowledged that "functional considerations were both substantial and important" and thus that the second factor "favors a finding of fair use." App., *infra*, 42a. But the Federal Circuit ignored how the functional nature of software interfaces affects the fair-use analysis as a whole.

A functional work is entitled only to thin copyright protection—which, in turn, makes it easier to prove a noninfringing, fair use of that work. This Court has recognized that "some works are closer to the core of intended copyright protection than others, with the consequence that fair use is more difficult to establish when the former works are copied." *Campbell*, 510 U.S. at 586. Unlike literary or artistic works, software interfaces "perform[] functions that are not entitled to copyright protection." *Sony*, 203 F.3d at 602. In light of that functional character, interfaces lie "at a distance from the core" of copy-

right protection and are thus owed a "lower degree of protection than more traditional literary works." *Id.* at 603; see *Sega*, 977 F.2d at 1527.

In its opinion on fair use, the Federal Circuit seemingly took a contrary position. While recognizing the functional nature of software interfaces, the Federal Circuit gave them the same copyright protection—and, as is relevant here, the same fair-use treatment—afforded to literary and artistic works. The Federal Circuit thus systematically erred when it discounted the particular characteristics of software interfaces.

b. Beyond that overarching flaw, the Federal Circuit applied the fair-use factors incorrectly and too rigidly. That is most apparent in its consideration of the first fair-use factor, the purpose and character of the use—and, in particular, a component of that factor, transformative use. Transformative use focuses on "whether the new work merely supersede[s] the objects of the original creation, or instead adds something new, with a further purpose or different character, altering the first with new expression, meaning, or message." *Campbell*, 510 U.S. at 579 (internal quotation marks and citations omitted).

The Federal Circuit's most fundamental mistake in analyzing transformative use was in fixating only on the material that Google reused (certain Java API declarations) and assessing whether that material was itself transformed in the new work. App., *infra*, 32a-37a. That is not the correct inquiry. As numerous courts have indicated, "a secondary work can be transformative in function or purpose without altering or actually adding to the original work." *American Society for Testing & Materials* v. *Public.Resource.Org, Inc.*, 896 F.3d 437, 450 (D.C. Cir. 2018); see *Swatch Group Management Services Ltd.* v. *Bloomberg L.P.*, 756 F.3d 73, 84 (2d Cir. 2014); *Seltzer* v. *Green Day*, 725 F.3d 1170, 1177 (9th Cir. 2013).

Instead, the correct inquiry focuses on the "new work" *as a whole*, asking whether the "new work * * * adds something new [to the copyrighted work], with a further purpose or different character." *Campbell*, 510 U.S. at 579. That new purpose or character, in turn, informs the degree and nature of the transformation necessary to qualify as fair use: for example, a parody "needs to mimic an original to make its point," which may include copying the heart of the original work to "make the object of its critical wit recognizable." *Id.* at 580-581, 588.

Here, both the new work Google created and its use of the Java API declarations were undoubtedly transformative. Google set out to create an entirely new platform for smartphones. In contrast, the original work, Java SE, was designed for desktop and server computers. The new Android platform would have to accommodate resource constraints, such as limited memory and battery life, that did not apply to Java SE. See p. 4, *supra*. The Federal Circuit therefore erred when it suggested that Google's new platform merely changed existing computer code from one medium or format to another. App., *infra*, 35a-37a.

The new work also used the Java API declarations in a transformative way. The declarations formed only a small part of the new work. Because the Android platform needed to be tailored to a new smartphone environment, Google had to develop all of its own implementing code for Android, including new implementing code for the Java API declarations. App., *infra*, 112a, 114a. Google created entirely new libraries, including new declarations and implementing code, for functions necessary to operate modern smartphones, such as touchscreens, web browsing, the built-in camera, and location awareness.

To be sure, Google did incorporate certain Java API declarations from Java SE. That limited incorporation allowed developers to use the Java language to build applications for Android. By the same token, Google's use of the declarations prevented Oracle from locking in developers familiar with the Java language into building applications only for Oracle's platforms. Notably, the government recognized those "legitimate concerns" of lock-in effects and interoperability in addressing Google's earlier petition for certiorari in this case, and it highlighted fair use as the proper way to accommodate those concerns. 14-410 U.S. Br. 17.

Such interoperability was critical for developers programming in the Java language. At Sun's and Oracle's encouragement, developers had invested in learning the Java language and had grown accustomed to using the well-known shorthand commands derived from the Java API declarations. The district court likened those declarations to the keys on a QWERTY keyboard. App., *infra*, 104a. Developers therefore wanted to use the Java API declarations to write code for Android applications in the Java language. To allow such code to run on Android, Google had to incorporate the applicable Java API declarations. By allowing applications written in the Java language to operate in the new environment, those declarations took on a "further purpose or different character" in Android that they did not have in Java SE. *Campbell*, 510 U.S. at 579.

In the Federal Circuit's view, Google's incorporation of the software interfaces into a new work was effectively irrelevant for fair-use purposes, because the Java API declarations performed the same function in the original and new works. App., *infra*, 33a. That is a dangerous misapplication of the fair-use doctrine with breathtakingly broad implications. If a mere identity of function were

enough to preclude fair use, the reuse of any preexisting computer code in new software would never fall within the fair-use defense. That is because computer code is essentially a set of instructions that performs the same function whenever it is used. See, *e.g.*, *Sega*, 977 F.2d at 1524.

Here, the Java API declarations operated via entirely new implementing code in Android. For instance, the declaration may call for something to be displayed, and the corresponding implementing code would display the output on the touchscreen of a smartphone, rather than the monitor of a desktop computer. The reuse of the Java API declarations in Android simply permitted developers to program Android applications in the free and open Java language.

The Federal Circuit effectively dismissed the concerns regarding lock-in effects and interoperability that the government has recognized as "substantial and important," 14-410 U.S. Br. 17, and that other courts have similarly emphasized, see, *e.g.*, *Sega*, 977 F.2d at 1526-1527. If Google had used entirely different interfaces in Android, developers would have had to learn the new interfaces to operate the prewritten methods that they already knew from the Java language. Developers could not have used the familiar, industry-standard Java shorthand commands to build Android applications and would therefore have been deterred from doing so.

As a practical matter, then, precluding Google's use of the Java API declarations would permit Oracle to accrue market power via copyright, locking in developers that had invested in learning the Java language and making it difficult for them to use those skills to program for new platforms. See *Lotus*, 49 F.3d at 821 (Boudin, J., concurring). Oracle's control, in turn, would effectively block competing platforms from accessing developers trained in the Java language. Because interfaces are central to how

developers operate software, control over interfaces gives rise to barriers to entry and implicates issues of competition and innovation that warrant this Court's review.

c. The Federal Circuit's approach to the fourth fair-use factor—the effect on the actual or potential market for the copyrighted work—was similarly flawed. As this Court has explained, the fair-use doctrine applies to "copying by others which does not materially impair the marketability of the work which is copied." *Harper & Row Publishers, Inc.* v. *Nation Enterprises*, 471 U.S. 539, 566-567 (1985). The Federal Circuit concluded that Google's use of the Java API declarations caused harm to Oracle's *actual* market because Java SE was purportedly also used in mobile phones before Android's debut. App., *infra*, 50a-51a. And even if there was no actual harm, the Federal Circuit concluded that there was potential harm because smartphones were a "traditional, reasonable, or likely to be developed market" for Java SE. *Id.* at 51a-52a.

The Federal Circuit reached those conclusions only by improperly revisiting and reversing the jury's implicit factual determinations. Whether and to what extent Java SE was used in mobile devices were disputed issues below. The Federal Circuit acknowledged that it "must assume that the jury resolved all factual issues relating to the historical facts in favor of the verdict." App., *infra*, 23a. Yet it rejected that assumption and instead made its own (erroneous) determination that Java SE was in fact used in early mobile devices comparable to Android before Android's release. See *id.* at 50a-51a. If the Federal Circuit had deferred to the jury, as it was required to do, it would have concluded that Java SE was never used in a modern smartphone and that Java SE and Android occupied different markets, which meant that there was no actual market harm.

The Federal Circuit further concluded that, simply because Oracle *could* have tried to adapt Java SE for use in smartphones, Google's use of the Java API declarations in a mobile platform caused harm to a potential market. App., *infra*, 51a-52a. To find market harm on that basis, however, is entirely circular. The fair-use issue arose in this case precisely because Google did not "pay a fee for the right to * * * use" the declarations. *Bill Graham Archives* v. *Dorling Kindersley Ltd.*, 448 F.3d 605, 614 (2d Cir. 2006). If the potential market for the copyrighted material is defined to include the market for licensing the exact use at issue, then potential market harm is baked into the market definition. Courts and commentators alike have warned against watering down the market-harm inquiry in that way. See, *e.g.*, 4 Nimmer § 13.05[A][4]; *Swatch Group*, 756 F.3d at 91.

In sum, the Federal Circuit's fair-use analysis is replete with errors and cannot be reconciled with the decisions of this Court and others. Further review is warranted on the fair-use question, as well as the copyrightability question.

C. The Questions Presented Are Exceptionally Important And Warrant Review In This Case

1. Above and beyond the broader implications for copyright law, this case warrants the Court's attention for its sheer practical importance. Repeatedly hailed as the "copyright lawsuit of the decade," this case presents a conflict between two giants of the technology industry, Google and Oracle. At the center of this dispute is Android, a platform used worldwide by billions of users. And what Oracle is seeking here is nothing less than complete control over a community of developers that have invested in learning the free and open Java language. That effort

aims to stifle rather than encourage the creation of new works.

Given the enormous stakes, it is unsurprising that this case has drawn widespread attention, including from the Court. The last time that Google sought review in this case—when the sole question involved copyrightability—this Court called for the views of the Solicitor General. While the government advised against granting certiorari while the case was in an interlocutory posture, it noted "substantial and important concerns" that in its view should be addressed through the fair-use doctrine. 14-410 U.S. Br. 17. Numerous industry groups, academics, and other interested parties filed amicus briefs in the proceedings below. And numerous commentators have highlighted the widespread impact of the Federal Circuit's decision and stressed the need for this Court's intervention. See, *e.g.*, Tony Dutra, *Oracle Victory in Copyright Case Has Seeds for a Google Appeal*, Bloomberg Law (Mar. 28, 2018) <tinyurl.com/dutraarticle>.

As those amici and commentators have warned, if allowed to stand, the Federal Circuit's approach would have a devastating impact on the development of computer software. Although this case involves software interfaces, the Federal Circuit's reasoning has implications for all computer code. In particular, if code must perform a different function in a new work for the fair-use defense to apply, then a developer will be foreclosed from reusing copyrighted code designed to execute one particular function.

The Federal Circuit's reasoning threatens the prevailing approach to building computer software. Developers are not coding programs entirely from scratch, as they may have been in the early days of programming. Instead, new programs now incorporate and rely on preexisting interfaces to trigger certain functions, which saves

the wasted effort of reinventing and retesting what came before. See David Orenstein, *Application Programming Interface*, ComputerWorld (Jan. 10, 2000) <tinyurl.com/orensteinarticle>. If the Federal Circuit's approach is allowed to stand, developers will be forced to abandon their traditional building-block approach to software development. At the very least, they will be left in confusion about whether and when their longstanding practices constitute copyright infringement.

Not only does the Federal Circuit's approach wreak havoc on copyright law, but it also risks disturbing the balance between copyright law and patent law, the two principal bodies of law that govern innovation. The Federal Circuit has effectively provided blanket copyright protection to an entire class of computer code. It has done so despite repeated warnings from other courts that patent law may be better equipped to address the functional aspects of computer code. See, *e.g.*, *Sony*, 203 F.3d at 605; *Computer Associates*, 982 F.2d at 712; *Sega*, 977 F.2d at 1526. As those courts have pointed out, if the creator of computer code "wishes to obtain a lawful monopoly on the *functional concepts* in its software, it must satisfy the more stringent standards of the patent laws," including novelty and nonobviousness. *Sony*, 203 F.3d at 605 (emphasis added). In contrast, "copyright registration—with its indiscriminating availability—is not ideally suited to deal with the highly dynamic technology of computer science." *Computer Associates*, 982 F.2d at 712.

The Federal Circuit's approach pays no heed to those warnings. In the opinions under review, the Federal Circuit afforded software interfaces a government-granted monopoly based on a more relaxed standard and for a much longer period than permitted by patent law. Software interfaces—the critical building blocks of software development—can now be kept out of the public domain

for at least 70 years after the creator's death. This Court should closely scrutinize the Federal Circuit's expansion of copyright law into the traditional territory of the patent system.

2. This case is an ideal vehicle for considering the questions presented. The Federal Circuit has now squarely held both that software interfaces are copyrightable and that petitioner's reuse of such interfaces did not constitute fair use as a matter of law. The questions presented have been exhaustively briefed by the parties and their amici below.

There would be no material benefit from further percolation on the questions presented. As to copyrightability, there has been a persistent circuit conflict, and the arguments for both sides have been amply considered by the courts of appeals. See pp. 12-17, *supra*. And as to fair use, the numerous errors in various aspects of the Federal Circuit's analysis warrant the Court's intervention. That is particularly true because a copyright holder will be able to invoke the Federal Circuit's jurisdiction through the simple expedient of including a patent claim. See, *e.g.*, Joe Mullin, *Cisco v. Arista Awaits a Jury Verdict Under the Oracle v. Google Shadow*, ArsTechnica (Dec. 14, 2016) <tinyurl.com/y9xxd4zf> (noting the use of the same strategy in another case).

In short, the Federal Circuit has deepened a widely recognized conflict on the copyrightability of software interfaces and effectively excluded their reuse from the fair-use defense. The courts of appeals have taken divergent approaches to the application of copyright law to computer software, a key driver of technological innovation. The Federal Circuit should not have the final word in this landmark case. This Court's review is unquestionably warranted.

CONCLUSION

The petition for a writ of certiorari should be granted.

Respectfully submitted.

THOMAS C. GOLDSTEIN
GOLDSTEIN & RUSSELL, P.C.
5225 Wisconsin Avenue, N.W., Suite 404
Washington, DC 20015

ROBERT A. VAN NEST
CHRISTA M. ANDERSON
EUGENE M. PAIGE
REID P. MULLEN
KEKER, VAN NEST & PETERS LLP
633 Battery Street
San Francisco, CA 94111

BRUCE W. BABER
MARISA C. MALECK
KING & SPALDING LLP
1180 Peachtree Street, N.E.
Atlanta, GA 30309

KANNON K. SHANMUGAM
CHARLES L. MCCLOUD
MENG JIA YANG
WILLIAMS & CONNOLLY LLP
725 Twelfth Street, N.W.
Washington, DC 20005
(202) 434-5000
kshanmugam@wc.com

MICHAEL S. KWUN
KWUN BHANSALI LAZARUS LLP
555 Montgomery Street, Suite 750
San Francisco, CA 94111

JANUARY 2019

APPENDIX

TABLE OF APPENDICES

Appendix A

United States Court of Appeals
for the Federal Circuit

ORACLE AMERICA, INC.,

Plaintiff-Appellant,

v.

GOOGLE LLC,

Defendant-Cross-Appellant.

2017-1118, 2017-1202

Appeals from the United States District Court for the Northern District of California in No. 3:10-cv-03561-WHA, Judge William H. Alsup.

Decided: March 27, 2018

---

* * *

Before: O'MALLEY, PLAGER, and TARANTO, *Circuit Judges.*

O'MALLEY, Circuit Judge.

This copyright case returns to us after a second jury trial, this one focusing on the defense of fair use. Oracle America, Inc. ("Oracle") filed suit against Google Inc.

("Google")[1] in the United States District Court for the Northern District of California, alleging that Google's unauthorized use of 37 packages of Oracle's Java application programming interface ("API packages") in its Android operating system infringed Oracle's patents and copyrights.

At the first trial, the jury found that Google infringed Oracle's copyrights in the Java Standard Edition platform, but deadlocked on the question of whether Google's copying was a fair use.[2] After the verdict, however, the district court found that the API packages were not copyrightable as a matter of law and entered judgment for Google. *Oracle Am., Inc. v. Google Inc.*, 872 F. Supp. 2d 974 (N.D. Cal. 2012). Oracle appealed that determination to this court, and we reversed, finding that declaring code and the structure, sequence, and organization ("SSO") of the Java API packages are entitled to copyright protection. *Oracle Am., Inc. v. Google Inc.*, 750 F.3d 1339, 1348 (Fed. Cir. 2014). We remanded with instructions to reinstate the jury's infringement verdict and for further proceedings on Google's fair use defense and, if appropriate, on damages. *Id.* at 1381.

Google subsequently filed a petition for certiorari on the copyrightability determination. The Supreme Court called for the views of the Solicitor General, who expressed agreement with our determination and recommended denying review. The Supreme Court

---

[1] In September 2017, Google converted from a corporation to a limited liability company and changed its name to Google LLC, as reflected in the amended caption.

[2] The jury found no patent infringement, and the patent claims are not at issue on appeal.

denied *certiorari* in 2015. *Google Inc. v. Oracle Am., Inc., 135 S. Ct. 2887 (2015) (Mem.).*

At the second jury trial, Google prevailed on its fair use defense. After the jury verdict, the district court denied Oracle's motion for judgment as a matter of law ("JMOL") and entered final judgment in favor of Google. *Oracle Am., Inc. v. Google Inc.*, No. C 10-03561, 2016 WL 3181206 (N.D. Cal. June 8, 2016) ("*Order Denying JMOL*"); Final Judgment, *Oracle Am., Inc. v. Google Inc.*, No. 3:10-cv-3561 (N.D. Cal. June 8, 2016), ECF No. 1989.

Oracle filed a renewed motion for JMOL and separately moved for a new trial. The district court denied both motions in a single order. *Oracle Am., Inc. v. Google Inc.*, No. C 10-03561, 2016 WL 5393938 (N.D. Cal. Sept. 27, 2016) ("*Order Denying Renewed JMOL/New Trial*"). Consistent with these determinations, no damages verdict was rendered.

Oracle now appeals from the district court's final judgment and its decisions denying Oracle's motions for JMOL and motion for a new trial. Google cross-appeals from the final judgment purportedly to "preserv[e] its claim that the declarations/SSO are not protected by copyright law," but advances no argument for why this court can or should revisit our prior decision on copyrightability. Cross-Appellant Br. 83.

Because we conclude that Google's use of the Java API packages was not fair as a matter of law, we reverse the district court's decisions denying Oracle's motions for JMOL and remand for a trial on damages. We also dismiss Google's cross-appeal.

I. BACKGROUND

A. The Technology

Oracle's predecessor, Sun Microsystems, Inc. ("Sun"), developed the Java platform for computer programming in the 1990s, and Oracle purchased Sun in 2010. The Java platform is software used to write and run programs in the Java programming language. It allows programmers to write programs that "run on different types of computer hardware without having to rewrite them for each different type." *Oracle*, 750 F.3d at 1348. With Java, programmers can "write once, run anywhere." *Id.*

The Java 2 Standard Edition ("Java SE") of the platform includes, among other things, the Java Virtual Machine and the Java Application Programming Interface ("API"). The Java API is a collection of "pre-written Java source code programs for common and more advanced computer functions." *Order Denying JMOL*, 2016 WL 3181206, at *3. These APIs "allow programmers to use the prewritten code to build certain functions into their own programs rather than write their own code to perform those functions from scratch. They are shortcuts." *Oracle*, 750 F.3d at 1349. The prewritten programs are organized into packages, classes, and methods. Specifically, an API package is a collection of classes and each class contains methods and other elements. "Each method performs a specific function, sparing a programmer the need to write Java code from scratch to perform that function." *Order Denying JMOL*, 2016 WL 3181206, at *3.

To include a particular function in a program, the programmer invokes the Java "declaring code." As the district court explained, the declaring code is the line or

lines of source code that "declares or defines (i) the method name and (ii) the input(s) and their type as expected by the method and the type of any outputs." *Id.* at *4. After the declaring code, each method includes "implementing code," which takes the input(s) and gives the computer step-by-step instructions to carry out the declared function.

By 2008, Java SE included 166 API packages divided into 3,000 classes containing more than 30,000 methods. At issue in this appeal are 37 API packages from Java SE Version 1.4 and Version 5.0. We have already concluded that the declaring code and the SSO of the 37 Java API packages at issue are entitled to copyright protection. *Oracle*, 750 F.3d at 1348.

The Java programming language itself is free and available for use without permission. At this stage, it is undisputed that, to write in the Java programming language, "62 classes (and some of their methods), spread across three packages *within* the Java API library, *must* be used. Otherwise the language itself will fail." *Order Denying JMOL*, 2016 WL 3181206, at *5. It is also undisputed that anyone using the Java programming language can write their own library of prewritten programs to carry out various functions.

Although Oracle makes the Java platform freely available to programmers building applications ("apps"), it devised a licensing scheme to attract programmers while simultaneously commercializing the platform. In relevant part, Oracle charges a licensing fee to those who want to use the APIs in a competing platform or embed them in an electronic device. To preserve the "write once, run anywhere" philosophy, Oracle imposes strict compatibility requirements on licensees. *Oracle*, 750 F.3d

at 1350. Oracle also made available without charge under an open source license a version of Java called "OpenJDK." *Order Denying JMOL*, 2016 WL 3181206, at *10. Oracle maintains, however, that OpenJDK came with an important catch: any company that improved on the packages in OpenJDK had to " 'give away those changes for free' to the Java community." Appellant Br. 53.

The evidence showed that Oracle licensed Java in 700 million PCs by 2005. Although Oracle never successfully developed its own smartphone platform using Java, it licensed Java SE for mobile devices. According to Oracle, the "mobile device market was particularly lucrative," and "Java quickly became the leading platform for developing and running apps on mobile phones." Appellant Br. 9.

B. Google's Android Platform

In 2005, Google acquired Android, Inc. as part of a plan to develop a software platform for mobile devices. That same year, Google and Sun began discussing the possibility of Google taking a license to use and adapt the Java platform for mobile devices. *Oracle*, 750 F.3d at 1350. The parties were unable to reach an agreement, in part because Google wanted device manufacturers to be able to use Oracle's APIs in Android for free with no limits on modifying the code, which would jeopardize the "write once, run anywhere" philosophy.

The jury heard evidence that Google wanted to move quickly to develop a platform that would attract Java developers to build apps for Android. The Android team had been working on creating its own APIs, but was unable to do so successfully. After negotiations between the parties reached an impasse, Google elected to "[d]o Java anyway and defend [its] decision, perhaps making

enemies along the way." *Order Denying JMOL*, 2016 WL 3181206, at *6. It is undisputed that Google copied verbatim the declaring code of the 37 Java API packages— 11,500 lines of Oracle's copyrighted code. It also copied the SSO of the Java API packages. Google then wrote its own implementing code.

Google announced its Android software platform for mobile devices in 2007, and the first Android phones went on sale the following year. Google provides the Android platform free of charge to smartphone manufacturers and publishes the source code for use without charge under an open source license. Although Google does not directly charge its users, Android has generated over $42 billion in revenue from advertising. Oracle explains that Android was "devastating" to its licensing strategy and that many of its customers switched to Android. Appellant Br. 15. Even customers who stayed with Oracle cited Android as a reason to demand discounts. The jury heard evidence that Amazon, which had entered into a license to use Java for its Kindle tablet device, switched to Android for the subsequently released Kindle Fire and then used the existence of Android to leverage a steep discount from Oracle on the next generation Kindle.

C. Remand Proceedings

In the first appeal, we held that the declaring code and the SSO of the 37 API packages are entitled to copyright protection and ordered the district court to reinstate the jury's infringement finding. *Oracle*, 750 F.3d at 1381. We also considered Oracle's argument that it was entitled to judgment as a matter of law on Google's fair use defense. Although we found that Oracle's position was "not without force," and that Google was overstating what could be fair use under the law, we found that the record evidence

regarding the relevant fair use factors was insufficiently developed for us to resolve the issue on appeal. *Oracle*, 750 F.3d at 1376. In doing so, we pointed to sharp disputes between the parties, both legal and factual, including whether Google's use was transformative, whether "functional aspects of the package" and Google's "desire to achieve commercial 'interoperability'" weighed in favor of the second and third factors, and whether Android caused market harm to Oracle. *Id.* at 1376-77. We concluded that "due respect for the limit of our appellate function" required remand. *Id.* at 1376.

During the pendency of the first appeal, Google's Android business expanded significantly. Android gained new users and developers, and Google "released modified implementations and derivatives of Android for use in numerous device categories, including wearable devices with small screens (Android Wear), dashboard interfaces in cars (Android Auto), television sets (Android TV), and everyday devices with Internet connectivity." *Oracle Am., Inc. v. Google Inc.*, No. C10-03561, 2016 WL 1743111, at *1 (N.D. Cal. May 2, 2016) ("*Order on Motion in Limine*").

When the case returned to the district court, Oracle filed a supplemental complaint adding allegations of market harm and damages resulting from new versions of Android released since the original complaint. Specifically, Oracle alleged that Google had launched new versions of Android for phones and tablets and had expanded Android into new device categories. *Id.* Google did not oppose the supplemental complaint, and the district court granted Oracle's motion to file it. But when Oracle served expert reports that addressed versions of

Java SE that were not at issue in the first trial, Google moved to strike those reports. *Id.*

When the parties were unable to agree on the scope of the retrial, the district court limited it to: (1) the two versions of Java SE that Oracle asserted in the first trial; and (2) released versions of Android used in smartphones and tablets "which Google . . . agreed would be subject to the prior jury's adverse finding of infringement and which Oracle identified in its supplemental complaint." *Id.* The court explained that Oracle retained the right to sue Google for infringement with respect to the other versions and implementations of Android in a separate trial or proceeding. Order re: Google's Motion to Strike at 2, *Oracle Am., Inc. v. Google Inc.*, No. 3:10-cv-3561 (N.D. Cal. Feb. 5, 2016), ECF No. 1479. The court also granted Google's motion in limine to exclude all evidence of the new Android products.

The district court bifurcated the issue of fair use from willfulness and monetary remedies, and the trial on fair use began on May 10, 2016. After roughly one week of evidence and several days of deliberations, the jury found that Google's use of the declaring lines of code and the SSO of the 37 API packages constituted fair use.

Oracle moved for JMOL, which the district court denied. At the outset, the court noted that Oracle stipulated before the jury "that it was fair to use the 62 'necessary' classes given that the Java programming language itself was free and open to use without a license." *Order Denying JMOL*, 2016 WL 3181206, at *5. "That the 62 'necessary' classes reside without any identification as such within the Java API library (rather than reside within the programming language)," the court explained, "supports Google's contention that the Java API library is

simply an extension of the programming language itself and helps explain why some view the Java API declarations as free and open for use as the programming language itself." *Id.* Because Android and Java both "presupposed the Java programming language in the first place," the court noted that a jury reasonably could have found that it "was better for both to share the same SSO insofar as they offered the same functionalities, thus maintaining usage consistency across systems and avoiding cross-system confusion." *Id.* at *6.

The district court then considered each of the four statutory fair use factors. As to factor one—the purpose and character of the use—the court concluded that a reasonable jury could have found that, although Google's use was commercial, it was transformative because Google integrated only selected elements for mobile smartphones and added its own implementing code. *Id.* at *7-9. With respect to factor two—the nature of the copyrighted work—the district court found that a reasonable jury could have concluded that, "while the declaring code and SSO were creative enough to qualify for copyright protection," they were not "highly creative," and that "functional considerations predominated in their design." *Id.* at *10.

As to factor three—the amount and substantiality of the portion used—the court concluded that a reasonable jury could have found that "Google copied only so much as was reasonably necessary for a transformative use," and that the number of lines duplicated was minimal. *Id.* Finally, as to factor four—market harm—the court concluded that the jury "could reasonably have found that use of the declaring lines of code (including their SSO) in Android caused no harm to the market for the copyrighted

works, which were for desktop and laptop computers." *Id.* The court determined that, on the record presented, the jury could have found for either side and that the jury was "reasonably within the record in finding fair use." *Id.* at *11.

Oracle subsequently renewed its motion for JMOL and separately moved for a new trial challenging several of the court's discretionary decisions at trial. The district court denied both motions in a single order. With respect to JMOL, the court simply stated that it denied Oracle's renewed motion for the same reasons it denied the original motion. With respect to the motion for a new trial, the court rejected Oracle's argument that the court abused its discretion by limiting the evidence at trial to Google's use of Android in smartphones and tablets.

The court also rejected Oracle's allegation that Google engaged in discovery misconduct by withholding evidence during discovery relating to Google's App Runtime for Chrome ("ARC"), which enabled laptops and desktops running Google's computer operating system to run certain Android applications. *Order Denying Renewed JMOL/New Trial*, 2016 WL 5393938, at *5. The court found that Google had produced relevant documents during discovery and that, in any event, those documents pertained to issues beyond the scope of the retrial. *Id.* at *7-8.

Finally, the district court rejected Oracle's argument that certain of the court's evidentiary rulings were abuses of discretion. The court explained that it: (1) redacted one line from an email because it was "too inflammatory and without foundation;" and (2) excluded other documents because Oracle had withheld them as privileged until trial. *Id.* at *9-12.

On June 8, 2016, the district court entered final judgment in favor of Google and against Oracle. Oracle timely appealed from the district court's judgment against it, including the court's underlying decisions denying its motions for JMOL and for a new trial. Google timely cross-appealed from all adverse orders and rulings underlying that final judgment.

This court has exclusive jurisdiction over all appeals in actions involving patent claims, including where, as here, an appeal raises only non-patent issues. 28 U.S.C. § 1295(a)(1). Because copyright law is not within this court's exclusive jurisdiction, we apply the law of the regional circuit in which the district court sits; here, the Ninth Circuit. *Atari Games Corp. v. Nintendo of Am., Inc.*, 975 F.2d 832, 837 (Fed. Cir. 1992).

II. ORACLE'S APPEAL

A. Legal Framework

It is undisputed that Google copied Oracle's declaring code and SSO for the 37 API packages verbatim. The question is whether that copying was fair. "From the infancy of copyright protection, some opportunity for fair use of copyrighted materials has been thought necessary to fulfill copyright's very purpose, 'to promote the Progress of Science and useful Arts.' " *Campbell v. Acuff-Rose Music, Inc.*, 510 U.S. 569, 575 (1994) (quoting U.S. Const., art. I, § 8, cl. 8). As the Supreme Court noted in *Campbell*, "[i]n truth, in literature, in science and in art, there are, and can be, few, if any, things, which in an abstract sense, are strictly new and original throughout. Every book in literature, science and art, borrows, and must necessarily borrow, and use much which was well

known and used before." *Id.* (quoting *Emerson v. Davies*, 8 F. Cas. 615, 619 (C.C.D. Mass. 1845)).

The fair use defense began as a judge-made doctrine and was codified in Section 107 of the 1976 Copyright Act. *Id.* at 576. It operates as a limited exception to the copyright holder's exclusive rights and permits use of copyrighted work if it is "for purposes such as criticism, comment, news reporting, teaching . . ., scholarship, or research." 17 U.S.C. § 107. The "such as" language confirms that the listing "was not intended to be exhaustive," but nevertheless "give[s] some idea of the sort of activities the courts might regard as fair use under the circumstances." *Harper & Row Publishers, Inc. v. Nation Enters.*, 471 U.S. 539, 561 (1985) (citation omitted).

"Section 107 requires a case-by-case determination whether a particular use is fair, and the statute notes four nonexclusive factors to be considered." *Id.* at 549. Those factors include: (1) "the purpose and character of the use, including whether such use is of a commercial nature or is for nonprofit educational purposes;" (2) "the nature of the copyrighted work;" (3) "the amount and substantiality of the portion used in relation to the copyrighted work as a whole;" and (4) "the effect of the use upon the potential market for or value of the copyrighted work." 17 U.S.C. § 107. The Supreme Court has cautioned against adopting bright-line rules and has emphasized that all of the statutory factors "are to be explored, and the results weighed together, in light of the purposes of copyright." *Campbell*, 510 U.S. at 578.

The legislative history reveals that Congress intended § 107 " 'to restate the present judicial doctrine of fair use, not to change, narrow, or enlarge it in any way' and intended that courts continue the common-law tradition of

fair use adjudication." *Id.* at 577 (quoting H.R. Rep. No. 94-1476, at 66 (1976), S. Rep. No. 94-473 at 62 (1975), U.S. Code Cong. & Admin. News 5659, 5679 (1976)). Accordingly, in balancing the four statutory factors, courts consider "whether the copyright law's goal of 'promot[ing] the Progress of Science and useful Arts,' U.S. Const., art. 1, § 8, cl. 8, 'would be better served by allowing the use than by preventing it.' " *Castle Rock Entm't, Inc. v. Carol Publ'g Grp., Inc.*, 150 F.3d 132, 141 (2d Cir. 1998) (quoting *Arica Inst., Inc. v. Palmer*, 970 F.2d 1067, 1077 (2d Cir. 1992)).

Despite this guidance, the doctrine of fair use has long been considered "the most troublesome in the whole law of copyright." *Monge v. Maya Magazines, Inc.*, 688 F.3d 1164, 1170 (9th Cir. 2012) (quoting *Dellar v. Samuel Goldwyn, Inc.*, 104 F.2d 661, 662 (2d Cir. 1939) (per curiam)). It both permits and requires "courts to avoid rigid application of the copyright statute when, on occasion, it would stifle the very creativity which that law is designed to foster." *Campbell*, 510 U.S. at 577 (quoting *Stewart v. Abend*, 495 U.S. 207, 236 (1990)).

Because fair use is an affirmative defense to a claim of infringement, Google bears the burden to prove that the statutory factors weigh in its favor. *Id.* at 590. Not all of the four factors must fatevor Google, however. *See Wall Data Inc. v. L.A. Cty. Sheriff's Dep't*, 447 F.3d 769, 778 (9th Cir. 2006). Instead, "fair use is appropriate where a 'reasonable copyright owner' would have consented to the use, i.e., where the 'custom or public policy' at the time would have defined the use as reasonable." *Id.* (citation omitted).

On appeal, Oracle argues that each of the four statutory factors weighs against a finding of fair use.

Specifically, it submits that: (1) the purpose and character of Google's use was purely for commercial purposes; (2) the nature of Oracle's work is highly creative; (3) Google copied 11,330 more lines of code than necessary to write in a Java language-based program; and (4) Oracle's customers stopped licensing Java SE and switched to Android because Google provided free access to it. In the alternative, Oracle argues that it is entitled to a new trial because the district court made several errors that deprived it of a fair opportunity to present its case. Because, as explained below, we agree with Oracle that Google's copying was not fair use as a matter of law, we need not address Oracle's alternative arguments for a new trial.

B. Standards of Review

Before turning to a consideration of the four statutory factors and any relevant underlying factual determinations, we first address the standard of review we are to employ in that consideration. While this section of most appellate opinions presents easily resolvable questions, like much else in the fair use context, that is not completely the case here.

There are several components to this inquiry. First, which aspects of the fair use determination are legal in nature and which are factual? Particularly, is the ultimate question of fair use a legal inquiry which is to be reviewed de novo? Second, what factual questions are involved in the fair use determination and under what standard are those determinations to be reviewed? Finally, though neither party addresses the question in detail, we consider what, if any, aspects of the fair use determination are for the jury to decide.

The Supreme Court has said that fair use is a mixed question of law and fact. *Harper & Row*, 471 U.S. at 560 (citing *Pac. & S. Co. v. Duncan*, 744 F.2d 1490, 1495 n.8 (11th Cir. 1984)). Merely characterizing an issue as a mixed question of law and fact does not dictate the applicable standard of review, however. *See U.S. Bank Nat'l Ass'n ex rel. CWCapital Asset Mgmt. LLC*, No. 15-1509, 2018 WL 1143822, at *5 (U.S. Mar. 5, 2018).

The Supreme Court has recently explained how we are to determine what the standard of review should be in connection with any mixed question of law and fact. *Id.* Specifically, the Court made clear that an appellate court is to break mixed questions into their component parts and to review each under the appropriate standard of review. *Id.* at *5-7. In *U.S. Bank*, the Supreme Court considered the level of review to be applied to a Bankruptcy Court's determination of whether a creditor in a bankruptcy action qualified as a "non-statutory insider" for purposes of 11 U.S.C. § 1129(a). *Id.* at *3-4. The Court found that there were three components to that inquiry: (1) determining the legal standard governing the question posed and what types of historical facts are relevant to that standard; (2) finding what the historical facts in the case at hand are; and (3) assessing whether the historical facts found satisfy the legal test governing the question to be answered. *Id.* at *4-5. As the Court explained, the first of these three is a purely legal question to be reviewed de novo on appeal and the second involves factual questions which "are reviewable only for clear error." *Id.* at *4 (citing Fed. R. Civ. P. 52(a)(6) (clear error standard)). The third is what the Court characterized as the "mixed question." *Id.* at *5.

Importantly, the Court noted that "[m]ixed questions are not all alike." *Id.* The Court then held that "the standard of review for a mixed question all depends—on whether answering it entails primarily legal or factual work." *Id.* Where applying the law to the historical facts "involves developing auxiliary legal principles of use in other cases—appellate courts should typically review a decision de novo." *Id.* (citing *Salve Regina College v. Russell*, 499 U.S. 225, 231-33 (1991)). But where the mixed question requires immersion in case-specific factual issues that are so narrow as to "utterly resist generalization," the mixed question review is to be deferential. *Id.* (quoting *Pierce v. Underwood*, 487 U.S. 552, 561-62 (1988)). Ultimately, the Court found that review of the mixed question at issue in that bankruptcy context should be deferential because de novo review of the question would do little to "clarify legal principles or provide guidance to other courts resolving other disputes." *Id.* at *7.

While this may be the first time the Supreme Court has so clearly explained how appellate courts are to analyze mixed questions of law and fact, it is not the first time the Supreme Court has told us how to analyze the particular mixed question of law and fact at issue here. In other words, while the Supreme Court has not previously broken the fair use inquiry into its three analytical components as expressly as it did the question in *U.S. Bank*, it has made clear that both the first and third of those components are subject to de novo review.

In *Harper & Row*, the Court explained that, "[w]here the district court has found facts sufficient to evaluate each of the statutory factors, an appellate court 'need not remand for further factfinding but may conclude as a matter of law that the challenged use does not qualify as a

fair use of the copyrighted work.' " 471 U.S. at 560 (quoting *Pac. & S. Co.*, 744 F.2d at 1495) (internal alterations omitted)). The Ninth Circuit has resolved the question in the same way. Where fair use is resolved on summary judgment, the Ninth Circuit reviews the district court's ultimate determination de novo. *SOFA Entm't, Inc. v. Dodger Prods., Inc.*, 709 F.3d 1273, 1277 (9th Cir. 2013) ("Whether Dodger's use of the clip constitutes fair use is a mixed question of law and fact that we review de novo."). That court has explained that, "as fair use is a mixed question of fact and law, so long as the record is 'sufficient to evaluate each of the statutory factors,' we may reweigh on appeal the inferences to be drawn from that record.' " *Mattel Inc. v. Walking Mountain Prods.*, 353 F.3d 792, 800 (9th Cir. 2003) (quoting *L.A. News Serv. v. CBS Broad., Inc.*, 305 F.3d 924, 942 (9th Cir. 2002)).

This treatment of the ultimate question posed when a fair use defense is raised makes sense. The fair use question entails, in the words of *U.S. Bank*, a primarily legal exercise. It requires a court to assess the inferences to be drawn from the historical facts found in light of the legal standards outlined in the statute and relevant case law and to determine what conclusion those inferences dictate. Because, as noted below, the historical facts in a fair use inquiry are generally few, generally similar from case to case, and rarely debated, resolution of what any set of facts means to the fair use determination definitely does not "resist generalization." *See U.S. Bank*, 2018 WL 1143822, at *5. Instead, the exercise of assessing whether a use is fair in one case will help guide resolution of that question in all future cases.

For these reasons, we conclude that whether the court applied the correct legal standard to the fair use inquiry is a question we review de novo, whether the findings relating to any relevant historical facts were correct are questions which we review with deference, and whether the use at issue is ultimately a fair one is something we also review de novo.

We have outlined the legal standard governing fair use above. We consider below whether the court properly applied those standards in the course of its fair use analysis and whether it reached the correct legal conclusion with respect to fair use. Before doing so, we briefly discuss the historical facts relevant to the fair use inquiry and consider the jury's role in determining those facts.

The Supreme Court has described "historical facts" as "a recital of external events." *Thompson v. Keohane*, 516 U.S. 99, 110 (1995); *see also U.S. Bank*, 2018 WL 1143822, at *4 (describing the historical facts at issue there as facts relating to "the attributes of a particular relationship or the circumstances and terms of a prior transaction"). In the fair use context, historical facts include the "origin, history, content, and defendant's use" of the copyrighted work. *Fitzgerald v. CBS Broad., Inc.*, 491 F. Supp. 2d 177, 184 (D. Mass. 2007); *see also Lotus Dev. Corp. v. Borland Int'l, Inc.*, 788 F. Supp. 78, 95 (D. Mass 1992) (defining historical facts to include "who did what, where, and when"). When asked at oral argument to identify historical facts relevant to the fair use inquiry, counsel for Oracle agreed that they are the "who, what, where, when, how, [and] how much." Oral Arg. at 3:28-54, available at http://oralarguments.cafc.uscourts.gov/default.aspx?fl=2017-1118.mp3. Google did not dispute this

characterization. This is, in part, because, in most fair use cases, defendants concede that they have used the copyrighted work, and "there is rarely dispute over the history, content, or origin of the copyrighted work." *See* Ned Snow, *Judges Playing Jury: Constitutional Conflicts in Deciding Fair Use on Summary Judgment*, 44 U.C. Davis L. Rev. 483, 493 (2010).

While some courts once treated the entire question of fair use as factual, and, thus, a question to be sent to the jury, that is not the modern view.[3] Since *Harper & Row*, the Ninth Circuit has described fair use as an "equitable defense." *Fisher v. Dees*, 794 F.2d 432, 435 (9th Cir. 1986) ("The fair-use doctrine was initially developed by courts as an equitable defense to copyright infringement."). Indeed, the Supreme Court referred to fair use as "an equitable rule of reason" in *Harper & Row*. 471 U.S. at 560. Congress did the same when it codified the doctrine of fair use in 1976. *See* H.R. Rep. No. 94-1476, 94th Cong., 2d Sess. 65-66 (1976), U.S. Code Cong. & Admin. News 1976, 5659, 5679-80 ("[S]ince the doctrine [of fair use] is an equitable rule of reason, no generally applicable definition is possible, and each case raising the question must be decided on its own facts . . . ."). If fair use is equitable in nature, it would seem to be a question for the judge, not the jury, to decide, even when there are factual disputes regarding its application. *See Granite State Ins. Co. v.*

---

[3] In *DC Comics, Inc. v. Reel Fantasy, Inc.*, 696 F.2d 24, 28 (2d Cir. 1982), the Second Circuit found that "[t]he four factors listed in Section 107 raise essentially factual issues and, as the district court correctly noted, are normally questions for the jury." So too, Justice Joseph Story described fair use as a "question of fact to come to a jury" in 1845. *Emerson v. Davies*, 8 F. Cas. 615, 623–24 (C.C.D. Mass. 1845).

Smart Modular Techs., Inc., 76 F.3d 1023, 1027 (9th Cir. 1996) ("A litigant is not entitled to have a jury resolve a disputed affirmative defense if the defense is equitable in nature."). In that instance, it would be the judge's factual determinations that would receive a deferential review—being assessed for clear error on the record before the court.

That said, the Supreme Court has never clarified whether and to what extent the jury is to play a role in the fair use analysis. *Harper & Row* involved an appeal from a *bench* trial where the district court concluded that the use of the copyrighted material was not a fair use. *Harper & Row Publishers, Inc. v. Nation Enters.*, 723 F.2d 195, 199 (2d Cir. 1983). The Court, thus, had no reason to discuss a jury determination of fair use and has not since taken an opportunity to do so.

Perhaps because of this silence, even after *Harper & Row*, several courts—including the Ninth Circuit—have continued to accept the fact that the question of fair use may go to a jury, albeit without analysis of why it may. *Compaq Comput. Corp. v. Ergonome Inc.*, 387 F.3d 403, 411 (5th Cir. 2004) ("The evidence presented at trial and the reasonable inferences therefrom, when viewed through the lens of the statutory fair use factors, support the jury's fair use finding."); *Jartech, Inc. v. Clancy*, 666 F.2d 403, 407-08 (9th Cir. 1982) (concluding that substantial evidence supported the jury's verdict on fair use); *Fiset v. Sayles*, No. 90-16548, 1992 WL 110263, at *4 (9th Cir. May 22, 1992) (finding that a reasonable jury could have concluded that "the evidence supporting fair use was not substantial"); *see also BUC Int'l Corp. v. Int'l Yacht Council*, 489 F.3d 1129, 1137 (11th Cir. 2007) (noting that the fair use defense went to the jury); *N.Y. Univ. v.*

Planet Earth Found., 163 F. App'x 13, 14 (2d Cir. 2005) ("As to the copyright infringement claim, the evidence also supports the jury's finding of fair use, under the four-factored analysis prescribed by statute.").

The Ninth Circuit has clarified, however, that the jury role in this context is limited to determining disputed "historical facts," not the inferences or conclusions to be drawn from those facts. *See Fisher*, 794 F.2d at 436. In *Fisher*, for example, the court explained that "[n]o material historical facts are at issue in this case. The parties dispute only the ultimate conclusions to be drawn from the admitted facts. Because, under *Harper & Row*, these judgments are legal in nature, we can make them without usurping the function of the jury." *Id.*; *see also Seltzer v. Green Day, Inc.*, 725 F.3d 1170, 1175 (9th Cir. 2013) ("As in *Fisher*, '[n]o material historical facts are at issue in this case. The parties dispute only the ultimate conclusion to be drawn from the admitted facts.' " (citing *Fisher*, 794 F.2d at 436)); *Hustler Magazine, Inc. v. Moral Majority, Inc.*, 606 F. Supp. 1526, 1532 (C.D. Cal. 1985) (noting that "fair use normally is a question of fact for the jury," but concluding that "the issue of fair use, at least in the context of this case, presents primarily a question of law"). Accordingly, while inferences from the four-factor analysis and the ultimate question of fair use are "legal in nature," in the Ninth Circuit, disputed historical facts represent questions for the jury. *Fisher*, 794 F.2d at 436. Where there are no disputed material historical facts, fair use can be decided by the court alone. *Id.*

Despite this case law, all aspects of Google's fair use defense went to the jury with neither party arguing that it should not. Thus, the jury was asked not just what the historical facts were, but what the implications of those

facts were for the fair use defense. During the first appeal, Google argued to this court that there were disputed issues of material historical fact relevant to its fair use defense. As discussed below, the parties stipulated—or at least ceased to dispute—some of those facts, and presented the remaining disputed historical facts to the jury on remand. The jury returned a verdict in favor of Google on its fair use defense. Because the verdict form—though captioned as a "special verdict"—did not ask the jury to articulate its fact findings in any detail, we must assume that the jury resolved all factual issues relating to the historical facts in favor of the verdict.[4] Despite the posture of the fair use finding, we must break that finding into its constituent parts. We must then review the subsidiary and controverted findings of historical fact for substantial evidence. *See Seltzer*, 725 F.3d at 1175; *see also Brewer v. Hustler Magazine, Inc.*, 749 F.2d 527, 528 (9th Cir. 1984) ("We may disturb a jury verdict only if the evidence was insufficient as a matter of law.").

All jury findings relating to fair use other than its implied findings of historical fact must, under governing

---

[4] As counsel for Oracle noted at oral argument, this is similar to the standard we apply in obviousness cases. Oral Argument at 9:34–10:24. Because obviousness is a mixed question of law and fact, we "first presume that the jury resolved the underlying factual disputes in favor of the verdict [ ] and leave those presumed findings undisturbed if they are supported by substantial evidence. Then we examine the [ultimate] legal conclusion [of obviousness] de novo to see whether it is correct in light of the presumed jury fact findings." *Kinetic Concepts, Inc. v. Smith & Nephew, Inc.*, 688 F.3d 1342, 1356–57 (Fed. Cir. 2012) (quoting *Jurgens v. McKasy*, 927 F.2d 1552, 1557 (Fed. Cir. 1991)). Likewise, Google cited our decision in *Kinetic Concepts* for the proposition that we must "presume that the jury made all findings in support of the verdict that are supported by substantial evidence." Cross–Appellant Br. 35.

Supreme Court and Ninth Circuit case law, be viewed as advisory only. Accordingly, while we might assess the jury's role in the assessment of fair use differently if not bound by Ninth Circuit law, we proceed on the assumption both that: (1) it was not error to send the question to the jury, because the Ninth Circuit has at least implicitly endorsed doing so; and (2) we must assess all inferences to be drawn from the historical facts found by the jury and the ultimate question of fair use de novo, because the Ninth Circuit has explicitly said we must do so.

The parties have identified the following historical facts relating to Google's use of the copyrighted work:

- the history and origin of the copyrighted work, including what declaring code is;
- how much of the copyrighted work was copied;
- whether there were other ways to write the API packages;
- whether the copied material was used for the same purpose as in the original work;
- whether the use was commercial in nature;
- whether Google acted in bad faith in copying the work;
- whether there are functional aspects to the copyrighted work that make it less deserving of protection; and
- whether there was harm to the actual or potential markets for the copyrighted work.

The parties now agree on the resolution of the first four factual questions: (1) what the declaring code is and what it does in Java SE and Android, and that the code at

issue was a work created by Oracle; (2) how many lines of code were copied; (3) that there were other ways for Google to write API packages; and (4) that Google used the API packages in Android for the same purpose they were created for in Java. The parties dispute, however, the remaining historical facts they identified. We address those disputes in the context of our assessment of the statutory factors to which the respective historical fact is relevant.

C. Applying the Fair Use Factors

Factor 1: The Purpose and Character of the Use

The first factor in the fair use inquiry involves "the purpose and character of the use, including whether such use is of a commercial nature or is for nonprofit educational purposes." 17 U.S.C. § 107(1). This factor has two primary components: (1) whether the use is commercial in nature, rather than for educational or public interest purposes; and (2) "whether the new work is transformative or simply supplants the original." *Wall Data*, 447 F.3d at 778 (citing *Campbell*, 510 U.S. at 579). As explained below, the first is a question of fact and the second is a question of law. As Oracle points out, moreover, courts sometimes also consider whether the historical facts support the conclusion that the infringer acted in bad faith. *See Harper & Row*, 471 U.S. at 562. We address each component in turn.

a. Commercial Use

Analysis of the first factor requires inquiry into the commercial nature of the use. Use of the copyrighted work that is commercial "tends to weigh against a finding of fair use." *Harper & Row*, 471 U.S. at 562. Courts have recognized, however, that, "[s]ince many, if not most,

secondary users seek at least some measure of commercial gain from their use, unduly emphasizing the commercial motivation of a copier will lead to an overly restrictive view of fair use." *Am. Geophysical Union v. Texaco, Inc.*, 60 F.3d 913, 921 (2d Cir. 1994); *see also Infinity Broad. Corp. v. Kirkwood*, 150 F.3d 104, 109 (2d Cir. 1998) ("[N]otwithstanding its mention in the text of the statute, commerciality has only limited usefulness to a fair use inquiry; most secondary uses of copyrighted material, including nearly all of the uses listed in the statutory preamble, are commercial."). Accordingly, although the statute requires us to consider the "commercial nature" of the work, "the degree to which the new user exploits the copyright for commercial gain—as opposed to incidental use as part of a commercial enterprise—affects the weight we afford commercial nature as a factor." *Elvis Presley Enters., Inc. v. Passport Video*, 349 F.3d 622, 627 (9th Cir. 2003). "[I]t is undisputed that Google's use of the declaring code and SSO from 37 Java API packages served commercial purposes." *Order Denying JMOL*, 2016 WL 3181206, at *7. Although the jury was instructed that commercial use weighed against fair use, the district court explained that the jury "could reasonably have found that Google's decision to make Android available open source and free for all to use had non-commercial purposes as well (such as the general interest in sharing software innovation)." *Id.*

On appeal, Oracle argues that Android is "hugely profitable" and that "Google reaps billions from exploiting Java in Android." Appellant Br. 29. As such, Oracle

maintains that no reasonable jury could have found Android anything but "overwhelmingly commercial." *Id.*[5]

Google responds that: (1) because it gives Android away for free under an open source license the jury could have concluded that Android has non-commercial purposes; and (2) the jury could have reasonably found that Google's revenue flows from the advertisements on its search engine which preexisted Android. Neither argument has merit. First, the fact that Android is free of charge does not make Google's use of the Java API packages noncommercial. Giving customers "for free something they would ordinarily have to buy" can constitute commercial use. *A&M Records, Inc. v. Napster, Inc.*, 239 F.3d 1004, 1015 (9th Cir. 2001) (finding that "repeated and exploitative copying of copyrighted works, even if the copies are not offered for sale, may constitute a commercial use"). That Google might also have non-commercial motives is irrelevant as a matter of law. As the Supreme Court made clear when *The Nation* magazine published excerpts from Harper & Row's book, partly for the purpose of providing the public newsworthy

---

[5] Oracle also argues that Google conceded that its use was "entirely commercial" during oral argument to this court in the first appeal. *Order Denying JMOL*, 2016 WL 3181206, at *7 ("Q: But for purpose and character, though, you don't dispute that it was entirely a commercial purpose. A: No."). The district court treated this colloquy as a judicial admission that Google's use was "commercial." *Id.* (noting that the word "entirely" was "part of the give and take" of oral argument). The court therefore instructed the jury that Google's use was commercial, but that it was up to the jury to determine the extent of the commerciality. *Id.* at *8. Oracle does not challenge the district court's jury instructions on appeal. In any event, as the district court noted, "even a wholly commercial use may still constitute fair use." *Id.* at *7 (citing *Campbell*, 510 U.S. at 585).

information, the question "is not whether the sole motive of the use is monetary gain but whether the user stands to profit from exploitation of the copyrighted material without paying the customary price." *Harper & Row*, 471 U.S. at 562. Second, although Google maintains that its revenue flows from advertisements, not from Android, commerciality does not depend on how Google earns its money. Indeed, "[d]irect economic benefit is not required to demonstrate a commercial use." *A&M Records,* 239 F.3d at 1015. We find, therefore, that, to the extent we must assume the jury found Google's use of the API packages to be anything other than overwhelmingly commercial, that conclusion finds no substantial evidentiary support in the record. Accordingly, Google's commercial use of the API packages weighs against a finding of fair use.

b. Transformative Use

Although the Copyright Act does not use the word "transformative," the Supreme Court has stated that the "central purpose" of the first fair use factor is to determine "whether and to what extent the new work is transformative." *Campbell*, 510 U.S. at 579. Transformative works "lie at the heart of the fair use doctrine's guarantee of breathing space within the confines of copyright, and the more transformative the new work, the less will be the significance of other factors, like commercialism, that may weigh against a finding of fair use." *Id.* (internal citation omitted).

A use is "transformative" if it "adds something new, with a further purpose or different character, altering the first with new expression, meaning or message." *Id.* The critical question is "whether the new work merely supersede[s] the objects of the original creation . . . or

instead adds something new." *Id.* (citations and internal quotation marks omitted). This inquiry "may be guided by the examples given in the preamble to § 107, looking to whether the use is for criticism, or comment, or news reporting, and the like." *Id.* at 578-79. "The Supreme Court has recognized that parodic works, like other works that comment and criticize, are by their nature often sufficiently transformative to fit clearly under the fair use exception." *Mattel Inc. v. Walking Mountain Prods.*, 353 F.3d 792, 800 (9th Cir. 2003) (citing *Campbell*, 510 U.S. at 579).

"Although transformation is a key factor in fair use, whether a work is transformative is a often highly contentious topic." *Seltzer*, 725 F.3d at 1176. Indeed, a "leading treatise on this topic has lamented the frequent misuse of the transformation test, complaining that it has become a conclusory label which is 'all things to all people.' " *Id.* (quoting Melville B. Nimmer & David Nimmer, 4 Nimmer on Copyright § 13.05[A][1][b], 13168-70 (2011)).

To be transformative, a secondary work must either alter the original with new expression, meaning, or message or serve a new purpose distinct from that of the original work. *Campbell*, 510 U.S. at 579; *Elvis Presley Enters.*, 349 F.3d at 629. Where the use "is for the same intrinsic purpose as [the copyright holder's] . . . such use seriously weakens a claimed fair use." *Worldwide Church of God v. Phila. Church of God, Inc.*, 227 F.3d 1110, 1117 (9th Cir. 2000) (quoting *Weissmann v. Freeman*, 868 F.2d 1313, 1324 (2d Cir. 1989)).

Although "transformative use is not absolutely necessary for a finding of fair use, the goal of copyright, to promote science and the arts, is generally furthered by the creation of transformative works." *Campbell*, 510 U.S. at

579 (citation and footnote omitted). As such, "the more transformative the new work, the less will be the significance of other factors, like commercialism, that may weigh against a finding of fair use." *Id.* Importantly, in the Ninth Circuit, whether a work is transformative is a question of law. *See Mattel*, 353 F.3d at 801 (explaining that parody—a well-established species of transformative use—"is a question of law, not a matter of public majority opinion"); *see also Fox News Network, LLC v. TVEyes, Inc.*, No. 15-3885, 2018 WL 1057178, at *3-4 (2d Cir. Feb. 27, 2018) (reassessing whether the use in question was transformative and deciding it was as a matter of law).

In denying JMOL, the district court explained that "of course, the copied declarations serve the same function in both works, for by definition, declaring code in the Java programming language serves the [same] specific definitional purposes." *Order Denying JMOL*, 2016 WL 3181206, at *8.[6] The court concluded, however, that the jury could reasonably have found that Google's selection of some, but not all, of the Java API packages—"with new

[6] According to the district court, if this fact were sufficient to defeat fair use, "it would be impossible ever to duplicate declaring code as fair use and presumably the Federal Circuit would have disallowed this factor on the first appeal rather than remanding for a jury trial." *Id.* But in our prior decision, we remanded in part because Google represented to this court that there were disputes of fact regarding how Android was used and whether the APIs Google copied served the same function in Android and Java. *Oracle*, 750 F.3d at 1376. Without the benefit of briefs exploring the record on these issues, and Google's later agreement with respect to these facts, we concluded that we could not say that there were no material facts in dispute. *Id.* As explained previously, however, those facts are no longer in dispute. The only question that remains regarding transformative use is whether, on the now undisputed facts, Google's use of the APIs was, in fact, transformative.

implementing code adapted to the constrained operating environment of mobile smartphone devices," together with new "methods, classes, and packages written by Google for the mobile smartphone platform"—constituted "a fresh context giving new expression, meaning, or message to the duplicated code." *Id.* at *9.

On appeal, Oracle argues that Google's use was not transformative because it did not alter the APIs with "new expression, meaning, or message." Appellant Br. 29 (quoting *Campbell*, 510 U.S. at 579). Because Google concedes that it uses the API packages for the same purpose, Oracle maintains that it was unreasonable for either the jury or the court to find that Google sufficiently transformed the APIs to overcome its highly commercial use.

Google responds that a reasonable jury could have concluded that Google used a small portion of the Java API packages to create a new work in a new context—"Android, a platform for smartphones, not desktops and servers." Cross-Appellant Br. 37. Google argues that, although the declarations and SSO may perform the same functions in Android and Java, the jury could reasonably find that they have different purposes because the "point of Android was to create a groundbreaking platform for smartphones." *Id.* at 39.

Google's arguments are without merit. As explained below, Google's use of the API packages is not transformative as a matter of law because: (1) it does not fit within the uses listed in the preamble to § 107; (2) the purpose of the API packages in Android is the same as the purpose of the packages in the Java platform; (3) Google made no alteration to the expressive content or message

of the copyrighted material; and (4) smartphones were not a new context.

First, though not dispositive, we turn to the examples given in the preamble to § 107, "looking to whether the use is for criticism, or comment, or news reporting, and the like." *Campbell*, 510 U.S. at 578-79. Google's use of the Java API packages does not fit within the statutory categories, and Google does not suggest otherwise. Instead, Google cites *Sony Computer Entertainment, Inc. v. Connectix Corp.*, 203 F.3d 596 (9th Cir. 2000), for the proposition that the "Ninth Circuit has held other types of uses—specifically including uses of computer code—to be fair." Cross-Appellant Br. 41. In *Sony*, the court found that the defendant's reverse engineering and intermediate copying of Sony's copyrighted software system "was a fair use for the purpose of gaining access to the unprotected elements of Sony's software." 203 F.3d at 602. The court explained that Sony's software program contained unprotected functional elements and that the defendant could only access those elements through reverse engineering. *Id.* at 603. The defendant used that information to create a software program that let consumers play games designed for Sony's PlayStation console on their computers. The court found that the defendant's use was only "modestly transformative" where: (1) the defendant created "a wholly new product" with "entirely new . . . code," and (2) the intermediate copying was performed to "produce a product that would be compatible." *Id.* at 606- 07. As Oracle points out, even the "modest" level of transformation at issue in *Sony* is more transformative than what Google did here: copy code verbatim to attract programmers to Google's "new and incompatible platform." Appellant Response Br. 21.

It is undisputed that the API packages "serve the same function in both works." *Order Denying JMOL*, 2016 WL 3181206, at *8. And, as Oracle explains, the historical facts relevant to transformative use are also undisputed: what declaring code is, what it does in Java and in Android, how the audience of computer developers perceives it, how much Google took and added, what the added code does, and why Google used the declaring code and SSO. Indeed, Google conceded that "including the declarations (and their associated SSO) was for the benefit of developers, who—familiar with the Java programming language—had certain expectations regarding the language's APIs." Google's Opp. to Oracle's Rule 50(a) Motion for JMOL at 20, *Oracle Am., Inc. v. Google Inc.*, No. 3:10-cv-3561 (N.D. Cal. May 21, 2016), ECF No. 1935. The fact that Google created exact copies of the declaring code and SSO and used those copies for the same purpose as the original material "seriously weakens [the] claimed fair use." *See Wall Data*, 447 F.3d at 778 (finding that, where the "Sheriff's Department created exact copies of RUMBA's software . . . [and] put those copies to the identical purpose as the original software," the use was not transformative); *see also Campbell*, 510 U.S. at 580 (noting that where the alleged infringer merely seeks "to avoid the drudgery in working up something fresh," any "claim to fairness . . . diminishes accordingly").

Google argues that Android is transformative because Google selectively used the declarations and SSO of only 37 of the 166 Java SE API packages and wrote its own implementing code. But taking only select passages of a copyrighted work is, by itself, not transformative. *See L.A. News Serv. v. CBS Broad., Inc.*, 305 F.3d 924, 938-39 (9th Cir. 2002) ("Merely plucking the most visually arresting

excerpt from LANS's nine minutes of footage cannot be said to have added anything new."). While, as discussed below, the volume of work copied is relevant to the fair use inquiry generally, thought must be given to the quality and importance of the copied material, not just to its relative quantity vis-à-vis the overall work. *See Campbell*, 510 U.S. at 586-87. To hold otherwise would mean that verbatim copying could qualify as fair use as long as the plagiarist stops short of taking the entire work. That approach is inconsistent with settled law and is particularly troubling where, as here, the portion copied is qualitatively significant. *See Harper & Row*, 471 U.S. at 569 (finding that verbatim copying of 300 words from a manuscript of more than 200,000 words was not a fair use); *see also Folsom v. Marsh*, 9 F. Cas. 342, 345 (C.C.D. Mass 1841) (Story, J.) ("There must be real, substantial condensation of the materials, and intellectual labor and judgment bestowed thereon; and not merely the facile use of the scissors; or extracts of the essential parts, constituting the chief value of the original work.").

That Google wrote its own implementing code is irrelevant to the question of whether use of the APIs was transformative. As we noted in the prior appeal, "no plagiarist can excuse the wrong by showing how much of his work he did not pirate." *Oracle*, 750 F.3d at 1375 (quoting *Harper & Row*, 471 U.S. at 565). The relevant question is whether Google altered "the *expressive content or message* of the original work" that it copied—not whether it rewrote the portions it did not copy. *See Seltzer*, 725 F.3d at 1177 (explaining that a work is not transformative where the user "makes no alteration to the *expressive content or message* of the original work"). That said, even where the allegedly infringing work "makes few

physical changes to the original or fails to comment on the original," it will "typically [be] viewed as transformative as long as new expressive content or message is apparent." *Id.* Here, however, there is no suggestion that the new implementing code somehow changed the expression or message of the declaring code. While Google's use could have been transformative if it had copied the APIs for some other purpose—such as teaching how to design an API—merely copying the material and moving it from one platform to another without alteration is not transformative.

Google's primary argument on appeal is that Android is transformative because Google incorporated the declarations and SSO of the 37 API packages into a new *context*—smartphones. But the record showed that Java SE APIs were in smartphones before Android entered the market. Specifically, Oracle presented evidence that Java SE was in SavaJe mobile phones and that Oracle licensed Java SE to other smartphone manufacturers, including Danger and Nokia. Because the Java SE was already being used in smartphones, Google did not "transform" the copyrighted material into a new context and no reasonable jury could conclude otherwise.[7]

In any event, moving material to a new context is not transformative in and of itself—even if it is a "sharply different context." *TCA Television Corp. v. McCollum*, 839 F.3d 168, 181-83 (2d Cir. 2016) (finding that use "at

[7] Because we conclude that smartphones were not a new context, we need not address the argument, made by Oracle and certain amici, that the district court's order excluding evidence of Google's use of Android in multiple other circumstances—including laptops—tainted the jury's and the court's ability to fairly assess the character of the use.

some length, almost verbatim," of the copyrighted comedy routine "Who's on First?" in a dramatic play was not transformative where the play neither "imbued the Routine with any new expression, meaning, or message," nor added "any new dramatic purpose"). As previously explained, a use becomes transformative only if it serves a different purpose or alters the "expression, meaning, or message" of the original work. *Kelly*, 336 F.3d at 818. As such, "[c]ourts have been reluctant to find fair use when an original work is merely retransmitted in a different medium." *A&M Records,* 239 F.3d at 1015. Accordingly, although a change of format may be "useful," it "is not technically a transformation." *Infinity Broad.*, 150 F.3d at 108 n.2 (finding that retransmitting copyrighted radio transmissions over telephone lines was not transformative because there was no new expression, meaning, or message).

The Ninth Circuit has stated that "[a] use is considered transformative only where a defendant changes a plaintiff's copyrighted work or uses the plaintiff's copyrighted work in a different context such that the plaintiff's work is transformed into a new creation." *Perfect 10, Inc. v. Amazon.com, Inc.*, 508 F.3d 1146, 1165 (9th Cir. 2007) (quoting *Wall Data*, 447 F.3d at 778). In *Perfect 10*, for example, the court found Google's use of thumbnail versions of copyrighted images "highly transformative" because, "[a]lthough an image may have been created originally to serve an entertainment, aesthetic, or informative function, a search engine transforms the image into a pointer directing a user to a source of information." *Id.* Although the court discussed the change in context (moving the copyrighted images into the electronic reference tool), it emphasized that Google

used the images "in a new context to serve a different purpose." *Id.* In reaching this conclusion, the court reiterated that "even making an exact copy of a work may be transformative so long as the copy serves a different function than the original work." *Id.* (citing *Kelly*, 336 F.3d at 818-19). It is clear, therefore, that the change in context alone was not dispositive in *Perfect 10*; rather, the change in context facilitated the change in purpose, which made the use transformative.

To some extent, any use of copyrighted work takes place in a slightly different context than the original. And of course, there is no bright line identifying when a use becomes transformative. But where, as here, the copying is verbatim, for an identical function and purpose, and there are no changes to the expressive content or message, a mere change in format (e.g., from desktop and laptop computers to smartphones and tablets) is insufficient as a matter of law to qualify as a transformative use.[8]

c. Bad faith

In evaluating the "purpose and character" factor, the Ninth Circuit applies "the general rule that a party claiming fair use must act in a manner generally compatible with principles of good faith and fair dealing." *Perfect 10,* 508 F.3d at 1164 n.8 (citing *Harper & Row*, 471 U.S. at 562-63). In part, this is based on the fact that, in *Harper & Row,* the Supreme Court expressly stated that "[f]air use presupposes 'good faith' and 'fair dealing.' " 471

[8] As some amici note, to hold otherwise could encroach upon the copyright holder's right to "prepare derivative works based upon the copyrighted work." 17 U.S.C. § 106(2); *see* Br. of Amicus Curiae N.Y. Intell. Prop. L. Ass'n at 17–20.

U.S. at 562 (citation omitted). It is also in part true because, as the Ninth Circuit has said, one who acts in bad faith should be barred from invoking the equitable defense of fair use. *Fisher*, 794 F.2d at 436 (calling the principle of considering the alleged infringer's "bad conduct" as a "bar [to] his use of the equitable defense of fair use" a sound one).[9]

Consistent with this authority, and at Oracle's request, the district court instructed the jury that it could consider whether Google acted in bad faith (or not) as part of its assessment of the first fair use factor. *Order Denying JMOL*, 2016 WL 3181206, at *6. And, because Oracle was permitted to introduce evidence that Google acted in bad

[9] As the district court recognized, there is some debate about whether good or bad faith should remain relevant to the factor one inquiry. *Order Denying JMOL*, 2016 WL 3181206, at *2 ("[T]here is a respectable view that good or bad faith should no longer be a consideration after the Supreme Court's decision in *Campbell*."); *see also* Hon. Pierre N. Leval, *Toward a Fair Use Standard*, 103 Harv. L. Rev. 1105, 1128 (1990) ("Whether the secondary use is within the protection of the [fair use] doctrine depends on factors pertinent to the objectives of the copyright law and not on the morality or motives of either the secondary user or the copyright-owning plaintiff."). In *Campbell*, the Supreme Court expressed skepticism about "the weight one might place on the alleged infringer's state of mind." *Campbell*, 510 U.S. at 585 n.18. But the Ninth Circuit has not repudiated its view that " 'the propriety of the defendant's conduct' is relevant to the character of the use at least to the extent that it may knowingly have exploited a purloined work for free that could have been obtained for a fee." *L.A. News Serv. v. KCAL–TV Channel 9*, 108 F.3d 1119, 1122 (9th Cir. 1997) (quoting *Harper & Row*, 471 U.S. at 562). For that reason, and because we conclude in any event that the jury must have found that Google did not act in bad faith, we address that question and the parties' arguments relating thereto.

faith, the court permitted Google to try to prove its good faith. *Id.*

At trial, Oracle introduced evidence suggesting that "Google felt it needed to copy the Java API as an accelerant to bring Android to the market quicker" and knew that it needed a license to use Java. *Id.* For its part, Google presented evidence that it believed that the declaring code and SSO were "free to use and re-implement, both as a matter of developer practice and because the availability of independent implementations of the Java API enhanced the popularity of the Java programming language, which Sun promoted as free for all to use." *Id.* at *7. Given this conflicting evidence, the district court found that the jury could reasonably have concluded that "Google's use of parts of the Java API as an accelerant was undertaken based on a good faith belief that at least the declaring code and SSO were free to use (which it did use), while a license was necessary for the implementing code (which it did not use)." *Id.*

On appeal, Oracle argues that there was ample evidence that Google intentionally copied Oracle's copyrighted work and knew that it needed a license to use Java. Google responds that the jury heard sufficient evidence of Google's good faith based on industry custom and was entitled to credit that evidence.

But, while bad faith may weigh against fair use, a copyist's good faith cannot weigh in favor of fair use. Indeed, the Ninth Circuit has expressly recognized that "the innocent intent of the defendant constitutes no defense to liability." *Monge*, 688 F.3d at 1170 (quoting 4 Melville B. Nimmer & David Nimmer, *Nimmer on Copyright* § 13.08[B][1] (Matthew Bender rev. ed. 2011)). If it were clear, accordingly, that the jury found fair use

solely or even largely because it *approved* of Google's motives even if they were in bad faith, we would find such a conclusion improper. Because evidence of Google's good faith was relevant to rebut evidence of its bad faith, however, and there is no objection to the instructions to the jury on this or any other point, we must assume that the jury simply did not find the evidence of Google's bad faith persuasive.[10] We note, moreover, that merely "being denied permission to use a work does not weigh against a finding of fair use." *Campbell*, 510 U.S. at 585 n.18 ("If the use is otherwise fair, then no permission need be sought or granted."). Ultimately, we find that, even assuming the jury was unpersuaded that Google acted in bad faith, the highly commercial and non-transformative nature of the use strongly support the conclusion that the first factor weighs against a finding of fair use.

Factor 2: Nature of the Copyrighted Work

The second factor—the nature of the copyrighted work—"calls for recognition that some works are closer to the core of intended copyright protection than others, with the consequence that fair use is more difficult to establish when the former works are copied." *Campbell*, 510 U.S. at 586. This factor "turns on whether the work is informational or creative." *Worldwide Church of God*, 227 F.3d at 1118; *see also Harper & Row*, 471 U.S. at 563 ("The law generally recognizes a greater need to disseminate

[10] The jury was instructed that, "[i]n evaluating the extent to which Google acted in good faith or not, you may take into account, together with all other circumstances, the extent to which Google relied upon or contravened any recognized practices in the industry concerning reimplementation of API libraries." *Order Denying JMOL*, 2016 WL 3181206, at *3 n.2. Oracle has not challenged this instruction on appeal.

factual works than works of fiction or fantasy."). Creative expression "falls within the core of the copyright's protective purposes." *Campbell*, 510 U.S. at 586. Although "software products are not purely creative works," it is well established that copyright law protects computer software. *Wall Data*, 447 F.3d at 780 (citing *Sega Enters. Ltd. v. Accolade, Inc.*, 977 F.2d 1510, 1519 (9th Cir. 1992) ("[T]he 1980 amendments to the Copyright Act unambiguously extended copyright protection to computer programs.")).

Here, the district court found that the jury could have concluded that the process of designing APIs was "highly creative" and "thus at the core of copyright's protection" or it could "reasonably have gone the other way and concluded that the declaring code was not highly creative." *Order Denying JMOL*, 2016 WL 3181206, at *10. While the jury heard testimony from Google's own expert that API design is "an art, not a science," other witnesses emphasized the functional role of the declaring code and the SSO and minimized the creative aspects. *Id.* Accordingly, the district court concluded that the "jury could reasonably have found that, while the declaring code and SSO were creative enough to qualify for copyright protection, functional considerations predominated in their design." *Id.*

On appeal, Oracle emphasizes that designing the APIs was a highly creative process and that the organization of the packages was not mandated by function. Indeed, this court has already held that the declaring code and the SSO of the 37 API packages at issue were sufficiently creative and original to qualify for copyright protection. *Oracle*, 750 F.3d at 1356. According to Oracle, the district court

erred in assuming that, because the APIs have a "functional role," they cannot be creative.

As Google points out, however, all we found in the first appeal was that the declarations and SSO were sufficiently creative to provide the "minimal degree of creativity," *Feist Publ'ns, Inc. v. Rural Tel. Serv. Co.*, 499 U.S. 340, 345 (1991), that is required for copyrightability. We also recognized that a reasonable jury could find that "the functional aspects of the packages" are "relevant to Google's fair use defense." *Oracle*, 750 F.3d at 1369, 1376-77. On remand, Oracle stipulated that some of the declarations were necessary to use the Java language and presented no evidence explaining how the jury could distinguish the functionality and creativity of those declarations from the others. Google maintains that it presented evidence that the declarations and SSO were functional and the jury was entitled to credit that evidence.

Although it is clear that the 37 API packages at issue involved some level of creativity—and no reasonable juror could disagree with that conclusion—reasonable jurors could have concluded that functional considerations were both substantial and important. Based on that assumed factual finding, we conclude that factor two favors a finding of fair use.

The Ninth Circuit has recognized, however, that this second factor "typically has not been terribly significant in the overall fair use balancing." *Dr. Seuss Enters., L.P. v. Penguin Books USA, Inc.*, 109 F.3d 1394, 1402 (9th Cir. 1997) (finding that the "creativity, imagination and originality embodied in The Cat in the Hat and its central character tilts the scale against fair use"); *Mattel*, 353 F.3d at 803 (similar). Other circuits agree. *Fox News*

Network, 2018 WL 1057178, at *5 ("This factor 'has rarely played a significant role in the determination of a fair use dispute,' and it plays no significant role here." (quoting *Authors Guild v. Google, Inc.*, 804 F.3d 202, 220 (2d Cir. 2015))). We note, moreover, that allowing this one factor to dictate a conclusion of fair use in all cases involving copying of software could effectively negate Congress's express declaration—continuing unchanged for some forty years—that software is copyrightable. Accordingly, though the jury's assumed view of the nature of the copyrighted work weighs in favor of finding fair use, it has less significance to the overall analysis.

Factor 3: Amount and Substantiality of the Portion Used

The third factor focuses on the "amount and substantiality of the portion used in . . . the context of the copyrighted work, not the infringing work." *Oracle*, 750 F.3d at 1375. Indeed, the statutory language makes clear that "a taking may not be excused merely because it is insubstantial with respect to the *infringing* work." *Harper & Row*, 471 U.S. at 565. "[T]he fact that a substantial portion of the infringing work was copied verbatim [from the original work] is evidence of the qualitative value of the copied material, both to the originator and to the plagiarist who seeks to profit from marketing someone else's copyrighted expression." *Id.* Thus, while "wholesale copying does not preclude fair use per se, copying an entire work militates against a finding of fair use." *Worldwide Church of God*, 227 F.3d at 1118 (citation and quotation marks omitted). But, there is no relevance to the opposite—i.e., adding substantial content to the copyrighted work is not evidence that what was copied was insubstantial or unimportant.

The inquiry under this third factor "is a flexible one, rather than a simple determination of the percentage of the copyrighted work used." *Monge*, 688 F.3d at 1179. The Ninth Circuit has explained that this third factor looks to the quantitative amount and qualitative value of the original work used in relation to the justification for its use. *Seltzer*, 725 F.3d at 1178. The percentage of work copied is not dispositive where the portion copied was qualitatively significant. *Harper & Row*, 471 U.S. at 566 ("In view of the expressive value of the excerpts and their key role in the infringing work, we cannot agree with the Second Circuit that the 'magazine took a meager, indeed an infinitesimal amount of Ford's original language.' "(citation omitted)). Google is correct that the Ninth Circuit has said that, "this factor will not weigh against an alleged infringer, even when he copies the whole work, if he takes no more than is necessary for his intended use." *Id.* (citing *Kelly v. Arriba Soft Corp.*, 336 F.3d 811, 820-21 (9th Cir. 2003)). But the Ninth Circuit has only said that is true where the intended use was a transformative one, because the "extent of permissible copying varies with the purpose and character of the use." *Id.* (quoting *Campbell*, 510 U.S. at 586-87). Here, we have found that Google's use was not transformative and Google has conceded both that it could have written its own APIs and that the purpose of its copying was to make Android attractive to programmers. "Necessary" in the context of the cases upon which Google relies does not simply mean easier.

In assessing factor three, the district court explained that the "jury could reasonably have found that Google duplicated the bare minimum of the 37 API packages, just enough to preserve inter-system consistency in usage,

namely the declarations and their SSO only, and did not copy any of the implementing code," such that Google "copied only so much as was reasonably necessary." *Order Denying JMOL*, 2016 WL 3181206, at *10. In reaching this conclusion, the court noted that the jury could have found that the number of lines of code Google duplicated was a "tiny fraction of one percent of the copyrighted works (and even less of Android, for that matter)." *Id.* We disagree that such a conclusion would have been reasonable or sufficient on this record.

On remand, the parties stipulated that only 170 lines of code were necessary to write in the Java language. It is undisputed, however, that Google copied 11,500 lines of code—11,330 more lines than necessary to write in Java. That Google copied more than necessary weighs against fair use. *See Monge*, 688 F.3d at 1179 (finding that, where the copyist "used far more than was necessary" of the original work, "this factor weighs against fair use"). And, although Google emphasizes that it used a small percentage of Java (11,500 lines of declarations out of roughly 2.86 million lines of code in the Java SE libraries), it copied the SSO for the 37 API packages in its entirety.

The district court emphasized Google's desire to "preserve inter-system consistency" to "avoid confusion among Java programmers as between the Java system and the Android system." *Order Denying JMOL*, 2016 WL 3181206, at *10-11. As we noted in the prior appeal, however, Google did not seek to foster any "inter-system consistency" between its platform and Oracle's Java platform. *Oracle*, 750 F.3d at 1371. And Google does not

rely on any interoperability arguments in this appeal.[11] Google sought "to capitalize on the fact that software developers were already trained and experienced in using the Java API packages at issue." *Id.* But there is no inherent right to copy in order to capitalize on the popularity of the copyrighted work or to meet the expectations of intended customers. Taking those aspects of the copyrighted material that were familiar to software developers to create a similar work designed to be popular with those same developers is not fair use. *See Dr. Seuss Enters.*, 109 F.3d at 1401 (copying the most famous and well recognized aspects of a work "to get attention" or "to avoid the drudgery in working up something fresh" is not a fair use (quoting *Campbell*, 510 U.S. at 580)).

Even assuming the jury accepted Google's argument that it copied only a small portion of Java, no reasonable jury could conclude that what was copied was qualitatively insignificant, particularly when the material copied was important to the creation of the Android platform. Google conceded as much when it explained to the jury the importance of the APIs to the developers it wished to attract. *See* Tr. of Proceedings held on 5/16/16 at 106:8-14, *Oracle Am., Inc. Google Inc.*, No. 3:10-cv-3561 (N.D. Cal. May 20, 2016), ECF No. 1930; *Id.* at 134:6-11. Indeed, Google's own expert conceded that "it was a sound

---

[11] In the prior appeal, we noted that "Google's competitive desire to achieve commercial 'interoperability' . . . may be relevant to a fair use analysis." *Oracle*, 750 F.3d at 1376–77. But, although several amici in this appeal discuss interoperability concerns, Google has abandoned the arguments it once made about interoperability. This change in course is not surprising given the unrebutted evidence that Google specifically designed Android to be *incompatible* with the Java platform and not allow for interoperability with Java programs. *Id.* at 1371.

business practice for Google to leverage the existing community of developers, minimizing the amount of new material and maximizing existing knowledge," even though Google also conceded that it could have written the APIs differently to achieve the same functions. *Id.* at 144:5-10. For these reasons, we find that the third factor is, at best, neutral in the fair use inquiry, and arguably weighs against such a finding.

Factor 4: Effect Upon the Potential Market

The fourth and final factor focuses on "the effect of the use upon the potential market for or value of the copyrighted work." 17 U.S.C. § 107(4). This factor reflects the idea that fair use "is limited to copying by others which does not materially impair the marketability of the work which is copied." *Harper & Row*, 471 U.S. at 566-67. It requires that courts "consider not only the extent of market harm caused by the particular actions of the alleged infringer, but also whether unrestricted and widespread conduct of the sort engaged in by the defendant . . . would result in a substantially adverse impact on the potential market for the original." *Campbell*, 510 U.S. at 590 (citation and quotation marks omitted).

The Supreme Court once said that factor four is "undoubtedly the single most important element of fair use." *Harper & Row*, 471 U.S. at 566. In its subsequent opinion in *Campbell*, however, the Court emphasized that none of the four factors can be viewed in isolation and that "[a]ll are to be explored, and the results weighed together, in light of the purposes of copyright." 510 U.S. at 578; *see also Infinity Broad.*, 150 F.3d at 110 ("Historically, the fourth factor has been seen as central to fair use analysis, although the Supreme Court appears to have backed away

from this position." (internal citation omitted)). The Court has also explained that "[m]arket harm is a matter of degree, and the importance of this factor will vary, not only with the amount of harm, but also with the relative strength of the showing on the other factors." *Campbell*, 510 U.S. at 590 n.21.

The Ninth Circuit recently indicated that likely market harm can be presumed where a use is "commercial and not transformative." *Disney Enters., Inc. v. VidAngel, Inc.*, 869 F.3d 848, 861 (9th Cir. 2017) (citing *Leadsinger*, 512 F.3d at 531, for the proposition that, where a use "was commercial and not transformative, it was not error to presume likely market harm"). That presumption allegedly traces back to *Sony Corp. of America v. University City Studios, Inc.*, 464 U.S. 417, 451 (1984), where the Supreme Court stated that, "[i]f the intended use is for commercial gain, that likelihood [of future harm] may be presumed. But if it is for a noncommercial purpose, the likelihood must be demonstrated." The Supreme Court has since clarified that market impact, "no less than the other three [factors], may be addressed only through a 'sensitive balancing of interests' " and that earlier interpretations of *Sony* to the contrary were incorrect. *Campbell*, 510 U.S. at 590 n.21 (quoting *Sony*, 464 U.S. at 455 n.40);[12] *see also Monge*, 688 F.3d at 1181 (cautioning against overemphasis on a presumption of market harm after *Campbell*). On this point, we must apply clear Supreme Court precedent

[12] The Court noted, however, that "what *Sony* said simply makes common sense: when a commercial use amounts to mere duplication of the entirety of an original, it clearly 'supersede[s] the objects,' of the original and serves as a market replacement for it, making it likely that cognizable market harm to the original will occur." *Id.* at 591.

rather than the more recent Ninth Circuit's statements to the contrary.

In evaluating the fourth factor, courts consider not only harm to the actual or potential market for the copyrighted work, but also harm to the "market for potential derivative uses," including "those that creators of original works would in general develop or license others to develop." *Campbell*, 510 U.S. at 592; *see also A&M Records*, 239 F.3d at 1017 ("[L]ack of harm to an established market cannot deprive the copyright holder of the right to develop alternative markets for the works."). A court can therefore consider the challenged use's "impact on potential licensing revenues for traditional, reasonable, or likely to be developed markets." *Swatch Grp. Mgmt. Servs. Ltd. v. Bloomberg L.P.*, 756 F.3d 73, 91 (2d Cir. 2014) (citation omitted); *see also Seltzer*, 725 F.3d at 1179 ("This factor also considers any impact on 'traditional, reasonable, or likely to be developed markets.' " (citation omitted)).

Also relevant to the inquiry is the fact that a copyright holder has the exclusive right to determine "when, 'whether and in what form to release' " the copyrighted work into new markets, whether on its own or via a licensing agreement. *Monge*, 688 F.3d at 1182 (quoting *Harper & Row*, 471 U.S. at 553). Indeed, the Ninth Circuit has recognized that "[e]ven an author who had disavowed any intention to publish his work during his lifetime" was entitled to copyright protection because: (1) "the relevant consideration was the 'potential market' " and (2) "he has the right to change his mind." *Worldwide Church*, 227 F.3d at 1119 (citing *Salinger v. Random House, Inc.*, 811 F.2d 90, 99 (2d Cir. 1987)); *see also Micro Star v. Formgen Inc.*, 154 F.3d 1107, 1113 (9th Cir. 1998) (noting that only

the copyright holder "has the right to enter that market; whether it chooses to do so is entirely its business").

Here, the district court concluded that the jury "could reasonably have found that use of the declaring lines of code (including their SSO) in Android caused no harm to the market for the copyrighted works, which were for desktop and laptop computers." *Order Denying JMOL*, 2016 WL 3181206, at *10. In reaching this conclusion, the district court noted that, before Android was released, Sun made all of the Java API packages available for free and open source under the name OpenJDK, subject only to the terms of a general public license. *Id.* According to the district court, the jury could have concluded that "Android's impact on the market for the copyrighted works paralleled what Sun already expected via its OpenJDK." *Id.*

On appeal, Oracle argues that the evidence of actual and potential harm stemming from Google's copying was "overwhelming," and that the district court erred as a matter of law in concluding otherwise. Appellant Br. 52. We agree.

First, with respect to actual market harm, the evidence showed that Java SE had been used for years in mobile devices, including early smartphones, prior to Android's release. Specifically, the jury heard testimony that Java SE was already in smartphones, including Blackberry, SavaJe, Danger, and Nokia. That Android competed directly with Java SE in the market for mobile devices is sufficient to undercut Google's market harm arguments. With respect to tablets, the evidence showed that Oracle licensed Java SE for the Amazon Kindle. After Android's release, however, Amazon was faced with two competing

options—Java SE and Android—and selected Android.[13] The jury also heard evidence that Amazon later used the fact that Android was free to negotiate a steep discount to use Java SE in its newer e-reader. In other words, the record contained substantial evidence that Android was used as a substitute for Java SE and had a direct market impact. Given this evidence of actual market harm, no reasonable jury could have concluded that there was no market harm to Oracle from Google's copying.

Even if there were a dispute about whether Oracle was licensing Java SE in smartphones at the time Android launched, moreover, "fair use focuses on *potential*, not just actual, market harm." *Monge*, 688 F.3d at 1181. Accordingly, although the district court focused exclusively on the market it found that Oracle had already entered—desktops and laptops—it should have considered how Google's copying affected potential markets Oracle might enter or derivative works it might create or license others to create. *See Campbell*, 510 U.S. at 590. Licensing Java SE for smartphones with increased processing capabilities was one such potential new market. And the fact that Oracle and Google engaged in lengthy licensing negotiations demonstrates that Oracle was attempting to license its work for mobile devices, including smartphones.[14] Smartphones were, therefore, a

---

[13] Google submits that the jury could have discounted this evidence because the Java SE APIs were available for free through OpenJDK. But Amazon moved from Java to Android—not to OpenJDK. And the evidence of record makes clear that device manufacturers did not view OpenJDK as a commercially viable alternative to using Java SE because any improvement to the packages in OpenJDK had to be given away for free to the Java community.

[14] Of course, the fact that those negotiations were not successful does not factor into the analysis. *Campbell*, 510 U.S. at 585 n.18, ("If the

"traditional, reasonable, or likely to be developed market." *See Swatch Grp.*, 756 F.3d at 91; *see also Seltzer*, 725 F.3d at 1179.

Google argues that a reasonable jury could have concluded that Java SE and Android did not compete in the same market because Oracle: (1) was not a device maker; and (2) had not yet built its own smartphone platform. Neither argument has merit. That Oracle never built a smartphone device is irrelevant because potential markets include licensing others to develop derivative works. *See Campbell*, 510 U.S. at 592. The fact that Oracle had not yet developed a smartphone platform is likewise irrelevant as a matter of law because, as Oracle submits, a market is a potential market even where the copyright owner has no immediate plans to enter it or is unsuccessful in doing so. *See Worldwide Church*, 227 F.3d at 1119; *Micro Star*, 154 F.3d at 1113. Even assuming a reasonable jury could have found no *current* market harm, the undisputed evidence showed, at an minimum, that Oracle intended to license Java SE in smartphones; there was no evidence in the record to support any contrary conclusion. Because the law recognizes and protects a copyright owner's right to enter a "potential market," this fact alone is sufficient to establish market impact.

Given the record evidence of actual and potential harm, we conclude that "unrestricted and widespread conduct of the sort engaged in by" Google would result in "a substantially adverse impact on the potential market for the original" and its derivatives. *See Campbell*, 510

---

use is otherwise fair, then no permission need be sought or granted. Thus, being denied permission to use a work does not weigh against a finding of fair use."). Such evidence was only relevant to show Oracle's interest in the potential market for smartphones.

U.S. at 590 (citation and quotation marks omitted). Accordingly, the fourth factor weighs heavily in favor of Oracle.

Balancing the Four Factors

Having undertaken a case-specific analysis of all four factors, we must weigh the factors together "in light of the purposes of copyright." *Campbell*, 510 U.S. at 578. We conclude that allowing Google to commercially exploit Oracle's work will not advance the purposes of copyright in this case. Although Google could have furthered copyright's goals of promoting creative expression and innovation by developing its own APIs, or by licensing Oracle's APIs for use in developing a new platform, it chose to copy Oracle's creative efforts instead. There is nothing fair about taking a copyrighted work verbatim and using it for the same purpose and function as the original in a competing platform.

Even if we ignore the record evidence and assume that Oracle was not already licensing Java SE in the smartphone context, smartphones were undoubtedly a potential market. Android's release effectively replaced Java SE as the supplier of Oracle's copyrighted works and prevented Oracle from participating in developing markets. This superseding use is inherently unfair.

On this record, factors one and four weigh heavily against a finding of fair use, while factor two weighs in favor of such a finding and factor three is, at best, neutral. Weighing these factors together, we conclude that Google's use of the declaring code and SSO of the 37 API packages was not fair as a matter of law.

We do not conclude that a fair use defense could never be sustained in an action involving the copying of

computer code. Indeed, the Ninth Circuit has made it clear that some such uses can be fair. *See Sony*, 203 F.3d at 608; *Sega*, 977 F.2d at 1527-28. We hold that, given the facts relating to the copying at issue here—which differ materially from those at issue in *Sony* and *Sega*—Google's copying and use of this particular code was not fair as a matter of law.

III. GOOGLE'S CROSS-APPEAL

Google cross-appeals from the district court's final judgment solely to "preserv[e] its claim that the declarations/SSO are not protected by copyright law." Cross-Appellant Br. 83. Specifically, Google maintains that the declaring code and SSO are: (1) an unprotected "method of operation" under 17 U.S.C. § 102(b), because they allow programmers to operate the pre-written programs of the Java language; and (2) subject to the merger doctrine. We resolved these issues against Google in the first appeal, finding that the declaring code and the SSO of the 37 API packages at issue are entitled to copyright protection. *Oracle*, 750 F.3d at 1354.

Google did not petition this court for rehearing and instead filed a petition for a writ of certiorari asking the Supreme Court to determine whether our copyrightability determination was in error. Oracle responded to the petition, and the Supreme Court invited the Solicitor General to express the views of the United States. The government agreed that Oracle's computer code is copyrightable, and the Supreme Court denied Google's petition in June 2015. *Google, Inc. v. Oracle Am., Inc.*, 135 S. Ct. 2887 (2015).

Google neither asks the panel for relief on the copyrightability issue nor offers any arguments on that

issue. We remain convinced that our earlier copyrightability decision was consistent with Congress's repeated directives on the subject. Accordingly, we provide no relief to Google on its cross-appeal, finding a ruling on it unnecessary.

IV. CONCLUSION

For the foregoing reasons, we conclude that Google's use of the 37 Java API packages was not fair as a matter of law. We therefore reverse the district court's decisions denying Oracle's motions for JMOL and remand for a trial on damages. The district court may determine the appropriate vehicle for consideration of infringement allegations regarding additional uses of Android. We dismiss Google's cross-appeal.

REVERSED AND REMANDED;

CROSS-APPEAL DISMISSED

COSTS

No costs.

Appendix B

IN THE UNITED STATES DISTRICT COURT FOR THE NORTHERN DISTRICT OF CALIFORNIA

ORACLE AMERICA, INC., Plaintiff, v. GOOGLE INC., Defendant.	No. C 10-03561 WHA

ORDER DENYING RENEWED MOTION FOR JUDGMENT AS A MATTER OF LAW AND MOTION FOR A NEW TRIAL

INTRODUCTION

In this copyright infringement action, the jury found the accused infringement constituted fair use. The copyright owner now renews its motion for judgment as a matter of law and separately moves for a new trial. For the reasons stated below, both motions are **DENIED**.

STATEMENT

The history of this case appears earlier (Dkt. No. 1988). In brief, Oracle America, Inc., formerly Sun Microsystems, Inc., has sued Google Inc. for copyright infringement with respect to Google's "reimplementation" of certain API packages in copyrighted Java 2 Standard Edition Versions 1.4 and 5. Following remand from the Federal Circuit, this action proceeded to a second jury trial on fair use, infringement otherwise having been established in the first trial as to certain uses. A pretrial

order divided the second trial into phases. Phase one addressed defendant Google's fair use defense. Had the jury found for Oracle during phase one, the same jury would have determined willfulness and monetary remedies in phase two. A third phase, before the judge only, would have determined whether Oracle deserved equitable remedies, including whether Google had equitable defenses.

In phase one, the ten-person jury returned a unanimous verdict finding that Google had carried its burden on the defense of fair use. A comprehensive order denied both sides' motions for judgment as a matter of law, so judgment was entered in Google's favor (Dkt. No. 1988).

Oracle now repeats its motion for judgment, adding a further motion for a new trial under Rule 59. This order follows full briefing, oral argument, and supplemental declarations addressing discovery issues raised in support of a new trial request.

ANALYSIS

1. RENEWED MOTION FOR JUDGMENT AS A MATTER OF LAW.

Oracle's new Rule 50 motion is denied for the same reasons as its old one (Dkt. No. 1988).[1]

2. MOTION FOR A NEW TRIAL.

[1] Oracle's argument that it is entitled to a new trial because the verdict was against the weight of the evidence, which incorporates by reference its brief on the motion for judgment as a matter of law, fails for the same reason.

Pursuant to Rule 59(a)(1)(A), a court may grant a new trial "for any reason for which a new trial has heretofore been granted in an action at law." Rule 61 provides that "no error in admitting or excluding evidence" constitutes a ground for granting a new trial "unless justice so requires." A district court has broad discretion in deciding whether to admit or exclude evidence. *Ruvalcaba v. City of Los Angeles*, 64 F.3d 1323, 1328 (9th Cir. 1995). A district court also has broad discretion in deciding whether to bifurcate a trial. *See Danjaq LLC v. Sony Corp.*, 263 F.3d 942, 961–62 (9th Cir. 2001). To warrant a new trial on these grounds, the movant must show that the Court's rulings constituted an abuse of discretion plus caused it substantial prejudice.

Oracle's motion for a new trial challenges several discretionary decisions made at trial. Oracle's primary argument, however, is that Google perpetrated discovery-concealment misconduct. The charged misconduct, Oracle says, rates as a "game changer." For important context, however, this order first addresses Oracle's related contention that the Court abused its discretion in limiting the trial to Android as used in smartphones and tablets, postponing all other uses to later trials.

A. New Device Categories and Scope of Trial.

The original trial in 2010 covered Android versions called 1.0, 1.1, Cupcake, Donut, Eclair, and Froyo, as used in smartphones and tablets. The original jury found those version infringed but deadlocked over fair use. On remand, the issue arose whether to retry that same case taking the infringement verdict as a given and postponing later developments to a future trial versus whether to expand the retrial to include post-2010 developments, a question that came into focus as follows.

After the remand, Oracle sought leave to file a supplemental complaint. Oracle's eventual motion for leave to file a supplemental complaint drew no opposition, and the motion was granted. The supplemental complaint identified six further versions of Android released since the original complaint. It further alleged that Google had implemented Android in various new device categories, including automobiles, wristwatches, televisions, and household appliances (Dkt. No. 1292).

Disagreement surfaced when the parties served their new expert reports. Oracle's expert reports evaluated Google's alleged use of new API packages from Java 2 Standard Edition Versions 6 and 7. But those versions had never been asserted in any operative pleading, including even the supplemental complaint. Only versions 1.4 and 5 had been asserted. Only versions 1.4 and 5 had been presented to the original jury and found to have been infringed. Google moved to strike the overreaching passages of Oracle's expert reports. This led to a hearing that featured the peril of the retrial spinning out of control via a piling on of everexpanding "updating" issues. The Court expressed concern over the ever-mounting prolixity of this case and the need for a cutoff of new device implementations to be tried (without prejudice to trying the rest later). The Court observed (Dkt. No. 1470 at 9–10):

> There's a much cleaner way to deal with this. We can roll back the clock to the moment that that [earlier] trial took place, and try it on that set of facts and the circumstances then. And then all these new products by [Oracle] and these new products by Google would not be in play. And

> what that means is, over there on the Google side, that you're going to have to face another lawsuit downstream . . .

In other words, the practical approach remained retrying the very trial revived by the Federal Circuit, complicated as it already was, preserving the infringement verdict, and saving for a later day all of the subsequent developments.

Nevertheless, the retrial expanded in two important ways. First, in light of Google's stipulation that the earlier jury's finding of infringement should apply to all later versions of Android up through Lollipop, a pretrial order eventually held that our retrial would cover those versions. A later stipulation included Marshmallow as well, adding a total of seven new major releases of Android to the original six. The second expansion was to include the post-2010 time period covered by these versions.

These expansions, by themselves, led to a vast inflation of Oracle's claimed recovery. At the first trial, Oracle's claim for monetary remedies clocked in at much less than a billion dollars, but now they rose to nine billion. The vast inflation flowed from the longer time period of sales of smartphones and tablets as well as the longer list of implicated versions of Android. The vast inflation resulted even though the uses on trial for the fair use defense remained, as before, smartphones and tablets.

The trial was not, however, expanded to include certain other more recent uses like Android TV, Android Auto, Android Wear, or Brillo. They presented a messier problem and were excluded from the scope of the upcoming trial (without prejudice to a later trial to cover them). Notably, the parties couldn't agree on whether the

original verdict of infringement would have covered those uses (since they arose after the original verdict, and no evidence on them was presented at the original trial). Had those uses been included in the retrial, Oracle would have had the burden, Google urged, to prove that those uses infringed, rather than relying, as Oracle wished to do, solely on the original verdict of infringement and imposing on Google the burden to prove fair use. Oracle offered to move for summary judgment to establish that the original finding of infringement should be extended to these new implementations, but by the time of that offer, there wasn't sufficient time for the Court to pursue that alternative while sorting out the superabundancy of pretrial issues.

To repeat, all agree that under the pretrial orders, Oracle remained (and remains) free to pursue its claims for infringement arising from Google's implementations of Android in devices other than smartphones and tablets in a separate proceeding and trial.

The scope-of-trial issue surfaced in a second way. Oracle sought to introduce evidence of the excluded device categories at trial as part of its evidence of market harm under the fourth fair use factor. An order *in limine*, however, held that the only uses set for trial were smartphones and tablets (again without prejudice to a separate future trial as to other uses) (Dkt. No. 1781).

In its new trial motion, Oracle now argues that it was error to limit the device uses in play to smartphones and tablets. We should have had one mega-trial on all uses, it urges. This, however, ignores the fact that Oracle's earlier win on infringement in 2010 — the same win it wished to take as a given without relitigation — concerned only smartphones and tablets. And, it ignores the obvious —

one use might be a fair use but another use might not, and the four statutory factors are to be applied on a use-by-use basis. Significantly, the language of Section 107(4) of Title 17 of the United States Code directs us to consider "the effect of *the use* upon the potential market for or value of the copyrighted work." Oracle cites no authority whatsoever for the proposition that all uses must stand or fall together under the fair use test of Section 107.

True, the fourth fair use factor must consider "whether unrestricted and widespread conduct of the sort engaged in by the defendant would result in a substantially adverse impact on the potential market for the original." *Campbell v. Acuff-Rose Music, Inc.*, 510 U.S. 569, 590 (1994). The concern with widespread use, however, is *not* whether uses *distinct from* the accused uses — each of which must be subject to distinct transformativeness analyses — might harm the market for the copyrighted works. Rather, the concern is whether a use of the *same* sort, if multiplied via use by others, would cause market harm, even though the actual use by the infringer caused only minimal harm. That is not our case. Again, our trial concerned two very important uses — smartphones and tablets — uses that implicated many billions of dollars. All other uses remained open for litigation in further trials.

Oracle relies on decisions from our court of appeals holding that supplementation of a complaint "is favored." *E.g.*, *Planned Parenthood of Southern Arizona v. Neely*, 130 F.3d 400, 402 (9th Cir. 1997). It argues that postponing its claims relating to devices other than smartphones and tablets contravened the purpose of "promot[ing] as complete an adjudication of the dispute between the parties as is possible." *LaSalvia v. United Dairymen*, 804

F.2d 1113, 1119 (9th Cir. 1986). Oracle provides a five-page description of the various markets such as automobiles, healthcare devices, "Internet of Things," appliances, and machine-to-machine communication — all involving vastly different technology and functionality from smartphones and tablets — in which Oracle has allegedly suffered harm due to Google's Android-related offerings.

Allowing complaints to be supplemented is favored, but a district judge still has a separate responsibility to manage complex cases, including to decide which issues should be tried in which trial. Good reasons rooted in case and trial management favored the eventual scope of our trial.

Oracle itself, it must be said, successfully excluded at least one post-2010 development that would have helped Google. Specifically, a pretrial ruling obtained by Oracle excluded evidence tendered by Google with respect to Android Nougat. Significantly, this evidence would have shown that (back in 2008) all of the accused APIs could simply have been taken from OpenJDK, Sun's own open-source version of Java, apparently in full compliance with the open-source license. Put differently, Sun itself had given away Java (including all of the lines of code in suit) in 2008 via its open-source OpenJDK. In 2015, Google used OpenJDK to reimplement the Java APIs for the latest release of Android, which it called Nougat. Google wished to use this evidence under the fourth fair use factor to show that its infringement did no more market harm than Sun itself had already invited via its own OpenJDK release. Despite its importance, the Court excluded this development because it had not been presented by Google in time for effective rebuttal by Oracle. This exclusion was a major win for Oracle in the weeks leading up to trial.

Oracle also argues that the first trial was not expressly limited to smartphones and tablets, so it was inappropriate to impose that limitation for the retrial. This isn't correct. In 2012, at our first trial, Oracle presented no evidence of any uses beyond smartphones and tablets. The other alleged uses lay in the future and were not considered by our first jury. Google simply had not yet implemented any aspect of Android on any of the new devices at that time.

After considerable deliberation, the Court exercised its discretion to limit the scope of our trial to address the issue of whether the uses of the copyrighted materials considered at the first trial — smartphones and tablets — including all thirteen versions of Android enabling those uses were fair or not, saving for a future trial new and different uses. In this way, Oracle was allowed to take unquestioned advantage of the infringement verdict in the first trial while also taking full advantage of the subsequent revenue derived from those very device implementations — smartphones and tablets. That limitation also protected our second jury from needing to absorb ever greater complexity in technology and the business models of new and different uses. Oracle remains free to pursue those new and later uses in a future lawsuit, but it is not entitled to a new trial as to smartphones and tablets.[2]

B. The Charge of Discovery Misconduct and ARC++.

---

[2] After the verdict, the Court invited counsel to propose scheduling for exactly such a trial on the alleged new and different uses, but both sides preferred to enter a final judgment and proceed to appeals with the understanding that the alleged new and different uses were still open for future lawsuits (Dkt. Nos. 2049–50).

With the benefit of the foregoing history of the smartphones and tablets limitation, we turn to Oracle's charge of discovery misconduct. This charge is not anchored in any claimed error by the judge but is anchored in claimed misconduct by Google and its counsel.

At both trials, Google argued that Android's use of the copyrighted lines of code qualified as "transformative" (under the first fair use factor) because Java had been designed for desktops and laptops whereas Android transformed the code at issue to work in the then newly-emerging world of smartphones and tablets. Thus, Google drew a significant distinction between desktops and laptops (Java) and smartphones and tablets (Android). Oracle now Google of withholding evidence in discovery that allegedly would have shown that Google was, by the close of our retrial, expecting soon to implement Android on desktops and laptops too. This argument will now be set out in detail.

Throughout the supplemental discovery period following the remand, Oracle sought discovery into all Google products that incorporated the copyrighted lines at issue. In response, Google identified its App Runtime for Chrome ("ARC"), which enabled laptops and desktops running Google's computer operating system, Chrome OS, to run certain Android applications. Chrome OS was and remains a different operating system from Android (Lin Dep. at 14–19, 107–09). ARC operated on top of Chrome OS and offered all of the Android APIs reimplemented from the Java code at issue. A related project, ARC Welder, enabled Android app developers to repackage the code in their apps for use on Chrome OS devices via ARC.

One of Oracle's own technical experts, Robert Zeidman, addressed ARC in detail in his opening report (Zeidman Rep. ¶¶ 126–43). Oracle's damages expert, James Malackowski, opined in his opening report that Google's release of ARC and ARC Welder and the availability of some Android functionality on Chrome OS devices "means Google is now using Android to occupy the original, traditional market of the Java Platform" (Malackowski Rep. ¶ 172). Oracle, however, never sought to introduce any of the evidence on which these comments were based (or to introduce the expert testimony). Oracle does not accuse anyone of misconduct as to ARC, but ARC supplies relevant background.

Now we come to the crux of the matter. In 2015, Google began a new project, which it internally called "ARC++." Among the goals of ARC++ was to "[p]rovide Chrome OS users with Play Android apps on Chrome OS without developer action" (Anderson Decl., Exh. 7 at *785). That is, Google intended for ARC++ to make the "entire Android app ecosystem" available on Chrome OS devices, so that Android apps would "appear alongside Chrome apps" in the Chrome OS program menu (*id.*, Exh. 8 at *404, Exh. 10 at *396). With ARC++, Google planned to run "Android in an isolated container inside Chrome OS," and "[i]nside the container should be effectively another Linux environment, similar to on an actual device" (*id.*, Exh. 9 at *417). That is, ARC++ would run an isolated instance of Android (with *all* of Android's public APIs, including those reimplemented from Java) in order to allow users to run all Android apps on Chrome OS devices. Google planned to include its "Play Store" — Google's app wherein users could purchase and download

other Android apps — as part of ARC++ to facilitate access to those apps.

In 2015, Google produced to Oracle at least nine documents relating to ARC++ setting forth the information in the preceding paragraph (along with more extensive technical details) and tracking the development of the project (Anderson Decl. ¶¶ 16–20, Exhs. 6–14). This is a key fact in resolving the accusation at hand. Our trial began on May 9, 2016. Our last day of evidence was May 19, which happened also to be the second day of Google's annual developer conference. On that day, Google announced via a blog post that it would make all Android apps available for use on Chrome OS devices via the Play Store (*id.*, Exh. 15). Although the announcement did not refer to this new feature as ARC++ (no name was given), it reflected the same goals and technical details as the ARC++ project. The announcement stated the feature would first roll out on the experimental developer channel, though over time it would become generally available. The same day at the developer conference, Google demonstrated the use of the Play Store with several Android apps on Chrome OS devices. The presenters acknowledged the technical limitations of the earlier ARC, stating that Google was "building a whole new platform to run Android apps on Chromebooks," *i.e.*, on laptops and desktops (Bush Decl., Exh. J at 3:30). One presenter explained that the new feature ran Android "directly on top of the Linux kernel [of Chrome OS]." Users could "run all of Android Marshmallow within Chrome OS. This includes the Google Play Store" (*id.* at 7:10).

In short, the announcement indicated that the full functionality of Android would soon be working on

desktops and laptops, not just on smartphones and tablets.

Oracle now contends that Google's failure to supplement several responses to interrogatories, requests for admission, and requests for production of documents, as well as the deposition testimony of two witnesses to reflect developments in the ARC++ project constituted discovery misconduct warranting a new trial.

"The test to be applied when discovery misconduct is alleged in a Rule 59 motion must be borrowed from cases interpreting Rule 60(b)(3) . . . ." *Jones v. Aero/Chem Corp.*, 921 F.2d 875 (9th Cir. 1990). Rule 60(b)(3) provides for relief from judgment for "fraud (whether previously called intrinsic or extrinsic), misrepresentation, or misconduct by an opposing party . . . ." To establish misconduct under Rule 60(b)(3), a moving party must:

> (1) prove by clear and convincing evidence that the verdict was obtained through fraud, misrepresentation, or other misconduct.
>
> (2) establish that the conduct complained of prevented the losing party from fully and fairly presenting his case or defense. Although when the case involves the withholding of information called for by discovery, the party need not establish that the result in the case would be altered.

Ibid. (quoting *Bunch v. United States*, 680 1271, 1283 (9th Cir. 1982)). A movant need not show that there would have been a different outcome without the alleged misconduct but need only demonstrate "'substantial interference' by showing 'the material's likely worth as trial evidence or by elucidating its value as a tool for obtaining meaningful

discovery.'" *Ibid.* (quoting *Anderson v. Cryovac, Inc.*, 862 F.2d 910 (1st Cir. 1988)).

Our court of appeals has recognized a "presumption of substantial interference if [the moving party] can demonstrate the misconduct was sufficiently knowing, deliberate or intentional." *Ibid.* Although *Jones* did not expressly lay out the framework for applying that presumption, it stated that *Anderson*, a decision from the First Circuit, "summarized the applicable standards and burdens of proof." *Ibid. Anderson*, 862 F.2d at 925, held that the presumption of substantial interference "may be refuted by clear and convincing evidence demonstrating that the withheld material was in fact inconsequential."

The oral argument on Oracle's motion for a new trial, which lasted two hours, focused almost exclusively on Oracle's "game changer" allegation of discovery misconduct. Following the hearing, counsel for both sides were ordered to file sworn declarations detailing Oracle's discovery requests on this point and Google's responses. After reviewing the parties' submissions, the Court called for sworn replies.

Throughout the briefing and argument on this motion, Oracle left the distinct impression — more accurately distinct misimpression — that Google had stonewalled and had completely concealed the ARC++ project. This was an unfair argument.

In fact, Google timely produced at least nine documents discussing the goals and technical details of ARC++ and did so back in 2015, at least five months before trial. Counsel for Oracle now acknowledges their legal team never reviewed those documents until the supplemental briefing on this motion (Hurst Reply Decl. ¶

12). The Court is disappointed that Oracle fostered this impression that no discovery had been timely provided on the ARC++ project eventually announced on May 19.[3]

Rule 26(e) requires a party to supplement discovery responses in a timely manner only "if the additional or corrective information has not otherwise been made known to the other parties during the discovery process or in writing" (or if otherwise ordered by the Court). This creates a "'duty to supplement,' not a right." *Luke v. Fam. Care and Urgent Med. Clinics*, 323 Fed. Appx. 496, 500 (9th Cir. 2009). Nevertheless, Google had no duty to supplement responses with new information that had already been disclosed in the ARC++ documents already produced.

Oracle should have known that items produced in response to its own document requests potentially contained information that supplemented Google's earlier

---

[3] Oracle contends that Google should have produced source code for the ARC++ project in response to a request for source code that "can be used to facilitate use of Android" on devices other than smartphones and tablets or that it should have identified ARC++ in an interrogatory seeking identification of "any software based on or derived from" Android that incorporated the 37 reimplemented Java API packages, among other similar requests. Google objected to vague language in those requests, and it was not clear to Google whether ARC++, which was in its early stages of development, would have been responsive to requests for information about "products," "software," or versions that were "developed or released," all of which are directed to completed projects. Indeed, the parties met and conferred about discovery responses and discussed Google's objections to Oracle's vague references to efforts to "port Android to desktop," but Oracle did not follow up on Google's objections (Anderson Decl. ¶¶ 30–39).

written discovery responses. Oracle's failure to review the ARC++ documents is its own fault.

It's important, most of all, to step back and remember the scope of our trial. Significantly, any evidence relating to implementations of Android on devices *other than* smartphones and tablets fell outside the scope of our trial, which was limited to uses on smartphones and tablets. Within the scope of our trial, therefore, Google fairly argued that Android was transformative because it took the declaring code in question, which had been designed for desktops and laptops, and reimplemented it for use in a new context, smartphones and tablets. It may well be true that the use of the copyrighted APIs in ARC++ (or any other later use) will not qualify as a fair use, but that will not and does not mean that Google's argument on transformative use as to the original uses on trial (smartphones and tablets) was improper. That Oracle failed to detect the ARC++ documents in its possession had no consequence within the defined scope of our trial.

Google committed a "fraud on the court," Oracle contends, by eliciting testimony that Android had not caused any harm to the market for the copyrighted works because it was not used on laptops and desktops. As stated, however, this remained a fair argument so long as the trial was focused, as it was, on the original uses — smartphones and tablets — and it remained a fair argument for the time period on trial (the blog announcement came later). The testimony and argument in question fell within the defined scope of our trial. Had Oracle brought up ARC or ARC++, the witnesses would plainly have clarified that their testimony related to the *accused* uses on trial.

Oracle further notes that the order denying its motion for judgment as a matter of law held that the jury could reasonably have found that "Android caused no harm to the market for the copyrighted works, which were for desktop and laptop computers" (Dkt. No. 1988 at 17). Again, "Android" in that context plainly referred to the accused original implementations of Android within the defined scope of our trial.

Google's launch of the full Android system on Chrome OS also remains, even now, in preliminary stages, available only to developers and on a limited set of devices. Oracle *already had evidence* of ARC++, but didn't realize it. Thus, to the extent Google's recent announcement had any value at our trial (or in discovery), Oracle already had evidence of the same project (and its predecessor), and it passed on any opportunity to introduce that evidence.

Nor would evidence of ARC++ have caused any interference relating to the Court's rulings limiting the scope of the trial. Indeed, in the briefing and argument on the scope of trial, Oracle never once mentioned ARC, ARC++, or any other use on laptop and desktop computers (neither did Google) (Dkt. Nos. 1559, 1612-3, 1643, 1682). This was so even though Oracle Expert Malackowski had already opined that the release of ARC "means Google is now using Android to occupy the original, traditional market of the Java Platform" (Malackowski Rep. ¶ 172). Instead, at oral argument, Attorney Lisa Simpson for Oracle identified "Android Auto" (not ARC or ARC++) as the most important implementation (to Oracle) that Oracle wished to add (Dkt. No. 1682, Tr. at 123). Oracle contends that the technical differences between ARC and ARC++ meant the latter presented a more compelling narrative both in

pretrial motion practice and at trial, but both projects made the same 37 reimplemented Java API packages available for use on Chrome OS; any differences between ARC and ARC++ remained peripheral to Oracle's interest in the projects.

Oracle's purported "game changer" would not have changed anything at all, because the scope of the "game" was smartphones and tablets, postponing new and later uses to a later contest. ARC++ was not yet on trial. Thus, any failure to produce such evidence could not have substantially interfered with Oracle's preparation for our trial. On the contrary, it clearly and convincingly would have been inconsequential.[4] Oracle insists on taking depositions and document discovery into Google's failure to supplement all discovery responses to reflect the imminent release of a developer version of ARC++ and to present its findings at an evidentiary hearing. Oracle cites *Jones v. Aero/Chem Corp.*, 921 F.2d 875 (9th Cir. 1990), for the proposition that failure to hold an evidentiary hearing on this issue would be reversible error. This type of fishing expedition will not be allowed, and *Jones* in no way requires such a course.

In *Jones*, two days after a jury found there had been no defect in the defendants' product, a third-party defendant produced a letter it received from one of the primary defendants nearly a decade earlier indicating that the primary defendant had known of the claimed defect and had explored remedial measures. The plaintiff moved

---

[4] Out of caution, this order makes clear that the test under Rule 59 is "substantial interference," not "game changer." The phrase "game changer" is Oracle's phrase, even if it expresses a less favorable test than here applicable. This order applies the correct test, "substantial interference."

for a new trial, claiming, *inter alia*, that the defendants had engaged in prejudicial discovery misconduct by withholding the correspondence. "At the hearing on the motion [for a new trial], the district court indicated it might later hold a hearing to determine whether [the] failure to produce the documents involved misconduct." *Id.* at 877.

Our court of appeals held that the district court improperly decided the motion based on= whether the withheld evidence would have resulted in a "different outcome," rather than whether it caused "substantial interference," as required by decisions interpreting Rule 60(b)(3). The failure to hold a separate "hearing" — the court of appeals never referenced an "evidentiary hearing," contrary to Oracle — on the issue was a background circumstance. The actual error was in the *standard* applied, not the procedure for applying that standard. Notably, the court of appeals did not even *require* the district court to hold a subsequent hearing, but rather directed it to hold "appropriate proceedings to determine" whether discovery misconduct had occurred according to the proper standard.

In our case, the Court did hold "appropriate proceedings" and did hold a hearing at which the proper standard — Rule 60(b)(3) — was considered, and it further required sworn statements from counsel for both sides and then invited and considered sworn replies, all detailing the discovery conduct at issue. After reviewing many pages and exhibits, the Court finds that no misconduct has been shown (or would likely be shown even with the benefit of a fishing expedition). Nor could any omission of evidence relating to ARC++ have interfered with Oracle's case *at all*, much less substantially. Contrary

to Oracle, ARC++ documents were in fact timely produced. They laid out the basic goals and technical details of the very product referenced on May 19. Since Oracle had that information, there was no need to supplement the written discovery to the extent evidence of ARC++ was responsive at all. Moreover, any further disclosure of ARC++ would have been of no consequence in Oracle's preparation for our trial or its presentation at trial, which later became limited in scope to smartphones and tablets. This ground for a new trial is rejected.

C. Steffano Mazzocchi.

Oracle next contends that a new trial is warranted due to the exclusion of minor evidence and testimony from Stefano Mazzocchi, a member of the board of directors of the Apache Software Foundation in 2008. Back then, Mazzocchi volunteered as a mentor overseeing the Apache Harmony Project and as a member of its Project Management Committee, which sought to create and offer an open-source reimplementation of the Java API. Google eventually used portions of the Harmony project in its reimplementation of 37 Java API packages in Android. Later on, Mazzocchi went to work for Google, but at the relevant time, he worked for neither side.

At our trial, Google presented evidence first (having the burden of proof), but it did notcall Mazzocchi as a witness. Nevertheless, Google otherwise introduced evidence of Harmony to support its position that reimplementation of APIs without licenses flowered in the industry.

Oracle never properly designated Mazzocchi as a trial witness under Rule 26(a). Oracle wished to lay before the jury an email that Mazzocchi had sent in April 2008 during

the development of Apache Harmony. (In fact, the exhibit was an email from the vice president of legal affairs at Apache and incorporated and responded to an email from Mazzocchi.) Despite Oracle's Rule 26 violation, the Court acquiesced in allowing Oracle to present almost everything it wished to present, including Mazzocchi and the email, save and except for two minor items.

Mazzocchi's email went to a mailing list of members of Apache (TX 5046). It expressed concern that Apache could not distribute Harmony without a license from Sun, even with new implementing code, because "the copyright on the API is real and hard to ignore." Mazzocchi added, "[s]o, we are, in fact, infringing on the spec lead copyright if we distribute something that has not passed the TCK and *we know that*." Our jury heard Mazzocchi's testimony regarding this email, and the entire email itself, including the quotations above, went into evidence, subject to one redaction.

That redaction is now the basis for Oracle's first assignment of error. Its second is that Oracle was precluded from eliciting testimony that Mazzocchi worked for Google at the time of the *trial*, though he had worked elsewhere when he sent the email.

(i) Redaction.

The Court held that Mazzocchi could testify and that his emails would be admitted, over Google's objection, subject to redaction of the following sentence in the email (TX 5046):

> This makes us *already* doing illegal things (in fact, Android using Harmony is illegal as well).

An exchange regarding that redaction occurred (outside the presence of the jury) as follows (Tr. at 1588):

> THE COURT: However, the one sentence that I think is too inflammatory and without foundation and should come out is the one sentence that says "This makes us *already* doing illegal things (in fact, Android using Harmony code is illegal as well)." That should not be used. But the two paragraphs that I think you're more interested in, they can be used. So that one sentence about "This makes us *already* doing illegal things (in fact, Android using Harmony code is illegal as well)" that should be deleted or at least redacted.
>
> MS. HURST (for Oracle): We'll redact that, Your Honor.

Although, as just shown, Oracle's counsel readily accepted that redaction and the email, as redacted, went before the jury, Oracle later — only after Mazzocchi had finished his testimony and had been excused — requested that the Court remove the redaction (Dkt. No. 1925). This was denied, a denial that forms a basis for the new trial motion.

Oracle now argues that sufficient foundation existed because Mazzocchi had "corresponded with the Apache Foundation's VP of Legal Affairs regarding legal issues related to use of copyrighted Java APIs in the Harmony Project" (Pl.'s Mtn. at 16) (citing Tr. at 1712–13).

The so-called "correspondence" with the lawyer, it turns out, went into evidence as the thread leading up to the "Mazzocchi email" (TX 5046; Tr. at 1715). So, whatever foundation existed for the redacted sentence made its way to the jury anyway. (Perhaps this hearsay from the lawyer

shouldn't have been admissible at all, but no objection on that ground was made.)

Significantly, nowhere in any passage written by any lawyer did anything come close to what Mazzocchi said in the redacted sentence. So, the thread itself supplied inadequate foundation. Even if Mazzocchi had consulted a lawyer beyond the thread itself (and no such consultation was ever intimated), Mazzocchi himself was *not* a lawyer, so merely repeating what some lawyer might have told him would have been hearsay (within hearsay).

Indeed, Mazzocchi's testimony before the jury demonstrated that his legal conclusion was utterly without qualification (Tr. at 1727–28):

> [MR. KWUN (for Google)]. So thinking back to April of 2008, what, if anything, did you know about fair use in copyright law?
>
> A. I don't recall knowing anything about that.
>
> Q. Did you know what the legal standard is for fair use?
>
> A. I don't — didn't and still don't.
>
> Q. After the email exchange with Mr. Ruby, did you resign as a member from the Apache Software Foundation?
>
> A. No.
>
> Q. And what, if anything, do you conclude from the fact that you did not resign your membership after that email?
>
> A. I really cared about my involvement in Apache. I mean, this was all volunteer work, and I really wanted the foundation to do the right

> thing for protection of the membership and also for protection of the users.
>
> I would have left slamming the door if I thought that what the foundation was doing was causing harm or doing any illegal things.
>
> So since I wrote these email [sic], I must have changed my mind, something must have changed my mind whether that was the case.
>
> And I didn't leave.

Notwithstanding Mazzocchi's lack of training in the law, the Court allowed Oracle to make hay with "the copyright on the API is real and hard to ignore" and that releasing Harmony's reimplementation of the Java API code without passing the compatibility test would have constituted "infringing on the spec lead."[5]

It is worth stressing that the email made no mention of "fair use." It had nothing to do with the fair use issue our jury had to decide. Mazzocchi admitted that he knew nothing about fair use. The Court had already told the jury that Android infringed the copyright subject only to the fair use defense, so a good case existed for excluding the entire email. Nevertheless, virtually all of it came in.

Nor did Mazzocchi's testimony, elicited by Google, that he "would have left slamming the door [at Apache] if [he] thought that what the foundation was doing was

---

[5] The Court similarly restricted Google from eliciting legal conclusions from former Sun CEO, Jonathan Schwartz, about whether Sun had any legal claim against Google. After his testimony veered too close to that conclusion, the Court issued a corrective instruction and allowed Oracle to question Schwartz about a document that Oracle had improperly clawed back as privileged (Tr. at 508–10, 526). (Schwartz could not recall the document, so it was not admitted into evidence.)

causing harm or doing any illegal things" open the door to using the redaction. Mazzocchi's testimony already responded to his understanding that Apache was infringing on Oracle's copyright, and by noting that something "changed [his] mind," he acknowledged that his email reflected initial concern about the legality of Apache's work anyway. Admission of the redaction would have been cumulative.

(ii) Mazzocchi's Employment.

Oracle also contends that it should have been permitted to cross-examine Mazzocchi based on his alleged bias as a current employee of Google. When the Court initially allowed Oracle, despite its inadequate Rule 26 disclosure and over Google's strenuous objection, to call Mazzocchi as a witness, the Court did so to allow presentation of his views when he worked for Apache in 2008 and ruled as follows (Tr. at 1589):

> And don't bring up that he works at Google now unless bias becomes a problem. If it appears he's been coached to say things that may not be true, possibly then I would allow you to bring up that he works for Google and that Google — he has met with the lawyers and so forth. But for the time being, you should steer clear of that. And you may treat him as an adverse witness.

During direct examination before the jury, and without seeking leave to address the issue, counsel for Oracle asked Mazzocchi (after he denied recollection of the email containing the "illegal things statement") whether he had met with Google's trial lawyers, which he confirmed he had (Tr. at 1724). The Court allowed the questions over Google's objection.

On cross-examination by Google, as stated, Mazzocchi testified that following the email addressing the issue of Oracle's copyright in the Java APIs with regard to Harmony "something must have changed my mind whether that was the case" (Tr. at 1727). When Google passed the witness back for redirect, Oracle requested a sidebar to be allowed to elicit the fact that Mazzocchi became employed at Google the following year, in order to suggest it was his later employment with Google that had "changed his mind" about the legal status of the Apache Harmony project.

At the sidebar, the Court reviewed Mazzocchi's testimony and concluded that he testified that he would have left Apache sooner than 2009 if he had believed it had been doing something illegal, while he didn't begin his employment with Google until 2010. Contrary to Oracle, Mazzocchi's testimony suggested that something changed his mind *before* he began working at Google.

Even so, Oracle was able to offer evidence of Mazzocchi's purported bias by eliciting testimony that Mazzocchi spoke with Google's counsel before testifying (Tr. at 1724). Thus, the probative value of evidence of Mazzocchi's then-current employment was minimal, particularly in light of the substantial risk that the jury would mistakenly ascribe Mazzocchi's state of mind while at Apache to Google. (Indeed, Oracle sought to ascribe Mazzocchi's *shift* in his state of mind to Google, although it predated his employment with Google.)

In the larger picture, the jury heard evidence, pro and con, from both Sun (Oracle) and Google personnel concerning the extent to which reimplementation of APIs occurred in the industry. In view of this sea of evidence, the Mazzocchi email was cumulative. Nevertheless,

virtually all of the email came into evidence, including his statement that reimplementing the Java API in particular constituted infringement of the copyright.

Thus, Oracle's contention that it is entitled to a new trial on the basis of the excluded evidence relating to Mazzocchi is rejected.

D. European Commission Response.

Oracle next contends that the Court improperly excluded a document containing responses to questions posed by the European Commission in connection with its 2009 review of Oracle's acquisition of Sun. The question called for an explanation of "the conflict between Sun and Google with regard to Google's Android" (TX 5295 at 39). Oracle sought to admit its response, which read, "Sun believes that the Dalvic [sic] virtual machine plus class libraries, which together constitute Android runtime environment, are an unauthorized derivative work of Java SE" (*ibid.*). Oracle wished to lay this response before the jury to meet testimony by Sun's former CEO, Jonathan Schwartz, that Sun had welcomed Google's then-recent announcement of Android as part of the Java community, and that industry reimplementations of the Java API had promoted rather than hindered Sun's business plan.

To avoid the self-serving hearsay problem, Oracle attempted to lay foundation for the response through the testimony of its CEO, Safra Catz, who oversaw the acquisition and testified that Sun (not Oracle) had supplied the answer. Out of the presence of the jury, the Court stated it would consider allowing Oracle to admit the response if it had originated with Sun rather than Oracle (Tr. at 1314).

The next morning, out of the presence of the jury, Oracle proffered several drafts of the response to the European Commission. These drafts purportedly traced earlier versions of the response. They originated from Sun's in-house intellectual property counsel. Google protested that these drafts had long been withheld from Google as privileged until the previous night, so that it had had no opportunity to vet Oracle's representations about the drafts. Counsel for Oracle responded that Oracle would waive the privilege. This after-the-deadline waiver, Google replied, failed to cure the prejudice. Temporizing, the Court warned Oracle that its disclosure of privileged documents would constitute an extraordinary waiver (Tr. at 1328).

Nevertheless, still out of the presence of the jury and using the privileged documents, counsel for Oracle traced the internal development of the response to the European Commission.

One draft stated, colorfully, "[a] recidivist bank robber should not complain, at least to the authorities, that the bank's new owner might increase security measures around the bank" (Tr. at 1330). A subsequent email from Sun's in-house counsel noted that Oracle's corporate counsel had removed the colorful language and stated "Re Android, we liked our recidivist bank robber analogy" (Tr. at 1331). In light of its document tracing, Oracle proposed that Catz be permitted to testify that the response to the European Commission originated with Sun (how she would have known that on her own was never explained).

The Court rejected that proposal, a rejection that now serves as a ground for the Rule 59 motion.

It is true that Google presented evidence at trial that Sun had embraced a custom of reimplementation of APIs and that Sun's CEO had welcomed Android to the Java community. It is further true that Google argued to the jury that this welcoming attitude reversed only after Oracle took over Sun and brought this suit. Oracle was free to present counterevidence (and did) but the extraordinary after-the-deadline waiver of privilege was too timewise prejudicial to Google, should not have been allowed, and was not.[6]

Oracle's gamesmanship deprived Google of a fair opportunity to vet the privileged documents and to verify the supposed chain of authorship. Anyway, the timing of the emails (at a time when Sun's employees had cause to curry favor with their new boss) suggested that any response "from Sun" was really "from Oracle." This ground for a new trial is rejected.

E. Self-Serving In-House Presentations

Oracle was barred from placing in evidence certain self-serving in-house materials, offered supposedly to show how Android had hurt Oracle's markets for Java. Specifically, as part of its evidence on market harm under the fourth fair use factor, Oracle sought to admit Trial Exhibits 5961, 6431, and 6470, which were in-house slide show presentations at Oracle. They were used "as [Oracle's] way of planning for [the] next year. They're also used to educate [Oracle's executives] about what is going on in the business" (Tr. at 1356). The presentations

[6] Counsel for Oracle contended they could offer an email from 2008 in which someone internal to Sun stated Google's conduct constituted copyright infringement, but no such document was ever shown to the Court or offered into evidence.

included slides that discussed the purported impact of Android on Oracle's revenue.

Oracle invoked Rule 803(6) of the Federal Rules of Evidence, which provides an exception to the rule excluding hearsay evidence for records of a regularly conducted activity, as follows:

> A record of an act, event, condition, opinion, or diagnosis if:
>
> (A) the record was made at or near the time by — or from information transmitted by — someone with knowledge;
>
> (B) the record was kept in the course of a regularly conducted activity of a business, organization, occupation, or calling, whether or not for profit;
>
> (C) making the record was a regular practice of that activity;
>
> (D) all these conditions are shown by the testimony of the custodian or another qualified witness, or by a certification that complies with Rule 902(11) or (12) or with a statute permitting certification; and
>
> (E) the opponent does not show that the source of information or the method or circumstances of preparation indicate a lack of trustworthiness.

The Oracle-made documents contained slides with "highlights" and "lowlights" of certain fiscal years, identified "priorities and key messages," summarized revenue data, forecasts, and budgets, identified market challenges, and mapped out product strategies (Bush Decl., Exhs. 26, 27, 29). As to Trial Exhibit 5961, Oracle

offered the testimony of its CEO, Safra Catz, to lay the foundation that the presentation had been prepared as part of Oracle's annual budget review (Tr. at 1357). When Oracle moved to admit that exhibit into evidence, Google objected, and the Court sustained the objection because it remained simply a slide show of internal self-serving propositions (even worse, created pending this lawsuit). The Court stated, "if it was just a financial statement, I would allow it, but there are too many slide shows in that document to qualify it as a business record" (Tr. at 1357). Counsel for Oracle sought to admit just page 21 of the exhibit, but that page, titled "FY11 Priorities and Key Messages — Java" suffered from the same self-serving problems. Indeed, that page addressed "integration-specific concerns" regarding the integration of Sun into Oracle — hardly a regularly-conducted activity.[7]

Oracle sought to admit similar presentations, Trial Exhibits 6431 and 6470, through the testimony of its former vice president of worldwide original electronic manufacturer sales, Neal Civjan, but those presentations were excluded on similar grounds.

Rule 803(6) is not an open window through which any self-serving in-house internal hearsay sails into evidence at the author's behest:

> The element of unusual reliability of business records is said variously to be supplied by

[7] In its brief, Oracle describes this page as "a spreadsheet of revenue and expenses for the first two quarters of fiscal year 2011 for Java embedded and forecasts for the third quarter" (Pl.'s Mtn. at 23). Page 21 does not meet that description. It is possible, it now appears, that counsel for Oracle intended to direct Catz and the Court to page 23, but that error by Oracle then would not now be a reason to grant a new trial.

> systematic checking, by regularity and continuity which produce habits of precision, by actual experience of business in relying upon them, or by a duty to make an accurate record as part of a continuing job or occupation.

N.L.R.B. v. First Termite Control Co., Inc., 646 F.2d 424, 427 (9th Cir. 1981), *opinion amended on reh'g sub nom. Natl. Lab. Rel. Bd. v. First Termite Control Co. Inc.* (9th Cir. Aug. 5, 1981); *see also* Advisory Committee Notes, 1972 Proposed Rules, Note to Paragraph (6).

The Oracle presentations sought to be admitted were not the kinds of records that could be assured of their reliability due to systematic checking or habits of precision. On the contrary, the documents contained narrative, analysis, and commentary — *i.e.*, self-serving argument. The only "regularity" of the self-serving presentations was that they arose as part of an annual budget review, but the statements themselves had not derived from such a systematic habit of precision. They otherwise lacked the indicia of trustworthiness sought by Rule 803(6). They were properly excluded as hearsay.

F. Bifurcation.

A pretrial order bifurcated the issues of fair use from willfulness and monetary remedies (Dkt. No. 1321 at 13). This prejudiced Oracle, it asserts, because "important market harm testimony never made it to the jury because it was relegated to the damages phase" and because "bifurcation provided a structural incentive for the jury to return a defense verdict" (Pl.'s Mtn. at 20).

Oracle's argument that bifurcation precluded it from presenting its market harm evidence is simply untrue. Nothing about the bifurcation precluded Oracle in phase

one from presenting evidence of Oracle's lost revenue attributable to Android. Indeed, Oracle presented extensive evidence in phase one directed at the issue of market harm to the copyrighted works, the fourth fair use factor.

Although there was some overlap in the evidence relevant to market harm and Oracle's actual damages (and Oracle remained free to present it in phase one and did), the most complex evidence on Oracle's remedies — *the disgorgement of Google's profits* — had virtually no relevance to the market harm/fair use inquiry. Section 107(4) on fair use focuses on the "effect of the use upon the potential market for or value of the copyrighted work" (*i.e.*, harm to Oracle). Section 504(b) on remedies allows a copyright owner to recover "the actual damages suffered . . . as a result of the infringement" (again, harm to Oracle) *as well as* "any profits of the infringer that are attributable to the infringement" (*Google's* profits from infringement) — to the extent the awards are not duplicative. Put differently, phase one focused on market harm to the copyrighted work whereas phase two focused on Oracle's damages from that market harm *and* possible disgorgement of Google's profits attributable to the infringement. *Oracle's claim for disgorgement of Google's profits totaled more than ten times Oracle's claimed actual damages and thus would have dominated Oracle's case in phase two.*[8]

---

[8] Google contended that the issue of disgorgement should not be presented to a jury. An order held that the jury would rule on disgorgement, but the Court would resolve Google's argument after the verdict, possibly treating the jury's verdict as advisory, if not conclusive (Dkt. No. 1769).

The disgorgement issue presented extraordinary complexity — complexity unrelated to market harm to the copyrighted works. For one, Google never directly *sold* Android. Instead, Google offered it free to all comers as open source. Google benefited indirectly. It used Android as a platform for its *other* services, which earned revenue from advertisements and sales of apps and media. But these other services (like its popular search engine) had already been operating and earning revenue well before Android. Oracle conceded this but contended that Android had multiplied that revenue. Thus, to isolate profits attributable to use of Oracle's code, the jury would have been required to apportion, first of all, the revenue between the pre-existing technology already in place versus Android.

Next the jury would have had to further apportion between the accused lines of code versus the unaccused lines of code within Android. The infringing part of Android constituted only a small fraction of one percent of Android. Oracle conceded this but contended that this sliver held the key to the success of Android. These apportionment difficulties were just two examples of many posed by the disgorgement claim for our jury.

Thus, phase two was poised to present bone-crushing analytics on how to apportion any Android profits attributable to the infringement versus profits attributable to non-infringement. To meet this challenge, the parties presented dueling economic models yielding massively different answers. Again, unlike Oracle's lost profits segment, the apportionment/disgorgement problems had virtually no relevance to market harm and fair use.

In the Court's judgment and discretion, our trial was best managed by postponing that mind-bender to phase two, so that the jury could give its undivided attention in phase one to the critical issue of fair use. Dividing the trial further served the important purpose of saving the resources of the Court and the jury (and the parties) in the event that the jury decided against Oracle on fair use.

To repeat, Oracle was free to present its lost profits and other market harm evidence in phase one — and it did so at length. (In phase two, all previously admitted evidence would still have been deemed in evidence.)

Turning to Oracle's structural incentive argument, the Court instructed the jury not to allow any desire to conclude the trial sooner to influence its decision. We must presume the jurors followed the instruction, and there is nothing to indicate otherwise. *Richardson v. Marsh*, 481 U.S. 200, 206 (1987). Oracle's structural incentive argument, such as it is, would undermine every bifurcation of damages from liability. Yet the law plainly allows bifurcation.[9]

It deserves to be said, in favor of our jury, that the ten who served were as punctual, attentive, and diligent in note-taking as any jury this district judge has seen in seventeen years of service. They had all cleared their

---

[9] The Court instructed the jury as follows (Dkt. No. 1950 ¶ 46):

> Once you render a verdict on the fair use question, we may proceed to the shorter and final phase of the trial on damages issues, depending on your answer to the fair use question. This would still be within the June 10 end date stated earlier. Please do not allow any desire to complete trial sooner to influence your thinking. Once you render your verdict on the fair use issue, it will be final and may not be re-visited or modified during the second phase.

calendars. We were on target to meet or beat the time estimate given to the jury. Those with hardships had already been excused during jury selection. It is impossible to even suspect that bifurcation somehow steered the jury to rule as it did. The Court remains completely convinced that the verdict rested, after three days of deliberation, solely on the jury's sincere assessment of the evidence and the instructions of law.

CONCLUSION

For the reasons stated above, Oracle's motion for a new trial and its motion for judgment as a matter of law are **DENIED**.

IT IS SO ORDERED.

Dated: September 27, 2016.

/s/

WILLIAM ALSUP, UNITED STATES DISTRICT JUDGE

Appendix C

IN THE UNITED STATES DISTRICT COURT FOR THE NORTHERN DISTRICT OF CALIFORNIA

ORACLE AMERICA, INC., Plaintiff, v. GOOGLE INC., Defendant.	No. C 10-03561 WHA

ORDER DENYING RULE 50 MOTIONS

In this copyright case, the Federal Circuit remanded for a second jury trial on the issue of fair use, rejecting the argument of Oracle America, Inc., that the first trial record entitled it to judgment as a matter of law and that a remand on that issue would be "pointless" (Br. at 68). Now, after an adverse verdict in the second trial, Oracle again asserts that it is entitled to judgment as a matter of law on fair use. For the same reasons as before, Oracle is wrong in saying that no reasonable jury could find against it.

Under the law as stated in the final charge and on our trial record, our jury could reasonably have found for either side on the fair use issue. Our trial presented a series of credibility calls for our jury. Both sides are wrong in saying that all reasonable balancings of the statutory factors favor their side only. To the extent either side now quarrels with the law as stated in the final charge (Dkt. No. 1950), the time for those arguments was at or before the charging conference or eventually on appeal. For now,

at the district court, the jury instructions control. Based on those instructions, the Rule 50 motions must be **DENIED**. Since an appeal is promised, however, it may be of assistance to leave a few important observations.

1. The fair use instructions followed largely the review of fair use law as set forth in the Federal Circuit's opinion except for modifications urged by counsel and to account for how the case was actually tried. The final jury charge culminated an exhaustive and iterative process of proposals by the judge followed by critiques by counsel. Months before trial, the Court informed both sides that it expected to use the Federal Circuit's opinion canvassing fair use law as the starting point and requested briefing from the parties addressing what modifications should be made (Dkt. Nos. 1518, 1519 at 51). After reviewing those comments, the Court circulated a first proposed charge on fair use and requested critiques (Dkt. No. 1615). Counsel submitted their critiques a week later with replies the following week. In light of the critiques, a second draft made substantial revisions (Dkt. Nos. 1688, 1716), asking counsel to meet and confer to reach an agreed-on instruction in light of that proposal and to submit briefs and responses regarding the areas of disagreement. After reviewing the further briefs and responses, the Court next circulated "penultimate instructions on fair use," a third draft, and invited a third round of comment (Dkt. No. 1790). Those critiques also led to modifications and a final notice of the pre-instruction on fair use to be read to the jury before the start of the evidence (Dkt. No. 1828). Counsel (and the jury) were advised that the final instructions at the end of the evidence would possibly be adjusted to reflect the way the case was tried (and, in fact, some minor modifications did occur). During the trial, the

judge sought briefs on several issues in play as the evidence came in. Based thereon, a notice of the proposed final charge circulated the night before the close of evidence (Dkt. No. 1923). At the charging conference, counsel raised both new points and old ones (although they were permitted to rest on prior critiques). Final modifications followed. The jury was charged accordingly (Dkt. No. 1950) evidence (Dkt. No. 1828). Counsel (and the jury) were advised that the final instructions at the end of the evidence would possibly be adjusted to reflect the way the case was tried (and, in fact, some minor modifications did occur). During the trial, the judge sought briefs on several issues in play as the evidence came in. Based thereon, a notice of the proposed final charge circulated the night before the close of evidence (Dkt. No. 1923). At the charging conference, counsel raised both new points and old ones (although they were permitted to rest on prior critiques). Final modifications followed. The jury was charged accordingly (Dkt. No. 1950).

2. On fair use, Oracle's most emphatic argument remains the "propriety of the defendant's conduct," meaning the subjective awareness by Google Inc. of the copyrights and, construing its internal e-mails in a light most unfavorable to Google, its "bad faith." This, however, underscores an important point for the appeal. Although the Federal Circuit opinion omitted any reference to the "propriety of the defendant's conduct" (good faith versus bad faith) as a consideration under any part of the four-factor test for fair use, Oracle insisted on remand that the jury be told that it could consider bad faith by our accused infringer as a subfactor under Factor One. This Court acquiesced in Oracle's view and did so despite an omission — conceivably a studied omission — of any such

consideration in the Federal Circuit opinion and despite the fact that there is a respectable view that good or bad faith should no longer be a consideration after the Supreme Court's decision in *Campbell v. Acuff-Rose Music, Inc.*, 510 U.S. 569, 585 n.18 (1994). *See* 2 Paul Goldstein, *Goldstein on Copyright* § 12.2.2, at 12:44.5–12:45 (3d ed. 2016). Put differently, either a use is objectively fair or it is not and subjective worry over the issue arguably should not penalize the user.[1]

Still, Oracle is correct that the Supreme Court in *Harper & Row Publishers, Inc. v. Nation Enterprises*, 471 U.S. 539, 562 (1985), called out the propriety of the defendant's conduct as a consideration in that case. Footnote 18 in *Campbell* later questioned whether or not propriety should persist as a consideration but did not rule it out. At the district court level, we must treat *Harper & Row* as still the law and leave it to the appellate courts to revise (although our instructions included a modification based on *Campbell*). This is no small point in this case, for

---

[1] On appeal, Oracle argued as follows (Br. 72):

> Finally, "[f]air use presupposes good faith and fair dealing." *Harper & Row*, 471 U.S. at 562 (internal quotation marks omitted). Google considered, negotiated, and ultimately rejected the opportunity to license the packages, deciding to "[d]o Java anyway and defend our decision, perhaps making enemies along the way." A1166. That Google knew it needed a license, and then sought but did not obtain one, weighs heavily in showing "the character of the use" was not fair. *Los Angeles News Serv. v. KCAL-TV Channel 9*, 108 F.3d 1119, 1122 (9th Cir. 1997). Google "knowingly . . . exploited a purloined work for free that could have [otherwise] been obtained." *Id.*

Despite this argument, the Federal Circuit did *not* include this consideration in its discussion of factors on fair use, remaining silent on the issue.

no Oracle jury argument received more airtime than its argument that Google "knew" it needed a license and chose in bad faith to "make enemies" instead.

This leads to a reciprocal key point. Given that Oracle was allowed to try to prove Google acted in bad faith, Google was allowed to try to prove good faith. Its witnesses testified that they had understood that "re-implementing" an API library was a legitimate, recognized practice so long as all that was duplicated was the "declaring code" and so long as the duplicator supplied its own "implementing code," that is, the methods were "re-implemented." In this way, Java programmers using the Android API could call on functionalities with the same Java command statements needed to call the same functionalities in the Java API, thereby avoiding splintering of the ways that identical functionalities became invoked by Java programmers.

Google asked to go a step further and asked for an instruction on "custom," citing *Wall Data Inc. v. Los Angeles County Sheriff's Dept.*, 447 F.3d 769, 778 (9th Cir. 2006), which stated "fair use is appropriate where a 'reasonable copyright owner' would have consented to the use, *i.e.*, where the 'custom or public policy' at the time would have defined the use as reasonable" (quoting Subcomm. on Patents, Trademarks & Copyrights of the Sen. Comm. on the Judiciary, 86th Cong., 2d Sess., Study No. 14, *Fair Use of Copyrighted Works 15* (Latman) (Comm. Print 1960)). Oracle objected on the ground that custom was omitted from the Federal Circuit opinion. It was omitted from that opinion — true. But neither was there any mention in that opinion of the propriety of the accused infringer's conduct. So, that omission by itself wasn't a good reason to ignore a pertinent statement by

the Ninth Circuit, the law applicable in this copyright case arising in the Ninth Circuit. Oracle also argued that "custom" had to be vastly more entrenched than the "practice" evidence Google wished to present.

Whether or not the evidence would have warranted a *Wall Data* instruction on custom, the fact remained that once Oracle endeavored to prove bad faith, it opened the door for Google to prove good faith, so Google explained its mental state and explained that it believed it had followed a recognized practice in freely re-implementing API libraries by duplicating only declaring code. Oracle vigorously tried to impeach this testimony. Whether or not the practice rose to the level of an entrenched custom under *Wall Data* fell by the wayside. Paragraph 27 of the instructions allowed the jury to consider, in evaluating good faith or not, together with all other circumstances, the extent to which Google's conduct followed or contravened any recognized practice in the industry. The instructions were not adjusted to insert a further reference to custom or *Wall Data* in a second place (presumably in the concluding paragraph on the fair use).[2]

---

[2]Paragraph 27 stated:

> Also relevant to the first statutory factor is the propriety of the accused infringer's conduct because fair use presupposes good faith and fair dealing. Where, for example, the intended purpose is to supplant the copyright holder's commercially valuable right of first publication, good faith is absent. In evaluating the question of the propriety of Google's conduct, meaning good faith or not, you may only consider evidence up to the commencement of this lawsuit on August 12, 2010, and may not consider events thereafter. Your decision as to fair use, however, will govern as to all versions of Android at issue in this case, regardless of their date of issue. Again, in evaluating good faith or not, you should limit your consideration to

Mentioning it twice would have elevated practice and custom to a higher profile than deserved over and above the other fair use factors. Google's point was adequately subsumed under the discussion of propriety of the accused infringer's conduct.

3. Deserving notice before turning to Oracle's main challenges is a tediously undramatic yet highly practical point. Our jury could reasonably have concluded as follows. Sun developed the Java programming language and made it free for all to use without a license. Sun further accumulated the copyrighted Java API library of pre-written code, including its implementing code, to carry out common and more advanced functions and made it available for all to use with a license, although the

---

events before August 12, 2010, and disregard any evidence you have heard after that date. This evidence cut-off date applies only to the issue of good faith or not. In evaluating the extent to which Google acted in good faith or not, you may take into account, together with all other circumstances, the extent to which Google relied upon or contravened any recognized practices in the industry concerning re-implementation of API libraries. You have heard evidence concerning the possibility of Google seeking a license from Oracle. Under the law, if the accused use is otherwise fair, then no permission or license need be sought or granted. Thus, seeking or being denied permission to use a work does not weigh against a finding of fair use. Similarly, you have heard evidence about various licenses from the Apache Foundation, the Apache Harmony Project involving Java, and the General Public License. These are relevant in some ways, but Google concedes it had no license from Sun or Oracle, and it is important to remember that Google makes no claim that its use was pursuant to a license from Sun or Oracle, directly or indirectly. Instead, Google claims that its use was a fair use and therefore required no license at all.

question for our jury was the extent to which, if at all, the declaring code and its structure, sequence, and organization ("SSO") could be carried over into the Android platform without a license under the statutory right of fair use.

The Java API library contains, as stated, pre-written Java source code programs for common and more advanced computer functions. They are organized into "packages," "classes," and "methods." A "package" is a collection of "classes," and in turn, each "class" is a collection of "methods" (and other elements). Each method performs a specific function, sparing a programmer the need to write Java code from scratch to perform that function. Put the other way, various methods are grouped under various classes with the classes grouped under various packages, as in "java.lang.Math" with "java.lang" being the package and "java.lang.Math" being the class. The particular taxonomy adopted for the Java API reflects its unique file system, that is, its SSO.

Significantly, under the rules of the language itself, each method must begin with a "declaration," usually referred to herein as "declaring code." This declares or defines (i) the method name and (ii) the input(s) and their type as expected by the method and the type of any outputs. After the declaration, each method next includes "implementing code," *i.e.*, the pre-written program, which takes the input(s) and, using step-by-step code, carries out the function. The implementing code is set off by special punctuation.

A simple example of a pre-written method is one that finds a square root. At the place in a developer's own program that needs a square root (of say 81), he or she inserts a line (or a "statement") in the specified format

invoking a method pre-written to find square roots. When the computer runs the program and reaches this line, the computer calls upon the pre-written method in its file in the Java API library, provides the method with the input (81), steps through the "implementing code" to the end of the method, and finally "returns" the square root (9) to the program in progress.

The two definitional purposes of "declaring code" are critical. The first declares the precise name of the method (so that the right file will be accessed). The second specifies the input(s) and their type (so that the implementing code will receive the input(s) in the way expected) as well as the type of the output. In our simple square root example, there is only one input, but it must be in parentheses after "sqrt," which is the name of the method. In the Java API library, that particular input is a special kind of floating decimal point number (rather than a whole integer) known as a "double." That part of the method declaration would look like this:

public static double sqrt(double x)

wherein sqrt is the method name, the input is a floating decimal number in the form of a "double" (like 81.0 or 11.56), and the method will return a "double" (like 9.0 or 3.4). (For present purposes, we may ignore the words "public" and "static.")

In writing his or her own Java program, a programmer may only invoke a method with a statement using the precise form defined by the declaring code for the method, both as to name of method and the input format specification. To repeat, the precise name finds the precise file containing the pre-written code for that method. The precise inputs must match the format expected by the

method. In our programmer's own program, the statement calling upon the method might look like the second line just below:

x = 81.0

y = Math.sqrt(x)

wherein the right side of the statement is dictated by the declaring code and the left side y is a variable choice made by the programmer (so that y would be set to the square root, here 9.0).[3]

Regardless of the approach taken by the implementing code to solving the problem addressed by the method (*e.g.*, getting the square root), the input(s) to the method, to repeat, must be of the type as specified in the method declaration, so that the implementing code will receive the inputs in the type expected. Similarly, the method will return an output in the type specified in the method declaration.

Many thousands of pre-written methods have been written for Java, so many that thick books (*see, e.g.*, TX 980) are needed to explain them, organized by packages, classes, and methods. For each method, the book sets forth the precise declaring code but does not (and need not) set forth any implementing code. In other words, the book duplicates all of the method declarations (organized by packages and classes) together with plain English explanations. A Java user can study the book and learn the exact method name and inputs needed to invoke a method for use in his or her own program. The overall set of declarations is called the Java Application Program

[3] "Math.sqrt" corresponds to "Class.method." The package need not be specified in our example.

Interface or Java API. Again, all that the Java programmer need master are the declarations. The implementing code remains a "black box" to the programmer.

In this important sense, the declarations are "interfaces," meaning precise doorways to command access to the pre-written methods and their implementation code performing the actual work of the methods. Java users (and Android users for that matter) must invoke the methods using command statements conforming to the specifications declared by the declarations.

Oracle has portrayed the Java programming language as distinct from the Java API library, insisting that only the language itself was free for all to use. Turns out, however, that in order to write at all in the Java programming language, 62 classes (and some of their methods), spread across three packages *within* the Java API library, *must* be used. Otherwise, the language itself will fail. The 62 "necessary" classes are mixed with "unnecessary" ones in the Java API library and it takes experts to comb them out. As a result, Oracle has now stipulated before the jury that it was fair to use the 62 "necessary" classes given that the Java programming language itself was free and open to use without a license (Tr. 1442–43; TX 9223).[4]

---

[4] Java 2 SE Version 5.0 (one of the copyright works), included 166 API packages. Those packages included over three thousand classes and interfaces, which, in turn, included a total of more than ten thousand methods. Android used the declaring code and SSO of 37 of those API packages including more than six hundred classes (and other elements) which, in turn, included more than six thousand methods. As stated, the implementing code was not copied.

That the 62 "necessary" classes reside without any identification as such within the Java API library (rather than reside within the programming language) supports Google's contention that the Java API library is simply an extension of the programming language itself and helps explain why some view the Java API declarations as free and open for use as the programming language itself. At least to the extent of the 62 "necessary" classes, Oracle agrees.[5]

All this said, our fair use issue, as presented to our jury, came down to whether someone using the Java programming language to build their own library of Java packages was free to duplicate, not just the "necessary" functions in the Java API library but also to duplicate any other functions in it and, in doing so, use the same interfaces, *i.e.*, declaring code, to specify the methods — so long as they supplied their own implementing code.

Oracle's argument in the negative amounts to saying: Yes, all were free to use the Java programming language. Yes, all were free to use the 62 necessary classes from the Java API. Yes, all were free to duplicate the same functionality of any and all methods in the Java API library so long as they "re-implemented" (since copyright does not protect functionality or ideas, only expression). But, Oracle would say, anyone doing so should have scrambled the functionalities among a different taxonomy

---

[5] Trial Exhibit 980, *The Java Application Programming Interface, Volume 1*, is a book that covers four packages and refers to them as the "core packages." According to the back cover of the book, these four packages "are the foundation of the Java language. These libraries include java.lang, java.io, java.util, and java.net. These are the general purpose libraries fundamental to every Java program."

of packages and classes (except as to the 62 "necessary" classes). That is, they should have used a different SSO.

Here, the undramatic yet practical point comes into sharp focus. If, as it was entitled to do, Google had simply reorganized the same functionality of the 37 re-implemented Java packages into a different SSO (taking care, however, not to disturb the 62 necessary classes and their three respective packages), then Java programmers, in order to use the Java system as well as the reorganized Android system, would have had to master and keep straight two different SSO's as they switched between the two systems for different projects. Our jury could reasonably have found that this incompatibility would have fomented confusion and error to the detriment of *both* Java-based systems and to the detriment of Java programmers at large. By analogy, all typewriters use the same QWERTY keyboard — imagine the confusion and universal disservice if every typewriter maker had to scramble the keyboard. Since both systems presupposed the Java programming language in the first place, it was better for both to share the same SSO insofar as they offered the same functionalities, thus maintaining usage consistency across systems and avoiding cross-system confusion, just as all typewriter keyboards should use the QWERTY layout — or so our jury could reasonably have found.

The same could have been reasonably found for the second purpose of the declaring code — specifying the inputs, outputs, and their type. To the extent a specification could be written in more than one way to carry out a given function, it was nevertheless better for all using the Java language to master a single specification rather than having to master, for the same function,

different specifications, one for each system, with the attendant risk of error in switching between systems — or so our jury could reasonably have found.

In terms of the four statutory factors, this consideration bears significantly upon the nature and character of the use (the First Factor), the functional character of the declaring code (the Second Factor), and the limited extent of copying (the Third Factor), that is, Google copied only so much declaring code as was necessary to maintain inter-system consistency among Java users. Google supplied its own code for the rest. Overall, avoiding cross-system babel promoted the progress of science and useful arts — or so our jury could reasonably have found.[6]

This order will now turn to specific arguments raised by Oracle, the losing party, in its challenge to the verdict.

4. With respect to Factor One, Oracle presses hard its view that Google copied in bad faith disregard of Sun/Oracle's property rights. As stated, there remains an ongoing debate regarding whether the "propriety of the use" is a cognizable consideration in any fair use inquiry. Nevertheless, our jury was instructed, as requested by Oracle, to consider whether Google acted in good faith or not as part of its consideration of the first statutory factor. Although mental state is a classic question reserved to the jury, and in our trial mental state was much contested,

---

[6] This point of inter-system consistency, by the way, differs from the interoperability point criticized by the Federal Circuit. 750 F.3d at 1371. The immediate point of cross-system consistency focuses on avoiding confusion in usage between the two systems, both of which are Java-based, not on one program written for one system being operable on the other, the point addressed by the Federal Circuit.

Oracle now insists that our jury could not reasonably have concluded that Google acted in good faith.

Oracle cites numerous examples of internal documents and trial testimony that suggested that Google felt it needed to copy the Java API as an accelerant to bring Android to the market quicker. It points to the breakdown in negotiations between Google and Sun seeking to form a full partnership leaving Google with Java class libraries that were "half-ass at best. [It] need[ed] another half of an ass" (TX 215). In light of that breakdown, Google elected to "[d]o Java anyway and defend [its] decision, perhaps making enemies along the way" (TX 7 at 2). Oracle further notes that even after Sun's CEO at the time publicly praised Android, Andy Rubin (head of Google's Android team) instructed representatives at a trade show, "don't demonstrate [Android] to any [S]un employees or lawyers" (TX 29). Finally, Oracle points to internal communications indicating that Google believed it needed a license to use Java (TX 10; *see also* TX 409 (discussing the possibility of buying Sun to "solve all these lawsuits we're facing")).

On the other hand, Google presented evidence that many at Google (and Sun) understood that at least the declaring code and their SSO were free to use and re-implement, both as a matter of developer practice and because the availability of independent implementations of the Java API enhanced the popularity of the Java programming language, which Sun promoted as free for all to use (Schmidt Testimony, Tr. 361; Page Testimony, Tr. 1846; Rubin Testimony, Tr. 639; Rubin Testimony, Tr. 1088–89).

Sun's own CEO at the time, Jonathan Schwartz, testified on Google's behalf at trial and supported Google's

view that a practice of duplicating declarations existed and that the competition was on implementations. Oracle's harsh cross-examination focused on character assassination and showing that Schwartz resented Oracle for its treatment of Schwartz after the buyout. That Oracle resorted to such impeachment underscores how fact-bound the issue was, another classic role of a jury to resolve.

In light of the foregoing, our jury could reasonably have concluded that Google's use of parts of the Java API as an accelerant was undertaken based on a good faith belief that at least the declaring code and SSO were free to use (which it did use), while a license was necessary for the implementing code (which it did not use). Our jury could reasonably have concluded that Google's concern about making an enemy of Sun reflected concern about the parties' business relationship in light of the failed negotiations that would have brought Sun in as a major partner in Android, rather than concerns about litigation. Mental state was and remains a classic province of the jury.

5. With respect to the Factor One and commercialism, it is undisputed that Google's use of the declaring code and SSO from 37 Java API packages served commercial purposes and our jury was so instructed, including an instruction that a commercial use weighed against fair use. Nevertheless, our jury could reasonably have found that Google's decision to make Android available open source and free for all to use had non-commercial purposes as well (such as the general interest in sharing software innovation). Indeed, Sun itself acknowledged (before Android launched) that making OpenJDK available as open source, as Sun did, could undermine its

own commercial efforts with Java SE licensing (TX 971 at 14). Thus, even though Google's use was commercial, which weighed against fair use, the jury could reasonably have found the open-source character of Android tempered Google's overall commercial goals.

Of course, even a *wholly* commercial use may still constitute a fair use. *Campbell*, 510 U.S. at 585. Thus, in the alternative, our jury could reasonably have found that Google's use of the declaring code and SSO from 37 Java API packages constituted a fair use despite even a heavily commercial character of that use.

It is true that in the first appeal, the following exchange occurred at oral argument between Circuit Judge Kathleen O'Malley and counsel for Google:

> *Judge O'Malley:* But for purpose and character, though, you don't dispute that it was entirely a commercial purpose.
>
> *Van Nest:* No.

Oral Arg., *Oracle Am., Inc. v. Google Inc.*, Nos. 2013-1021, 2013-1022 (Fed. Circ.) 1:02:54–1:03:00.

On remand, Oracle sought to convert this colloquy to a judicial admission that Google's use was "entirely commercial." It is for the district court, in its discretion, to determine the extent, if any, of a judicial admission. *American Title Ins. Co. v. Lacelaw Corp.*, 861 F.2d 224, 226 (9th Cir. 1988). As set forth in the final pretrial order (Dkt. No. 1760), the undersigned examined the colloquy (and all other statements of record on the point) and determined that the "commercial" part would be treated as a judicial admission, but the "entirely" part would not be. The word "entirely" was part of the give and take of an

oral argument. In light of all statements by counsel and in light of the free and open availability of Android, the word "entirely" would have been too conclusive, inaccurate, and unfair. The district court exercised its discretion to limit the admission to "commercial" and let the jury decide for itself how commercial, according to the evidence.

Accordingly, our jury was instructed that Google's use was commercial, but that it was up to the jury to determine the extent of the commerciality, as follows (Dkt. No. 1981 ¶ 21) (emphasis added):

In evaluating the first statutory factor, the extent of the commercial nature of the accused use must be considered. *In this case, all agree that Google's accused use was commercial in nature but disagree over the extent*. Commercial use weighs against a finding of fair use, but even a commercial use may be found (or not found, as the case may be) to be sufficiently transformative that the first statutory factor, on balance, still cuts in favor of fair use. To put it differently, the more transformative an accused work, the more other factors, such as commercialism, will recede in importance. By contrast, the less transformative the accused work, the more other factors like commercialism will dominate.

Our jury could reasonably have agreed with Oracle that the evidence showed the use was entirely commercial (yet still ruled for Google), but it could also have reasonably found that the use, while commercial, served non-commercial purposes as well, *i.e.*, as part of a free and open software platform, namely Android.[7]

---

[7] Although Google gives Android away for free, Oracle argues that the "Android Ecosystem" has generated over forty billion dollars in revenue and thus Android has had a massive commercial benefit to

6. With respect to the Factor One and "transformativeness," a use is transformative if it "adds something new, with a further purpose or different character, altering the first with new expression, meaning, or message." *Campbell v. Acuff-Rose Music, Inc.*, 510 U.S. 569, 579 (1994). Oracle argues that no jury could reasonably find that Google's use of the declaring code and SSO from 37 Java API packages in Android imbued the copyrighted works with new expression, meaning, or message. Specifically, Oracle argues that the copied code served the same function in Android as it did in Java, inasmuch as the code served as an interface for accessing methods in both systems (*see* Astrachan Testimony, Tr. 1265; Bloch Testimony, Tr. 997).

It should go without saying (but it must be said anyway) that, of course, the words copied will always be the same (or virtually so) in a copyright case — otherwise there can be no copyright problem in the first place. And, of course, the copied declarations serve the same function in both works, for by definition, declaring code in the Java programming language serves the specific definitional purposes explained above. If this were enough to defeat fair use, it would be impossible ever to duplicate declaring code as fair use and presumably the Federal Circuit would have disallowed this factor on the first appeal rather than remanding for a jury trial.

Google. There is no doubt that Android has contributed to a large expansion of smartphones but the revenue benefit to Google flows from the ad revenue generated by its search engine which pre existed Android. In other words, our jury could reasonably have found that without Android the void would have been filled by other mobile platforms, yet those platforms would still have led to more Google search requests and ad revenue.

With respect to transformativeness, our jury could reasonably have found that (i) Google's selection of 37 out of 166 Java API packages (ii) re-implemented with new implementing code adapted to the constrained operating environment of mobile smartphone devices with small batteries, and (iii) combined with brand new methods, classes, and packages written by Google for the mobile smartphone platform — all constituted a fresh context giving new expression, meaning, or message to the duplicated code.[8] (The copyrighted works were designed and used for desktop and laptop computers.)

In *Campbell*, the accused work (a rap parody song) used the same bass riff and an identical first line of Roy Orbison's "Oh, Pretty Woman." The parody also included exact copies of certain phrases in subsequent lines and maintained the same structure and rhyme scheme throughout. The copied elements served the same function in the accused work as in the original. Nevertheless, the Supreme Court acknowledged that the transformative purpose of parody had a "need to mimic an original to make its point," and thus, warranted copying some exact elements. *Id.* at 580–81. The question of the *extent* of the copying permissible to serve that function was the subject of the inquiry of the third statutory fair use factor. So too here.

---

[8] As stated, the Android core libraries included over one hundred new API packages that had never been part of the Java API. Those packages enabled functionality specifically intended for use in a mobile smartphone environment, and like the 37 Java API packages at issue here, they were written in the Java programming language (Rubin Testimony, Tr. 670). Some additional functionality in Android, however, was performed by a separate set of libraries written in C or C++ for performance purposes (Douglas Schmidt Testimony, Tr. 1602).

Android did not merely incorporate the copyrighted work "as part of a broader work," without any change to the purpose, message, or meaning of the underlying work (*see* Dkt. No. 1780). Android did not merely adopt the Java platform wholesale as part of a broader software platform without any changes. Instead, it integrated selected elements, namely declarations from 37 packages to interface with all new implementing code optimized for mobile smartphones and added entirely new Java packages written by Google itself. This enabled a purpose distinct from the desktop purpose of the copyrighted works — or so our jury could reasonably have found.

In light of the foregoing, our jury could reasonably have concluded that Google's use of the declaring code and SSO of 37 API packages from the desktop platform work in a full-stack, open-source mobile operating system for smartphones was transformative.[9]

7. With respect to Factor Two, the "nature of the copyrighted work," the final charge to the jury stated "[t]his factor recognizes that traditional literary works are closer than informational works, such as instruction

[9] The instructions on "transformativeness" deleted a point from the Federal Circuit opinion that might have favored Google, which had requested an instruction defining "transformative" as the incorporation of copyrighted material "as part of a broader work," relying on a parenthetical snippet in the Federal Circuit opinion. This Court denied Google's request and explained why (Dkt. 1780). In brief, the parenthetical snippet was taken from our court of appeal's decision in *Monge v. Maya Magazines*, 688 F.3d 1164, 1176 (9th Cir. 2012). But as our court of appeals there explained, the incorporation of a copyrighted material into a larger work, such as the arrangement of a work in a photo montage, *could be* transformative and fair use, not that it *must be*. Please see the order at Docket Number 1780 for the reasoning.

manuals, to the core of intended copyright protection. Creative writing and expression lie at the very heart of copyright protection, so fair use is generally more difficult to establish for copying of traditional literary works than for copying of informational works" (Dkt. No. 1981 ¶ 28); *see also Campbell*, 510 U.S. at 586.[10]

The Java programming language itself requires the package-class-method hierarchy, an idea on which Oracle does not claim any copyright. Oracle instead argues that because there were countless ways to name and organize the packages in Java and because Google could have used a completely new taxonomy in Android (except as to the 62 "necessary" classes), our jury should have concluded that the process of designing APIs must have been "highly creative" and thus at the core of copyright's protection. Of course, such a conclusion would have been within the evidence, but our jury could reasonably have gone the other way and concluded that the declaring code was not highly creative.

Oracle highlights Google's own witness, Joshua Bloch, who designed many of the Java APIs while working at Sun and who later worked at Google on the Android team. Bloch testified that one of the challenges he faced in designing API was "the complexity of figuring out how best to express what it is that the programmer wants done" (Tr. 1007). Oracle focuses on Bloch's use of the word "express" to demonstrate the expressive nature of API

---

[10] "[I]f a work is largely functional, it receives only weak protection. 'This result is neither unfair nor unfortunate. It is the means by which copyright advances the progress of science and art.'" *Sega Enterprises, Ltd. v. Accolade, Inc.*, 977 F.2d 1510, 1527 (9th Cir. 1992) (quoting *Feist Publications, Inc. v. Rural Tel. Serv. Co., Inc.*, 499 U.S. 340, 350 (1991)).

design but it ignores the fact that he addressed the challenge of expressing a particular *function*. Similarly, Oracle notes that Bloch described API design as "an art not a science" and cites his eloquence regarding "design principles" (Tr. 971).

In citing this, Oracle resorts to the time-honored tactic of emphasizing a concession by one of the other side's witness. But other witnesses (*e.g.*, Dr. Owen Astrachan, among others) emphasized the functional role of the declaring lines of code and their SSO and minimized the "creative" aspect. Our jury could reasonably have found that, while the declaring code and SSO were creative enough to qualify for copyright protection, functional considerations predominated in their design, and thus Factor Two was not a strong factor in favor of Oracle after all.

8. With respect to Factor Three, our jury could reasonably have found that Google duplicated the bare minimum of the 37 API packages, just enough to preserve inter-system consistency in usage, namely the declarations and their SSO only, and did not copy any of the implementing code, thus finding that Google copied only so much as was reasonably necessary for a transformative use. The number of lines of code duplicated constituted a tiny fraction of one percent of the copyrighted works (and even less of Android, for that matter).

9. With respect to Factor Four, our jury could reasonably have found that use of the declaring lines of code (including their SSO) in Android caused no harm to the market for the copyrighted works, which were for desktop and laptop computers. As to Java ME, our jury could reasonably have found that Java ME eventually

declined in revenue just as predicted by Sun before Android was even released, meaning that Android had no further negative impact on Java ME beyond the tailspin already predicted within Sun.

Also, before Android was released, Sun made all of the Java API available as free and open source under the name OpenJDK, subject only to the lax terms of the General Public License Version 2 with Classpath Exception. This invited anyone to subset the API. Anyone could have duplicated, for commercial purposes, the very same 37 packages as wound up in Android with the very same SSO and done so without any fee, subject only to lenient "give-back" conditions of the GPLv2+CE. Although Google didn't acquire the 37 packages via OpenJDK, our jury could reasonably have found that Android's impact on the market for the copyrighted works paralleled what Sun already expected via its OpenJDK.

10. Stepping back, it seems hard to reconcile Oracle's current position with the one it took just as the trial was getting underway, namely, that fair use is an equitable rule of reason and each case requires its own balancing of factors. In its critique of the first proposed jury instructions on fair use (Dkt. No. 1663 at 1), Oracle argued that the Court's draft characterization of the policy of fair use contravened the legislative history, and Oracle cited the following language from a Senate report on the 1976 Copyright Act (which language was repeated in the House Report):

Although the courts have considered and ruled upon the fair use doctrine over and over again, no real definition of the concept has ever emerged. Indeed, since the doctrine is an equitable rule of reason, no generally

applicable definition is possible, and each case raising the question must be decided on its own facts.

S.Rep. No. 94-473 at 62 (1975). The Court adopted Oracle's proposed instruction in the next draft as well as in the final charge to the jury, stating: "Since the doctrine of fair use is an equitable rule of reason, no generally accepted definition is possible, and each case raising the question must be decided on its own facts" (Dkt. No. 1981 ¶ 21).

Now, Oracle argues instead that this case must be decided as a matter of law, and not "on its own facts." Oracle argues that Google's copying fails to resemble any of the statutory examples of fair use listed in the precatory language of Section 107, again contradicting its earlier position that "no generally applicable definition is possible."

In applying an "equitable rule of reason," our jury could reasonably have given weight to the fact that cross-system confusion would have resulted had Google scrambled the SSO and specifications. Java programmers and science and the useful arts were better served by a common set of command-type statements, just as all typists are better served by a common QWERTY keyboard.

11. In summary, on Factor One, our jury could reasonably have found that while the use was commercial, the commercial use was outweighed by a transformative use, namely use of the declaring code as one component in a full stack platform for highly advanced smartphones, a different context in which (i) 37 of the 166 API packages were selected, (ii) all of the implementing code was re-implemented for a mobile low-power platform, and (iii)

many new packages original with Android were added. Despite Google's internal e-mails, our jury could reasonably have found that most of them pertained to earlier negotiations for a joint venture to use the *entire* Java system, including the implementing code, and that, after those discussions failed, Google acted in good faith by duplicating only the declarations to 37 packages to maintain inter-system consistency in usage and by supplying its own implementing code. On Factor Two, our jury could reasonably have found that the code copied was not highly creative, was mainly functional, and was less deserving of protection. On Factor Three, our jury could reasonably have found that Google duplicated only the declaring code, a tiny fraction of the copyrighted works, duplicated to avoid confusion among Java programmers as between the Java system and the Android system. On Factor Four, our jury could have found that Android caused no harm to the desktop market for the copyrighted works or to any mobile derivative, as borne out by Sun's own records. Of course, Oracle had arguments going the other way, but the jury was reasonably within the record in finding fair use.

This order cannot cover all the myriad ways that the jury could reasonably have balanced the statutory factors and found in favor of fair use. The possibilities above represent but one take on the evidence. Witness credibility was much challenged. Plainly, many more variations and balancings could have reasonably led to the same verdict.

12. A final word about a separate issue that arose during trial. In their joint final pretrial submission, both sides agreed that no reference would be made before the jury to the prior proceedings in this case (Dkt. No. 1709 at

8). As this trial developed, however, Oracle left the impression before the jury that all the way up to the present, Google had uniformly acted in bad faith. Problem was, during a substantial part of this period (2012–2014), Google had been entitled to rely on the judgment of the district court that the material asserted was not copyrightable. *Kamar Int'l, Inc. v. Russ Berrie & Co.*, 752 F.2d 1326, 1330 (column two) (9th Cir. 1984) stated (emphasis added):

> We affirm the district court's holding that the sales by Russ Berrie of its stuffed animals immediately following the first judgment do not count as infringements after notice. *Kamar's* supposed citation to the contrary . . . is wholly inapposite. In its first judgment, the district court held Russ Berrie's animals noninfringing. Kamar did not obtain any stay pending appeal. *Russ was entitled to rely on the judgment at that time.*

In response, Oracle contended that Judge Alex Kozinski's opinion for our court of appeals in *Micro Star v. Formgen, Inc.*, 154 F.3d 1107 (9th Cir. 1998), had been so at odds with the decision by this Court holding that the declaring code and their structure, sequence and organization were not copyrightable that Google could not reasonably have believed that this Court's holding on uncopyrightability was correct (Trial Tr. at 1591). The short answer was that *Micro Star* provided no holding or dictum whatsoever on copyrightability — none. Copyrightability was not there raised. (It was a fair use case.) Indeed, in our earlier trial whencopyrightability was debated, no one, including Oracle, ever cited *Micro Star* on copyrightability. Nor was it raised on appeal.

To resolve this problem of the 2012-2014 interregnum period as best as could be done with minimal strain on the parties' stipulation, the Court gave the following instruction:

In evaluating the question of the propriety of Google's conduct, meaning good faith or not, you may only consider evidence up to the commencement of this lawsuit on August 12, 2010, and may not consider events thereafter. Your decision as to fair use, however, will govern as to all versions of Android at issue in this case, regardless of their date of issue. Again, in evaluating good faith or not, you should limit your consideration to events before August 12, 2010, and disregard any evidence you have heard after that date. This evidence cut-off date applies only to the issue of good faith or not.

No mention was made to the jury about the earlier judgment rejecting copyrightability. The problem was largely solved by the date cut-off, which allowed Oracle to use all of Google's "bad" e-mails. To mitigate the problem of speculation regarding prior testimony read in at the second trial, the following instruction was given:

You may have heard from a witness that there was a prior trial in this case. It is true that there was a prior trial. We have heard evidence in this trial of a prior proceeding, which is the earlier trial that occurred in this case. Do not speculate about what happened in the prior trial. No determination on fair use was made one way or the other in that trial. It is up to you, the jury, to determine fair use based on the evidence you have heard in this trial and my instructions of the law.

Unfortunately, this might not have eliminated all of the prejudice to Google from the suggestion made before the

jury by Oracle, but it went most of the way and was the best the Court could do in light of the stipulation made by the parties at the outset.

* * *

All Rule 50 motions are **DENIED**. Judgment will be entered in accordance with the jury's verdict. **IT IS SO ORDERED.**

Dated: June 8, 2016.

/s/

WILLIAM ALSUP, UNITED STATES DISTRICT JUDGE

Appendix D

United States Court of Appeals
for the Federal Circuit

ORACLE AMERICA, INC.,

Plaintiff-Appellant,

v.

GOOGLE INC.,

Defendant-Cross-Appellant.

2013-1021, -1022

Appeals from the United States District Court for the Northern District of California in No. 10-CV-3561, Judge William H. Alsup.

Decided: May 9, 2014

---

* * *

Before O'MALLEY, PLAGER, and TARANTO, *Circuit Judges*

O'MALLEY, *Circuit Judge*.

This copyright dispute involves 37 packages of computer source code. The parties have often referred to these groups of computer programs, individually or collectively, as "application programming interfaces," or

API packages, but it is their content, not their name, that matters. The predecessor of Oracle America, Inc. ("Oracle") wrote these and other API packages in the Java programming language, and Oracle licenses them on various terms for others to use. Many software developers use the Java language, as well as Oracle's API packages, to write applications (commonly referred to as "apps") for desktop and laptop computers, tablets, smartphones, and other devices.

Oracle filed suit against Google Inc. ("Google") in the United States District Court for the Northern District of California, alleging that Google's Android mobile operating system infringed Oracle's patents and copyrights. The jury found no patent infringement, and the patent claims are not at issue in this appeal. As to the copyright claims, the parties agreed that the jury would decide infringement, fair use, and whether any copying was de minimis, while the district judge would decide copyrightability and Google's equitable defenses. The jury found that Google infringed Oracle's copyrights in the 37 Java packages and a specific computer routine called "rangeCheck," but returned a noninfringement verdict as to eight decompiled security files. The jury deadlocked on Google's fair use defense.

After the jury verdict, the district court denied Oracle's motion for judgment as a matter of law ("JMOL") regarding fair use as well as Google's motion for JMOL with respect to the rangeCheck files. Order on Motions for Judgment as a Matter of Law, *Oracle Am., Inc. v. Google Inc.*, No. 3:10-cv-3561 (N.D. Cal. May 10, 2012), ECF No. 1119. Oracle also moved for JMOL of infringement with respect to the eight decompiled security files. In granting that motion, the court found that: (1) Google admitted to

copying the eight files; and (2) no reasonable jury could find that the copying was de minimis. *Oracle Am., Inc. v. Google Inc.*, No. C 10-3561, 2012 U.S. Dist. LEXIS 66417 (N.D. Cal. May 11, 2012) ("*Order Granting JMOL on Decompiled Files*").

Shortly thereafter, the district court issued its decision on copyrightability, finding that the replicated elements of the 37 API packages—including the declaring code and the structure, sequence, and organization—were not subject to copyright protection. *Oracle Am., Inc. v. Google Inc.*, 872 F. Supp. 2d 974 (N.D. Cal. 2012) ("*Copyrightability Decision*"). Accordingly, the district court entered final judgment in favor of Google on Oracle's copyright infringement claims, except with respect to the rangeCheck code and the eight decompiled files. Final Judgment, *Oracle Am., Inc. v. Google Inc.*, No. 3:10-cv-3561 (N.D. Cal. June 20, 2012), ECF No. 1211. Oracle appeals from the portion of the final judgment entered against it, and Google cross-appeals from the portion of that same judgment entered in favor of Oracle as to the rangeCheck code and eight decompiled files.

Because we conclude that the declaring code and the structure, sequence, and organization of the API packages are entitled to copyright protection, we reverse the district court's copyrightability determination with instructions to reinstate the jury's infringement finding as to the 37 Java packages. Because the jury deadlocked on fair use, we remand for further consideration of Google's fair use defense in light of this decision. With respect to Google's cross-appeal, we affirm the district court's decisions: (1) granting Oracle's motion for JMOL as to the eight decompiled Java files that Google copied into Android; and (2) denying Google's motion for JMOL with

respect to the rangeCheck function. Accordingly, we affirm-in-part, reverse-in-part, and remand for further proceedings.

BACKGROUND

A. The Technology

Sun Microsystems, Inc. ("Sun") developed the Java "platform" for computer programming and released it in 1996.[1] The aim was to relieve programmers from the burden of writing different versions of their computer programs for different operating systems or devices. "The Java platform, through the use of a virtual machine, enable[d] software developers to write programs that [we]re able to run on different types of computer hardware without having to rewrite them for each different type." *Copyrightability Decision*, 872 F. Supp. 2d at 977. With Java, a software programmer could "write once, run anywhere."

The Java virtual machine ("JVM") plays a central role in the overall Java platform. The Java programming language itself—which includes words, symbols, and other units, together with syntax rules for using them to create instructions—is the language in which a Java programmer writes source code, the version of a program that is "in a human-readable language." *Id.* For the instructions to be executed, they must be converted (or compiled) into binary machine code (object code) consisting of 0s and 1s understandable by the particular computing device. In the Java system, "source code is first converted into 'bytecode,' an intermediate form, before it is then converted into binary machine code by the Java virtual

[1] Oracle acquired Sun in 2010.

machine" that has been designed for that device. *Id.* The Java platform includes the "Java development kit (JDK), javac compiler, tools and utilities, runtime programs, class libraries (API packages), and the Java virtual machine." *Id.* at 977 n.2.

Sun wrote a number of ready-to-use Java programs to perform common computer functions and organized those programs into groups it called "packages." These packages, which are the application programming interfaces at issue in this appeal, allow programmers to use the prewritten code to build certain functions into their own programs, rather than write their own code to perform those functions from scratch. They are shortcuts. Sun called the code for a specific operation (function) a "method." It defined "classes" so that each class consists of specified methods plus variables and other elements on which the methods operate. To organize the classes for users, then, it grouped classes (along with certain related "interfaces") into "packages." *See id.* at 982 (describing organization: "[e]ach package [i]s broken into classes and those in turn [are] broken into methods"). The parties have not disputed the district court's analogy: Oracle's collection of API packages is like a library, each package is like a bookshelf in the library, each class is like a book on the shelf, and each method is like a how-to chapter in a book. *Id.* at 977.

The original Java Standard Edition Platform ("Java SE") included "eight packages of pre-written programs." *Id.* at 982. The district court found, and Oracle concedes to some extent, that three of those packages—java.lang, java.io, and java.util—were "core" packages, meaning that programmers using the Java language had to use them "in order to make any worthwhile use of the language." *Id.* By

2008, the Java platform had more than 6,000 methods making up more than 600 classes grouped into 166 API packages. There are 37 Java API packages at issue in this appeal, three of which are the core packages identified by the district court.[2] These packages contain thousands of individual elements, including classes, subclasses, methods, and interfaces.

Every package consists of two types of source code—what the parties call (1) declaring code; and (2) implementing code. Declaring code is the expression that identifies the prewritten function and is sometimes referred to as the "declaration" or "header." As the district court explained, the "main point is that this header line of code introduces the method body and specifies very precisely the inputs, name and other functionality." *Id.* at 979–80. The expressions used by the programmer from the declaring code command the computer to execute the associated implementing code, which gives the computer the step-by-step instructions for carrying out the declared function.

To use the district court's example, one of the Java API packages at issue is "java.lang." Within that package is a class called "math," and within "math" there are several

---

[2] The 37 API packages involved in this appeal are: java.awt.font, java.beans, java.io, java.lang, ja va.lang.annotation, java.lang.ref, java.lang.reflect, java. net, java.nio, java.nio.channels, java.nio.channels.spi, java.nio.charset, java.nio.charset.spi, java.security, java. security.acl, java.security.cert, java.security.interfaces, java.security.spec, java.sql, java.text, java.util, java. util.jar, java.util.logging, java.util.prefs, java. util.regex, java.util.zip, javax.crypto, javax.crypto.interfaces, javax.crypto.spec, javax.net, javax.net.ssl, javax.security.auth, javax. security.auth.callback, javax.security.auth.login, javax.security.auth.x500, javax.security.cert, and javax. sql.

methods, including one that is designed to find the larger of two numbers: "max." The declaration for the "max" method, as defined for integers, is: "public static int max(int x, int y)," where the word "public" means that the method is generally accessible, "static" means that no specific instance of the class is needed to call the method, the first "int" indicates that the method returns an integer, and "int x" and "int y" are the two numbers (inputs) being compared. *Copyrightability Decision*, 872 F. Supp. 2d at 980–82. A programmer calls the "max" method by typing the name of the method stated in the declaring code and providing unique inputs for the variables "x" and "y." The expressions used command the computer to execute the implementing code that carries out the operation of returning the larger number.

Although Oracle owns the copyright on Java SE and the API packages, it offers three different licenses to those who want to make use of them. The first is the General Public License, which is free of charge and provides that the licensee can use the packages—both the declaring and implementing code—but must "contribute back" its innovations to the public. This arrangement is referred to as an "open source" license. The second option is the Specification License, which provides that the licensee can use the declaring code and organization of Oracle's API packages but must write its own implementing code. The third option is the Commercial License, which is for businesses that "want to use and customize the full Java code in their commercial products and keep their code secret." Appellant Br. 14. Oracle offers the Commercial License in exchange for royalties. To maintain Java's "write once, run anywhere" motto, the Specification and Commercial Licenses require that the

licensees' programs pass certain tests to ensure compatibility with the Java platform.

The testimony at trial also revealed that Sun was licensing a derivative version of the Java platform for use on mobile devices: the Java Micro Edition ("Java ME"). Oracle licensed Java ME for use on feature phones and smartphones. Sun/Oracle has never successfully developed its own smartphone platform using Java.

B. Google's Accused Product: Android

The accused product is Android, a software platform that was designed for mobile devices and competes with Java in that market. Google acquired Android, Inc. in 2005 as part of a plan to develop a smartphone platform. Later that same year, Google and Sun began discussing the possibility of Google "taking a license to use and to adapt the entire Java platform for mobile devices." *Copyrightability Decision*, 872 F. Supp. 2d at 978. They also discussed a "possible co-development partnership deal with Sun under which Java technology would become an open-source part of the Android platform, adapted for mobile devices." *Id.* The parties negotiated for months but were unable to reach an agreement. The point of contention between the parties was Google's refusal to make the implementation of its programs compatible with the Java virtual machine or interoperable with other Java programs. Because Sun/Oracle found that position to be anathema to the "write once, run anywhere" philosophy, it did not grant Google a license to use the Java API packages.

When the parties' negotiations reached an impasse, Google decided to use the Java programming language to design its own virtual machine—the Dalvik virtual

machine ("Dalvik VM")—and "to write its own implementations for the functions in the Java API that were key to mobile devices." *Id.* Google developed the Android platform, which grew to include 168 API packages—37 of which correspond to the Java API packages at issue in this appeal.

With respect to the 37 packages at issue, "Google believed Java application programmers would want to find the same 37 sets of functionalities in the new Android system callable by the same names as used in Java." *Id.* To achieve this result, Google copied the declaring source code from the 37 Java API packages verbatim, inserting that code into parts of its Android software. In doing so, Google copied the elaborately organized taxonomy of all the names of methods, classes, interfaces, and packages—the "overall system of organized names—covering 37 packages, with over six hundred classes, with over six thousand methods." *Copyrightability Decision*, 872 F. Supp. 2d at 999. The parties and district court referred to this taxonomy of expressions as the "structure, sequence, and organization" or "SSO" of the 37 packages. It is undisputed, however, that Google wrote its own implementing code, except with respect to: (1) the rangeCheck function, which consisted of nine lines of code; and (2) eight decompiled security files.

As to rangeCheck, the court found that the Sun engineer who wrote it later worked for Google and contributed two files he created containing the rangeCheck function—"Timsort.java" and "ComparableTimsort"—to the Android platform. In doing so, the nine-line rangeCheck function was copied directly into Android. As to the eight decompiled files, the district court found that they were copied and used as test files

but "never found their way into Android or any handset." *Id.* at 983.

Google released the Android platform in 2007, and the first Android phones went on sale the following year. Although it is undisputed that certain Android software contains copies of the 37 API packages' declaring code at issue, neither the district court nor the parties specify in which programs those copies appear. Oracle indicated at oral argument, however, that all Android phones contain copies of the accused portions of the Android software. Oral Argument at 1:35, *available at* http://www.cafc.uscourts.gov/oral-argument-recordings/2013-1021/all. Android smartphones "rapidly grew in popularity and now comprise a large share of the United States market." *Copyrightability Decision*, 872 F. Supp. 2d at 978. Google provides the Android platform free of charge to smartphone manufacturers and receives revenue when customers use particular functions on the Android phone. Although Android uses the Java programming language, it is undisputed that Android is not generally Java compatible. As Oracle explains, "Google ultimately designed Android to be *incompatible* with the Java platform, so that apps written for one will not work on the other." Appellant Br. 29.

C. Trial and Post-Trial Rulings

Beginning on April 16, 2012, the district court and the jury—on parallel tracks—viewed documents and heard testimony from twenty-four witnesses on copyrightability, infringement, fair use, and Google's other defenses. Because the parties agreed the district court would decide copyrightability, the court instructed the jury to assume that the structure, sequence, and organization of the 37 API packages was copyrightable. And, the court informed

the jury that Google conceded that it copied the declaring code used in the 37 packages verbatim. The court also instructed the jury that Google conceded copying the rangeCheck function and the eight decompiled security files, but that Google maintained that its use of those lines of code was de minimis. *See* Final Charge to the Jury (Phase One), *Oracle Am., Inc. v. Google Inc.*, 3:10-cv-3561 (N.D. Cal. Apr. 30, 2012), ECF No. 1018 at 14 ("With respect to the infringement issues concerning the rangeCheck and other similar files, Google agrees that the accused lines of code and comments came from the copyrighted material but contends that the amounts involved were so negligible as to be de minimis and thus should be excused.").

On May 7, 2012, the jury returned a verdict finding that Google infringed Oracle's copyright in the 37 Java API packages and in the nine lines of rangeCheck code, but returned a noninfringement verdict as to eight decompiled security files. The jury hung on Google's fair use defense.

The parties filed a number of post-trial motions, most of which were ultimately denied. In relevant part, the district court denied Oracle's motion for JMOL regarding fair use and Google's motion for JMOL as to the rangeCheck files. Order on Motions for Judgment as a Matter of Law, *Oracle Am., Inc. v. Google Inc.*, No. 3:10-cv-3561 (N.D. Cal. May 10, 2012), ECF No. 1119. The district court granted Oracle's motion for JMOL of infringement as to the eight decompiled files, however. In its order, the court explained that: (1) Google copied the files in their entirety; (2) the trial testimony revealed that the use of those files was "significant"; and (3) no reasonable jury could find the copying de minimis. *Order*

Granting JMOL on Decompiled Files, 2012 U.S. Dist. LEXIS 66417, at *6.

On May 31, 2012, the district court issued the primary decision at issue in this appeal, finding that the replicated elements of the Java API packages—including the declarations and their structure, sequence, and organization—were not copyrightable. As to the declaring code, the court concluded that "there is only one way to write" it, and thus the "merger doctrine bars anyone from claiming exclusive copyright ownership of that expression." *Copyrightability Decision*, 872 F. Supp. 2d at 998. The court further found that the declaring code was not protectable because "names and short phrases cannot be copyrighted." *Id.* As such, the court determined that "there can be no copyright violation in using the identical declarations." *Id.*

As to the overall structure, sequence, and organization of the Java API packages, the court recognized that "nothing in the rules of the Java language . . . required that Google replicate the same groupings even if Google was free to replicate the same functionality." *Id.* at 999. Therefore, the court determined that "Oracle's best argument . . . is that while no single name is copyrightable, Java's overall system of organized names—covering 37 packages, with over six hundred classes, with over six thousand methods—is a 'taxonomy' and, therefore, copyrightable." *Id.*

Although it acknowledged that the overall structure of Oracle's API packages is creative, original, and "resembles a taxonomy," the district court found that it "is nevertheless a command structure, a system or method of operation—a long hierarchy of over six thousand commands to carry out pre-assigned functions"—that is

not entitled to copyright protection under Section 102(b) of the Copyright Act. *Id.* at 999–1000. In reaching this conclusion, the court emphasized that, "[o]f the 166 Java packages, 129 were not violated in any way." *Id.* at 1001. And, of the 37 Java API packages at issue, "97 percent of the Android lines were new from Google and the remaining three percent were freely replicable under the merger and names doctrines." *Id.* On these grounds, the court dismissed Oracle's copyright claims, concluding that "the particular elements replicated by Google were free for all to use under the Copyright Act." *Id.*

On June 20, 2012, the district court entered final judgment in favor of Google and against Oracle on its claim for copyright infringement, except with respect to the rangeCheck function and the eight decompiled files. As to rangeCheck and the decompiled files, the court entered judgment for Oracle and against Google in the amount of zero dollars, per the parties' stipulation. Final Judgment, *Oracle Am., Inc. v. Google Inc.*, No. 3:10-cv-3561 (N.D. Cal. June 20, 2012), ECF No. 1211. Oracle timely appealed from the portion of the district court's final judgment entered against it and Google timely crossappealed with respect to rangeCheck and the eight decompiled files. Because this action included patent claims, we have jurisdiction pursuant to 28 U.S.C. § 1295(a)(1).

DISCUSSION

I. ORACLE'S APPEAL

It is undisputed that the Java programming language is open and free for anyone to use. Except to the limited extent noted below regarding three of the API packages, it is also undisputed that Google could have written its own

API packages using the Java language. Google chose not to do that. Instead, it is undisputed that Google copied 7,000 lines of declaring code and generally replicated the overall structure, sequence, and organization of Oracle's 37 Java API packages. The central question before us is whether these elements of the Java platform are entitled to copyright protection. The district court concluded that they are not, and Oracle challenges that determination on appeal. Oracle also argues that the district court should have dismissed Google's fair use defense as a matter of law.

According to Google, however, the district court correctly determined that: (1) there was only one way to write the Java method declarations and remain "interoperable" with Java; and (2) the organization and structure of the 37 Java API packages is a "command structure" excluded from copyright protection under Section 102(b). Google also argues that, if we reverse the district court's copyrightability determination, we should direct the district court to retry its fair use defense.

"When the questions on appeal involve law and precedent on subjects not exclusively assigned to the Federal Circuit, the court applies the law which would be applied by the regional circuit." *Atari Games Corp. v. Nintendo of Am., Inc.*, 897 F.2d 1572, 1575 (Fed. Cir. 1990). Copyright issues are not exclusively assigned to the Federal Circuit. *See* 28 U.S.C. § 1295. The parties agree that Ninth Circuit law applies and that, in the Ninth Circuit, whether particular expression is protected by

copyright law is "subject to de novo review." *Ets-Hokin v. Skyy Spirits, Inc.*, 225 F.3d 1068, 1073 (9th Cir. 2000).[3]

We are mindful that the application of copyright law in the computer context is often a difficult task. *See Lotus Dev. Corp. v. Borland Int'l, Inc.*, 49 F.3d 807, 820 (1st Cir. 1995) (Boudin, J., concurring) ("Applying copyright law to computer programs is like assembling a jigsaw puzzle whose pieces do not quite fit."). On this record, however, we find that the district court failed to distinguish between the threshold question of what is copyrightable—which

[3] The Supreme Court has not addressed whether copyrightability is a pure question of law or a mixed question of law and fact, or whether, if it is a mixed question of law and fact, the factual components of that inquiry are for the court, rather than the jury. Relatedly, it has not decided the standard of review that applies on appeal. Ten years ago, before finding it unnecessary to decide whether copyrightability is a pure question of law or a mixed question of law and fact, the Seventh Circuit noted that it had "found only a handful of appellate cases addressing the issue, and they are split." *Gaiman v. McFarlane*, 360 F.3d 644, 648 (7th Cir. 2004). And, panels of the Ninth Circuit have defined the respective roles of the jury and the court differently where questions of originality were at issue. *Compare North Coast Indus. v. Jason Maxwell, Inc.*, 972 F.2d 1031, 1035 (9th Cir. 1992), *with Ets-Hokin*, 225 F.3d at 1073. More recently, several district courts within the Ninth Circuit have treated copyrightability as a question for only the court, regardless of whether it is a pure question of law. *See Stern v. Does*, No. 09-1986, 2011 U.S. Dist. LEXIS 37735, *7 (C.D. Cal. Feb. 10, 2011); *Jonathan Browning, Inc. v. Venetian Casino Resort LLC*, No. C 07-3983, 2009 U.S. Dist. LEXIS 57525, at *2 (N.D. Cal. June 19, 2009); *see also Pivot Point Int'l, Inc. v. Charlene Prods., Inc.*, 932 F. Supp. 220, 225 (N.D. Ill. 1996) (Easterbrook, J.) (citing to *Markman v. Westview Instruments, Inc.*, 517 U.S. 370 (1996), and concluding that whether works are copyrightable is a question which the "jury has nothing to do with"). We need not address any of these questions, because the parties here agreed that the district court would decide copyrightability, and both largely agree that we may undertake a review of that determination de novo.

presents a low bar—and the scope of conduct that constitutes infringing activity. The court also erred by importing fair use principles, including interoperability concerns, into its copyrightability analysis.

For the reasons that follow, we conclude that the declaring code and the structure, sequence, and organization of the 37 Java API packages are entitled to copyright protection. Because there is an insufficient record as to the relevant fair use factors, we remand for further proceedings on Google's fair use defense.

A. Copyrightability

The Copyright Act provides protection to "original works of authorship fixed in any tangible medium of expression," including "literary works." 17 U.S.C. § 102(a). It is undisputed that computer programs—defined in the Copyright Act as "a set of statements or instructions to be used directly or indirectly in a computer in order to bring about a certain result," 17 U.S.C. § 101—can be subject to copyright protection as "literary works." *See Atari Games Corp. v. Nintendo of Am., Inc.*, 975 F.2d 832, 838 (Fed. Cir. 1992) ("As literary works, copyright protection extends to computer programs."). Indeed, the legislative history explains that "literary works" includes "computer programs to the extent that they incorporate authorship in the programmer's expression of original ideas, as distinguished from the ideas themselves." H.R. Rep. No. 1476, 94th Cong., 2d Sess. 54, *reprinted in* 1976 U.S.C.C.A.N. 5659, 5667.

By statute, a work must be "original" to qualify for copyright protection. 17 U.S.C. § 102(a). This "originality requirement is not particularly stringent," however. *Feist Publ'ns, Inc. v. Rural Tel. Serv. Co.*, 499 U.S. 340, 358

(1991). "Original, as the term is used in copyright, means only that the work was independently created by the author (as opposed to copied from other works), and that it possesses at least some minimal degree of creativity." *Id.* at 345.

Copyright protection extends only to the expression of an idea—not to the underlying idea itself. *Mazer v. Stein*, 347 U.S. 201, 217 (1954) ("Unlike a patent, a copyright gives no exclusive right to the art disclosed; protection is given only to the expression of the idea—not the idea itself."). This distinction—commonly referred to as the "idea/expression dichotomy"—is codified in Section 102(b) of the Copyright Act, which provides:

> In no case does copyright protection for an original work of authorship extend to any idea, procedure, process, system, method of operation, concept, principle, or discovery, regardless of the form in which it is described, explained, illustrated, or embodied in such work.

17 U.S.C. § 102(b); *see Golan v. Holder*, 132 S. Ct. 873, 890 (2012) ("The idea/expression dichotomy is codified at 17 U.S.C. § 102(b).").

The idea/expression dichotomy traces back to the Supreme Court's decision in *Baker v. Selden*, 101 U.S. 99, 101 (1879). In *Baker*, the plaintiff Selden wrote and obtained copyrights on a series of books setting out a new system of bookkeeping. *Id.* at 100. The books included an introductory essay explaining the system and blank forms with ruled lines and headings designed for use with that system. *Id.* Baker published account books employing a system with similar forms, and Selden filed suit alleging copyright infringement. According to Selden, the "ruled

lines and headings, given to illustrate the system, are a part of the book" and "no one can make or use similar ruled lines and headings, or ruled lines and headings made and arranged on substantially the same system, without violating the copyright." *Id.* at 101.

The Supreme Court framed the issue on appeal in *Baker* as "whether the exclusive property in a system of book-keeping can be claimed, under the law of copyright, by means of a book in which that system is explained." *Id.* In reversing the circuit court's decision, the Court concluded that the "copyright of a book on book-keeping cannot secure the exclusive right to make, sell, and use account-books prepared upon the plan set forth in such book." *Id.* at 104. Likewise, the "copyright of a work on mathematical science cannot give to the author an exclusive right to the methods of operation which he propounds." *Id.* at 103. The Court found that, although the copyright protects the way Selden "explained and described a peculiar system of book-keeping," it does not prevent others from using the system described therein. *Id.* at 104. The Court further indicated that, if it is necessary to use the forms Selden included in his books to make use of the accounting system, that use would not amount to copyright infringement. *See id.* (noting that the public has the right to use the account-books and that, "in using the art, the ruled lines and headings of accounts must necessarily be used as incident to it").

Courts routinely cite Baker as the source of several principles incorporated into Section 102(b) that relate to this appeal, including that: (1) copyright protection extends only to expression, not to ideas, systems, or processes; and (2) "those elements of a computer program that are necessarily incidental to its function are . . .

unprotectable." *See Computer Assocs. Int'l v. Altai*, 982 F.2d 693, 704–05 (2d Cir. 1992) ("*Altai*") (discussing *Baker*, 101 U.S. at 103–04).

It is well established that copyright protection can extend to both literal and non-literal elements of a computer program. *See Altai*, 982 F.2d at 702. The literal elements of a computer program are the source code and object code. *See Johnson Controls, Inc. v. Phoenix Control Sys., Inc.*, 886 F.2d 1173, 1175 (9th Cir. 1989). Courts have defined source code as "the spelled-out program commands that humans can read." *Lexmark Int'l, Inc. v. Static Control Components, Inc.*, 387 F.3d 522, 533 (6th Cir. 2004). Object code refers to "the binary language comprised of zeros and ones through which the computer directly receives its instructions." *Altai*, 982 F.2d at 698. Both source and object code "are consistently held protected by a copyright on the program." *Johnson Controls*, 886 F.2d at 1175; *see also Altai*, 982 F.2d at 702 ("It is now well settled that the literal elements of computer programs, i.e., their source and object codes, are the subject of copyright protection."). Google nowhere disputes that premise. *See, e.g.*, Oral Argument at 57:38.

The non-literal components of a computer program include, among other things, the program's sequence, structure, and organization, as well as the program's user interface. *Johnson Controls*, 886 F.2d at 1175. As discussed below, whether the non-literal elements of a program "are protected depends on whether, on the particular facts of each case, the component in question qualifies as an expression of an idea, or an idea itself." *Id.*

In this case, Oracle claims copyright protection with respect to both: (1) literal elements of its API packages—the 7,000 lines of declaring source code; and (2) non-literal

elements—the structure, sequence, and organization of each of the 37 Java API packages.

The distinction between literal and non-literal aspects of a computer program is separate from the distinction between literal and non-literal copying. *See Altai*, 982 F.2d at 701–02. "Literal" copying is verbatim copying of original expression. "Non-literal" copying is "paraphrased or loosely paraphrased rather than word for word." *Lotus Dev. Corp. v. Borland Int'l*, 49 F.3d 807, 814 (1st Cir. 1995). Here, Google concedes that it copied the declaring code verbatim. Oracle explains that the lines of declaring code "embody the structure of each [API] package, just as the chapter titles and topic sentences represent the structure of a novel." Appellant Br. 45. As Oracle explains, when Google copied the declaring code in these packages "it also copied the 'sequence and organization' of the packages (i.e., the three-dimensional structure with all the chutes and ladders)" employed by Sun/Oracle in the packages. Appellant Br. 27. Oracle also argues that the nonliteral elements of the API packages—the structure, sequence, and organization that led naturally to the implementing code Google created—are entitled to protection. Oracle does not assert "literal" copying of the entire SSO, but, rather, that Google literally copied the declaring code and then paraphrased the remainder of the SSO by writing its own implementing code. It therefore asserts non-literal copying with respect to the entirety of the SSO.

At this stage, it is undisputed that the declaring code and the structure and organization of the Java API packages are original. The testimony at trial revealed that designing the Java API packages was a creative process and that the Sun/Oracle developers had a vast range of

options for the structure and organization. In its copyrightability decision, the district court specifically found that the API packages are both creative and original, and Google concedes on appeal that the originality requirements are met. See *Copyrightability Decision*, 872 F. Supp. 2d at 976 ("The overall name tree, of course, has creative elements . . . ."); *Id.* at 999 ("Yes, it is creative. Yes, it is original."); Appellee Br. 5 ("Google does not dispute" the district court's finding that "the Java API clears the low originality threshold."). The court found, however, that neither the declaring code nor the SSO was entitled to copyright protection under the Copyright Act.

Although the parties agree that Oracle's API packages meet the originality requirement under Section 102(a), they disagree as to the proper interpretation and application of Section 102(b). For its part, Google suggests that there is a two-step copyrightability analysis, wherein Section 102(a) grants copyright protection to original works, while Section 102(b) takes it away if the work has a functional component. To the contrary, however, Congress emphasized that Section 102(b) "in no way enlarges or contracts the scope of copyright protection" and that its "purpose is to restate . . . that the basic dichotomy between expression and idea remains unchanged." *Feist*, 499 U.S. at 356 (quoting H.R. Rep. No. 1476, 94th Cong., 2d Sess. 54, *reprinted in* 1976 U.S.C.C.A.N. 5659, 5670). "Section 102(b) does not extinguish the protection accorded a particular expression of an idea merely because that expression is embodied in a method of operation." *Mitel, Inc. v. Iqtel, Inc.*, 124 F.3d 1366, 1372 (10th Cir. 1997). Section 102(a) and 102(b) are to be considered collectively so that certain expressions

are subject to greater scrutiny. *Id.* In assessing copyrightability, the district court is required to ferret out apparent expressive aspects of a work and then separate protectable expression from "unprotectable ideas, facts, processes, and methods of operation." *See Atari*, 975 F.2d at 839.

Of course, as with many things, in defining this task, the devil is in the details. Circuit courts have struggled with, and disagree over, the tests to be employed when attempting to draw the line between what is protectable expression and what is not. *Compare Whelan Assocs., Inc. v. Jaslow Dental Lab., Inc.*, 797 F.2d 1222, 1236 (3d Cir. 1986) (everything not necessary to the purpose or function of a work is expression), *with Lotus*, 49 F.3d at 815 (methods of operation are means by which a user operates something and any words used to effectuate that operation are unprotected expression). When assessing whether the non-literal elements of a computer program constitute protectable expression, the Ninth Circuit has endorsed an "abstraction-filtration-comparison" test formulated by the Second Circuit and expressly adopted by several other circuits. *Sega Enters. Ltd. v. Accolade, Inc.*, 977 F.2d 1510, 1525 (9th Cir. 1992) ("In our view, in light of the essentially utilitarian nature of computer programs, the Second Circuit's approach is an appropriate one."). This test rejects the notion that anything that performs a function is necessarily uncopyrightable. *See Mitel*, 124 F.3d at 1372 (rejecting the *Lotus* court's formulation, and concluding that, "although an element of a work may be characterized as a method of operation, that element may nevertheless contain expression that is eligible for copyright protection."). And it also rejects as flawed the *Whelan* assumption that, once any separable

idea can be identified in a computer program everything else must be protectable expression, on grounds that more than one idea may be embodied in any particular program. *Altai*, 982 F.2d at 705–06.

Thus, this test eschews bright line approaches and requires a more nuanced assessment of the particular program at issue in order to determine what expression is protectable and infringed. As the Second Circuit explains, this test has three steps. In the abstraction step, the court "first break[s] down the allegedly infringed program into its constituent structural parts." *Id.* at 706. In the filtration step, the court "sift[s] out all non-protectable material," including ideas and "expression that is necessarily incidental to those ideas." *Id.* In the final step, the court compares the remaining creative expression with the allegedly infringing program.[4]

In the second step, the court is first to assess whether the expression is original to the programmer or author. *Atari*, 975 F.2d at 839. The court must then determine whether the particular inclusion of any level of abstraction is dictated by considerations of efficiency, required by factors already external to the program itself, or taken from the public domain—all of which would render the expression unprotectable. *Id.* These conclusions are to be informed by traditional copyright principles of originality,

---

[4] Importantly, this full analysis only applies where a copyright owner alleges infringement of the non-literal aspects of its work. Where "admitted literal copying of a discrete, easily-conceptualized portion of a work" is at issue—as with Oracle's declaring code—a court "need not perform a complete abstraction-filtration-comparison analysis" and may focus the protectability analysis on the filtration stage, with attendant reference to standard copyright principles. *Mitel*, 124 F.3d at 1372–73.

merger, and scenes a faire. *See Mitel*, 124 F.3d at 1372 ("Although this core of expression is eligible for copyright protection, it is subject to the rigors of filtration analysis which excludes from protection expression that is in the public domain, otherwise unoriginal, or subject to the doctrines of merger and scenes a faire.").

In all circuits, it is clear that the first step is part of the copyrightability analysis and that the third is an infringement question. It is at the second step of this analysis where the circuits are in less accord. Some treat all aspects of this second step as part of the copyrightability analysis, while others divide questions of originality from the other inquiries, treating the former as a question of copyrightability and the latter as part of the infringement inquiry. *Compare Lexmark*, 387 F.3d at 537–38 (finding that the district court erred in assessing principles of merger and scenes a faire in the infringement analysis, rather than as a component of copyrightability), *with Kregos*, 937 F.2d at 705 (noting that the Second Circuit has considered the merger doctrine "in determining whether actionable infringement has occurred, rather than whether a copyright is valid"); *see also Lexmark*, 387 F.3d at 557 (Feikens, J., dissenting-in-part) (noting the circuit split and concluding that, where a court is assessing merger of an expression with a method of operation, "I would find the merger doctrine can operate only as a defense to infringement in that context, and as such has no bearing on the question of copyrightability."). We need not assess the wisdom of these respective views because there is no doubt on which side of this circuit split the Ninth Circuit falls.

In the Ninth Circuit, while questions regarding originality are considered questions of copyrightability,

concepts of merger and scenes a faire are affirmative defenses to claims of infringement. *Ets-Hokin*, 225 F.3d at 1082; *Satava v. Lowry*, 323 F.3d 805, 810 n.3 (9th Cir. 2003) ("The Ninth Circuit treats scenes a faire as a defense to infringement rather than as a barrier to copyrightability."). The Ninth Circuit has acknowledged that "there is some disagreement among courts as to whether these two doctrines figure into the issue of copyrightability or are more properly defenses to infringement." *Ets-Hokin*, 225 F.3d at 1082 (citations omitted). It, nonetheless, has made clear that, in that circuit, these concepts are to be treated as defenses to infringement. *Id.* (citing *Kregos*, 937 F.2d at 705 (holding that the merger doctrine relates to infringement, not copyrightability); *Reed-Union Corp. v. Turtle Wax, Inc.*, 77 F.3d 909, 914 (7th Cir. 1996) (explaining why the doctrine of scenes a faire is separate from the validity of a copyright)).

With these principles in mind, we turn to the trial court's analysis and judgment and to Oracle's objections thereto. While the trial court mentioned the abstraction-filtration-comparison test when describing the development of relevant law, it did not purport to actually apply that test. Instead, it moved directly to application of familiar principles of copyright law when assessing the copyrightability of the declaring code and interpreted Section 102(b) to preclude copyrightability for any functional element "essential for interoperability" "regardless of its form." *Copyrightability Decision*, 872 F. Supp. 2d at 997.

Oracle asserts that all of the trial court's conclusions regarding copyrightability are erroneous. Oracle argues that its Java API packages are entitled to protection

under the Copyright Act because they are expressive and could have been written and organized in any number of ways to achieve the same functions. Specifically, Oracle argues that the district court erred when it: (1) concluded that each line of declaring code is uncopyrightable because the idea and expression have merged; (2) found the declaring code uncopyrightable because it employs short phrases; (3) found all aspects of the SSO devoid of protection as a "method of operation" under 17 U.S.C. § 102(b); and (4) invoked Google's "interoperability" concerns in the copyrightability analysis. For the reasons explained below, we agree with Oracle on each point.

1. Declaring Source Code

First, Oracle argues that the district court erred in concluding that each line of declaring source code is completely unprotected under the merger and short phrases doctrines. Google responds that Oracle waived its right to assert copyrightability based on the 7,000 lines of declaring code by failing "to object to instructions and a verdict form that effectively eliminated that theory from the case." Appellee Br. 67. Even if not waived, moreover, Google argues that, because there is only one way to write the names and declarations, the merger doctrine bars copyright protection.

We find that Oracle did not waive arguments based on Google's literal copying of the declaring code. Prior to trial, both parties informed the court that Oracle's copyright infringement claims included the declarations of the API elements in the Android class library source code. *See* Oracle's Statement of Issues Regarding Copyright, *Oracle Am., Inc. v. Google Inc.*, No. 3:10-cv-3561 (N.D. Cal. Apr. 12, 2012), ECF No. 899-1, at 3 (Oracle accuses the "declarations of the API elements in the Android class

library source code and object code that implements the 37 API packages" of copyright infringement.); *see also* Google's Proposed Statement of Issues Regarding Copyright, *Oracle Am., Inc. v. Google Inc.*, No. 3:10-cv-3561 (N.D. Cal. Apr. 12, 2012), ECF No. 901, at 2 (Oracle accuses the "declarations of the API elements in Android class library source code and object code that implements the 37 API packages.").

While Google is correct that the jury instructions and verdict form focused on the structure and organization of the packages, we agree with Oracle that there was no need for the jury to address copying of the declaring code because Google conceded that it copied it verbatim. Indeed, the district court specifically instructed the jury that "Google agrees that it uses the same names and declarations" in Android. *Final Charge to the Jury* at 10.

That the district court addressed the declaring code in its post-jury verdict copyrightability decision further confirms that the verbatim copying of declaring code remained in the case. The court explained that the "identical lines" that Google copied into Android "are those lines that specify the names, parameters and functionality of the methods and classes, lines called 'declarations' or 'headers.'" *Copyrightability Decision*, 872 F. Supp. 2d at 979. The court specifically found that the declaring code was not entitled to copyright protection under the merger and short phrases doctrines. We address each in turn.

a. Merger

The merger doctrine functions as an exception to the idea/expression dichotomy. It provides that, when there are a limited number of ways to express an idea, the idea

is said to "merge" with its expression, and the expression becomes unprotected. *Altai*, 982 F.2d at 707–08. As noted, the Ninth Circuit treats this concept as an affirmative defense to infringement. *Ets-Hokin*, 225 F.3d at 1082. Accordingly, it appears that the district court's merger analysis is irrelevant to the question of whether Oracle's API packages are copyrightable in the first instance. Regardless of when the analysis occurs, we conclude that merger does not apply on the record before us.

Under the merger doctrine, a court will not protect a copyrighted work from infringement if the idea contained therein can be expressed in only one way. *Satava v. Lowry*, 323 F.3d 805, 812 n.5 (9th Cir. 2003). For computer programs, "this means that when specific [parts of the code], even though previously copyrighted, are the only and essential means of accomplishing a given task, their later use by another will not amount to infringement." *Altai*, 982 F.2d at 708 (citation omitted). We have recognized, however, applying Ninth Circuit law, that the "unique arrangement of computer program expression . . . does not merge with the process so long as alternate expressions are available." *Atari*, 975 F.2d at 840.

In *Atari*, for example, Nintendo designed a program—the 10NES—to prevent its video game system from accepting unauthorized game cartridges. 975 F.2d at 836. Nintendo "chose arbitrary programming instructions and arranged them in a unique sequence to create a purely arbitrary data stream" which "serves as the key to unlock the NES." *Id.* at 840. Because Nintendo produced expert testimony "showing a multitude of different ways to generate a data stream which unlocks the NES console," we concluded that Nintendo's specific choice of code did not merge with the process. *Id.*

Here, the district court found that, "no matter how creative or imaginative a Java method specification may be, the entire world is entitled to use the same method specification (inputs, outputs, parameters) so long as the line-by-line implementations are different." *Copyrightability Decision*, 872 F. Supp. 2d at 998. In its analysis, the court identified the method declaration as the idea and found that the implementation is the expression. *Id.* ("The method specification is the *idea*. The method implementation is the *expression*. No one may monopolize the *idea*.") (emphases in original). The court explained that, under the rules of Java, a programmer must use the identical "declaration or method header lines" to "declare a method specifying the *same* functionality." *Id.* at 976. Because the district court found that there was only one way to write the declaring code for each of the Java packages, it concluded that "the merger doctrine bars anyone from claiming exclusive copyright ownership" of it. *Id.* at 998. Accordingly, the court held there could be "no copyright violation in using the identical declarations." *Id.*

Google agrees with the district court that the implementing code is the expression entitled to protection—not the declaring code. Indeed, at oral argument, counsel for Google explained that, "it is not our position that none of Java is copyrightable. Obviously, Google spent two and a half years . . . to write from scratch all of the implementing code." Oral Argument at 33:16.[5]

[5] It is undisputed that Microsoft and Apple developed mobile operating systems from scratch, using their own array of software packages. When asked whether Google could also copy all of Microsoft or Apple's declaring code—codes that obviously differ from those at issue here—counsel for Google responded: "Yes, but only the structure, sequence, and organization. Only the command structure—

Because it is undisputed that Google wrote its own implementing code, the copyrightability of the precise language of that code is not at issue on appeal. Instead, our focus is on the declaring code and structure of the API packages.

On appeal, Oracle argues that the district court: (1) misapplied the merger doctrine; and (2) failed to focus its analysis on the options available to the original author. We agree with Oracle on both points. First, we agree that merger cannot bar copyright protection for any lines of declaring source code unless Sun/Oracle had only one way, or a limited number of ways, to write them. *See Satava*, 323 F.3d at 812 n.5 ("Under the merger doctrine, courts will not protect a copyrighted work from infringement if the idea underlying the copyrighted work can be expressed in only one way, lest there be a monopoly on the underlying idea."). The evidence showed that Oracle had "unlimited options as to the selection and arrangement of the 7000 lines Google copied." Appellant Br. 50. Using the district court's "java.lang.Math.max" example, Oracle explains that the developers could have called it any number of things, including "Math.maximum" or "Arith.larger." This was not a situation where Oracle was selecting among preordained names and phrases to create its packages.[6] As the district court recognized, moreover,

---

what you need to access the functions. You'd have to rewrite all the millions of lines of code in Apple or in Microsoft which is what Google did in Android." Oral Argument at 36:00.

[6] In their brief as amici curiae in support of reversal, Scott McNealy and Brian Sutphin—both former executives at Sun who were involved in the development of the Java platform—provide a detailed example of the creative choices involved in designing a Java package. Looking at the "java.text" package, they explain that it "contains 25 classes, 2 interfaces, and hundreds of methods to handle text, dates, numbers,

"the Android method and class names could have been different from the names of their counterparts in Java and still have worked." *Copyrightability Decision*, 872 F. Supp. 2d at 976. Because "alternative expressions [we]re available," there is no merger. *See Atari*, 975 F.2d at 840.

We further find that the district court erred in focusing its merger analysis on the options available to Google at the time of copying. It is well-established that copyrightability and the scope of protectable activity are to be evaluated at the time of creation, not at the time of infringement. *See Apple Computer, Inc. v. Formula Int'l, Inc.*, 725 F.2d 521, 524 (9th Cir. 1984) (quoting National Commission on New Technological Uses of Copyrighted Works, Final Report at 21 (1979) ("CONTU Report") (recognizing that the Copyright Act was designed "to protect all works of authorship from the moment of their fixation in any tangible medium of expression")). The focus is, therefore, on the options that were available to Sun/Oracle at the time it created the API packages. Of course, once Sun/Oracle created "java.lang.Math.max," programmers who want to use that particular package have to call it by that name. But, as the court acknowledged, nothing prevented Google from writing its

---

and messages in a manner independent of natural human languages . . . ." Br. of McNealy and Sutphin 14–15. Java's creators had to determine whether to include a java.text package in the first place, how long the package would be, what elements to include, how to organize that package, and how it would relate to other packages. *Id.* at 16. This description of Sun's creative process is consistent with the evidence presented at trial. *See* Appellant Br. 12–13 (citing testimony that it took years to write some of the Java packages and that Sun/Oracle developers had to "wrestle with what functions to include in the package, which to put in other packages, and which to omit entirely").

own declaring code, along with its own implementing code, to achieve the same result. In such circumstances, the chosen expression simply does not merge with the idea being expressed.[7]

It seems possible that the merger doctrine, when properly analyzed, would exclude the three packages identified by the district court as core packages from the scope of actionable infringing conduct. This would be so if the Java authors, at the time these packages were created, had only a limited number of ways to express the methods and classes therein if they wanted to write in the Java language. In that instance, the idea may well be merged with the expression in these three packages.[8] Google did

---

[7] The district court did not find merger with respect to the structure, sequence, and organization of Oracle's Java API packages. Nor could it, given the court's recognition that there were myriad ways in which the API packages could have been organized. Indeed, the court found that the SSO is original and that "nothing in the rules of the Java language . . . required that Google replicate the same groupings." *Copyrightability Decision*, 872 F. Supp. 2d at 999. As discussed below, however, the court nonetheless found that the SSO is an uncopyrightable "method of operation."

[8] At oral argument, counsel for Oracle was asked whether we should view the three core packages "differently vis-à-vis the concept of a method of operation than the other packages." *See* Oral Argument at 7:43. He responded: "I think not your Honor. I would view them differently with respect to fair use . . . . It's not that they are more basic. It's that there are just several methods, that is, routines, within just those three packages that are necessary to 'speak the Java language.' Nothing in the other thirty-four packages is necessary in order to speak in Java, so to speak." *Id.* Counsel conceded, however, that this issue "might go to merger. It might go to the question whether someone—since we conceded that it's okay to use the language—if it's alright to use the language that there are certain things that the original developers had to say in order to use that

not present its merger argument in this way below and does not do so here, however. Indeed, Google does not try to differentiate among the packages for purposes of its copyrightability analysis and does not appeal the infringement verdict as to the packages. For these reasons, we reject the trial court's merger analysis.

b. Short Phrases

The district court also found that Oracle's declaring code consists of uncopyrightable short phrases. Specifically, the court concluded that, "while the Android method and class names could have been different from the names of their counterparts in Java and still have worked, copyright protection never extends to names or short phrases as a matter of law." *Copyrightability Decision*, 872 F. Supp. 2d at 976.

The district court is correct that "[w]ords and short phrases such as names, titles, and slogans" are not subject to copyright protection. 37 C.F.R. § 202.1(a). The court failed to recognize, however, that the relevant question for copyrightability purposes is not whether the work at issue contains short phrases—as literary works often do—but, rather, whether those phrases are creative. *See Soc'y of Holy Transfiguration Monastery, Inc. v. Gregory*, 689 F.3d 29, 52 (1st Cir. 2012) (noting that "not all short phrases will automatically be deemed uncopyrightable"); *see also* 1 Melville B. Nimmer & David Nimmer, Nimmer on Copyright § 2.01[B] (2013) ("[E]ven a short phrase may command copyright protection if it exhibits sufficient creativity."). And, by dissecting the individual lines of declaring code at issue into short phrases, the district

---

language, arguably, although I still think it's really a fair use analysis." *Id.*

court further failed to recognize that an original *combination* of elements can be copyrightable. *See Softel, Inc. v. Dragon Med. & Scientific Commc'ns*, 118 F.3d 955, 964 (2d Cir. 1997) (noting that, in Feist, "the Court made quite clear that a compilation of nonprotectible elements can enjoy copyright protection even though its constituent elements do not").

By analogy, the opening of Charles Dickens' *A Tale of Two Cities* is nothing but a string of short phrases. Yet no one could contend that this portion of Dickens' work is unworthy of copyright protection because it can be broken into those shorter constituent components. The question is not whether a short phrase or series of short phrases can be extracted from the work, but whether the manner in which they are used or strung together exhibits creativity.

Although the district court apparently focused on individual lines of code, Oracle is not seeking copyright protection for a specific short phrase or word. Instead, the portion of declaring code at issue is 7,000 lines, and Google's own "Java guru" conceded that there can be "creativity and artistry even in a single method declaration." Joint Appendix ("J.A.") 20,970. Because Oracle "exercised creativity in the selection and arrangement" of the method declarations when it created the API packages and wrote the relevant declaring code, they contain protectable expression that is entitled to copyright protection. *See Atari*, 975 F.2d at 840; *see also* 17 U.S.C. §§ 101, 103 (recognizing copyright protection for "compilations" which are defined as work that is "selected, coordinated, or arranged in such a way that the resulting work as a whole constitutes an original work of authorship"). Accordingly, we conclude that the district

court erred in applying the short phrases doctrine to find the declaring code not copyrightable.

c. Scenes a Faire

The scenes a faire doctrine, which is related to the merger doctrine, operates to bar certain otherwise creative expression from copyright protection. *Apple Computer, Inc. v. Microsoft Corp.*, 35 F.3d 1435, 1444 (9th Cir. 1994). It provides that "expressive elements of a work of authorship are not entitled to protection against infringement if they are standard, stock, or common to a topic, or if they necessarily follow from a common theme or setting." *Mitel*, 124 F.3d at 1374. Under this doctrine, "when certain commonplace expressions are indispensable and naturally associated with the treatment of a given idea, those expressions are treated like ideas and therefore [are] not protected by copyright." *Swirsky v. Carey*, 376 F.3d 841, 850 (9th Cir. 2004). In the computer context, "the scene a faire doctrine denies protection to program elements that are dictated by external factors such as 'the mechanical specifications of the computer on which a particular program is intended to run' or 'widely accepted programming practices within the computer industry.'" *Softel*, 118 F.3d at 963 (citation omitted).

The trial court rejected Google's reliance on the scenes a faire doctrine. It did so in a footnote, finding that Google had failed to present evidence to support the claim that either the grouping of methods within the classes or the code chosen for them "would be so expected and customary as to be permissible under the scenes a faire doctrine." *Copyrightability Decision*, 872 F. Supp. 2d at 999 n.9. Specifically, the trial court found that "it is impossible to say on this record that *all* of the classes and their contents are typical of such classes and, on this

record, this order rejects Google's global argument based on *scenes a faire*." *Id.*

On appeal, Google refers to scenes a faire concepts briefly, as do some amici, apparently contending that, because programmers have become accustomed to and comfortable using the groupings in the Java API packages, those groupings are so commonplace as to be indispensable to the expression of an acceptable programming platform. As such, the argument goes, they are so associated with the "idea" of what the packages are accomplishing that they should be treated as ideas rather than expression. *See* Br. of Amici Curiae Rackspace US, Inc., et al. at 19–22.

Google cannot rely on the scenes a faire doctrine as an alternative ground upon which we might affirm the copyrightability judgment of the district court. This is so for several reasons. First, as noted, like merger, in the Ninth Circuit, the scenes a faire doctrine is a component of the infringement analysis. "[S]imilarity of expression, whether literal or non-literal, which necessarily results from the fact that the common idea is only capable of expression in more or less stereotyped form, will preclude a finding of actionable similarity." 4 Nimmer on Copyright § 13.03[B][3]. Thus, the expression is not excluded from copyright protection; it is just that certain copying is forgiven as a necessary incident of any expression of the underlying idea. *See Satava*, 323 F.3d at 810 n.3 ("The Ninth Circuit treats scenes a faire as a defense to infringement rather than as a barrier to copyrightability.").

Second, Google has not objected to the trial court's conclusion that Google failed to make a sufficient factual record to support its contention that the groupings and

code chosen for the 37 Java API packages were driven by external factors or premised on features that were either commonplace or essential to the idea being expressed. Google provides no record citations indicating that such a showing was made and does not contend that the trial court erred when it expressly found it was not. Indeed, Google does not even make this argument with respect to the core packages.

Finally, Google's reliance on the doctrine below and the amici reference to it here are premised on a fundamental misunderstanding of the doctrine. Like merger, the focus of the scenes a faire doctrine is on the circumstances presented to the creator, not the copier. *See Mitel*, 124 F.3d at 1375 (finding error to the extent the trial court discussed "whether external factors such as market forces and efficiency considerations justified Iqtel's copying of the command codes"). The court's analytical focus must be upon the external factors that dictated Sun's selection of classes, methods, and code—not upon what Google encountered at the time it chose to copy those groupings and that code. *See id.* "[T]he scenes a faire doctrine identifies and excludes from protection against infringement expression whose creation 'flowed naturally from considerations external to the author's creativity.'" *Id.* (quoting Nimmer § 13.03[F][3], at 13-131 (1997)). It is this showing the trial court found Google failed to make, and Google cites to nothing in the record which indicates otherwise.

For these reasons, the trial court was correct to conclude that the scenes a faire doctrine does not affect the copyrightability of either the declaring code in, or the SSO of, the Java API packages at issue.

2. The Structure, Sequence, and Organization of the API Packages

The district court found that the SSO of the Java API packages is creative and original, but nevertheless held that it is a "system or method of operation . . . and, therefore, cannot be copyrighted" under 17 U.S.C. § 102(b). *Copyrightability Decision*, 872 F. Supp. 2d at 976–77. In reaching this conclusion, the district court seems to have relied upon language contained in a First Circuit decision: *Lotus Development Corp. v. Borland International, Inc.*, 49 F.3d 807 (1st Cir. 1995), *aff'd without opinion by equally divided court*, 516 U.S. 233 (1996)[9]

In *Lotus*, it was undisputed that the defendant copied the menu command hierarchy and interface from Lotus 1-2-3, a computer spreadsheet program "that enables users to perform accounting functions electronically on a computer." 49 F.3d at 809. The menu command hierarchy referred to a series of commands—such as "Copy," "Print," and "Quit"—which were arranged into more than 50 menus and submenus. *Id.* Although the defendant did not copy any Lotus source code, it copied the menu command hierarchy into its rival program. The question before the court was "whether a computer menu command hierarchy is copyrightable subject matter." *Id.*

Although it accepted the district court's finding that Lotus developers made some expressive choices in

---

[9] The Supreme Court granted certiorari in *Lotus*, but, shortly after oral argument, the Court announced that it was equally divided and that Justice Stevens took no part in the consideration or decision of the case. The Court therefore left the First Circuit's decision undisturbed. *See Lotus*, 516 U.S. at 233–34.

selecting and arranging the command terms, the First Circuit found that the command hierarchy was not copyrightable because, among other things, it was a "method of operation" under Section 102(b). In reaching this conclusion, the court defined a "method of operation" as "the means by which a person operates something, whether it be a car, a food processor, or a computer." *Id.* at 815.[10] Because the Lotus menu command hierarchy provided "the means by which users control and operate Lotus 1-2-3," it was deemed unprotectable. *Id.* For example, if users wanted to copy material, they would use the "Copy" command and the command terms would tell the computer what to do. According to the *Lotus* court, the "fact that Lotus developers could have designed the Lotus menu command hierarchy differently is immaterial to the question of whether it is a 'method of operation.'" *Id.* at 816. (noting that "our initial inquiry is not whether the Lotus menu command hierarchy incorporates any expression"). The court further indicated that, "[i]f specific words are essential to operating something, then they are part of a 'method of operation' and, as such, are unprotectable." *Id.*

On appeal, Oracle argues that the district court's reliance on *Lotus* is misplaced because it is distinguishable on its facts and is inconsistent with Ninth Circuit law. We agree. First, while the defendant in *Lotus* did not copy any of the underlying code, Google concedes that it copied portions of Oracle's declaring source code verbatim. Second, the *Lotus* court found that the commands at issue there (copy, print, etc.) were not creative, but it is undisputed here that the declaring code and the structure

---

[10] The *Lotus* majority cited no authority for this definition of "method of operation."

and organization of the API packages are both creative and original. Finally, while the court in *Lotus* found the commands at issue were "essential to operating" the system, it is undisputed that—other than perhaps as to the three core packages—Google did not need to copy the structure, sequence, and organization of the Java API packages to write programs in the Java language.

More importantly, however, the Ninth Circuit has not adopted the court's "method of operation" reasoning in *Lotus*, and we conclude that it is inconsistent with binding precedent.[11] Specifically, we find that *Lotus* is inconsistent with Ninth Circuit case law recognizing that the structure, sequence, and organization of a computer program is eligible for copyright protection where it qualifies as an expression of an idea, rather than the idea itself. *See Johnson Controls*, 886 F.2d at 1175–76. And while the court in Lotus held "that expression that is part of a 'method of operation' cannot be copyrighted," 49 F.3d at 818, this court—applying Ninth Circuit law—reached the exact opposite conclusion, finding that copyright protects "the expression of [a] process or method," *Atari*, 975 F.2d at 839.

We find, moreover, that the hard and fast rule set down in *Lotus* and employed by the district court here—i.e., that elements which perform a function can never be copyrightable—is at odds with the Ninth Circuit's

---

[11] As Oracle points out, the Ninth Circuit has cited *Lotus* only one time, on a procedural issue. *See Danjaq LLC v. Sony Corp.*, 263 F.3d 942, 954 (9th Cir. 2001) (citing *Lotus* for the proposition that delay "has been held permissible, among other reasons, when it is necessitated by the exhaustion of remedies through the administrative process . . . when it is used to evaluate and prepare a complicated claim").

endorsement of the abstraction-filtration-comparison analysis discussed earlier. As the Tenth Circuit concluded in expressly rejecting the *Lotus* "method of operation" analysis, in favor of the Second Circuit's abstraction-filtration-comparison test, "although an element of a work may be characterized as a method of operation, that element may nevertheless contain expression that is eligible for copyright protection." *Mitel*, 124 F.3d at 1372. Specifically, the court found that Section 102(b) "does not extinguish the protection accorded a particular expression of an idea merely because that expression is embodied in a method of operation at a higher level of abstraction." *Id.*

Other courts agree that components of a program that can be characterized as a "method of operation" may nevertheless be copyrightable. For example, the Third Circuit rejected a defendant's argument that operating system programs are "per se" uncopyrightable because an operating system is a "method of operation" for a computer. *Apple Computer, Inc. v. Franklin Computer Corp.*, 714 F.2d 1240, 1250–52 (3d Cir. 1983). The court distinguished between the "method which instructs the computer to perform its operating functions" and "the instructions themselves," and found that the instructions were copyrightable. *Id.* at 1250–51. In its analysis, the court noted: "[t]hat the words of a program are used ultimately in the implementation of a process should in no way affect their copyrightability." *Id.* at 1252 (quoting CONTU Report at 21). The court focused "on whether the idea is capable of various modes of expression" and indicated that, "[i]f other programs can be written or created which perform the same function as [i]n Apple's operating system program, then that program is an expression of the idea and hence copyrightable." *Id.* at

1253. Notably, no other circuit has adopted the First Circuit's "method of operation" analysis.

Courts have likewise found that classifying a work as a "system" does not preclude copyright for the particular expression of that system. *See Toro Co. v. R & R Prods. Co.*, 787 F.2d 1208, 1212 (8th Cir. 1986) (rejecting the district court's decision that "appellant's parts numbering system is not copyrightable because it is a 'system'" and indicating that Section 102(b) does not preclude protection for the "particular expression" of that system); *see also Am. Dental Ass'n v. Delta Dental Plans Ass'n*, 126 F.3d 977, 980 (7th Cir. 1997) ("A dictionary cannot be called a 'system' just because new novels are written using words, all of which appear in the dictionary. Nor is word-processing software a 'system' just because it has a command structure for producing paragraphs.").

Here, the district court recognized that the SSO "resembles a taxonomy," but found that "it is nevertheless a command structure, a system or method of operation—a long hierarchy of over six thousand commands to carry out pre-assigned functions." *Copyrightability Decision*, 872 F. Supp. 2d at 999–1000.[12] In other words, the court concluded that, although the SSO is expressive, it is not copyrightable because it is also functional. The problem with the district court's approach is that computer programs are by definition functional—they are all designed to accomplish some task. Indeed, the statutory definition of "computer program" acknowledges that they function "to bring about a certain result." *See* 17 U.S.C.

[12] This analogy by the district court is meaningful because taxonomies, in varying forms, have generally been deemed copyrightable. *See, e.g.*, *Practice Mgmt. Info. Corp. v. Am. Med. Ass'n*, 121 F.3d 516, 517–20 (9th Cir. 1997); *Am. Dental*, 126 F.3d at 978–81.

§ 101 (defining a "computer program" as "a set of statements or instructions to be used directly or indirectly in a computer in order to bring about a certain result"). If we were to accept the district court's suggestion that a computer program is uncopyrightable simply because it "carr[ies] out pre-assigned functions," no computer program is protectable. That result contradicts Congress's express intent to provide copyright protection to computer programs, as well as binding Ninth Circuit case law finding computer programs copyrightable, despite their utilitarian or functional purpose. Though the trial court did add the caveat that it "does not hold that the structure, sequence and organization of all computer programs may be stolen," *Copyrightability Decision*, 872 F. Supp. 2d at 1002, it is hard to see how its method of operation analysis could lead to any other conclusion.

While it does not appear that the Ninth Circuit has addressed the precise issue, we conclude that a set of commands to instruct a computer to carry out desired operations may contain expression that is eligible for copyright protection. *See Mitel*, 124 F.3d at 1372. We agree with Oracle that, under Ninth Circuit law, an original work—even one that serves a function—is entitled to copyright protection as long as the author had multiple ways to express the underlying idea. Section 102(b) does not, as Google seems to suggest, automatically deny copyright protection to elements of a computer program that are functional. Instead, as noted, Section 102(b) codifies the idea/expression dichotomy and the legislative history confirms that, among other things, Section 102(b) was "intended to make clear that the expression adopted by the programmer is the copyrightable element in a computer program." H.R. Rep.

No. 1476, 94th Cong., 2d Sess. 54, *reprinted in* 1976 U.S.C.C.A.N. 5659, 5670. Therefore, even if an element directs a computer to perform operations, the court must nevertheless determine whether it contains any separable expression entitled to protection.

On appeal, Oracle does not—and concedes that it cannot—claim copyright in the idea of organizing functions of a computer program or in the "package-class-method" organizational structure in the abstract. Instead, Oracle claims copyright protection only in its *particular* way of naming and organizing each of the 37 Java API packages.[13] Oracle recognizes, for example, that it "cannot copyright the idea of programs that open an internet connection," but "it can copyright the precise strings of code used to do so, at least so long as 'other language is available' to achieve the same function." Appellant Reply Br. 13–14 (citation omitted). Thus, Oracle concedes that Google and others could employ the Java language—much like anyone could employ the English language to write a paragraph without violating the copyrights of other English language writers. And, that Google may employ the "package-class-method" structure much like authors can employ the same rules of grammar chosen by other authors without fear of infringement. What Oracle contends is that, beyond that point, Google, like any author, is not permitted to employ the precise phrasing or precise structure chosen by Oracle to flesh out the

[13] At oral argument, counsel for Oracle explained that it "would never claim that anyone who uses a package-class-method manner of classifying violates our copyright. We don't own every conceivable way of organizing, we own only our specific expression—our specific way of naming each of these 362 methods, putting them into 36 classes, and 20 subclasses." Oral Argument at 16:44.

substance of its packages—the details and arrangement of the prose.

As the district court acknowledged, Google could have structured Android differently and could have chosen different ways to express and implement the functionality that it copied.[14] Specifically, the court found that "the very same functionality could have been offered in Android without duplicating the exact command structure used in Java." *Copyrightability Decision*, 872 F. Supp. 2d at 976. The court further explained that Google could have offered the same functions in Android by "rearranging the various methods under different groupings among the various classes and packages." *Id.* The evidence showed, moreover, that Google designed many of its own API packages from scratch, and, thus, could have designed its own corresponding 37 API packages if it wanted to do so.

Given the court's findings that the SSO is original and creative, and that the declaring code could have been

---

[14] Amici McNealy and Sutphin explain that "a quick examination of other programming environments shows that creators of other development platforms provide the same functions with wholly different creative choices." Br. of McNealy and Sutphin 17. For example, in Java, a developer setting the time zone would call the "setTime-Zone" method within the "DateFormat" class of the java. text package. *Id.* Apple's iOS platform, on the other hand, "devotes an entire class to set the time zone in an application—the 'NSTimeZone' class" which is in the "Foundation framework." *Id.* at 17–18 (noting that a "framework is Apple's terminology for a structure conceptually similar to Java's 'package'"). Microsoft provides similar functionality with "an entirely different structure, naming scheme, and selection." *Id.* at 18 ("In its Windows Phone development platform, Microsoft stores its time zone programs in the 'TimeZoneInfo' class in its 'Systems' namespace (Microsoft's version of a 'package' or 'framework')."). Again, this is consistent with the evidence presented at trial.

written and organized in any number of ways and still have achieved the same functions, we conclude that Section 102(b) does not bar the packages from copyright protection just because they also perform functions.

3. Google's Interoperability Arguments are Irrelevant to Copyrightability

Oracle also argues that the district court erred in invoking interoperability in its copyrightability analysis. Specifically, Oracle argues that Google's interoperability arguments are only relevant, if at all, to fair use—not to the question of whether the API packages are copyrightable. We agree.

In characterizing the SSO of the Java API packages as a "method of operation," the district court explained that "[d]uplication of the command structure is necessary for interoperability." *Copyrightability Decision*, 872 F. Supp. 2d at 977. The court found that, "[i]n order for at least some of [the pre-Android Java] code to run on Android, Google was required to provide the same java. package.Class.method() command system using the same names with the same 'taxonomy' and with the same functional specifications." *Id.* at 1000 (emphasis omitted). And, the court concluded that "Google replicated what was necessary to achieve a degree of interoperability—but no more, taking care, as said before, to provide its own implementations." *Id.* In reaching this conclusion, the court relied primarily on two Ninth Circuit decisions: *Sega Enterprises v. Accolade, Inc.*, 977 F.2d 1510 (9th Cir. 1992), and *Sony Computer Entertainment, Inc. v. Connectix, Corp.*, 203 F.3d 596 (9th Cir. 2000).

Both *Sega* and *Sony* are fair use cases in which copyrightability was addressed only tangentially. In *Sega*,

for example, Sega manufactured a video game console and game cartridges that contained hidden functional program elements necessary to achieve compatibility with the console. Defendant Accolade: (1) reverse-engineered Sega's video game programs to discover the requirements for compatibility; and (2) created its own games for the Sega console. *Sega*, 977 F.2d at 1514–15. As part of the reverse-engineering process, Accolade made intermediate copies of object code from Sega's console. *Id.* Although the court recognized that the intermediate copying of computer code may infringe Sega's copyright, it concluded that "disassembly of copyrighted object code is, as a matter of law, a fair use of the copyrighted work if such disassembly provides the only means of access to those elements of the code that are not protected by copyright and the copier has a legitimate reason for seeking such access." *Id.* at 1518. The court agreed with Accolade that its copying was necessary to examine the unprotected functional aspects of the program. *Id.* at 1520. And, because Accolade had a legitimate interest in making its cartridges compatible with Sega's console, the court found that Accolade's intermediate copying was fair use.

Likewise, in *Sony*, the Ninth Circuit found that the defendant's reverse engineering and intermediate copying of Sony's copyrighted software program "was a fair use for the purpose of gaining access to the unprotected elements of Sony's software." *Sony*, 203 F.3d at 602. The court explained that Sony's software program contained unprotected functional elements and that the defendant could only access those elements through reverse engineering. *Id.* at 603. The defendant used that information to create a software program that let consumers play games designed for Sony's PlayStation

console on their computers. Notably, the defendant's software program did not contain any of Sony's copyrighted material. *Id.* at 598.

The district court characterized *Sony* and *Sega* as "close analogies" to this case. *Copyrightability Decision*, 872 F. Supp. 2d at 1000. According to the court, both decisions "held that interface procedures that were necessary to duplicate in order to achieve interoperability were functional aspects not copyrightable under Section 102(b)." *Id.* The district court's reliance on *Sega* and *Sony* in the copyrightability context is misplaced, however.

As noted, both cases were focused on fair use, not copyrightability. In *Sega*, for example, the only question was whether Accolade's intermediate copying was fair use. The court never addressed the question of whether Sega's software code, which had functional elements, also contained separable creative expression entitled to protection. Likewise, although the court in *Sony* determined that Sony's computer program had functional elements, it never addressed whether it also had expressive elements. *Sega* and *Sony* are also factually distinguishable because the defendants in those cases made intermediate copies to understand the functional aspects of the copyrighted works and then created new products. *See Sony*, 203 F.3d at 606–07; *Sega*, 977 F.2d at 1522–23. This is not a case where Google reverse-engineered Oracle's Java packages to gain access to unprotected functional elements contained therein. As the former Register of Copyrights of the United States pointed out in his brief amicus curiae, "[h]ad Google reverse engineered the programming packages to figure out the ideas and functionality of the original, and then created its own structure and its own literal code, Oracle

would have no remedy under copyright whatsoever." Br. for Amicus Curiae Ralph Oman 29. Instead, Google chose to copy both the declaring code and the overall SSO of the 37 Java API packages at issue.

We disagree with Google's suggestion that *Sony* and *Sega* created an "interoperability exception" to copyrightability. *See* Appellee Br. 39 (citing *Sony* and *Sega* for the proposition that "compatibility elements are not copyrightable under section 102(b)" (emphasis omitted)). Although both cases recognized that the software programs at issue there contained unprotected functional elements, a determination that some elements are unprotected is not the same as saying that the entire work loses copyright protection. To accept Google's reading would contradict Ninth Circuit case law recognizing that both the literal and non-literal components of a software program are eligible for copyright protection. *See Johnson Controls*, 886 F.2d at 1175. And it would ignore the fact that the Ninth Circuit endorsed the abstraction-filtration-comparison inquiry in *Sega* itself.

As previously discussed, a court must examine the software program to determine whether it contains creative expression that can be separated from the underlying function. *See Sega*, 977 F.2d at 1524–25. In doing so, the court filters out the elements of the program that are "ideas" as well as elements that are "dictated by considerations of efficiency, so as to be necessarily incidental to that idea; required by factors external to the program itself." *Altai*, 982 F.2d at 707.

To determine "whether certain aspects of an allegedly infringed software are not protected by copyright law, the focus is on external factors that influenced the choice of

the creator of the infringed product." *Dun & Bradstreet Software Servs., Inc. v. Grace Consulting, Inc.*, 307 F.3d 197, 215 (3d Cir. 2002) (citing *Altai*, 982 F.2d at 714; *Mitel*, 124 F.3d at 1375). The Second Circuit, for example, has noted that programmers are often constrained in their design choices by "extrinsic considerations" including "the mechanical specifications of the computer on which a particular program is intended to run" and "compatibility requirements of other programs with which a program is designed to operate in conjunction." *Altai*, 982 F.2d at 709–10 (citing 3 Melville B. Nimmer & David Nimmer, Nimmer on Copyright § 13.01 at 13-66-71 (1991)). The Ninth Circuit has likewise recognized that: (1) computer programs "contain many logical, structural, and visual display elements that are dictated by . . . external factors such as compatibility requirements and industry demands"; and (2) "[i]n some circumstances, even the exact set of commands used by the programmer is deemed functional rather than creative for purposes of copyright." *Sega*, 977 F.2d at 1524 (internal citation omitted).

Because copyrightability is focused on the choices available to the plaintiff at the time the computer program was created, the relevant compatibility inquiry asks whether the plaintiff's choices were dictated by a need to ensure that its program worked with existing third-party programs. *Dun & Bradstreet*, 307 F.3d at 215; *see also Atari*, 975 F.2d at 840 ("External factors did not dictate the design of the 10NES program."). Whether a defendant later seeks to make its program interoperable with the plaintiff's program has no bearing on whether the software the plaintiff created had any design limitations dictated by external factors. *See Dun & Bradstreet*, 307 F.3d at 215 (finding an expert's testimony on

interoperability "wholly misplaced" because he "looked at externalities from the eyes of the plagiarist, not the eyes of the program's creator"). Stated differently, the focus is on the compatibility needs and programming choices of the party claiming copyright protection—not the choices the defendant made to achieve compatibility with the plaintiff's program. Consistent with this approach, courts have recognized that, once the plaintiff creates a copyrightable work, a defendant's desire "to achieve total compatibility . . . is a commercial and competitive objective which does not enter into the . . . issue of whether particular ideas and expressions have merged." *Apple Computer*, 714 F.2d at 1253.

Given this precedent, we conclude that the district court erred in focusing its interoperability analysis on Google's desires for its Android software. *See Copyrightability Decision*, 872 F. Supp. 2d at 1000 ("Google replicated what was necessary to achieve a degree of interoperability" with Java.). Whether Google's software is "interoperable" in some sense with any aspect of the Java platform (although as Google concedes, certainly not with the JVM) has no bearing on the threshold question of whether Oracle's software is copyrightable. It is the interoperability and other needs of Oracle—not those of Google—that apply in the copyrightability context, and there is no evidence that when Oracle created the Java API packages at issue it did so to meet compatibility requirements of other pre-existing programs.

Google maintains on appeal that its use of the "Java class and method names and declarations was 'the only and essential means' of achieving a degree of interoperability with existing programs written in the

[Java language]." Appellee Br. 49. Indeed, given the record evidence that Google designed Android so that it would not be compatible with the Java platform, or the JVM specifically, we find Google's interoperability argument confusing. While Google repeatedly cites to the district court's finding that Google had to copy the packages so that an app written in Java could run on Android, it cites to no evidence in the record that any such app exists and points to no Java apps that either pre-dated or post-dated Android that could run on the Android platform.[15] The compatibility Google sought to foster was not with Oracle's Java platform or with the JVM central to that platform. Instead, Google wanted to capitalize on the fact that software developers were already trained and experienced in using the Java API packages at issue. The district court agreed, finding that, as to the 37 Java API packages, "Google believed Java application programmers would want to find the same 37 sets of functionalities in the new Android system callable by the same names as used in Java." *Copyrightability Decision*, 872 F. Supp. 2d at 978. Google's interest was in accelerating its development process by "leverag[ing] Java for its existing base of developers." J.A. 2033, 2092. Although this competitive objective might be relevant to the fair use inquiry, we conclude that it is irrelevant to the

[15] During oral argument, Google's counsel stated that "a program written in the Java language can run on Android if it's only using packages within the 37. So if I'm a developer and I have written a program, I've written it in Java, I can stick an Android header on it and it will run in Android because it is using the identical names of the classes, methods, and packages." Oral Argument at 31:31. Counsel did not identify any programs that use only the 37 API packages at issue, however, and did not attest that any such program would be useful. Nor did Google cite to any record evidence to support this claim.

copyrightability of Oracle's declaring code and organization of the API packages.

Finally, to the extent Google suggests that it was entitled to copy the Java API packages because they had become the effective industry standard, we are unpersuaded. Google cites no authority for its suggestion that copyrighted works lose protection when they become popular, and we have found none.[16] In fact, the Ninth Circuit has rejected the argument that a work that later becomes the industry standard is uncopyrightable. *See Practice Mgmt. Info. Corp. v. Am. Med. Ass'n*, 121 F.3d 516, 520 n.8 (9th Cir. 1997) (noting that the district court found plaintiff's medical coding system entitled to copyright protection, and that, although the system had become the industry standard, plaintiff's copyright did not prevent competitors "from developing comparative or better coding systems and lobbying the federal government and private actors to adopt them. It simply prevents wholesale copying of an existing system.").

---

[16] Google argues that, in the same way a formerly distinctive trademark can become generic over time, a program element can lose copyright protection when it becomes an industry standard. But "it is to be expected that phrases and other fragments of expression in a highly successful copyrighted work will become part of the language. That does not mean they lose all protection in the manner of a trade name that has become generic." *Warner Bros., Inc. v. Am. Broadcasting Cos.*, 720 F.2d 231, 242 (2d Cir. 1983) ("No matter how well known a copyrighted phrase becomes, its author is entitled to guard against its appropriation to promote the sale of commercial products."). Notably, even when a patented method or system becomes an acknowledged industry standard with acquiescence of the patent owner, any permissible use generally requires payment of a reasonable royalty, which Google refused to do here. *See generally In re Innovatio IP Ventures, LLC*, No. 11-C-9308, 2013 U.S. Dist. LEXIS 144061 (N.D. Ill. Sept. 27, 2013).

Google was free to develop its own API packages and to "lobby" programmers to adopt them. Instead, it chose to copy Oracle's declaring code and the SSO to capitalize on the preexisting community of programmers who were accustomed to using the Java API packages. That desire has nothing to do with copyrightability. For these reasons, we find that Google's industry standard argument has no bearing on the copyrightability of Oracle's work.

B. Fair Use

As noted, the jury hung on Google's fair use defense, and the district court declined to order a new trial given its conclusion that the code and structure Google copied were not entitled to copyright protection. On appeal, Oracle argues that: (1) a remand to decide fair use "is pointless"; and (2) this court should find, as a matter of law, that "Google's commercial use of Oracle's work in a market where Oracle already competed was not fair use." Appellant Br. 68.

Fair use is an affirmative defense to copyright infringement and is codified in Section 107 of the Copyright Act. *Golan*, 132 S. Ct. at 890 ("[T]he fair use defense, is codified at 17 U.S.C. §107."). Section 107 permits use of copyrighted work if it is "for purposes such as criticism, comment, news reporting, teaching (including multiple copies for classroom use), scholarship, or research." 17 U.S.C. § 107. The fair use doctrine has been referred to as "'the most troublesome in the whole law of copyright.'" *Monge v. Maya Magazines, Inc.*, 688 F.3d 1164, 1170 (9th Cir. 2012) (quoting *Dellar v. Samuel Goldwyn, Inc.*, 104 F.2d 661, 662 (2d Cir. 1939) (per curiam)). It both permits and requires "courts to avoid rigid application of the copyright statute when, on occasion, it would stifle the very creativity which that law

is designed to foster." *Campbell v. Acuff-Rose Music, Inc.*, 510 U.S. 569, 577 (1994) (quoting *Stewart v. Abend*, 495 U.S. 207, 236 (1990)).

"Section 107 requires a case-by-case determination whether a particular use is fair, and the statute notes four nonexclusive factors to be considered." *Harper & Row Publishers, Inc. v. Nation Enters.*, 471 U.S. 539, 549 (1985). Those factors are: (1) "the purpose and character of the use, including whether such use is of a commercial nature or is for nonprofit educational purposes;" (2) "the nature of the copyrighted work;" (3) "the amount and substantiality of the portion used in relation to the copyrighted work as a whole;" and (4) "the effect of the use upon the potential market for or value of the copyrighted work." 17 U.S.C. § 107. The Supreme Court has explained that all of the statutory factors "are to be explored, and the results weighed together, in light of the purpose[] of copyright," which is "[t]o promote the Progress of Science and useful Arts." *Campbell*, 510 U.S. at 578, 575 (internal citations omitted).

"Fair use is a mixed question of law and fact." *Harper & Row*, 471 U.S. at 560. Thus, while subsidiary and controverted findings of fact must be reviewed for clear error under Rule 52 of the Federal Rules of Civil Procedure, the Ninth Circuit reviews the ultimate application of those facts de novo. *See Seltzer v. Green Day, Inc.*, 725 F.3d 1170, 1175 (9th Cir. 2013) (citing *SOFA Entm't, Inc. v. Dodger Prods., Inc.*, 709 F.3d 1273, 1277 (9th Cir. 2013)). Where there are no material facts at issue and "the parties dispute only the ultimate conclusions to be drawn from those facts, we may draw those conclusions without usurping the function of the jury." *Id.* (citing *Fisher v. Dees*, 794 F.2d 432, 436 (9th Cir. 1986)). Indeed,

the Supreme Court has specifically recognized that, "[w]here the district court has found facts sufficient to evaluate each of the statutory factors, an appellate court 'need not remand for further factfinding . . . [but] may conclude as a matter of law that [the challenged use] [does] not qualify as a fair use of the copyrighted work.'" *Harper & Row*, 471 U.S. at 560 (citation omitted).

Of course, the corollary to this point is true as well—where there are material facts in dispute and those facts have not yet been resolved by the trier of fact, appellate courts may not make findings of fact in the first instance. *See Shawmut Bank, N.A. v. Kress Assocs.*, 33 F.3d 1477, 1504 (9th Cir. 1994) ("[W]e must avoid finding facts in the first instance."); *see also Golden Bridge Tech., Inc. v. Nokia, Inc.*, 527 F.3d 1318, 1323 (Fed. Cir. 2008) ("Appellate courts review district court judgments; we do not find facts."). Here, it is undisputed that neither the jury nor the district court made findings of fact to which we can refer in assessing the question of whether Google's use of the API packages at issue was a "fair use" within the meaning of Section 107. Oracle urges resolution of the fair use question by arguing that the trial court should have decided the question as a matter of law based on the undisputed facts developed at trial, and that we can do so as well. Google, on the other hand, argues that many critical facts regarding fair use are in dispute. It asserts that the fact that the jury could not reach a resolution on the fair use defense indicates that at least some presumably reasonable jurors found its use to be fair. And, Google asserts that, even if it is true that the district court erred in discussing concepts of "interoperability" when considering copyrightability, those concepts are still

relevant to its fair use defense. We turn first to a more detailed examination of fair use.

The first factor in the fair use inquiry involves "the purpose and character of the use, including whether such use is of a commercial nature or is for nonprofit educational purposes." 17 U.S.C. § 107(1). This factor involves two sub-issues: (1) "whether and to what extent the new work is transformative," *Campbell*, 510 U.S. at 579 (citation and internal quotation marks omitted); and (2) whether the use serves a commercial purpose.

A use is "transformative" if it "adds something new, with a further purpose or different character, altering the first with new expression, meaning or message." *Id.* The critical question is "whether the new work merely supersede[s] the objects of the original creation . . . or instead adds something new." *Id.* (citations and internal quotation marks omitted). This inquiry "may be guided by the examples given in the preamble to § 107, looking to whether the use is for criticism, or comment, or news reporting, and the like." *Id.* at 578–79. "The Supreme Court has recognized that parodic works, like other works that comment and criticize, are by their nature often sufficiently transformative to fit clearly under the fair use exception." *Mattel Inc. v. Walking Mountain Prods.*, 353 F.3d 792, 800 (9th Cir. 2003) (citing *Campbell*, 510 U.S. at 579).

Courts have described new works as "transformative" when "the works use copy-righted material for purposes distinct from the purpose of the original material." *Elvis Presley Enters., Inc. v. Passport Video*, 349 F.3d 622, 629 (9th Cir. 2003) ("Here, Passport's use of many of the television clips is transformative because they are cited as historical reference points in the life of a remarkable

entertainer."), *overruled on other grounds by Flexible Lifeline Sys., Inc. v. Precision Lift, Inc.*, 654 F.3d 989, 995 (9th Cir. 2011) (per curiam); *see also Bouchat v. Baltimore Ravens Ltd. P'ship*, 619 F.3d 301, 309–10 (4th Cir. 2010) (quoting *A.V. ex rel. Vanderhyge v. iParadigms, LLC*, 562 F.3d 630, 638 (4th Cir. 2009) ("[A] transformative use is one that 'employ[s] the quoted matter in a different manner or for a different purpose from the original.'")). "A use is considered transformative only where a defendant changes a plaintiff's copyrighted work or uses the plaintiff's copyrighted work in a different context such that the plaintiff's work is transformed into a new creation." *Perfect 10, Inc. v. Amazon.com, Inc.*, 508 F.3d 1146, 1165 (9th Cir. 2007) (quoting *Wall Data Inc. v. L.A. County Sheriff's Dep't*, 447 F.3d 769, 778 (9th Cir. 2006), and finding that Google's use of thumbnail images in its search engine was "highly transformative").

A work is not transformative where the user "makes no alteration to the *expressive content or message* of the original work." *Seltzer*, 725 F.3d at 1177; *see also Wall Data*, 447 F.3d at 778 ("The Sheriff's Department created exact copies of RUMBA's software. It then put those copies to the identical purpose as the original software. Such a use cannot be considered transformative."); *Monge*, 688 F.3d at 1176 (finding that a magazine's publication of photographs of a secret celebrity wedding "sprinkled with written commentary" was "at best minimally transformative" where the magazine "did not transform the photos into a new work . . . or incorporate the photos as part of a broader work"); *Elvis Presley Enters.*, 349 F.3d at 629 (finding that use of copyrighted clips of Elvis's television appearances was not transformative where "some of the clips [we]re played

without much interruption, if any . . . [and] instead serve[d] the same intrinsic entertainment value that is protected by Plaintiffs' copyrights."). Where the use "is for the same intrinsic purpose as [the copyright holder's] . . . such use seriously weakens a claimed fair use." *Worldwide Church of God v. Phila. Church of God, Inc.*, 227 F.3d 1110, 1117 (9th Cir. 2000) (quoting *Weissmann v. Freeman*, 868 F.2d 1313, 1324 (2d Cir. 1989)).

Analysis of the first factor also requires inquiry into the commercial nature of the use. Use of the copyrighted work that is commercial "tends to weigh against a finding of fair use." *Harper & Row*, 471 U.S. at 562 ("The crux of the profit/nonprofit distinction is not whether the sole motive of the use is monetary gain but whether the user stands to profit from exploitation of the copyrighted material without paying the customary price."). "[T]he more transformative the new work, the less will be the significance of other factors, like commercialism, that may weigh against a finding of fair use." *Campbell*, 510 U.S. at 579.

The second factor—the nature of the copyrighted work—"calls for recognition that some works are closer to the core of intended copyright protection than others, with the consequence that fair use is more difficult to establish when the former works are copied." *Id.* at 586. This factor "turns on whether the work is informational or creative." *Worldwide Church of God*, 227 F.3d at 1118; *see also Harper & Row*, 471 U.S. at 563 ("The law generally recognizes a greater need to disseminate factual works than works of fiction or fantasy."). Creative expression "falls within the core of the copyright's protective purposes." *Campbell*, 510 U.S. at 586. Because computer programs have both functional and expressive

components, however, where the functional components are themselves unprotected (because, e.g., they are dictated by considerations of efficiency or other external factors), those elements should be afforded "a lower degree of protection than more traditional literary works." *Sega*, 977 F.2d at 1526. Thus, where the nature of the work is such that purely functional elements exist in the work and it is necessary to copy the expressive elements in order to perform those functions, consideration of this second factor arguably supports a finding that the use is fair.

The third factor asks the court to examine "the amount and substantiality of the portion used in relation to the copyrighted work as a whole." 17 U.S.C. § 107(3). Analysis of this factor is viewed in the context of the copyrighted work, not the infringing work. Indeed, the statutory language makes clear that "a taking may not be excused merely because it is insubstantial with respect to the *infringing* work." *Harper & Row*, 471 U.S. at 565. "As Judge Learned Hand cogently remarked, 'no plagiarist can excuse the wrong by showing how much of his work he did not pirate.'" *Id.* (quoting *Sheldon v. Metro-Goldwyn Pictures Corp.*, 81 F.2d 49, 56 (2d Cir. 1936)). In contrast, "the fact that a substantial portion of the infringing work was copied verbatim is evidence of the qualitative value of the copied material, both to the originator and to the plagiarist who seeks to profit from marketing someone else's copyrighted expression." *Id.* The Ninth Circuit has recognized that, while "wholesale copying does not preclude fair use per se, copying an entire work militates against a finding of fair use." *Worldwide Church of God*, 227 F.3d at 1118 (internal citation and quotation omitted). "If the secondary user only copies as much as is necessary

for his or her intended use, then this factor will not weigh against him or her." *Kelly v. Arriba Soft Corp.*, 336 F.3d 811, 820–21 (9th Cir. 2003). Under this factor, "attention turns to the persuasiveness of a parodist's justification for the particular copying done, and the enquiry will harken back to the first of the statutory factors . . . [because] the extent of permissible copying varies with the purpose and character of the use." *Campbell*, 510 U.S. at 586–87.

The fourth and final factor focuses on "the effect of the use upon the potential market for or value of the copyrighted work." *Harper & Row*, 471 U.S. at 566. This factor reflects the idea that fair use "is limited to copying by others which does not materially impair the marketability of the work which is copied." *Id.* at 566–67. The Supreme Court has said that this factor is "undoubtedly the single most important element of fair use." *Id.* at 566. It requires that courts "consider not only the extent of market harm caused by the particular actions of the alleged infringer, but also whether unrestricted and widespread conduct of the sort engaged in by the defendant . . . would result in a substantially adverse impact on the potential market for the original." *Campbell*, 510 U.S. at 590 (citation and quotation marks omitted). "Market harm is a matter of degree, and the importance of this factor will vary, not only with the amount of harm, but also with the relative strength of the showing on the other factors." *Id.* at 590 n.21.

Oracle asserts that all of these factors support its position that Google's use was not "fair use"—Google knowingly and illicitly copied a creative work to further its own commercial purposes, did so verbatim, and did so to the detriment of Oracle's market position. These undisputable facts, according to Oracle, should end the

fair use inquiry. Oracle's position is not without force. On many of these points, Google does not debate Oracle's characterization of its conduct, nor could it on the record evidence.

Google contends, however, that, although it admittedly copied portions of the API packages and did so for what were purely commercial purposes, a reasonable juror still could find that: (1) Google's use was transformative; (2) the Java API packages are entitled only to weak protection; (3) Google's use was necessary to work within a language that had become an industry standard; and (4) the market impact on Oracle was not substantial.

On balance, we find that due respect for the limit of our appellate function requires that we remand the fair use question for a new trial. First, although it is undisputed that Google's use of the API packages is commercial, the parties disagree on whether its use is "transformative." Google argues that it is, because it wrote its own implementing code, created its own virtual machine, and incorporated the packages into a smartphone platform. For its part, Oracle maintains that Google's use is not transformative because: (1) "[t]he same code in Android . . . enables programmers to invoke the same pre-programmed functions in exactly the same way;" and (2) Google's use of the declaring code and packages does not serve a different function from Java. Appellant Reply Br. 47. While Google overstates what activities can be deemed transformative under a correct application of the law, we cannot say that there are no material facts in dispute on the question of whether Google's use is "transformative," even under a correct reading of the law. As such, we are unable to resolve this issue on appeal.

Next, while we have concluded that it was error for the trial court to focus *unduly* on the functional aspects of the packages, and on Google's competitive desire to achieve commercial "interoperability" when deciding whether Oracle's API packages are entitled to copyright protection, we expressly noted that these factors may be relevant to a fair use analysis. While the trial court erred in concluding that these factors were sufficient to overcome Oracle's threshold claim of copyrightability, reasonable jurors might find that they are relevant to Google's fair use defense under the second and third factors of the inquiry. *See Sega*, 977 F.2d at 1524–25 (discussing the Second Circuit's approach to "break[ing] down a computer program into its component subroutines and sub-subroutines and then identif[ying] the idea or core functional element of each" in the context of the second fair use factor: the nature of the copyrighted work). We find this particularly true with respect to those core packages which it seems may be necessary for anyone to copy if they are to write programs in the Java language. And, it may be that others of the packages were similarly essential components of any Java language-based program. So far, that type of filtration analysis has not occurred.

Finally, as to market impact, the district court found that "Sun and Oracle never successfully developed its own smartphone platform using Java technology." *Copyrightability Decision*, 872 F. Supp. 2d at 978. But Oracle argues that, when Google copied the API packages, Oracle was licensing in the mobile and smartphone markets, and that Android's release substantially harmed those commercial opportunities as well as the potential market for a Java smartphone device. Because there are

material facts in dispute on this factor as well, remand is necessary.

Ultimately, we conclude that this is not a case in which the record contains sufficient factual findings upon which we could base a de novo assessment of Google's affirmative defense of fair use. Accordingly, we remand this question to the district court for further proceedings. On remand, the district court should revisit and revise its jury instructions on fair use consistent with this opinion so as to provide the jury with a clear and appropriate picture of the fair use defense.[17]

II. GOOGLE'S CROSS-APPEAL

Google cross-appeals from the portion of the district court's final judgment entered in favor of Oracle on its

---

[17] Google argues that, if we allow it to retry its fair use defense on remand, it is entitled to a retrial on infringement as well. We disagree. The question of whether Google's copying constituted infringement of a copyrighted work is "distinct and separable" from the question of whether Google can establish a fair use defense to its copying. *See Gasoline Prods. Co. v. Champlin Refining Co.*, 283 U.S. 494, 500 (1931) ("Where the practice permits a partial new trial, it may not properly be resorted to unless it clearly appears that the issue to be retried is so distinct and separable from the others that a trial of it alone may be had without injustice."). Indeed, we have emphasized more than once in this opinion the extent to which the questions are separable, and the confusion and error caused when they are blurred. The issues are not "interwoven" and it would not create "confusion and uncertainty" to reinstate the infringement verdict and submit fair use to a different jury. *Id.* We note, moreover, that, because Google only mentions this point in passing, with no development of an argument in support of it, under our case law, it has not been properly raised. *See SmithKline Beecham Corp. v. Apotex Corp.*, 439 F.3d 1312, 1320 (Fed. Cir. 2006) (when a party provides no developed argument on a point, we treat that argument as waived) (collecting cases).

claim for copyright infringement as to the nine lines of rangeCheck code and the eight decompiled files. Final Judgment, *Oracle Am., Inc. v. Google Inc.*, No. 3:10-cv-3561 (N.D. Cal. June 20, 2012), ECF No. 1211. Specifically, Google appeals from the district court's decisions: (1) granting Oracle's motion for JMOL of infringement as to the eight decompiled Java files that Google copied into Android; and (2) denying Google's motion for JMOL with respect to rangeCheck.

When reviewing a district court's grant or denial of a motion for JMOL, we apply the procedural law of the relevant regional circuit, here the Ninth Circuit. *Trading Techs. Int'l, Inc. v. eSpeed, Inc.*, 595 F.3d 1340, 1357 (Fed. Cir. 2010). The Ninth Circuit reviews a district court's JMOL decision de novo, applying the same standard as the district court. *Mangum v. Action Collection Serv., Inc.*, 575 F.3d 935, 938 (9th Cir. 2009). To grant judgment as a matter of law, the court must find that "the evidence presented at trial permits only one reasonable conclusion" and that "no reasonable juror could find in the non-moving party's favor." *Id.* at 938–39 (citation and internal quotation marks omitted).

Oracle explains that the eight decompiled files at issue "contain security functions governing access to network files" while rangeCheck "facilitates an important sorting function, frequently called upon during the operation of Java and Android." Oracle Response to Cross-Appeal 60–61. At trial, Google conceded that it copied the eight decompiled Java code files and the nine lines of code referred to as rangeCheck into Android. Its only defense was that the copying was de minimis. Accordingly, the district court instructed the jury that, "[w]ith respect to the infringement issues concerning the rangeCheck and

other similar files, Google agrees that the accused lines of code and comments came from the copyrighted materials but contends that the amounts involved were so negligible as to be de minimis and thus should be excluded." Final Charge to the Jury (Phase One), *Oracle Am., Inc. v. Google, Inc.*, No. 3:10-cv-3561 (N.D. Cal. Apr. 30, 2012), ECF No. 1018, at 14.

Although the jury found that Google infringed Oracle's copyright in the nine lines of code comprising rangeCheck, it returned a noninfringement verdict as to eight decompiled security files. But because the trial testimony was that Google's use of the decompiled files was significant—and there was no testimony to the contrary—the district court concluded that "[n]o reasonable jury could find that this copying was de minimis." *Order Granting JMOL on Decompiled Files*, 2012 U.S. Dist. LEXIS 66417, at *6. As such, the court granted Oracle's motion for JMOL of infringement as to the decompiled security files.

On appeal, Google maintains that its copying of rangeCheck and the decompiled security files was de minimis and thus did not infringe any of Oracle's copyrights. According to Google, the district court should have denied Oracle's motion for JMOL "because substantial evidence supported the jury's verdict that Google's use of eight decompiled test files was de minimis." Cross-Appellant Br. 76. Google further argues that the court should have granted its motion for JMOL as to rangeCheck because the "trial evidence revealed that the nine lines of rangeCheck code were both quantitatively and qualitatively insignificant in relation to the [Java] platform." *Id.* at 78.

In response, Oracle argues that the Ninth Circuit does not recognize a de minimis defense to copyright infringement and that, even if it does, we should affirm the judgments of infringement on grounds that Google's copying was significant. Because we agree with Oracle on its second point, we need not address the first, except to note that there is some conflicting Ninth Circuit precedent on the question of whether there is a free-standing de minimis defense to copyright infringement or whether the substantiality of the alleged copying is best addressed as part of a fair use defense. *Compare Norse v. Henry Holt & Co.*, 991 F.2d 563, 566 (9th Cir. 1993) (indicating that "even a small taking may sometimes be actionable" and the "question of whether a copying is substantial enough to be actionable may be best resolved through the fair use doctrine"), *with Newton v. Diamond*, 388 F.3d 1189, 1192–93 (9th Cir. 2003) ("For an unauthorized use of a copyrighted work to be actionable, the use must be significant enough to constitute infringement. This means that even where the fact of copying is conceded, no legal consequences will follow from that fact unless the copying is substantial.") (internal citation omitted)).[18]

Even assuming that the Ninth Circuit recognizes a stand-alone de minimis defense to copyright infringement, however, we conclude that: (1) the jury reasonably found

---

[18] At least one recent district court decision has recognized uncertainty in Ninth Circuit law on this point. *See Brocade Commc'ns Sys. v. A10 Networks, Inc.*, No. 10-cv-3428, 2013 U.S. Dist. LEXIS 8113, at *33 (N.D. Cal. Jan. 10, 2013) ("The Ninth Circuit has been unclear about whether the de minimis use doctrine serves as an affirmative defense under the Copyright Act's fair use exceptions or whether the doctrine merely highlights plaintiffs' obligation to show that 'the use must be significant enough to constitute infringement.'") (citing *Newton*, 388 F.2d at 1193; *Norse*, 991 F.2d at 566).

that Google's copying of the rangeCheck files was more than de minimis; and (2) the district court correctly concluded that the defense failed as a matter of law with respect to the decompiled security files.

First, the unrebutted testimony at trial revealed that rangeCheck and the decompiled security files were significant to both Oracle and Google. Oracle's expert, Dr. John Mitchell, testified that Android devices call the rangeCheck function 2,600 times just in powering on the device. Although Google argues that the eight decompiled files were insignificant because they were used only to test the Android platform, Dr. Mitchell testified that "using the copied files even as test files would have been significant use" and the district court specifically found that "[t]here was no testimony to the contrary." *Order Granting JMOL on Decompiled Files*, 2012 U.S. Dist. LEXIS 66417, at *6. Given this testimony, a reasonable jury could not have found Google's copying de minimis.

Google emphasizes that the nine lines of rangeCheck code "represented an infinitesimal percentage of the 2.8 million lines of code in the 166 Java packages—let alone the millions of lines of code in the entire [Java] platform." Google Cross-Appeal Br. 78–79. To the extent Google is arguing that a certain minimum number of lines of code must be copied before a court can find infringement, that argument is without merit. *See Baxter v. MCA, Inc.*, 812 F.2d 421, 425 (9th Cir. 1987) ("[N]o bright line rule exists as to what quantum of similarity is permitted."). And, given the trial testimony that both rangeCheck and the decompiled security files are qualitatively significant and Google copied them in their entirety, Google cannot show that the district court erred in denying its motion for JMOL.

We have considered Google's remaining arguments and find them unpersuasive. Accordingly, we affirm both of the JMOL decisions at issue in Google's cross-appeal.

III. GOOGLE'S POLICY-BASED ARGUMENTS

Many of Google's arguments, and those of some amici, appear premised on the belief that copyright is not the correct legal ground upon which to protect intellectual property rights to software programs; they opine that patent protection for such programs, with its insistence on non-obviousness, and shorter terms of protection, might be more applicable, and sufficient. Indeed, the district court's method of operation analysis seemed to say as much. *Copyrightability Decision*, 872 F. Supp. 2d at 984 (stating that this case raises the question of "whether the copyright holder is more appropriately asserting an exclusive right to a functional system, process, or method of operation that belongs in the realm of patents, not copyrights"). Google argues that "[a]fter *Sega*, developers could no longer hope to protect [software] interfaces by copyright . . . *Sega* signaled that the only reliable means for protecting the functional requirements for achieving interoperability was by patenting them." Appellee Br. 40 (quoting Pamela Samuelson, *Are Patents on Interfaces Impeding Interoperability*? 93 Minn. L. Rev. 1943, 1959 (2009)). And, Google relies heavily on articles written by Professor Pamela Samuelson, who has argued that "it would be best for a commission of computer program experts to draft a new form of intellectual property law for machine-readable programs." Pamela Samuelson, *CONTU Revisited: The Case Against Copyright Protection for Computer Programs in Machine-Readable Form*, 1984 Duke L.J. 663, 764 (1984). Professor Samuelson has more recently argued that "*Altai* and *Sega*

contributed to the eventual shift away from claims of copyright in program interfaces and toward reliance on patent protection. Patent protection also became more plausible and attractive as the courts became more receptive to software patents." Samuelson, 93 Minn. L. Rev. at 1959.

Although Google, and the authority on which it relies, seem to suggest that software is or should be entitled to protection only under patent law—not copyright law—several commentators have recently argued the exact opposite. *See* Technology Quarterly, *Stalking Trolls*, ECONOMIST, Mar. 8, 2014, http://www.economist.com/news/technology-quarterly/21598321-intellectual-property-after-being-blamed-stymying-innovation-america-vague ("[M]any innovators have argued that the electronics and software industries would flourish if companies trying to bring new technology (software innovations included) to market did not have to worry about being sued for infringing thousands of absurd patents at every turn. A perfectly adequate means of protecting and rewarding software developers for their ingenuity has existed for over 300 years. It is called copyright."); Timothy B. Lee, *Will the Supreme Court save us from software patents?*, WASH. POST, Feb. 26, 2014, 1:13 PM, http://www.washingtonpost.com/blogs/the-switch/wp/2014/02/26/will-the-supreme-court-save-us-from-softwarepatents/ ("If you write a book or a song, you can get copyright protection for it. If you invent a new pill or a better mousetrap, you can get a patent on it. But for the last two decades, software has had the distinction of being potentially eligible for both copyright and patent protection. Critics say that's a mistake. They argue that the complex and expensive patent system is a terrible fit

for the fast-moving software industry. And they argue that patent protection is unnecessary because software innovators already have copyright protection available.").

Importantly for our purposes, the Supreme Court has made clear that "[n]either the Copyright Statute nor any other says that because a thing is patentable it may not be copyrighted." *Mazer v. Stein*, 347 U.S. 201, 217 (1954). Indeed, the thrust of the CONTU Report is that copyright is "the most suitable mode of legal protection for computer software." Peter S. Menell, *An Analysis of the Scope of Copyright Protection for Application Programs*, 41 Stan. L. Rev. 1045, 1072 (1989); *see also* CONTU Report at 1 (recommending that copyright law be amended "to make it explicit that computer programs, to the extent that they embody an author's original creation, are proper subject matter of copyright"). Until either the Supreme Court or Congress tells us otherwise, we are bound to respect the Ninth Circuit's decision to afford software programs protection under the copyright laws. We thus decline any invitation to declare that protection of software programs should be the domain of patent law, and only patent law.

CONCLUSION

For the foregoing reasons, we conclude that the declaring code and the structure, sequence, and organization of the 37 Java API packages at issue are entitled to copyright protection. We therefore reverse the district court's copyrightability determination with instructions to reinstate the jury's infringement verdict. Because the jury hung on fair use, we remand Google's fair use defense for further proceedings consistent with this decision.

With respect to Google's cross-appeal, we affirm the district court's decisions: (1) granting Oracle's motion for JMOL as to the eight decompiled Java files that Google copied into Android; and (2) denying Google's motion for JMOL with respect to the rangeCheck function. Accordingly, we affirm-in-part, reverse-in-part, and remand for further proceedings.

AFFIRMED-IN-PART, REVERSED-IN-PART, AND REMANDED

Appendix E

IN THE UNITED STATES DISTRICT COURT FOR THE NORTHERN DISTRICT OF CALIFORNIA

ORACLE AMERICA, INC., Plaintiff, v. GOOGLE INC., Defendant.	No. C 10-03561 WHA

ORDER PARTIALLY GRANTING AND PARTIALLY DENYING DEFENDANT'S MOTION FOR SUMMARY JUDGMENT ON COPYRIGHT CLAIM

INTRODUCTION

In this patent and copyright infringement action involving features of Java and Android, defendant moves for summary judgment on the copyright infringement claim. With one exception described below, the motion is DENIED.

STATEMENT

Oracle America Inc. accuses Google Inc. of infringing some of Oracle's Java-related copyrights in portions of Google's Android software platform. Specifically, Oracle accuses twelve code files and 37 specifications for application programming interface packages. The Java technology and the basics of object-oriented programming were explained in the claim construction order (Dkt. No.

137). An overview of application programming interfaces and their role in Java and Android is provided here.

1. APPLICATION PROGRAMMING INTERFACES (APIS).

Conceptually, an API is what allows software programs to communicate with one another. It is a set of definitions governing how the services of a particular program can be called upon, including what types of input the program must be given and what kind of output will be returned. APIs make it possible for programs (and programmers) to use the services of a given program without knowing *how* the service is performed. APIs also insulate programs from one another, making it possible to change the way a given program performs a service without disrupting other programs that use the service.

APIs typically are composed of "methods," also known as "functions," which are software programs that perform particular services. For example, a programmer might write a software program method *A*, which calculates the area of a room when given the shape and dimensions of the room. A second programmer then could write a program method called *B*, which calculates the square footage of an entire house when given the shape and dimensions of each room. Rather than reinventing a new way to calculate area, the second programmer could simply write an instruction in *B*, "for each room, ask program A to calculate the area; then add all of the return values," using, of course, real programming language. As long as the second programmer knows what *A* is named, what type of "arguments" *A* must be given as inputs, and what return A outputs, the second programmer can write a program that will call on the services of *A*. The second programmer does not need to know how A actually works, or is "implemented." There may in fact be multiple ways

to implement *A* — for example, different ways to divide an oddly shaped room into geometric components — and the first programmer may refine his implementation of program A without disrupting program *B*.

A method must be defined before it can be used. A method can be "declared" (*i.e.*, defined) in a programming language such as Java by stating its name and describing its argument(s) and return(s) according to syntax conventions. Once a method has been declared, it can documented and implemented. *Documentation* is not code; it is a reference item that provides programmers with information about the method, its requirements, and its use. An *implementation* is code that actually tells the computer how to carry out the method. Often, as in the example above, multiple implementations are possible for a given method.

In object-oriented programming, methods are grouped into "classes." A class file typically contains several methods and related data. Classes, in turn, are grouped into "packages" known as API packages. Whereas a class generally corresponds to a single file, a package is more like a folder or directory providing an organizational structure for the class files. A given API package could contain many sub-packages, each with its own classes and sub-classes, which in turn contain their own methods. These elements generally are named and grouped in ways that help human programmers find, understand, and use them. A well developed set of API packages, sometimes called a "class library," is a powerful tool for software developers; as such, it can help attract developers to a particular platform.

The specification for a class library — much like the specification for an automobile — is an item of detailed

documentation that explains the organization and function of all packages, classes, methods, and data fields in the library. The class library specification for a given software platform, sometimes called the "API Specification" is an important reference item for programmers. In order to make effective use of the APIs, a programmer must be able to find the portion of the specification describing the particular package, class, and method needed for a given programming task.

2. JAVA AND ANDROID.

As explained in previous orders, Java and Android are both complex software platforms with many components. For example, the Java platform includes the Java programming language, Java class libraries, the Java virtual machine, and other elements. The Java programming language has been made freely available for use by anyone without charge. Both sides agree on this. Other aspects of the Java platform, however, such as the virtual machine and class libraries, allegedly are protected by patents and copyrights.

The Android platform uses the Java programming language; thus, software developers already familiar with the Java language do not have to learn a new language in order to write programs for Android. In contrast to Java, the Android platform uses the Dalvik virtual machine instead of the Java virtual machine, provides Android class libraries, and has other non-Java components. The Java platform has been used primarily on desktop computers, but it also has been used on cell phones and other mobile computing devices. Android, on the other hand, was designed specifically for mobile devices. Java and Android compete in the market for mobile computing software.

According to Oracle, Android is an unauthorized and incompatible Java implementation. The Java platform and the Android platform each includes class libraries with more than one hundred API packages. Android allegedly supports some, but not all, of the APIs defined for the Java platform. Thus, some programs written for the Java platform will not run properly on the Android platform, even though both use the Java language. Similarly, the Android platform allegedly includes additional APIs that are not part of the Java platform. Thus, some programs written for the Android platform will not run properly on the Java platform, even though they are written in the Java language. This so-called fragmentation undermines the "write once, run anywhere" concept underlying the Java system and supposedly damages Oracle by decreasing Java's appeal to software developers.

3. TERMINOLOGY

The term API is slippery. It has been used by the parties and in the industry as shorthand to refer to many related concepts, ranging from individual methods to code implementations to entire class libraries and specifications. In this order, the term API will be used only to refer to the abstract concept of an application programming interface. API *documentation* (*e.g.*, the specification for a class library or for an API package within the library) and API *implementations* (*e.g.*, the source code relating to a particular method within a class file) will be referenced as such. Having clarified this linguistic point, this order proceeds to consider the specific items accused of copyright infringement in this

action: twelve files of code, and 37 API package specifications.[1]

ANALYSIS

Summary judgment is proper when "there is no genuine dispute as to any material fact and the movant is entitled to judgment as a matter of law." FRCP 56(a). Where the party moving for summary judgment would bear the burden of proof at trial, that party bears the initial burden of producing evidence that would entitle it to a directed verdict if uncontroverted at trial. *See C.A.R. Transp. Brokerage Co. v. Darden Rests., Inc.*, 213 F.3d 474, 480 (9th Cir. 2000). Where the party moving for summary judgment would not bear the burden of proof at trial, that party bears the initial burden of either producing evidence that negates an essential element of the non-moving party's claims, or showing that the non-moving party does not have enough evidence of an essential element to carry its ultimate burden of persuasion at trial. If the moving party satisfies its initial burden of production, then the non-moving party must produce admissible evidence to show there exists a genuine issue of material fact. *See Nissan Fire & Marine Ins. Co. v. Fritz Cos.*, 210 F.3d 1099, 1102–03 (9th Cir. 2000).

Copyright protection subsists in "original works of authorship fixed in any tangible medium of expression."

---

[1] At the hearing, counsel for Oracle suggested that Google's code *implementations* of the 37 API package specifications are unauthorized derivative works. This theory was disclosed by Oracle during discovery (Dkt. No. 263-3 at 11), but it was dismissed summarily in Google's summary judgment brief (Br. 9). Because the briefing does not address this theory, it will not be addressed herein.

17 U.S.C. 102. In order to succeed on a copyright infringement claim, a plaintiff must show that it owns the copyright and that the defendant copied protected elements of the work. Only expressive elements that are "original," *i.e.*, independently created, are protected. Copying can be proven by showing that the alleged infringer had access to the copyrighted work and that the protected portions of the works are substantially similar. *Jada Toys, Inc. v. Mattel, Inc.*, 518 F.3d 628, 636–37 (9th Cir. 2008). Google advances a number of arguments why Oracle supposedly cannot prove all or part of its copyright infringement claim. Google is entitled to summary judgment on only one issue.

1. THE CODE FILES

Regarding the twelve code files at issue, Google argues that its alleged copying was *de minimis* (Br. 22–24). In the copyright infringement context, "a taking is considered *de minimis* only if it is so meager and fragmentary that the average audience would not recognize the appropriation." *Fisher v. Dees*, 794 F.2d 432, 434 n.2 (9th Cir. 1986). The extent of the copying "is measured by considering the qualitative and quantitative significance of the copied portion in relation to the plaintiff's work as a whole." *Newton v. Diamond*, 388 F.3d 1189, 1195 (9th Cir. 2004).

Here, the parties dispute what constitutes the plaintiff's work as a whole. Google argues that its alleged copying should be compared to the entire Java platform, which Oracle registered as a single work (Br. 22–23; Kwun Exh. B). Oracle, on the other hand, argues that each of the twelve code files at issue is a separate work for purposes of this analysis (Opp. 23–24). Google has not shown that the Java platform is the proper basis for comparison. Google cites two provisions of the copyright regulations,

but neither one supports Google's position (Reply Br. 12–13).

First, Google misapplies 37 C.F.R. 202.3(b)(4)(i)(A). That provision states: "For the purpose of registration on a single application and upon payment of a single registration fee, the following shall be considered a single work: (A) In the case of published works: all copyrightable elements that are otherwise recognizable as self-contained works, that are included in a single unit of publication, and in which the copyright claimant is the same." The plain meaning of this provision is that when a single published unit contains multiple elements "that are otherwise recognizable as self-contained works," the unit is considered a single work *for the limited purpose of registration*, while its elements may be recognized as separate works for other purposes. Courts considering Section 202.3(b)(4)(i)(A) generally agree with this interpretation. *See, e.g.*, *Tattoo Art, Inc. v. TAT Int'l., LLC*, --- F. Supp. 2d. ---, No. 2:10cv323, 2011 WL 2585376, at *15–16 (E.D. Va. June 29, 2011) (interpreting Section 202.3(b)(4)(i)(A) to codify the principle that "the copyrights in multiple works may be registered on a single form, and thus considered one work *for the purposes of registration* while still qualifying as separate 'works' for purposes of awarding statutory damages"). Google relies on Section 202.3(b)(4)(i)(A) to show that the code files comprising the Java platform should be treated collectively as a single work *for purposes of an infringement analysis*. This interpretation is contrary to the plain language of the regulation and is not supported by any cited authority.

Second, Google cites to 37 C.F.R. 202.3(b)(3), which concerns continuation sheets. Continuation sheets are

used "only in submissions for which a paper application is used and where additional space is needed by the applicant to provide all relevant information." 37 C.F.R 202.3(b)(3). The regulation requires use of a separate continuation sheet "to list contents titles, *i.e.*, titles of independent works in which copyright is being claimed and which appear within a larger work." *Ibid.* It does not, however, state that a failure to list individual titles precludes an applicant from later asserting those titles as separate works in infringement litigation. Nor does it address works registered by means other than a paper application. Google does not provide enough factual context to show that Section 202.3(b)(3) applies to the works at issue, and Google does not explain how it might bear upon the dispute at hand, even if it does apply.

Google cites no other authority. This order finds that, at least on the present record, Google has not shown that the Java platform as a whole is the work to which Google's alleged copying should be compared. Because all of Google's *de minimis* arguments compare the accused material in the code files to the entire Java platform as a whole, this order need not consider the *de minimis* question further.

2. THE API PACKAGE SPECIFICATIONS.

Regarding the 37 API package specifications at issue, which are reference items and not code, Google argues that the only similarities between the accused works and the asserted works are elements that are not subject to copyright protection. Google, however, does not specify which elements it views as similar. Google instead presents an array of theories why various *categories* of specification elements do not merit copyright protection. With one exception, this broad categorical approach fails.

Google's other arguments regarding the API package specifications — that the disputed works are not virtually identical or substantially similar, and that Google's alleged copying was fair use — also fail to earn summary judgment for Google.

A. Names.

"Words and short phrases such as names, titles, and slogans" are "not subject to copyright." 37 C.F.R. 202.1(a); *Planesi v. Peters*, No. 04-16936, slip op. at *1 (9th Cir. Aug. 15, 2005). Google argues that "the names of the Java language API files, packages, classes, and methods are not protectable as a matter of law" (Br. 17). This order agrees. Because names and other short phrases are not subject to copyright, the names of the various items appearing in the disputed API package specifications are not protected. *See Sega Enters. Ltd. v. Accolade, Inc.*, 977 F.2d 1510, 1524 n.7 (9th Cir. 1992) ("Sega's security code is of such de minimis length that it is probably unprotected under the words and short phrases doctrine.").

Oracle argues that it is entitled to a "presumption that the names in the Java API specifications are original" (Opp. 14). Not so. The decision Oracle cites for this proposition shows only that a certificate of registration may entitle its holder to a presumption of copyright validity as to the registered work. *Swirsky v. Carey*, 376 F.3d 841, 851 (9th Cir. 2004) (citing 17 U.S.C. 410(c)). Oracle cites no authority requiring a presumption of originality as to *specific elements* of a registered work.

Oracle also argues that its selection and arrangement of component names within the specifications is entitled to copyright protection (Opp. 15). This argument is non-responsive. Copyright protection for the selection and

arrangement of elements within a work is a separate question from whether the elements themselves are protected by copyright. In finding that the names of the various items appearing in the disputed API package specifications are not protected by copyright, this order does not foreclose the possibility that the selection or arrangement of those names is subject to copyright protection. *See Lamps Plus, Inc. v. Seattle Lighting Fixture Co.*, 345 F.3d 1140, 1147 (9th Cir. 2003) ("[A] combination of *unprotectable elements* is eligible for copyright protection only if those elements are numerous enough and their selection and arrangement original enough that their combination constitutes an original work of authorship.") (emphasis added).

Having found that the names of the various items appearing in the disputed API package specifications are not protected by copyright on account of the words and short phrases doctrine, this order need not consider Google's alternative theory that the names are unprotected because they are the result of customary programming practices.

B. *Scenes a Faire* and the Merger Doctrine.

"Under the scenes a faire doctrine, when certain commonplace expressions are indispensable and naturally associated with the treatment of a given idea, those expressions are treated like ideas and therefore not protected by copyright." *Swirsky v. Carey*, 376 F.3d at 850. "Under the merger doctrine, courts will not protect a copyrighted work from infringement if the idea underlying the copyrighted work can be expressed in only one way, lest there be a monopoly on the underlying idea." *Satava v. Lowry*, 323 F.3d 805, 812 n.5 (9th Cir. 2003).

Google argues that "[t]he API declarations are unprotectable *scenes a faire* or unprotectable under the merger doctrine" (Br. 14). Google, however, does not specify what it means by "API declarations." Google applies this argument to all of "[t]he allegedly copied elements of the Java language API packages," providing only a few examples: "the names of packages and methods and definitions" (*id.* at 14–16). To the extent Google directs this argument to names, it is moot in light of the above ruling. To the extent Google directs this argument to other elements of the API package specifications, it is not adequately supported.

Google's lack of specificity is fatal. If Google believes, for example, that a particular method declaration is a scene a faire or is the only possible way to express a given function, then Google should provide evidence and argument supporting its views as to that method declaration. Instead, Google argues — relying mostly on non-binding authority[2] — that entire *categories* of elements in API specifications do not merit copyright protection. This approach ignores the possibility that some method declarations (for example) may be subject to the merger doctrine or may be *scenes a faire*, whereas other method declarations may be creative contributions subject to copyright protection. Google has not justified the sweeping ruling it requests. Google has not even identified which categories of specification elements it deems unprotectable under these doctrines. This order declines to hold that API package specifications, or any

[2] The only binding authority Google cites is the Sega decision. The cited discussion addresses computer program code, not documentation. Google has not justified applying the Sega rationale to documentation such as the API package specifications at issue here.

particular category of elements they contain, are unprotectable under the *scenes a faire* or merger doctrines.

C. Methods of Operation.

"In no case does copyright protection for an original work of authorship extend to any idea, procedure, process, system, *method of operation*, concept, principle, or discovery, regardless of the form in which it is described, explained, illustrated, or embodied in such work." 17 U.S.C. 102(b) (emphasis added). Google argues that "APIs for a programming language" are unprotected methods of operation (Br. 13). Google, however, does not use the term API consistently in the relevant portions of its briefs, so it is unclear precisely what Google is attempting to characterize as a method of operation. Google states that all "elements common to Oracle's Java language APIs and the Android APIs are unprotectable methods of operation," but Google does not specify which elements it views as common (id. at 12). Context suggests two possible interpretations for Google's use of the term APIs. Both of Google's apparent arguments are unavailing.

First, Google appears to direct its methods-of-operation argument to APIs themselves as the term is used in this order — that is, to the abstract concept of an interface between programs. In its reply brief, Google distinguishes APIs both from their *implementation* in libraries of code ("the APIs are not the libraries themselves") and from their *documentation* in reference materials ("The APIs do not 'tell' how to use the libraries, they are the *means by which one uses* the libraries; the *documentation* for the APIs 'tells' how to use the libraries.") (Reply Br. 2–3). Google's argument that APIs are unprotectable methods of operation attacks a straw

man. *It is not the APIs but rather the specifications for 37 API packages that are accused.* Even if Google can show that *APIs* are methods of operation not subject to copyright protection, that would not defeat Oracle's infringement claim concerning the accused *specifications*.

Google may be trying to head off a possible argument by Oracle that the APIs described in the specifications are nonliteral elements of the specifications subject to copyright protection. It is unclear whether Oracle is advancing such an argument. Oracle's opposition brief seems to use the term API to refer to API packages and API package *specifications*. If this interpretation is correct, then the parties' arguments concerning whether "APIs" are methods of operation simply swipe past each other, with each party using the term in a different way. Because the issue is not properly teed up for summary judgment, this order does not decide whether APIs are methods of operation.

Second, Google also states that "API *specifications* are methods of operation" (Br. 14). This conclusion does not follow from Google's argument that APIs — meaning conceptual interfaces between programs — are methods of operation. No other supporting argument is provided. *API specifications are written documentation.* Even if Google could show that APIs are methods of operation, that would not mean that a written work that describes or embodies APIs is automatically exempt from copyright protection. This order finds that the API package specifications at issue are not "methods of operation" under 17 U.S.C. 102(b).

D. Degree of Similarity.

The copying element of copyright infringement generally can be proven by showing that the alleged infringer had access to the copyrighted work and that the protected portions of the works are substantially similar. *Jada Toys*, 518 F.3d at 636–37. "When the range of protectable and unauthorized expression is narrow," however, "the appropriate standard for illicit copying is virtual identity" rather than substantial similarity. *Apple Computer, Inc. v. Microsoft Corp.*, 35 F.3d 1435, 1439 (9th Cir. 1994).

Google argues that "[g]iven the substantial unprotected elements in the documentation (such as the API method declarations), the 'virtual identity' standard applies here" (Br. 24). This order agrees with Google that the *names* of the various items appearing in the disputed API package specifications are not protected by copyright. Google, however, has not shown that any other elements of the specifications are exempt from copyright protection. Because Google has not proven that a substantial portion of the specifications is unprotected, Google's justification for applying the virtual identity standard fails. This order therefore need not consider Google's arguments that the disputed Java and Android API package specifications are not virtually identical. In particular, Google analyzes the selection and arrangement of elements within the specifications under only the virtual identity standard (Br. 24–25).

As a fallback position, Google argues that even under the substantial similarity standard, the disputed Java and Android API package specifications are not sufficiently similar to show copying. Google analogizes the specifications to dictionary definitions whose similarities

are driven by external constraints, and Google cites an expert opinion that the Java and Android platforms are not substantially similar (Br. 24; Astrachan Exh. 1 at 77). Predictably, Oracle presents an opposing expert opinion that the API package specifications at issue *are* substantially similar (Mitchell Exh. 1 at 45). This conflicting expert testimony highlights a factual issue that precludes summary judgment; a reasonable trier of fact might agree with either expert's analysis of the degree of similarity between the asserted and accused specifications.

Google argues that Oracle's expert testimony is not sufficient to defeat summary judgment. Google criticizes the expert for offering a "summary 'conclusion'" based on a "single illustrative example," which Google interprets differently (Reply Br. 11). In his report, however, the expert provides multiple examples and explains that he conducted a detailed comparison of each of the API package specification pairs at issue (Mitchell Exh. 1 at 60–63). His opinion that the Android specifications are substantially similar to their Java counterparts is not a mere "[c]onclusory statement[] without factual support." *See Surrell v. Cal. Water Serv. Co.*, 518 F.3d 1097, 1103 (9th Cir. 2008). If Google disputes the basis for the opinion by Oracle's expert or his analysis of the specifications, then Google should raise its critiques during crossexamination at trial. Google has not earned summary judgment of no copying under either of the possible standards for comparison — virtual identity or substantial similarity.

E. Fair Use.

The following factors are considered in determining whether the use made of a work is a fair use: (1) the

purpose and character of the use, including whether such use is of a commercial nature or is for nonprofit educational purposes; (2) the nature of the copyrighted work; (3) the amount and substantiality of the portion used in relation to the copyrighted work as a whole; and (4) the effect of the use upon the potential market for or value of the copyrighted work. 17 U.S.C. 107. Google argues that its alleged use of elements from the Java API package specifications in its Android API specifications was fair (Br. 19–22). Evaluation of the fair use factors, however, depends upon disputed questions of material fact. As such, no finding of fair use can be made on the summary judgment record.

For example, with respect to factor four, Google argues that "Android has contributed positively to the market for the copyrighted works by increasing the number of Java language developers" (Br. 21). Google cites positive reactions by Sun executives at the time when Android was first released in 2007. These statements do not prove anything about Android's actual impact on the Java market since that time. Moreover, Oracle presents sworn testimony that Android fragmented the Java platform and locked Java out of the smartphone market (Swoopes Exh. 6 at 111–12). Oracle and Google both employ complex business models for their respective products. The question of damages is one of the most complicated and hotly contested issues in this action. On the present record, a reasonable fact finder could disagree with Google's rosy depiction of Android's impact on the Java market.

Because fact issues preclude a summary judgment finding of fair use, this order does not reach the parties' arguments on all of the fair use factors.

* * *

This order finds that the names of the various items appearing in the disputed API package specifications are not protected by copyright. This order makes no finding as to whether any other elements of the API package specifications (or their selection or arrangement) are protected or infringed.

3. INDIRECT INFRINGEMENT.

Google argues that Oracle's indirect copyright infringement theories fail because Oracle cannot establish any underlying direct copyright infringement (Br. 25). Because Google is not entitled to summary judgment on direct infringement, Google also is not entitled to summary judgment on indirect infringement.

CONCLUSION

For the foregoing reasons, defendant's motion for summary judgment on the copyright infringement claim is **GRANTED IN PART AND DENIED IN PART**. This order finds that the names of the various items appearing in the disputed API package specifications are not protected by copyright. To that extent, the motion is **GRANTED**. All of defendant's other summary judgment theories regarding the copyright claim are **DENIED**. Plaintiff's evidentiary objections to the Bornstein declaration and the Astrachan declaration are **MOOT**.

IT IS SO ORDERED.

Dated: September 15, 2011.

/s/

WILLIAM ALSUP, UNITED STATES DISTRICT JUDGE

Appendix F

IN THE UNITED STATES DISTRICT COURT FOR THE NORTHERN DISTRICT OF CALIFORNIA

ORACLE AMERICA, INC., Plaintiff, v. GOOGLE INC., Defendant.	No. C 10-03561 WHA

ORDER ON MOTIONS FOR JUDGMENT AS A MATTER OF LAW

For the reasons stated at the May 9 hearing, Oracle's motion for judgment as a matter of law regarding fair use, API documentation, and comment-copied files is **DENIED**; Google's motion for judgment as a matter of law regarding rangeCheck is **DENIED**.

IT IS SO ORDERED.

Dated: May 10, 2012.

/s/

WILLIAM ALSUP, UNITED STATES DISTRICT JUDGE

Appendix G

IN THE UNITED STATES DISTRICT COURT FOR THE NORTHERN DISTRICT OF CALIFORNIA

ORACLE AMERICA, INC., Plaintiff, v. GOOGLE INC., Defendant.	No. C 10-03561 WHA

ORDER RE COPYRIGHTABILITY OF CERTAIN REPLICATED ELEMENTS OF THE JAVA APPLICATION PROGRAMMING INTERFACE

INTRODUCTION

This action was the first of the so-called "smartphone war" cases tried to a jury. This order includes the findings of fact and conclusions of law on a central question tried simultaneously to the judge, namely the extent to which, if at all, certain replicated elements of the structure, sequence and organization of the Java application programming interface are protected by copyright.

PROCEDURAL HISTORY

In 2007, Google Inc., announced its Android software platform for mobile devices. In 2010, Oracle Corporation acquired Sun Microsystems, Inc., and thus acquired Sun's interest in the popular programming language known as Java, a language used in Android. Sun was renamed Oracle America, Inc. Shortly thereafter, Oracle America (hereinafter simply "Oracle") sued defendant Google and

accused its Android platform as infringing Oracle's Java-related copyrights and patents. Both Java and Android are complex platforms. Both include "virtual machines," development and testing kits, and application programming interfaces, also known as APIs. Oracle's copyright claim involves 37 packages in the Java API. Copyrightability of the elements replicated is the only issue addressed by this order.

Due to complexity, the Court decided that the jury (and the judge) would best understand the issues if the trial was conducted in phases. The first phase covered copyrightability and copyright infringement as well as equitable defenses. The second phase covered patent infringement. The third phase would have dealt with damages but was obviated by stipulation and verdicts.

For the first phase, it was agreed that the judge would decide issues of copyrightability and Google's equitable defenses and that the jury would decide infringement, fair use, and whether any copying was de minimis. Significantly, all agreed that Google had not literally copied the software but had instead come up with its own implementations of the 37 API packages. Oracle's central claim, rather, was that Google had replicated the structure, sequence and organization of the overall code for the 37 API packages.

For their task of determining infringement and fair use, the jury was told it should take for granted that the structure, sequence and organization of the 37 API packages as a whole was copyrightable. This, however, was not a final definitive legal ruling. One reason for this instruction was so that if the judge ultimately ruled, after hearing the phase one evidence, that the structure, sequence and organization in question was not protectable

but was later reversed in this regard, the court of appeals might simply reinstate the jury verdict. In this way, the court of appeals would have a wider range of alternatives without having to worry about an expensive retrial. Counsel were so informed but not the jury.

Each side was given seventeen hours of "air time" for phase one evidence (not counting openings, closings or motion practice). In phase one, as stated, the parties presented evidence on copyrightability, infringement, fair use, and the equitable defenses. As to the compilable code for the 37 Java API packages, the jury found that Google infringed but deadlocked on the follow-on question of whether the use was protected by fair use. As to the documentation for the 37 Java API packages, the jury found no infringement. As to certain small snippets of code, the jury found only one was infringing, namely, the nine lines of code called "rangeCheck." In phase two, the jury found no patent infringement across the board. (Those patents, it should be noted, had nothing to do with the subject addressed by this order.) The entire jury portion of the trial lasted six weeks.[1]

This order addresses and resolves the core premise of the main copyright claims, namely, whether the elements replicated by Google from the Java system were protectable by copyright in the first place. No law is directly on point. This order relies on general principles of

---

[1] After the jury verdict, the Court granted Oracle's Rule 50 motion for judgment as a matter of law of infringement of eight decompiled computer files, which were literally copied. Google admitted to copying eight computer files by decompiling the bytecode from eight Java files into source code and then copying the source code. These files were not proven to have ever been part of Android.

copyright law announced by Congress, the Supreme Court and the Ninth Circuit.

* * *

Counsel on both sides have supplied excellent briefing and the Court wishes to recognize their extraordinary effort and to thank counsel, including those behind the scenes burning midnight oil in law libraries, for their assistance.

SUMMARY OF RULING

So long as the specific code used to implement a method is different, anyone is free under the Copyright Act to write his or her own code to carry out exactly the same function or specification of any methods used in the Java API. It does not matter that the declaration or method header lines are identical. Under the rules of Java, they *must be identical* to declare a method specifying the *same* functionality — even when the implementation is different. When there is only one way to express an idea or function, then everyone is free to do so and no one can monopolize that expression. And, while the Android method and class names could have been different from the names of their counterparts in Java and still have worked, copyright protection never extends to names or short phrases as a matter of law.

It is true that the very same functionality could have been offered in Android without duplicating the exact command structure used in Java. This could have been done by re-arranging the various methods under different groupings among the various classes and packages (even if the same names had been used). In this sense, there were many ways to group the methods yet still duplicate the same range of functionality.

But the names are more than just names — they are symbols in a command structure wherein the commands take the form

java.package.Class.method()

Each command calls into action a pre-assigned function. The overall name tree, of course, has creative elements but it is also a precise command structure — a utilitarian and functional set of symbols, each to carry out a pre-assigned function. This command structure is a system or method of operation under Section 102(b) of the Copyright Act and, therefore, cannot be copyrighted. Duplication of the command structure is necessary for interoperability.

STATEMENT OF FINDINGS

1. JAVA AND ANDROID.

Java was developed by Sun, first released in 1996, and has become one of the world's most popular programming languages and platforms.[2] The Java platform, through the use of a virtual machine, enables software developers to write programs that are able to run on different types of computer hardware without having to rewrite them for each different type. Programs that run on the Java platform are written in the Java language. Java was

[2] For purposes of this order, the term "Java" means the Java platform, sometimes abbreviated to "J2SE," which includes the Java development kit (JDK), javac compiler, tools and utilities, runtime programs, class libraries (API packages), and the Java virtual machine.

developed to run on desktop computers and enterprise servers.[3]

The Java language, like C and C++, is a human-readable language. Code written in a human-readable language — "source code" — is not readable by computer hardware. Only "object code," which is not human-readable, can be used by computers. Most object code is in a binary language, meaning it consists entirely of 0s and 1s. Thus, a computer program has to be converted, that is, compiled, from source code into object code before it can run, or "execute." In the Java system, source code is first converted into "bytecode," an intermediate form, before it is then converted into binary machine code by the Java virtual machine.

The Java language itself is composed of keywords and other symbols and a set of pre-written programs to carry out various commands, such as printing something on the screen or retrieving the cosine of an angle. The set of pre-written programs is called the application programming interface or simply API (also known as class libraries).

In 2008, the Java API had 166 "packages," broken into more than six hundred "classes," all broken into over six

---

[3] Rather than merely vet each and every finding and conclusion proposed by the parties, this order has navigated its own course through the evidence and arguments, although many of the proposals have found their way into this order. Any proposal that has been expressly agreed to by the opposing side, however, shall be deemed adopted (to the extent agreed upon) even if not expressly adopted herein. It is unnecessary for this order to cite the record for all of the findings herein. In the findings, the phrase "this order finds . . ." is occasionally used to emphasize a point. The absence of this phrase, however, does not mean (and should not be construed to mean) that a statement is not a finding. All declarative fact statements set forth in the order are factual findings.

thousand "methods." This is very close to saying the Java API had 166 "folders" (packages), all including over six hundred pre-written programs (classes) to carry out a total of over six thousand subroutines (methods). Google replicated the exact names and exact functions of virtually all of these 37 packages but, as stated, took care to use different code to implement the six thousand-plus subroutines (methods) and six-hundred-plus classes.

An API is like a library. Each package is like a bookshelf in the library. Each class is like a book on the shelf. Each method is like a how-to-do-it chapter in a book. Go to the right shelf, select the right book, and open it to the chapter that covers the work you need. As to the 37 packages, the Java and Android libraries are organized in the same basic way but all of the chapters in Android have been written with implementations different from Java but solving the same problems and providing the same functions. Every method and class is specified to carry out precise desired functions and, thus, the "declaration" (or "header") line of code stating the specifications must be identical to carry out the given function.[4]

The accused product is Android, a software platform developed by Google for mobile devices. In August 2005, Google acquired Android, Inc., as part of a plan to develop a smartphone platform. Google decided to use the Java language for the Android platform. In late 2005, Google began discussing with Sun the possibility of taking a license to use and to adapt the entire Java platform for

---

[4] The term "declaration" was used throughout trial to describe the headers (non-implementing code) for methods and classes. While "header" is the more technically accurate term, this order will remain consistent with the trial record and use "declaration" and "header" interchangeably.

mobile devices. They also discussed a possible co-development partnership deal with Sun under which Java technology would become an open-source part of the Android platform, adapted for mobile devices. Google and Sun negotiated over several months, but they were unable to reach a deal.

In light of its inability to reach agreement with Sun, Google decided to use the Java language to design its own virtual machine via its own software and to write its own implementations for the functions in the Java API that were key to mobile devices. Specifically, Google wrote or acquired its own source code to implement virtually all the functions of the 37 API packages in question. Significantly, all agree that these implementations — which account for 97 percent of the lines of code in the 37 API packages — are different from the Java implementations. In its final form, the Android platform also had its own virtual machine (the so-called Dalvik virtual machine), built with software code different from the code for the Java virtual machine.

As to the 37 packages at issue, Google believed Java application programmers would want to find the same 37 sets of functionalities in the new Android system callable by the same names as used in Java. Code already written in the Java language would, to this extent, run on Android and thus achieve degree of interoperability.

The Android platform was released in 2007. The first Android phones went on sale the following year. Android-based mobile devices rapidly grew in popularity and now comprise a large share of the United States market. The Android platform is provided free of charge to smartphone manufacturers. Google receives revenue through advertisement whenever a consumer uses

particular functions on an Android smartphone. For its part, Sun and Oracle never successfully developed its own smartphone platform using Java technology. All agree that Google was and remains free to use the Java language itself.

All agree that Google's virtual machine is free of any copyright issues. All agree that the six-thousand-plus method implementations by Google are free of copyright issues. The copyright issue, rather, is whether Google was and remains free to replicate the names, organization of those names, and functionality of 37 out of 166 packages in the Java API, which has sometimes been referred to in this litigation as the "structure, sequence and organization" of the 37 packages.

The Android platform has its own API. It has 168 packages, 37 of which are in contention. Comparing the 37 Java and Android packages side by side, only three percent of the lines of code are the same. The identical lines are those lines that specify the names, parameters and functionality of the methods and classes, lines called "declarations" or "headers." In particular, the Android platform replicated the same package, method and class names, definitions and parameters of the 37 Java API packages from the Java 2SE 5.0 platform. This three percent is the heart of our main copyright issue.

A side-by-side comparison of the 37 packages in the J2SE 5.0 version of Java versus in the Froyo version of Android shows that the former has a total of 677 classes (plus interfaces) and 6508 methods wherein the latter has 616 and 6088, respectively. Twenty-one of the packages have the same number of classes, interfaces and methods, although, as stated, the method implementations differ.

The three percent of source code at issue includes "declarations." Significantly, the rules of Java dictate the precise form of certain necessary lines of code called declarations, whose precise and necessary form explains why Android and Java *must be* identical when it comes to those particular lines of code. That is, since there is only one way to declare a given method functionality, everyone using that function must write that specific line of code in the same way. The same is true for the "calls," the commands that invoke the methods. To see why this is so, this order will now review some of the key rules for Java programming. This explanation will start at the bottom and work its way upward.

2. THE JAVA LANGUAGE AND ITS API—IMPORTANT DETAILS.

Java syntax includes *separators* (*e.g.*, {, }, ;), *operators* (*e.g.*, +, -, *, /, <, >), *literal values* (*e.g.*, 123, 'x', "Foo"), and *keywords* (*e.g.*, if, else, while, return). These elements carry precise predefined meanings. Java syntax also includes *identifiers* (*e.g.*, String, java.lang.Object), which are used to name specific values, fields, methods, and classes as described below.

These syntax elements are used to form statements, each statement being a single command executed by the Java compiler to take some action. Statements are run in the sequence written. Statements are commands that tell the computer to do work.

A method is like a subroutine. Once declared, it can be invoked or "called on" elsewhere in the program. When a method is called on elsewhere in the program or in an application, "arguments" are usually passed to the method as inputs. The output from the method is known as the

"return." An example is a method that receives two numbers as inputs and returns the greater of the two as an output. Another example is a method that receives an angle expressed in degrees and returns the cosine of that angle. Methods can be much more complicated. A method, for example, could receive the month and day and return the Earth's declination to the sun for that month and day.

A method consists of the method header and the method body. A method header contains the name of the method; the number, order, type and name of the parameters used by the method; the type of value returned by the method; the checked exceptions that the method can throw; and various method modifiers that provide additional information about the method. At the trial, witnesses frequently referred to the method header as the "declaration." This discrepancy has no impact on the ultimate analysis. The main point is that this header line of code introduces the method body and specifies very precisely its inputs, name and other functionality. Anyone who wishes to supply a method with the same functionality must write this line of code in the same way and must do so no matter how different the implementation may be from someone else's implementation.

The method body is a block of code that then implements the method. If a method is declared to have a return type, then the method body must have a statement and the statement must include the expression to be returned when that line of code is reached. During trial, many witnesses referred to the method body as the "implementation." It is the method body that does the heavy lifting, namely the actual work of taking the inputs, crunching them, and returning an answer. The method body can be short or long. Google came up with its own

implementations for the method bodies and this accounts for 97 percent of the code for the 37 packages.

Once the method is written, tested and in place, it can be called on to do its work. A method call is a line of code *somewhere else*, such as in a different program that calls on (or invokes) the method and specifies the arguments to be passed to the method for crunching. The method would be called on using the command format "java.package.Class.method()" where () indicates the inputs passed to the method. For example, a = java.package.Class.method() would set the field "a" to equal the return of the method called. (The words "java.package.Class.method" would in a real program be other names like "java.lang.Math.max"; "java.package.Class.method" is used here simply to explain the format.)

After a method, the next higher level of syntax is the class. A class usually includes fields that hold values (such as pi = 3.141592) and methods that operate on those values. Classes are a fundamental structural element in the Java language. A Java program is written as one or more classes. More than one method can be in a class and more than one class can be in a package. All code in a Java program must be placed in a class. A class declaration (or header) is a line that includes the name of the class and other information that define the class. The body of the class includes fields and methods, and other parameters.

Classes can have subclasses that "inherit" the functionality of the class itself. When a new subclass is defined, the declaration line uses the word "extends" to alert the compiler that the fields and methods of the parent class are inherited automatically into the new

subclass so that only additional fields or methods for the subclass need to be declared.

The Java language does not allow a class to extend (be a subclass of) more than one parent class. This restrictiveness may be problematic when one class needs to inherit fields and methods from two different non-related classes. The Java programming language alleviates this dilemma through the use of "interfaces," which refers to something different from the word "interface" in the API acronym. An interface is similar to a class. It can also contain methods. It is also in its own source code file. It can also be inherited by classes. The distinction is that a class may inherit from more than one interface whereas, as mentioned, a class can only inherit from one other class.

For convenience, classes and interfaces are grouped into "packages" in the same way we all group files into folders on our computers. There is no inheritance function within packages; inheritance occurs only at the class and interface level.

Here is a simple example of source code that illustrates methods, classes and packages. The italicized comments on the right are merely explanatory and are not compiled:

```
package java.lang;              // Declares package java.lang

public class Math {             // Declares class Math

public static int max           // Declares method max
(int x, int y) {

    if (x > y) return x;        // Implementation, returns x or
```

```
    else return y;          // Implementation, returns y
    }                       // Closes method
}                           // Closes class
```

To invoke this method from another program (or class), the following call could be included in the program:

```
int a = java.lang.Math.max (2, 3);
```

Upon reaching this statement, the computer would go and find the max method under the Math class in the java.lang package, input "2" and "3" as arguments, and then return a "3," which would then be set as the value of "a."

The above example illustrates a point critical to our first main copyright issue, namely that the declaration line beginning "public static" is entirely dictated by the rules of the language. In order to declare a particular *functionality*, the language *demands* that the method declaration take a particular form. There is no choice in how to express it. To be specific, that line reads:

```
public static int max (int x, int y) {
```

The word "public" means that other programs can call on it. (If this instead says "private," then it can only be accessed by other methods inside the same class.) The word "static" means that the method can be invoked without creating an instance of the class. (If this instead is an instance method, then it would always be invoked with respect to an object.) The word "int" means that an integer is returned by the method. (Other alternatives are "boolean," "char," and "String" which respectively mean "true/false," "single character," and "character string.")

Each of these three parameters is drawn from a short menu of possibilities, each possibility corresponding to a very specific functionality. The word "max" is a name and while any name (other than a reserved word) could have been used, names themselves cannot be copyrighted, as will be shown. The phrase "(int x, int y)" identifies the arguments that must be passed into the method, stating that they will be in integer form. The "x" and the "y" could be "a" and "b" or "arg1" and "arg2," so there is a degree of creativity in naming the arguments. Again, names cannot be copyrighted. (Android did not copy all of the particular argument names used in Java but did so as to some arguments.) Finally, "{" is the beginning marker that tells the compiler that the method body is about to follow. The marker is mandatory. The foregoing description concerns the rules for the language itself. Again, each parameter choice other than the names has a precise functional choice. If someone wants to implement a particular function, the declaration specification can only be written in one way.

Part of the declaration of a method can list any exceptions. When a program violates the semantic constraints of the Java language, the Java virtual machine will signal this error to the program as an exception for special handling. These are specified via "throw" statements appended at the end of a declaration. Android and Java are not identical in their throw designations but they are very similar as to the 37 packages at issue.

A Java program must have at least one class. A typical program would have more than one method in a class. Packages are convenient folders to organize the classes.

This brings us to the application programming interface. When Java was first introduced in 1996, the API

included eight packages of pre-written programs. At least three of these packages were "core" packages, according to Sun, fundamental to being able to use the Java language at all. These packages were java.lang, java.io, and java.util. As a practical matter, anyone free to use the language itself (as Oracle concedes all are), must also use the three core packages in order to make any worthwhile use of the language. Contrary to Oracle, there is no bright line between the language and the API.

Each package was broken into classes and those in turn broken into methods. For example, java.lang (a package) included Math (a class) which in turn included max (a method) to return the greater of two inputs, which was (and remains) callable as java.lang.Math.max with appropriate arguments (inputs) in the precise form required (see the example above).

After Java's introduction in 1996, Sun and the Java Community Process, a mechanism for developing a standard specifications for Java classes and methods, wrote hundreds more programs to carry out various nifty functions and they were organized into coherent packages by Sun to become the Java application programming interface. In 2008, as stated, the Java API had grown from the original eight to 166 packages with over six hundred classes with over six thousand methods. All of it was downloadable from Sun's (now Oracle's) website and usable by anyone, including Java application developers, upon agreement to certain license restrictions. Java was particularly useful for writing programs for use via the Internet and desktop computers.

Although the declarations must be the same to achieve the same functionality, the names of the methods and the way in which the methods are grouped do not have to be

the same. Put differently, many different API organizations could supply the same overall range of functionality. They would not, however, be interoperable. Specifically, code written for one API would not run on an API organized differently, for the name structure itself dictates the precise form of command to call up any given method.

To write a fresh program, a programmer names a new class and adds fields and methods. These methods can call upon the pre-written functions in the API. Instead of re-inventing the wheels in the API from scratch, programmers can call on the tried-and-true pre-packaged programs in the API. These are ready-made to perform a vast menu of functions. This is the whole point of the API. For example, a student in high school can write a program that can call upon java.lang.Math.max to return the greater of two numbers, or to find the cosine of an angle, as one step in a larger homework assignment. Users and developers can supplement the API with their own specialized methods and classes.

The foregoing completes the facts necessary to decide the copyrightability issue but since Oracle has made much of two small items copied by Google, this order will now make findings thereon so that there will be proper context for the court of appeals.

3. RANGECHECK AND THE DE-COMPILED TEST FILES.

Oracle has made much of nine lines of code that crept into both Android and Java. This circumstance is so innocuous and overblown by Oracle that the actual facts, as found herein by the judge, will be set forth below for the benefit of the court of appeals.

Dr. Joshua Bloch worked at Sun from August 1996 through July 2004, eventually holding the title of distinguished engineer. While working at Sun, Dr. Bloch wrote a nine-line code for a function called "rangeCheck," which was put into a larger file, "Arrays.java," which was part of the class library for the 37 API packages at issue. The function of rangeCheck was to check the range of a list of values before sorting the list. This was a very simple function.

In 2004, Dr. Bloch left Sun to work at Google, where he came to be the "chief Java architect" and "Java guru." Around 2007, Dr. Bloch wrote the files, "Timsort.java" and "ComparableTimsort," both of which included the same rangeCheck function he wrote while at Sun. He wrote the Timsort files in his own spare time and not as part of any Google project. He planned to contribute Timsort and ComparableTimsort back to the Java community by submitting his code to an open implementation of the Java platform, OpenJDK, which was controlled by Sun. Dr. Bloch did, in fact, contribute his Timsort file to OpenJDK and Sun included Timsort as part of its Java J2SE 5.0 release.

In 2009, Dr. Bloch worked on Google's Android project for approximately one year. While working on the Android team, Dr. Bloch also contributed Timsort and ComparableTimsort to the Android platform. Thus, the nine-line rangeCheck function was copied into Google's Android. This was how the infringement happened to occur. When discovered, the rangeCheck lines were taken out of the then-current version of Android over a year ago. The rangeCheck block of code appeared in a class containing 3,179 lines of code. This was an innocent and

inconsequential instance of copying in the context of a massive number of lines of code.

Since the remainder of this order addresses only the issue concerning structure, sequence and organization, and since rangeCheck has nothing to do with that issue, rangeCheck will not be mentioned again, but the reader will please remember that it has been readily conceded that these nine lines of code found their way into an early version of Android.

Google also copied eight computer files by decompiling the bytecode from eight Java files back into source code and then using the source code. These files were merely used as test files and never found their way into Android or any handset. These eight files have been treated at trial as a single unit.

Line by line, Oracle tested all fifteen million lines of code in Android (and all files used to test along the way leading up to the final Android) and these minor items were the only items copied, save and except for the declarations and calls which, as stated, can only be written in one way to achieve the specified functionality.

ANALYSIS AND CONCLUSIONS OF LAW

1. NAMES AND SHORT PHRASES.

To start with a clear-cut rule, names, titles and short phrases are not copyrightable, according to the United States Copyright Office, whose rule thereon states as follows:

> Copyright law does not protect names, titles, or short phrases or expressions. Even if a name, title, or short phrase is novel or distinctive or lends itself to a play on words, it cannot be

> protected by copyright. The Copyright Office cannot register claims to exclusive rights in brief combinations of words such as:
>
> - Names of products or services.
> - Names of business organizations, or groups (including the names of performing groups).
> - Pseudonyms of individuals (including pen or stage names).
> - Titles of works.
> - Catchwords, catchphrases, mottoes, slogans, or short advertising expressions.
> - Listings of ingredients, as in recipes, labels, or formulas. When a recipe or formula is accompanied by an explanation or directions, the text directions may be copyrightable, but the recipe or formula itself remains uncopyrightable.

U.S. Copyright Office, Circular 34; *see* 37 C.F.R. 202.1(a).

This rule is followed in the Ninth Circuit. *Sega Enters., Ltd. v. Accolade, Inc.*, 977 F.2d 1510, 1524 n.7 (9th Cir. 1992). This has relevance to Oracle's claim of copyright ownership over names of methods, classes and packages.

2. THE DEVELOPMENT OF LAW ON THE COPYRIGHTABILITY OF COMPUTER PROGRAMS AND THEIR STRUCTURE, SEQUENCE AND ORGANIZATION.

Turning now to the more difficult question, this trial showcases a distinction between copyright protection and patent protection. It is an important distinction, for copyright exclusivity lasts 95 years whereas patent

exclusivity lasts twenty years. And, the Patent and Trademark Office examines applications for anticipation and obviousness before allowance whereas the Copyright Office does not. This distinction looms large where, as here, the vast majority of the code was *not* copied and the copyright owner must resort to alleging that the accused stole the "structure, sequence and organization" of the work. This phrase — structure, sequence and organization — does not appear in the Act or its legislative history. It is a phrase that crept into use to describe a residual property right where literal copying was absent. A question then arises whether the copyright holder is more appropriately asserting an exclusive right to a functional system, process, or method of operation that belongs in the realm of patents, not copyrights.

A. *Baker v. Seldon*.

The general question predates computers. In the Supreme Court's decision in *Baker v. Seldon*, 101 U.S. 99 (1879), the work at issue was a book on a new system of double-entry bookkeeping. It included blank forms, consisting of ruled lines, and headings, illustrating the system. The accused infringer copied the method of bookkeeping but used different forms. The Supreme Court framed the issue as follows:

> The evidence of the complainant is principally directed to the object of showing that Baker uses the same system as that which is explained and illustrated in Selden's books. It becomes important, therefore, to determine whether, in obtaining the copyright of his books, he secured the exclusive right to the use of the system or method of book-keeping which the said books are intended to illustrate and explain.

Id. at 101. *Baker* held that using the same accounting system would not constitute copyright infringement. The Supreme Court explained that only patent law can give an exclusive right to a method:

> To give to the author of the book an exclusive property in the art described therein, when no examination of its novelty has ever been officially made, would be a surprise and a fraud upon the public. That is the province of letters-patent, not of copyright. The claim to an invention or discovery of an art or manufacture must be subjected to the examination of the Patent Office before an exclusive right therein can be obtained; and it can only be secured by a patent from the government.

Id. at 102. The Supreme Court went on to explain that protecting the method under copyright law would frustrate the very purpose of publication:

> The copyright of a work on mathematical science cannot give to the author an exclusive right to the methods of operation which he propounds, or to the diagrams which he employs to explain them, so as to prevent an engineer from using them whenever occasion requires. The very object of publishing a book on science or the useful arts is to communicate to the world the useful knowledge which it contains. But this object would be frustrated if the knowledge could not be used without incurring the guilt of piracy of the book.

Id. at 103. Baker also established the "merger" doctrine for systems and methods intermingled with the texts or diagrams illustrating them:

> And where the art it teaches cannot be used without employing the methods and diagrams used to illustrate the book, or such as are similar to them, such methods and diagrams are to be considered as necessary incidents to the art, and given therewith to the public; not given for the purpose of publication in other works explanatory of the art, but for the purpose of practical application.

Ibid. It is true that Baker is aged but it is not passé. To the contrary, even in our modern era, *Baker* continues to be followed in the appellate courts, as will be seen below.

B. The Computer Age and Section 102(b) of the 1976 Act.

Almost a century later, Congress revamped the Copyright Act in 1976. By then, software for computers was just emerging as a copyright issue. Congress decided in the 1976 Act that computer programs would be copyrightable as "literary works." *See* H.R. REP. NO. 94-1476, at 54 (1976). There was, however, no express definition of a computer program until an amendment in 1980.

The 1976 Act also codified a *Baker*-like limitation on the scope of copyright protection in Section 102(b). *See Apple Computer, Inc. v. Microsoft Corp.*, 35 F.3d 1435, 1443 n.11 (9th Cir. 1994). Section 102(b) stated (and still states):

> In no case does copyright protection for an original work of authorship extend to any idea, procedure, process, system, method of operation, concept, principle, or discovery, regardless of the form in which it is described, explained, illustrated, or embodied in such work.

The House Report that accompanied Section 102(b) of the Copyright Act explained:

> Copyright does not preclude others from using the ideas or information revealed by the author's work. It pertains to the literary, musical, graphic, or artistic form in which the author expressed intellectual concepts. Section 102(b) makes clear that copyright protection does not extend to any idea, procedure, process, system, method of operation, concept, principle, or discovery, regardless of the form in which it is described, explained, illustrated, or embodied in such work.
>
> *Some concern has been expressed lest copyright in computer programs should extend protection to the methodology or processes adopted by the programmer, rather than merely to the 'writing' expressing his ideas. Section 102(b) is intended, among other things, to make clear that the expression adopted by the programmer is the copyrightable element in a computer program, and that the actual processes or methods embodied in the program are not within the scope of the copyright law.*
>
> Section 102(b) in no way enlarges or contracts the scope of copyright protection under the

> present law. Its purpose is to restate, in the context of the new single Federal system of copyright, that the basic dichotomy between expression and idea remains unchanged.

H.R. REP. NO. 94-1476, at 56–57 (1976) (emphasis added).[5]

Recognizing that computer programs posed novel copyright issues, Congress established the National Commission on New Technological Uses of Copyrighted Works (referred to as CONTU) to recommend the extent of copyright protection for software. The Commission consisted of twelve members with Judge Stanley Fuld as chairman and Professor Melville Nimmer as vice-chairman.

The Commission recommended that a definition of "computer program" be added to the copyright statutes. This definition was adopted in 1980 and remains in the current statute:

> A "computer program" is a set of statements or instructions to be used directly or indirectly in a computer in order to bring about a certain result.

17 U.S.C. 101. Moreover, the CONTU report stated that Section 102(b)'s preclusion of copyright protection for "procedure, process, system, method of operation" was reconcilable with the new definition of "computer program." The Commission explained the dichotomy

---

[5] The Court has reviewed the entire legislative history. The quoted material above is the only passage of relevance. This order includes a summary of the CONTU report but it came after-the-fact and had little impact on the Act other than to include a definition of "computer program."

between copyrightability and non-copyrightability as follows:

> Copyright, therefore, protects the program so long as it remains fixed in a tangible medium of expression but does not protect the electromechanical functioning of a machine. The way copyright affects games and game-playing is closely analogous: one may not adopt and republish or redistribute copyrighted game rules, but the copyright owner has no power to prevent others from playing the game.
>
> *Thus, one is always free to make a machine perform any conceivable process (in the absence of a patent), but one is not free to take another's program.*

NAT'L COMM'N ON NEW TECHNOLOGICAL USES OF COPYRIGHTED WORKS, FINAL REPORT 20 (1979) (emphasis added). The Commission also recognized the "merger" doctrine, a rule of importance a few pages below in this order (emphasis added):

> The "idea-expression identity" exception provides that copyrighted language may be copied without infringing when there is but a limited number of ways to express a given idea. This rule is the logical extension of the fundamental principle that copyright cannot protect ideas. *In the computer context this means that when specific instructions, even though previously copyrighted, are the only and essential means of accomplishing a given task, their later use by another will not amount to an infringement* . . . . [C]opyright protection for

> programs does not threaten to block the use of ideas or program language previously developed by others when that use is necessary to achieve a certain result. When other language is available, programmers are free to read copyrighted programs and use the ideas embodied in them in preparing their own works.

Ibid. The Commission realized that differentiating between the copyrightable form of a program and the uncopyrightable process was difficult, and expressly decided to leave the line drawing to federal courts:

> [T]he many ways in which programs are now used and the new applications which advancing technology will supply may make drawing the line of demarcation more and more difficult. To attempt to establish such a line in this report written in 1978 would be futile. . . . Should a line need to be drawn to exclude certain manifestations of programs from copyright, that line should be drawn on a case-by-case basis by the institution designed to make fine distinctions — the federal judiciary.

Id. at 22–23.

Congress prepared no legislative reports discussing the CONTU comments regarding Section 102(b). *See* H.R. REP. NO. 96-1307, at 23–24 (1980). Nevertheless, Congress followed CONTU's recommendations by adding the definition of computer programs to the statute and amending a section of the Act not relevant to this order. *See Apple Computer, Inc. v. Formula Intern. Inc.*, 725 F.2d 521, 522–25 (9th Cir. 1984).

Everyone agrees that no one can copy line-for-line someone else's copyrighted computer program. When the line-by-line listings are different, however, some copyright owners have nonetheless accused others of stealing the "structure, sequence and organization" of the copyrighted work. That is the claim here.

C. Decisions Outside the Ninth Circuit.

No court of appeals has addressed the copyrightability of APIs, much less their structure, sequence and organization. Nor has any district court. Nevertheless, a review of the case law regarding non-literal copying of software provides guidance. Circuit decisions outside the Ninth Circuit will be considered first.

The Third Circuit led off in *Whelan Associates, Inc. v. Jaslow Dental Laboratory, Inc.*, 797 F.2d 1222 (3d Cir. 1986). In that case, the claimant owned a program, Dentalab, that handled the administrative and bookkeeping tasks of dental prosthetics businesses. The accused infringer developed another program, Dentcom, using a different programming language. The Dentcom program handled the same tasks as the Dentalab program and had the following similarities:

> The programs were similar in three significant respects . . . most of the file structures, and the screen outputs, of the programs were virtually identical . . . five particularly important "subroutines" within both programs — order entry, invoicing, accounts receivable, end of day procedure, and end of month procedure — performed almost identically in both programs.

Id. at 1228. On these facts, the district court had found, after a bench trial, that the accused infringer copied the claimant's software program. *Id.* at 1228–29.

On appeal, the accused infringer argued that the structure of the claimant's program was not protectable under copyright. In rejecting this argument, the court of appeals created the following framework to deal with non-literal copying of software:

> [T]he line between idea and expression may be drawn with reference to the end sought to be achieved by the work in question. In other words, the purpose or function of a utilitarian work would be the work's idea, and everything that is not necessary to that purpose or function would be part of the expression of the idea.

Id. at 1236 (emphasis in original). Applying this test, *Whelan* found that the structure of Dentalab was copyrightable because there were many different ways to structure a program that managed a dental laboratory:

> [T]he idea of the Dentalab program was the efficient management of a dental laboratory (which presumably has significantly different requirements from those of other businesses). Because that idea could be accomplished in a number of different ways with a number of different structures, the structure of the Dentalab program is part of the program's expression, not its idea.

Id. at 1236 n.28. The phrase "structure, sequence and organization" originated in a passage in *Whelan* explaining that the opinion used those words interchangeably and that, although not themselves part of

the Act, they were intended to capture the thought that "sequence and order could be parts of the expression, not the idea, of a work." *Id.* at 1239, 1248.

To summarize, in affirming the district court's final judgment of infringement, *Whelan* held that the *structure* of the Dentalab program was copyrightable because there were many other ways to perform the same function of handling the administrative and bookkeeping tasks of dental prosthetics businesses with different structures and designs. *Id.* at 1238. Others were free to come up with their own version but could not appropriate the Dentalab structure. This decision plainly seems to have been the high-water mark of copyright protection for the structure, sequence and organization of computer programs. It was also the only appellate decision found by the undersigned judge that affirmed (or directed) a final judgment of copyrightability on a structure, sequence and organization theory.

Perhaps because it was the first appellate decision to wade into this problem, *Whelan* has since been criticized by subsequent treatises, articles, and courts, including our own court of appeals. *See Sega Enters., Ltd. v. Accolade, Inc.*, 977 F.2d 1510, 1524–25 (9th Cir. 1992). Instead, most circuits, including ours, have adopted some variation of an approach taken later by the Second Circuit. *See Apple Computer, Inc. v. Microsoft Corp.*, 35 F.3d 1435, 1445 (9th Cir. 1994).

In *Computer Associates International, Inc. v. Altai*, 982 F.2d 693 (2d Cir. 1992), the claimant owned a program designed to translate the language of another program into the particular language that the computer's operating system would be able to understand. The accused infringer developed its own program with substantially

similar structure but different source code (using the same programming language). The Second Circuit criticized *Whelan* for taking too narrow a view of the "idea" of a program. The Second Circuit adopted instead an "abstract-filtration-comparison" test. The test first dissected the copyrighted program into its structural components:

> In ascertaining substantial similarity under [the abstract-filtration-comparison test], a court would first break down the allegedly infringed program into its constituent structural parts. Then, by examining each of these parts for such things as incorporated ideas, expression that is necessarily incidental to those ideas, and elements that are taken from the public domain, a court would then be able to sift out all non-protectable material.

Id. at 706.

Then, the test filtered out structures that were not copyrightable. For this filtration step, the court of appeals relied on the premise that programmers fashioned structures "to maximize the program's speed, efficiency, as well as simplicity for user operation, while taking into consideration certain externalities such as the memory constraints of the computer upon which the program will be run." *Id.* at 698. Because these were "practical considerations," the court held that structures based on these considerations were not copyrightable expressions.

Thus, for the filtration step, the court of appeals outlined three types of structures that should be precluded from copyright protection. *First*, copyright

protection did not extend to structures dictated by efficiency. A court must inquire

> whether the use of this particular set of modules [is] necessary efficiently to implement that part of the program's process being implemented. If the answer is yes, then the expression represented by the programmer's choice of a specific module or group of modules has merged with their underlying idea and is unprotected.

Id. at 708 (emphasis in original). Paradoxically, this meant that non-efficient structures might be copyrightable while efficient structures may not be. Nevertheless, the Second Circuit explained its reasoning as follows:

> In the context of computer program design, the concept of efficiency is akin to deriving the most concise logical proof or formulating the most succinct mathematical computation. Thus, the more efficient a set of modules are, the more closely they approximate the idea or process embodied in that particular aspect of the program's structure
>
> While, hypothetically, there might be a myriad of ways in which a programmer may effectuate certain functions within a program — *i.e.*, express the idea embodied in a given subroutine — efficiency concerns may so narrow the practical range of choice as to make only one or two forms of expression workable options.

Ibid. Efficiency also encompassed user simplicity and ease of use. *Id.* at 708–09.

Second, copyright protection did not extend to structures dictated by external factors. The court explained this as follows:

> [I]n many instances it is virtually impossible to write a program to perform particular functions in a specific computing environment without employing standard techniques. This is a result of the fact that a programmer's freedom of design choice is often circumscribed by extrinsic considerations such as (1) the mechanical specifications of the computer on which a particular program is intended to run; (2) compatibility requirements of other programs with which a program is designed to operate in conjunction; (3) computer manufacturers' design standards; (4) demands of the industry being serviced; and (5) widely accepted programming practices within the computer industry.

Id. at 709–10.

Third, copyright protection did not extend to structures already found in the public domain. The court reasoned that materials in the public domain, such as elements of a computer program that have been freely accessible, cannot be appropriated. *Ibid.* Ultimately, in the case before it, the Second Circuit held that after removing unprotectable elements using the criteria discussed above, only a few lists and macros in accused product were similar to the copied product, and their impact on the program was not large enough to declare copyright infringement. *Id.* at 714–15. The copyright claim, in short, failed.

The Tenth Circuit elaborated on the abstract-filtration-comparison test in *Gates Rubber Co. v. Bando Chemical Industries, Ltd.*, 9 F.3d 823 (10th Cir. 1993). There, the claimant developed a computer program that determined the proper rubber belt for a particular machine by performing complicated calculations involving numerous variables. The program used published formulas in conjunction with certain mathematical constants developed by the claimant to determine belt size. The Tenth Circuit offered the following description of a software program's structure:

> The program's architecture or structure is a description of how the program operates in terms of its various functions, which are performed by discrete modules, and how each of these modules interact with each other.

Id. at 835. As had the Second Circuit, the Tenth Circuit held that filtration should eliminate the unprotectable elements of processes, facts, public domain information, merger material, *scenes a faire* material, and other unprotectable elements suggested by the particular facts of the program under examination. For Section 102(b) processes, the court gave the following description:

> Returning then to our levels of abstraction framework, we note that processes can be found at any level, except perhaps the main purpose level of abstraction. Most commonly, processes will be found as part of the system architecture, as operations within modules, or as algorithms.

Id. at 837. The court described the *scenes a faire* doctrine for computer programs as follows:

> The *scenes a faire* doctrine also excludes from protection those elements of a program that have been dictated by external factors. In the area of computer programs these external factors may include: hardware standards and mechanical specifications, software standards and compatibility requirements, *Sega Enterprises Ltd. v. Accolade, Inc.*, 977 F.2d 1510, 1525–27 (9th Cir. 1993), computer manufacturer design standards, target industry practices and demands, and computer industry programming practices.
>
> * * *
>
> We recognize that the *scenes a faire* doctrine may implicate the protectability of interfacing and that this topic is very sensitive and has the potential to effect [sic] widely the law of computer copyright. This appeal does not require us to determine the scope of the *scenes a faire* doctrine as it relates to interfacing and accordingly we refrain from discussing the issue.

Id. at 838 & n.14 (all citations omitted except *Sega*). Like the Second Circuit, the Tenth Circuit also listed many external considerations — such as compatibility, computer industry programming practices, and target industry practices and demands — that would exclude elements from copyright protection under the *scenes a faire* doctrine. Ultimately, the Tenth Circuit remanded because the district court had failed to make specific findings that fit this framework.

The First Circuit weighed in with its 1995 decision *Lotus Development Corp. v. Borland International, Inc.*,

49 F.3d 807 (1st Cir. 1995). In *Lotus*, the claimant owned the Lotus 1-2-3 spreadsheet program that enabled users to perform accounting functions electronically on a computer. Users manipulated and controlled the program via a series of menu commands, such as "Copy," "Print," and "Quit." In all, Lotus 1-2-3 had 469 commands arranged into more than 50 menus and submenus. Lotus 1-2-3 also allowed users to write "macros," whereby a user could designate a series of command choices (sequence of menus and submenus) with a single macro keystroke. Then, to execute that series of commands, the user only needed to type the single pre-programmed macro keystroke, causing the program to recall and perform the designated series of commands automatically. *Id.* at 809–10.

The accused infringer Borland developed a competing spreadsheet program. Borland included the Lotus menu command hierarchy in its program to make it compatible with Lotus 1-2-3 so that spreadsheet users who were already familiar with Lotus 1-2-3 would be able to switch to the Borland program without having to learn new commands or rewrite their Lotus macros. In so doing, Borland did not copy any of Lotus's underlying source or object code. (The opinion did not say whether the programs were written in the same language.)

The district court had ruled that the Lotus 1-2-3 menu command hierarchy was a copyrightable expression because there were many ways to construct a spreadsheet menu tree. Thus, the district court had concluded that the Lotus developers' choice and arrangement of command terms, reflected in the Lotus menu command hierarchy, constituted copyrightable expression. *Id.* at 810–11.

The First Circuit, however, held that the Lotus menu command hierarchy was not copyrightable because it was a method of operation under Section 102(b). The court explained:

> We think that "method of operation," as that term is used in § 102(b), refers to the means by which a person operates something, whether it be a car, a food processor, or a computer. Thus a text describing how to operate something would not extend copyright protection to the method of operation itself; other people would be free to employ that method and to describe it in their own words. Similarly, if a new method of operation is used rather than described, other people would still be free to employ or describe that method.

Id. at 815.

The court reasoned that because the menu command hierarchy was essential to make use of the program's functional capabilities, it should be properly categorized as a "method of operation" under Section 102(b). The court explained:

> The Lotus menu command hierarchy does not merely explain and present Lotus 1-2-3's functional capabilities to the user; it also serves as the method by which the program is operated and controlled . . . . In other words, to offer the same capabilities as Lotus 1-2-3, Borland did not have to copy Lotus's underlying code (and indeed it did not); to allow users to operate its programs in substantially the same way, however, Borland had to copy the Lotus menu command hierarchy.

> Thus the Lotus 1-2-3 code is not a uncopyrightable "method of operation."

Ibid. Thus, the court reasoned that although Lotus had made "expressive" choices of what to name the command terms and how to structure their hierarchy, it was nevertheless an uncopyrightable "method of operation." The *Lotus* decision was affirmed by an evenly divided Supreme Court (four to four).

The Federal Circuit had the opportunity to apply *Lotus* in an appeal originating from the District of Massachusetts in *Hutchins v. Zoll Medical Corp.*, 492 F.3d 1377 (Fed. Cir. 2007) (affirming summary judgment against copyright owner). In *Hutchins*, the claimant owned a program for performing CPR and argued that his copyright covered the "system of logic whereby CPR instructions are provided by computerized display, and [] the unique logic contained in [his] software program." *Id.* at 1384. The claimant argued that the accused program was similar because it "perform[ed] the same task in the same way, that is, by measuring heart activity and signaling the quantity and timing of CPR compressions to be performed by the rescuer." *Ibid.* The court of appeals rejected this argument, holding that copyright did not protect the "technologic method of treating victims by using CPR and instructing how to use CPR." *Ibid.* (citing *Lotus*).

D. Decisions in the Supreme Court and in our Circuit.

Our case is governed by the law in the Ninth Circuit and, of course, the Supreme Court. The Supreme Court missed the opportunity to address these issues in *Lotus* due to the four-to-four affirmance and has, thus, never

reached the general question. Nonetheless, *Baker*, which is still good law, provides guidance and informs how we should read Section 102(b).

Another Supreme Court decision, *Feist Publications, Inc. v. Rural Telephone Services Co., Inc.*, 499 U.S. 340 (1991), which dealt primarily with the copyrightability of purely factual compilations, provided some general principles. In *Feist*, the Supreme Court considered the copyrightability of a telephone directory comprised of names, addresses, and phone numbers organized in alphabetical order. The Supreme Court rejected the notion that copyright law was meant to reward authors for the "sweat of the brow." This meant that we should not yield to the temptation to award copyright protection merely because a lot of sweat went into the work. The Supreme Court concluded that protection only extended to the original components of an author's work. *Id.* at 353. The Supreme Court concluded:

> This inevitably means that the copyright in a factual compilation is thin. Notwithstanding a valid copyright, a subsequent compiler remains free to use the facts contained in another's publication to aid in preparing a competing work, so long as the competing work does not feature the same selection and arrangement.

Id. at 349.

Turning to our own Ninth Circuit, our court of appeals has recognized that non-literal components of a program, including the structure, sequence and organization and user interface, can be protectable under copyright depending on whether the structure, sequence and organization in question qualifies as an expression of an

idea rather than an idea itself. *Johnson Controls, Inc. v. Phoenix Control Sys., Inc.*, 886 F.2d 1173, 1175 (9th Cir. 1989). This decision arrived between the Third Circuit's *Whelan* decision and the Second Circuit's *Computer Associates* decision. *Johnson Controls* is one of Oracle's mainstays herein.

In *Johnson Controls*, the claimant developed a system of computer programs to control wastewater treatment plants. The district court found that the structure, sequence and organization of the program was expression and granted a preliminary injunction even though the accused product did not have similar source or object code. *Id.* at 1174. Therefore, the standard of review on appeal was limited to abuse of discretion and clear error. Our court of appeals affirmed the preliminary injunction, stating that the claimant's program was very sophisticated and each individual application was customized to the needs of the purchaser, indicating there may have been room for individualized expression in the accomplishment of common functions. Since there was some discretion and opportunity for creativity in the structure, the structure of the program was expression rather than an idea. *Id.* at 1175. *Johnson Controls*, however, did not elaborate on which particular structures deserved copyright protection.

In *Brown Bag Software v. Symantec Corp.*, 960 F.2d 1465 (9th Cir. 1992), our court of appeals outlined a two-part test for determining similarity between computer programs: the extrinsic and intrinsic tests. This pertained to infringement, not copyrightability. The claimant, who owned a computer program for outlining, alleged that an accused infringer copied his program's non-literal features. *Id.* at 1472. The claimant alleged that seventeen

specific features in the programs were similar. On summary judgment, the district court had found that each feature was either not protectable or not similar as a matter of law:

> The district court ruled that one group of features represented a claim of copyright in "concepts . . . fundamental to a host of computer programs" such as "the need to access existing files, edit the work, and print the work." As such, these features, which took the form of four options in the programs' opening menus, were held to be unprotectable under copyright.
>
> A second group of features involved "nine functions listed in the menu bar" and the fact that "virtually all of the functions of the PC-Outline program [ ] can be performed by Grandview." The district court declared that "these functions constitute the idea of the outlining program" and, furthermore, "[t]he expression of the ideas inherent in the features are . . . distinct." The court also held that "the similarity of using the main editing screen to enter and edit data . . . is essential to the very idea of a computer outlining program."
>
> The third group of features common to PC-Outline and Grandview concerned "the use of pull-down windows." Regarding these features, the district court made three separate rulings. The court first found that "[p]laintiffs may not claim copyright protection of an . . . expression that is, if not standard, then commonplace in the computer software industry" . . . . [and] that the

> pull-down windows of the two programs look different.

Id. at 1472–73. Our court of appeals affirmed the district court's order without elaborating on the copyrightability rulings quoted above.

In *Atari Games Corp. v. Nintendo of America Inc.*, 975 F.2d 832 (Fed. Cir. 1992), the Federal Circuit had occasion to interpret Ninth Circuit copyright precedent. In *Atari*, the claimant Nintendo sued Atari for copying the Nintendo 10NES program, which prevented the Nintendo game console from accepting unauthorized game cartridges. Atari deciphered the 10NES program through reverse engineering and developed its own program to unlock the Nintendo game console. Atari's new program generated signals indistinguishable from 10NES but was written in a different programming language. *Id.* at 835–36.

Applying our Ninth Circuit precedents, *Johnson Controls* and *Brown Bag*, the Federal Circuit affirmed the district court's preliminary injunction for copyright infringement. The Federal Circuit held that the 10NES program contained copyrightable expression because it had organization and sequencing unnecessary to the unlocking function:

> Nintendo's 10NES program contains more than an idea or expression necessarily incident to an idea. Nintendo incorporated within the 10NES program creative organization and sequencing *unnecessary* to the lock and key function. Nintendo chose arbitrary programming instructions and arranged them in a unique sequence to create a purely arbitrary data

> stream. This data stream serves as the key to unlock the NES. Nintendo may protect this creative element of the 10NES under copyright.

Id. at 840 (emphasis added). The Federal Circuit stated that there were creative elements in the 10NES program

> beyond the literal expression used to effect the unlocking process. The district court defined the unprotectable 10NES idea or process as the generation of a data stream to unlock a console. This court discerns no clear error in the district court's conclusion. The unique arrangement of computer program expression which generates that data stream does not merge with the process so long as alternate expressions are available. In this case, Nintendo has produced expert testimony showing a multitude of different ways to generate a data stream which unlocks the NES console.

Ibid. (citation omitted). Thus, the Federal Circuit held that the district court did not err in concluding that the 10NES program contained protectable expression and affirmed the preliminary injunction.

Next came two decisions holding that Section 102(b) bars from copyright software interfaces necessary for interoperability. The Section 102(b) holdings arose in the context of larger holdings that it had been fair use to copy software to reverse-engineer it so as to isolate the unprotectable segments. These two decisions will now be described in detail.

In *Sega Enterprises Ltd. v. Accolade, Inc.*, 977 F.2d 1510 (9th Cir. 1992), the accused infringer had to copy object code in order to understand the interface

procedures between the Sega game console and a game cartridge, that is, how the software in the game console interacted with the software in the game cartridge to achieve compatibility. *Id.* at 1515–16. After learning and documenting these interactions (interface procedures), the accused infringer wrote its own source code to mimic those same interface procedures in its own game cartridges so that its cartridges could run on the Sega console. Our court of appeals held that the copying of object code for the purpose of achieving compatibility was fair use. Notably, in its fair-use analysis, our court of appeals *expressly held that the interface procedures for compatibility were functional aspects not copyrightable under Section 102(b)*: "Accolade copied Sega's software solely in order to discover the functional requirements for compatibility with the Genesis console — aspects of Sega's programs that are not protected by copyright. 17 U.S.C. § 102(b)." *Id.* at 1522. The court used the phrase "interface procedures," a term describing the interface between applications, multiple times to describe the functional aspect of the interaction between software programs and summarized its analysis of copyrightability as follows:

> In summary, the record clearly establishes that disassembly of the object code in Sega's video game cartridges was necessary in order to understand the functional requirements for Genesis compatibility. The *interface procedures* for the Genesis console are distributed for public use only in object code form, and are not visible to the user during operation of the video game program. Because object code cannot be read by humans, it must be disassembled, either by hand or by machine. Disassembly of object code

> necessarily entails copying. Those facts dictate our analysis of the second statutory fair use factor. If disassembly of copyrighted object code is per se an unfair use, the owner of the copyright gains a de facto monopoly over *the functional aspects of his work — aspects that were expressly denied copyright protection by Congress*. 17 U.S.C. § 102(b). In order to enjoy a lawful monopoly over the idea or functional principle underlying a work, the creator of the work must satisfy the more stringent standards imposed by the patent laws. *Bonito Boats, Inc. v. Thunder Craft Boats, Inc.*, 489 U.S. 141, 159–64, 109 S.Ct. 971, 982–84, 103 L.Ed.2d 118 (1989). Sega does not hold a patent on the Genesis console.

Sega, 977 F.2d at 1526 (emphasis added). In *Sega*, the interface procedure that was required for compatibility was "20 bytes of initialization code plus the letters S–E–G–A." *Id.* at 1524 n.7. Our court of appeals found that this interface procedure was functional and therefore not copyrightable under Section 102(b). The accused infringer Accolade was free to copy this interface procedure for use in its own games to ensure compatibility with the Sega Genesis game console. Our court of appeals distinguished the *Atari* decision, where the Federal Circuit had found that the Nintendo's 10NES security system was infringed, because there was only one signal that unlocked the Sega console, unlike the "multitude of different ways to unlock" the Nintendo console:

> We therefore reject Sega's belated suggestion that Accolade's incorporation of the code which "unlocks" the Genesis III console is not a fair

> use. Our decision on this point is entirely consistent with *Atari v. Nintendo*, 975 F.2d 832 (Fed. Cir. 1992). Although Nintendo extended copyright protection to Nintendo's 10NES security system, that system consisted of an original program which generates an arbitrary data stream "key" which unlocks the NES console. Creativity and originality went into the design of that program. *See id.* at 840. Moreover, the federal circuit concluded that there is a "multitude of different ways to generate a data stream which unlocks the NES console." *Atari*, 975 F.2d at 839. The circumstances are clearly different here. Sega's key appears to be functional. It consists merely of 20 bytes of initialization code plus the letters S–E–G–A. There is no showing that there is a multitude of different ways to unlock the Genesis III console.

Sega, 977 F.2d at 1524 n.7.

This order reads *Sega* footnote seven (quoted above) as drawing a line between copying functional aspects necessary for compatibility (not copyrightable) versus copying functional aspects unnecessary for compatibility (possibly copyrightable). Our court of appeals explained that in *Atari*, the Nintendo game console's 10NES program had had functionality *unnecessary* to the lock-and-key function. *See also Atari*, 975 F.2d at 840. Since the accused infringer Atari had copied the entire 10NES program, it also had copied aspects of the 10NES program unnecessary for compatibility between the console and game cartridges. This was inapposite to the facts of *Sega*, where the accused infringer Accolade's final product duplicated *only* the aspect of Sega's program *necessary*

for compatibility between the console and game cartridges. Thus, the holding of our court of appeals was that the aspect of a program necessary for compatibility was unprotectable, specifically invoking Section 102(b), but copyrightable expression could still exist for aspects unnecessary for compatibility.

The *Sega* decision and its compatibility reasoning was followed in a subsequent reverse-engineering decision by our court of appeals, *Sony Computer Entertainment, Inc., v. Connectix Corporation*, 203 F.3d 596 (9th Cir. 2000). The facts were somewhat different in *Sony*. There, the accused infringer Connectix did not create its own games for Sony's Playstation game console; instead, the accused infringer created an emulated environment that duplicated the interface procedures of Sony's console so that games written for Sony's console could be played on a desktop computer running the emulator. In order to do this, the accused infringer copied object code for the Sony Playstation's operating software, its BIOS program, in order to discover signals sent between the BIOS and the rest of the game console. *Id.* at 600. After uncovering these signals (again, application interfaces), the accused infringer wrote its own source code to *duplicate these interfaces* in order to create its emulator for the desktop computer. Thus, games written for the Playstation console were playable on Connectix's emulator for the desktop computer. Citing Section 102(b) and *Sega*, our court of appeals stated that the Playstation BIOS contained "unprotected functional elements," and concluded that the accused infringer's intermediate step of copying object code was fair use because it was done for the "purpose of

gaining access to the unprotected elements of Sony's software." *Id.* at 602–03.[6]

* * *

With apology for its length, the above summary of the development of the law reveals a trajectory in which enthusiasm for protection of "structure, sequence and organization" peaked in the 1980s, most notably in the Third Circuit's *Whelan* decision. That phrase has not been re-used by the Ninth Circuit since *Johnson Controls* in 1989, a decision affirming preliminary injunction. Since then, the trend of the copyright decisions has been more cautious. This trend has been driven by fidelity to Section 102(b) and recognition of the danger of conferring a monopoly by copyright over what Congress expressly warned should be conferred only by patent. This is not to say that infringement of the structure, sequence and organization is a dead letter. To the contrary, it is not a dead letter. It is to say that the *Whelan* approach has given way to the *Computer Associates* approach, including in our own circuit. *See Sega Enters., Ltd. v. Accolade, Inc.*, 977 F.2d 1510, 1525 (9th Cir. 1992); *Apple Computer, Inc. v. Microsoft Corp.*, 35 F.3d 1435, 1445 (9th Cir. 1994).

In this connection, since the CONTU report was issued in 1980, the number of software patents in force in the United States has dramatically increased from barely a thousand in 1980 to hundreds of thousands today. *See*

---

[6] *Sega* and *Sony* are not the only Ninth Circuit decisions placing a premium on functionality as indicating uncopyrightability. Other such decisions were surveyed in the summary earlier in this order. *See also Triad Sys. Corp. v. Southeastern Exp. Co.*, 64 F.3d 1330, 1336 (9th Cir. 1995); *Apple Computer, Inc. v. Microsoft Corp.*, 35 F.3d 1435, 1444 (9th Cir. 1994); *Apple Computer, Inc. v. Formula Intern., Inc.*, 725 F.2d 521, 525 (9th Cir. 1984).

Iain Cockburn, *Patents, Tickets and the Financing of Early-Stage Firms: Evidence from the Software Industry*, 18 JOURNAL OF ECONOMICS & MANAGEMENT STRATEGY 729–73 (2009). This has caused at least one noted commentator to observe:

> As software patents gain increasingly broad protection, whatever reasons there once were for broad copyright protection of computer programs disappear. Much of what has been considered the copyrightable "structure, sequence and organization" of a computer program will become a mere incident to the patentable idea of the program or of one of its potentially patentable subroutines.

Mark Lemley, *Convergence in the Law of Software Copyright?*, 10 HIGH TECHNOLOGY LAW JOURNAL 1, 26–27 (1995). Both Oracle and Sun have applied for and received patents that claim aspects of the Java API. *See, e.g.*, U.S. Patents 6,598,093 and 7,006,855. (These were not asserted at trial.)[7]

---

[7] The issue has been debated in the journals. For example, Professor Pamela Samuelson has argued that Section 102(b) codified the *Baker* exclusion of procedures, processes, systems, and methods of operation for computer programs as well as the pre-*Baker* exclusion of high-level abstractions such as ideas, concepts, and principles. Pamela Samuelson, *Why Copyright Law Excludes Systems and Processes from the Scope of Protection*, 85 TEX. L. REV. 1921 (2007). In contrast, Professor David Nimmer (the son of Professor Melville Nimmer) has argued that Section 102(b) should not deny copyright protection to "the expression" of a work even if that work happens to consist of an idea, procedure or process. 1-2 NIMMER ON COPYRIGHT § 2.03[D] (internal citations omitted). Similarly, Professor Jane Ginsburg has argued that the Section 102(b) terms "process," "system," and "method of operation" should not be understood

* * *

In view of the foregoing, this order concludes that our immediate case is controlled by these principles of copyright law:

- Under the merger doctrine, when there is only one (or only a few) ways to express something, then no one can claim ownership of such expression by copyright.
- Under the names doctrine, names and short phrases are not copyrightable.
- Under Section 102(b), copyright protection never extends to any idea, procedure, process, system, method of operation or concept regardless of its form. Functional elements essential for interoperability are not copyrightable.
- Under *Feist*, we should not yield to the temptation to find copyrightability merely to reward an investment made in a body of intellectual property.

APPLICATION OF CONTROLLING LAW TO CONTROLLING FACTS

All agree that everyone was and remains free to program in the Java language itself. All agree that Google was free to use the Java language to write its own API.

---

literally for computer programs. Jane Ginsburg, *Four Reasons and a Paradox: The Manifest Superiority of Copyright Over Sui Generis Protection of Computer Software*, 94 COLUM. L. REV. 2559, 2569–70 (1994).

While Google took care to provide fresh line-by-line implementations (the 97 percent), it generally replicated the overall name organization and functionality of 37 packages in the Java API (the three percent). The main issue addressed herein is whether this violated the Copyright Act and more fundamentally whether the replicated elements were copyrightable in the first place.

This leads to the first holding central to this order and it concerns the method level. The reader will remember that a method is like a subroutine and over six thousand are in play in this proceeding. As long as the specific code written to implement a method is different, anyone is free under the Copyright Act to write his or her own method to carry out exactly the same function or specification of any and all methods used in the Java API. Contrary to Oracle, copyright law does not confer ownership over any and all ways to implement a function or specification, no matter how creative the copyrighted implementation or specification may be. The Act confers ownership only over the specific way in which the author wrote out his version. Others are free to write their own implementation to accomplish the identical function, for, importantly, ideas, concepts and functions cannot be monopolized by copyright.

To return to our example, one method in the Java API carries out the function of comparing two numbers and returning the greater. Google — and everyone else in the world — was and remains free to write its own code to carry out the identical function so long as the implementing code in the method body is different from the copyrighted implementation. This is a simple example, but even if a method resembles higher mathematics, everyone is still free to try their hand at writing a different

implementation, meaning that they are free to use the same inputs to derive the same outputs (while throwing the same exceptions) so long as the implementation in between is their own. The House Report, quoted above, stated in 1976 that "the actual processes or methods embodied in the program are not within the scope of the copyright law." H.R. REP. NO. 94-1476, at 57 (1976).

Much of Oracle's evidence at trial went to show that the design of methods in an API was a creative endeavor. Of course, that is true. Inventing a new method to deliver a new output can be creative, even inventive, including the choices of inputs needed and outputs returned. The same is true for classes. But such inventions — at the concept and functionality level — are protectable only under the Patent Act. The Patent and Trademark Office examines such inventions for validity and if the patent is allowed, it lasts for twenty years. Based on a single implementation, Oracle would bypass this entire patent scheme and claim ownership over any and all ways to carry out methods for 95 years — without any vetting by the Copyright Office of the type required for patents. This order holds that, under the Copyright Act, no matter how creative or imaginative a Java method specification may be, the entire world is entitled to use the same method specification (inputs, outputs, parameters) so long as the line-by-line implementations are different. To repeat the Second Circuit's phrasing, "there might be a myriad of ways in which a programmer may . . . express the idea embodied in a given subroutine." *Computer Associates*, 982 F.2d at 708. The method specification is the *idea*. The method

implementation is the *expression*. No one may monopolize the *idea*.[8]

To carry out any given function, the method specification as set forth in the declaration *must be identical* under the Java rules (save only for the choices of argument names). Any other declaration would carry out some *other* function. The declaration requires precision. Significantly, when there is only one way to write something, the merger doctrine bars anyone from claiming exclusive copyright ownership of that expression. Therefore, there can be no copyright violation in using the identical declarations. Nor can there be any copyright violation due to the *name* given to the method (or to the arguments), for under the law, names and short phrases cannot be copyrighted.

In sum, Google and the public were and remain free to write their own implementations to carry out exactly the same functions of all methods in question, using exactly the same method specifications and names. Therefore, at the method level — the level where the heavy lifting is done — Google has violated no copyright, it being undisputed that Google's implementations are different.

As for classes, the rules of the language likewise insist on giving names to classes and the rules insist on strict syntax and punctuation in the lines of code that declare a class. As with methods, for any desired functionality, the

---

[8] Each method has a singular purpose or function, and so, the basic function or purpose of a method will be an unprotectable process. *Gates Rubber Co. v. Bando Chemical Industries, Ltd.*, 9 F.3d 823, 836 (10th Cir. 1993); *see Apple Computer, Inc. v. Formula Intern. Inc.*, 725 F.2d 521, 525 (9th Cir. 1984) (holding that while a particular set of instructions is copyrightable, the underlying computer process is not).

declaration line will *always* read the same (otherwise the functionality would be different) — save only for the name, which cannot be claimed by copyright. Therefore, under the law, the declaration line cannot be protected by copyright. This analysis is parallel to the analysis for methods. This now accounts for virtually all of the three percent of similar code.

* * *

Even so, the second major copyright question is whether Google was and remains free to group its methods in the same way as in Java, that is, to organize its Android methods under the same class and package scheme as in Java. For example, the Math classes in both systems have a method that returns a cosine, another method that returns the larger of two numbers, and yet another method that returns logarithmic values, and so on. As Oracle notes, the rules of Java did not insist that these methods be grouped together in any particular class. Google could have placed its trigonometric function (or any other function) under a class other than Math class. Oracle is entirely correct that the rules of the Java language did not require that the same grouping pattern (or even that they be grouped at all, for each method could have been placed in a stand-alone class).[9]

---

[9] As to the groupings of methods within a class, Google invokes the *scenes a faire* doctrine. That is, Google contends that the groupings would be so expected and customary as to be permissible under the *scenes a faire* doctrine. For example, the methods included under the Math class are typical of what one would expect to see in a group of math methods. Just as one would expect certain items in the alcove for nuts, bolts and screws in a hardware store, one would expect the methods of the math class to be in, say, a typical math class. At trial, however, neither side presented evidence from which we can now say

Oracle's best argument, therefore, is that while no single name is copyrightable, Java's overall system of organized names — covering 37 packages, with over six hundred classes, with over six thousand methods — is a "taxonomy" and, therefore, copyrightable under *American Dental Association v. Delta Dental Plans Association*, 126 F.3d 977 (7th Cir. 1997). There was nothing in the rules of the Java language that required that Google replicate the same groupings even if Google was free to replicate the same functionality.[10]

The main answer to this argument is that while the overall scheme of file name organization resembles a taxonomy, it is *also* a command structure for a system or method of operation of the application programming interface. The commands are (and must be) in the form

java.package.Class.method()

---

that the same is true for all the other hundreds of classes at issue. Therefore, it is impossible to say on this record that *all* of the classes and their contents are typical of such classes and, on this record, this order rejects Google's global argument based on *scenes a faire*.

[10] This is a good place to point out that while the groupings appear to be the same, when we drill down into the detail code listings, we see that the actual sequences of methods in the listings are different. That is, the sequence of methods in the class Math in Android is different from the sequence in the same class in Java, although all of the methods in the Java version can be found somewhere in the Android version, at least as shown in their respective listings (TX 47.101, TX 623.101). The Court has not compared all six-hundred-plus classes. Nor has any witness or counsel so far on the record. Oracle does not, however, contend that the actual sequences would track method-for-method and it has not so proven. This detailed observation, however, does not change the fact that all of the methods in the Java version can be found somewhere in the Android version, classified under the same classes.

and each calls into action a pre-assigned function.[11]

To repeat, Section 102(b) states that "in no case does copyright protection for an original work of authorship extend to any idea, procedure, process, system, method of operation . . . regardless of the form . . . ." That a system or method of operation has thousands of commands arranged in a creative taxonomy does not change its character as a method of operation. Yes, it is creative. Yes, it is original. Yes, it resembles a taxonomy. But it is nevertheless a command structure, a system or method of operation — a long hierarchy of over six thousand commands to carry out pre-assigned functions. For that reason, it cannot receive copyright protection — patent protection perhaps — but not copyright protection.

* * *

Interoperability sheds further light on the character of the command structure as a system or method of operation. Surely, millions of lines of code had been written in Java before Android arrived. These programs necessarily used the java.package.Class.method( ) command format. These programs called on all or some of the specific 37 packages at issue and necessarily used the command structure of names at issue. Such code was owned by the developers themselves, not by Oracle. *In order for at least some of this code to run on Android, Google was required to provide the same java.package.Class.method() command system using the same names with the same "taxonomy" and with the same functional specifications*. Google replicated what was necessary to achieve a degree of interoperability —

[11] The parentheses indicate that inputs/arguments may be included in the command.

but no more, taking care, as said before, to provide its own implementations.

That interoperability is at the heart of the command structure is illustrated by Oracle's preoccupation with what it calls "fragmentation," meaning the problem of having imperfect interoperability among platforms. When this occurs, Java-based applications may not run on the incompatible platforms. For example, Java-based code using the replicated parts of the 37 API packages will run on Android but will not if a 38th package is needed. Such imperfect interoperability leads to a "fragmentation" — a Balkanization — of platforms, a circumstance which Sun and Oracle have tried to curb via their licensing programs. In this litigation, Oracle has made much of this problem, at times almost leaving the impression that if only Google had replicated *all* 166 Java API packages, Oracle would not have sued. While fragmentation is a legitimate business consideration, it begs the question whether or not a license was required in the first place to replicate some or all of the command structure. (This is especially so inasmuch as Android has not carried the Java trademark, and Google has not held out Android as fully compatible.) The immediate point is this: fragmentation, imperfect interoperability, and Oracle's angst over it illustrate the character of the command structure as a functional system or method of operation.

In this regard, the Ninth Circuit decisions in *Sega* and *Sony*, although not on all fours, are close analogies. Under these two decisions, interface procedures required for interoperability were deemed "functional requirements for compatibility" and were not copyrightable under Section 102(b). Both decisions held that interface procedures that were necessary to duplicate in order to

achieve interoperability were functional aspects not copyrightable under Section 102(b). Here, the command structure for the 37 packages (including inheritances and exception throws), when replicated, at least allows interoperability of code using the replicated commands. To the extent of the 37 packages — which, after all, is the extent of Oracle's copyright claim — *Sega* and *Sony* are analogous. Put differently, if someone could duplicate the interfaces of the Sony BIOS in order to run the Playstation games on desktops (taking care to write its own implementations), then Google was free to duplicate the command structure for the 37 packages in Android in order to accommodate third-party source code relying on the 37 packages (taking care to write its own implementations). Contrary to Oracle, "full compatibility" is not relevant to the Section 102(b) analysis. In *Sony*, the accused product implemented only 137 of the Playstation BIOS's 242 functions because those were the only functions invoked by the games tested. Connectix's Opening Appellate Brief at 18, available at 1999 WL 33623860, (9th Cir. May 27, 1999). Our court of appeals held that the accused product "itself infringe[d] no copyright." *Sony*, 203 F.3d at 608 n.11. This parallels Google's decision to implement some but not all of the Java API packages in Android.

* * *

This explains why *American Dental Association v. Delta Dental Plans Association*, 126 F.3d 977 (7th Cir. 1997), is not controlling. Assuming arguendo that a taxonomy is protectable by copyright in our circuit, *see Practice Mgmt. Info. Corp. v. Am. Med. Ass'n*, 121 F.3d 516 (9th Cir. 1997), the taxonomy in *ADA* had nothing to do with computer programs. It was not a system of

commands, much less a system of commands for a computer language. The taxonomy there subdivided the universe of all dental procedures into an outline of numbered categories with English-language descriptions created by the ADA. This was then to be used by insurance companies and dentists to facilitate billings. By contrast, here the taxonomy is composed entirely of a system of commands to carry out specified computer functions. For a similar reason, Oracle's analogy to stealing the plot and character from a movie is inapt, for movies involve no "system" or "method of operation" — scripts are entirely creative.

In *ADA*, Judge Frank Easterbrook (writing for the panel) suggested that a "system" under Section 102(b) had to come with "instructions for use." 126 F.3d at 980. Because the taxonomy there at issue had no instructions for use, among other reasons, it was held not to be a system. By contrast, the API at issue here does come with instructions for use, namely, the documentation and embedded comments that were much litigated at trial. They describe every package, class and method, what inputs they need, and what outputs they return — the classic form of instructions for use.

In our circuit, the structure, sequence and organization of a computer program may (or may not) qualify as a protectable element depending on the "particular facts of each case" and always subject to exclusion of unprotectable elements. *Johnson Controls v. Phoenix Control Sys.*, 886 F.2d 1173, 1175 (9th Cir. 1989). Contrary to Oracle, *Johnson Controls* did not hold that all structure, sequence and organization in all computer programs are within the protection of a copyright. On a motion for preliminary injunction, the district court found

that the structure, sequence and organization of the copyrighted program, on the facts there found, deserved copyright protection. (The structure, sequence and organization features found protectable were not described in the appellate decision.) On an appeal from the preliminary injunction, our court of appeals merely said no clear error had occurred. Again, the appellate opinion stated that the extent to which the structure, sequence and organization was protectable depended on the facts and circumstances of each case. The circumstances there are not the circumstances here.

In closing, it is important to step back and take in the breadth of Oracle's claim. Of the 166 Java packages, 129 were not violated in any way. Of the 37 accused, 97 percent of the Android lines were new from Google and the remaining three percent were freely replicable under the merger and names doctrines. Oracle must resort, therefore, to claiming that it owns, by copyright, the exclusive right to any and all possible implementations of the taxonomy-like command structure for the 166 packages and/or any subpart thereof — even though it copyrighted only one implementation. To accept Oracle's claim would be to allow anyone to copyright one version of code to carry out a system of commands and thereby bar all others from writing their own different versions to carry out all or part of the same commands. No holding has ever endorsed such a sweeping proposition.

CONCLUSION

This order does not hold that Java API packages are free for all to use without license. It does not hold that the structure, sequence and organization of all computer programs may be stolen. Rather, it holds on the specific facts of this case, the particular elements replicated by

Google were free for all to use under the Copyright Act. Therefore, Oracle's claim based on Google's copying of the 37 API packages, including their structure, sequence and organization is **DISMISSED**. To the extent stated herein, Google's Rule 50 motions regarding copyrightability are **GRANTED** (Dkt. Nos. 984, 1007). Google's motion for a new trial on copyright infringement is **DENIED AS MOOT** (Dkt. No. 1105).

IT IS SO ORDERED.

Dated: May 31, 2012.

/s/

WILLIAM ALSUP, UNITED STATES DISTRICT JUDGE

Appendix H

IN THE UNITED STATES DISTRICT COURT FOR THE NORTHERN DISTRICT OF CALIFORNIA

ORACLE AMERICA, INC., Plaintiff, v. GOOGLE INC., Defendant.	No. C 10-03561 WHA

FINDINGS OF FACT AND CONCLUSIONS OF LAW ON EQUITABLE DEFENSES

This order addresses Google's equitable defenses, (1) laches; (2) equitable estoppel; (3) implied license; and (4) waiver, for both copyright and patent infringement. In light of the Court's accompanying ruling that the structure, sequence and organization of the Java API packages are not copyrightable, and the jury's verdict of patent non-infringement, Google's equitable defenses are moot, at least pending appeal. Nonetheless, even in the event of a remand on one or more other liability issues, it is so unlikely that the remand could affect the calculus of the defenses of implied license and waiver that this order will go ahead and clear those away, leaving open the defenses of laches and equitable estoppel.

1. IMPLIED LICENSE.

An implied license requires a finding of an affirmative grant of consent or permission. Though rare, consent can be inferred from a course of conduct between parties. *Wang Labs., Inc. v. Mitsubishi Elecs.*, 103 F.3d 1571,

1581–82 (Fed. Cir. 1997). As with the other equitable defenses, there must be a nexus between the alleged conduct giving rise to the implied license and the infringing action. *Ibid.* In the context of both copyrights and patents, circumstances giving rise to an implied license are exceedingly narrow. *See Id.* at 1251–52; *A&M Records, Inc. v. Napster, Inc.*, 239 F.3d 1004, 1026 (9th Cir. 2001).

The requisite nexus between Oracle and/or Sun's conduct and Google's infringement has not been proved. Google agrees that Oracle and/or Sun did not specifically and affirmatively grant permission to Google to use the structure, sequence and arrangement of the 37 API packages (Dkt. No. 1079 ¶ 183). The same is true for the asserted patents. This leaves open only the "course of conduct" theory, which also fails.

Google's evidence of implied consent at most establishes Oracle's inaction. Google's equitable defenses rest primarily on a November 2007 blog post by Sun's CEO congratulating Google on the release of Android, as well as similar positive statements by Sun executives thereafter. Congratulatory statements do not fall under the narrow circumstances proscribed by our court of appeals. Even if Google understood Oracle and/or Sun's conduct to condone use of the Java API packages, the "course of conduct" must be assessed for an affirmative grant of such consent. None is apparent from the evidence Google presented here. Google has supplied no relevant authority that would support a finding in its favor on these facts. Furthermore, from the present record it would be impossible to determine the scope of any implied license. Under Google's theory, infringement is excused as to *any* aspect of Android because the whole of the platform was

generally applauded by Sun. Such a finding is not supported by precedent. The parties negotiated for a real license but the talks collapsed and no license was given. It would be most bizarre to somehow find an implied license in this scenario.

2. WAIVER.

To prevail on a waiver defense, Google must show by a preponderance of the evidence that Oracle and/or Sun, with full knowledge of the material facts, intentionally relinquished its rights to enforce the rights it now asserts. Waiver of a known right must be "manifested by some overt act indicating an intention to abandon that right." *Micro Star v. Formgen, Inc.*, 154 F.3d 1107, 1114 (9th Cir. 1998). The parties agree that inaction alone is insufficient to show waiver.

This order finds Google has not met its burden of proving an overt act by Oracle and/or Sun indicating its intention to abandon all rights to the Java platform, or to the specific technology at issue here. Google's best evidence on the issue of waiver is Jonathan Schwartz's testimony that Sun made a decision to not sue Google following the release of Android. This decision, however, is not an overt act. So long as it did not induce reliance by Google, Sun was free to change its mind and assert its rights within the statute of limitations period. The several congratulatory communications do not, as discussed above, constitute a clear indication that Oracle and/or Sun intended to relinquish its rights as to the entirety of its platform. Google concedes Oracle continued and continues to assert its rights as to other aspects of the platform such as the language specification and code (Dkt. No. 1079 ¶¶ 58–60). Save for a total relinquishment, Google has to prove an overt act by Oracle and/or Sun relaying its intent

to abandon rights as to the specific elements asserted here. The evidence is devoid of any such showing.

3. EQUITABLE ESTOPPEL AND LACHES.

There remains a possibility that these two equitable defenses can be revived on remand. Both these defenses are based, in part, on what intellectual property rights Sun and Oracle had in Java, and more specifically, rights to preventing others from using the structure, sequence and organization of the API packages. In the event of a remand, this could affect the calculus involving the defenses and the judge will reserve on deciding these defenses. If that occurs, those issues will likely be decided based on the existing trial record.

CONCLUSION

For the reasons stated, Google's defenses of implied license and waiver are rejected on the merits and Google's defenses of equitable estoppel and laches are denied as moot.

IT IS SO ORDERED.

Dated: May 31, 2012.

/s/

WILLIAM ALSUP, UNITED STATES DISTRICT JUDGE

Appendix I

IN THE UNITED STATES DISTRICT COURT FOR THE NORTHERN DISTRICT OF CALIFORNIA

ORACLE AMERICA, INC., Plaintiff, v. GOOGLE INC., Defendant.	No. C 10-03561 WHA

FINAL JUDGMENT

The pleadings in this action asserted the following: Oracle asserted infringement of seven patents, U.S. Patent Nos. 6,125,447; 6,192,476; 5,966,702; 7,426,720; RE38,104; 6,910,205; and 6,061,520. Oracle further asserted infringement of its copyrights in the code, documentation, specifications, libraries, and other materials that comprise the Java platform. Oracle alleged that the infringed elements included Java method and class names, definitions, organization, and parameters; the structure, organization and content of Java class libraries; and the content and organization of Java's documentation. In turn, Google asserted declaratory judgments of non-infringement and invalidity, and equitable defenses. Before trial, Oracle dismissed with prejudice all claims for relief based on the '447, '476, '702, '720, and '205 patents. During trial, Google abandoned claims for relief for invalidity declarations as to the '104 and '520 patents.

Based upon the verdicts by the jury and orders entered by the Court, it is now **ORDERED, ADJUDGED, AND DECREED** that:

With respect to Oracle's claim for relief and Google's counterclaim for declaratory judgment of non-infringement for the '520 and '104 patents, judgment is entered for Google and against Oracle. With respect to Google's counterclaims for declaratory judgment of invalidity for the '520 and '104 patents, judgment is entered for Oracle and against Google, such counterclaims having been abandoned during trial. With respect to the five remaining patents, claims for relief by Oracle were completely dismissed with prejudice by Oracle (and may not be resurrected except as indicated in the orders of May 3, 2011, and March 2, 2012, with respect to new products). In this regard, it is the intent of this judgment and order that general principles of merger of claims into the judgment and res judicata shall be applicable.

With respect to Oracle's claim for relief for copyright infringement, judgment is entered in favor of Google and against Oracle except as follows: the rangeCheck code in TimSort.java and ComparableTimSort.java, and the eight decompiled files (seven "Impl.java" files and one "ACL" file), as to which judgment for Oracle and against Google is entered in the amount of zero dollars (as per the parties' stipulation).

With respect to Google's equitable defenses, judgment is entered for Oracle and against Google as to waiver and implied license. As to equitable estoppel and laches, no ruling need be made due to mootness.

IT IS SO ORDERED.

Dated: June 20, 2012.

/s/

WILLIAM ALSUP, UNITED STATES DISTRICT JUDGE

Appendix J

IN THE UNITED STATES DISTRICT COURT FOR THE NORTHERN DISTRICT OF CALIFORNIA

ORACLE AMERICA, INC., Plaintiff, v. GOOGLE INC., Defendant.	No. C 10-03561 WHA

ORDER DENYING MOTION FOR JUDGMENT AS A MATTER OF LAW AND NEW TRIAL

Plaintiff Oracle America, Inc. moves for judgment as a matter of law under Rule 50(b), or in the alternative, for a new trial under Rule 59, on issues of patent and copyright infringement. Oracle's arguments are repetitive of its Rule 50(a) motions and rely on the same evidence. For reasons stated in prior orders (Dkt. Nos. 1119, 1165, 1201, 1202, 1203, 1211), Oracle's motion is **DENIED**. The hearing scheduled for July 26 is **VACATED**.

IT IS SO ORDERED.

Dated: July 13, 2012.

/s/

WILLIAM ALSUP, UNITED STATES DISTRICT JUDGE

Appendix K

IN THE UNITED STATES DISTRICT COURT FOR THE NORTHERN DISTRICT OF CALIFORNIA

ORACLE AMERICA, INC., Plaintiff, v. GOOGLE INC., Defendant.	No. C 10-03561 WHA

ORDER DENYING MOTION FOR JUDGMENT AS A MATTER OF LAW AND NEW TRIAL

Defendant Google Inc. moves for judgment as a matter of law under Rule 50(b), or in the alternative, for a new trial under Rule 59, on copyright issues regarding the rangeCheck function and decompiled files. Google's arguments are repetitive of its Rule 50(a) motion and rely on the same evidence. For reasons stated in the prior orders (Dkt. Nos. 1119, 1123), Google's motion is **DENIED**.

The Court takes this opportunity to state that it will take no further action regarding the subject of payments by the litigants to commentators and journalists and reassures both sides that no commentary has in any way influenced the Court's orders and ruling herein save and except for any treatise or article expressly cited in an order or ruling.

IT IS SO ORDERED.

Dated: September 4, 2012.

/s/

WILLIAM ALSUP, UNITED STATES DISTRICT JUDGE

Appendix L

United States Court of Appeals
for the Federal Circuit

ORACLE AMERICA, INC.,

Plaintiff-Appellant,

v.

GOOGLE LLC,

Defendant-Cross-Appellant.

2017-1118, 2017-1202

Appeals from the United States District Court for the Northern District of California in No. 3:10-cv-03561-WHA, Judge William H. Alsup.

ON PETITION FOR REHEARING EN BANC

---

* * *

Before: PROST, *Chief Judge*, NEWMAN, PLAGER*, LOURIE, DYK, MOORE, O'MALLEY, REYNA, WALLACH, TARANTO, CHEN, HUGHES, and STOLL, *Circuit Judges*.

PER CURIAM.

ORDER

The petition was first referred as a petition for rehearing to the panel that heard the appeal, and thereafter the petition for rehearing en banc was referred to the circuit judges who are in regular active service.

Upon consideration thereof,

IT IS ORDERED THAT:

The petition for panel rehearing is denied.

The petition for rehearing en banc is denied.

The mandate of the court will issue on September 4, 2018.

FOR THE COURT

August 28, 2018
Date

/s/Peter R. Marksteiner
Peter R. Marksteiner
Clerk of Court

Appendix M

17 U.S.C. § 101

Definitions

Except as otherwise provided in this title, as used in this title, the following terms and their variant forms mean the following:

An "anonymous work" is a work on the copies or phonorecords of which no natural person is identified as author.

An "architectural work" is the design of a building as embodied in any tangible medium of expression, including a building, architectural plans, or drawings. The work includes the overall form as well as the arrangement and composition of spaces and elements in the design, but does not include individual standard features.

"Audiovisual works" are works that consist of a series of related images which are intrinsically intended to be shown by the use of machines, or devices such as projectors, viewers, or electronic equipment, together with accompanying sounds, if any, regardless of the nature of the material objects, such as films or tapes, in which the works are embodied.

The "Berne Convention" is the Convention for the Protection of Literary and Artistic Works, signed at Berne, Switzerland, on September 9, 1886, and all acts, protocols, and revisions thereto.

The "best edition" of a work is the edition, published in the United States at any time before the date of deposit, that the Library of Congress determines to be most suitable for its purposes.

A person's "children" are that person's immediate offspring, whether legitimate or not, and any children legally adopted by that person.

A "collective work" is a work, such as a periodical issue, anthology, or encyclopedia, in which a number of contributions, constituting separate and independent works in themselves, are assembled into a collective whole.

A "compilation" is a work formed by the collection and assembling of preexisting materials or of data that are selected, coordinated, or arranged in such a way that the resulting work as a whole constitutes an original work of authorship. The term "compilation" includes collective works.

A "computer program" is a set of statements or instructions to be used directly or indirectly in a computer in order to bring about a certain result.

"Copies" are material objects, other than phonorecords, in which a work is fixed by any method now known or later developed, and from which the work can be perceived, reproduced, or otherwise communicated, either directly or with the aid of a machine or device. The term "copies" includes the material object, other than a phonorecord, in which the work is first fixed.

"Copyright owner", with respect to any one of the exclusive rights comprised in a copyright, refers to the owner of that particular right.

A "Copyright Royalty Judge" is a Copyright Royalty Judge appointed under section 802 of this title, and includes any individual serving as an interim Copyright Royalty Judge under such section.

A work is "created" when it is fixed in a copy or phonorecord for the first time; where a work is prepared over a period of time, the portion of it that has been fixed at any particular time constitutes the work as of that time, and where the work has been prepared in different versions, each version constitutes a separate work.

A "derivative work" is a work based upon one or more preexisting works, such as a translation, musical arrangement, dramatization, fictionalization, motion picture version, sound recording, art reproduction, abridgment, condensation, or any other form in which a work may be recast, transformed, or adapted. A work consisting of editorial revisions, annotations, elaborations, or other modifications which, as a whole, represent an original work of authorship, is a "derivative work".

A "device", "machine", or "process" is one now known or later developed.

A "digital transmission" is a transmission in whole or in part in a digital or other non-analog format.

To "display" a work means to show a copy of it, either directly or by means of a film, slide, television image, or any other device or process or, in the case of a motion picture or other audiovisual work, to show individual images nonsequentially.

An "establishment" is a store, shop, or any similar place of business open to the general public for the primary purpose of selling goods or services in which the majority of the gross square feet of space that is nonresidential is used for that purpose, and in which nondramatic musical works are performed publicly.

The term "financial gain" includes receipt, or expectation of receipt, of anything of value, including the receipt of other copyrighted works.

A work is "fixed" in a tangible medium of expression when its embodiment in a copy or phonorecord, by or under the authority of the author, is sufficiently permanent or stable to permit it to be perceived, reproduced, or otherwise communicated for a period of more than transitory duration. A work consisting of sounds, images, or both, that are being transmitted, is "fixed" for purposes of this title if a fixation of the work is being made simultaneously with its transmission.

A "food service or drinking establishment" is a restaurant, inn, bar, tavern, or any other similar place of business in which the public or patrons assemble for the primary purpose of being served food or drink, in which the majority of the gross square feet of space that is nonresidential is used for that purpose, and in which nondramatic musical works are performed publicly.

The "Geneva Phonograms Convention" is the Convention for the Protection of Producers of Phonograms Against Unauthorized Duplication of Their Phonograms, concluded at Geneva, Switzerland, on October 29, 1971.

The "gross square feet of space" of an establishment means the entire interior space of that establishment, and any adjoining outdoor space used to serve patrons, whether on a seasonal basis or otherwise.

The terms "including" and "such as" are illustrative and not limitative.

An "international agreement" is—

(1) the Universal Copyright Convention;

(2) the Geneva Phonograms Convention;

(3) the Berne Convention;

(4) the WTO Agreement;

(5) the WIPO Copyright Treaty;

(6) the WIPO Performances and Phonograms Treaty; and

(7) any other copyright treaty to which the United States is a party.

A "joint work" is a work prepared by two or more authors with the intention that their contributions be merged into inseparable or interdependent parts of a unitary whole.

"Literary works" are works, other than audiovisual works, expressed in words, numbers, or other verbal or numerical symbols or indicia, regardless of the nature of the material objects, such as books, periodicals, manuscripts, phonorecords, film, tapes, disks, or cards, in which they are embodied.

The term "motion picture exhibition facility" means a movie theater, screening room, or other venue that is being used primarily for the exhibition of a copyrighted motion picture, if such exhibition is open to the public or is made to an assembled group of viewers outside of a normal circle of a family and its social acquaintances.

"Motion pictures" are audiovisual works consisting of a series of related images which, when shown in succession, impart an impression of motion, together with accompanying sounds, if any.

To "perform" a work means to recite, render, play, dance, or act it, either directly or by means of any device or process or, in the case of a motion picture or other audiovisual work, to show its images in any sequence or to make the sounds accompanying it audible.

A "performing rights society" is an association, corporation, or other entity that licenses the public performance of nondramatic musical works on behalf of copyright owners of such works, such as the American Society of Composers, Authors and Publishers (ASCAP), Broadcast Music, Inc. (BMI), and SESAC, Inc.

"Phonorecords" are material objects in which sounds, other than those accompanying a motion picture or other audiovisual work, are fixed by any method now known or later developed, and from which the sounds can be perceived, reproduced, or otherwise communicated, either directly or with the aid of a machine or device. The term "phonorecords" includes the material object in which the sounds are first fixed.

"Pictorial, graphic, and sculptural works" include two-dimensional and three-dimensional works of fine, graphic, and applied art, photographs, prints and art reproductions, maps, globes, charts, diagrams, models, and technical drawings, including architectural plans. Such works shall include works of artistic craftsmanship insofar as their form but not their mechanical or utilitarian aspects are concerned; the design of a useful article, as defined in this section, shall be considered a pictorial, graphic, or sculptural work only if, and only to the extent that, such design incorporates pictorial, graphic, or sculptural features that can be identified separately from, and are capable of existing independently of, the utilitarian aspects of the article.

For purposes of section 513, a "proprietor" is an individual, corporation, partnership, or other entity, as the case may be, that owns an establishment or a food service or drinking establishment, except that no owner or operator of a radio or television station licensed by the Federal Communications Commission, cable system or satellite carrier, cable or satellite carrier service or programmer, provider of online services or network access or the operator of facilities therefor, telecommunications company, or any other such audio or audiovisual service or programmer now known or as may be developed in the future, commercial subscription music service, or owner or operator of any other transmission service, shall under any circumstances be deemed to be a proprietor.

A "pseudonymous work" is a work on the copies or phonorecords of which the author is identified under a fictitious name.

"Publication" is the distribution of copies or phonorecords of a work to the public by sale or other transfer of ownership, or by rental, lease, or lending. The offering to distribute copies or phonorecords to a group of persons for purposes of further distribution, public performance, or public display, constitutes publication. A public performance or display of a work does not of itself constitute publication.

To perform or display a work "publicly" means—

(1) to perform or display it at a place open to the public or at any place where a substantial number of persons outside of a normal circle of a family and its social acquaintances is gathered; or

(2) to transmit or otherwise communicate a performance or display of the work to a place specified by clause (1) or to the public, by means of any device or process, whether the members of the public capable of receiving the performance or display receive it in the same place or in separate places and at the same time or at different times.

"Registration", for purposes of sections 205 (c)(2), 405, 406, 410 (d), 411, 412, and 506 (e), means a registration of a claim in the original or the renewed and extended term of copyright.

"Sound recordings" are works that result from the fixation of a series of musical, spoken, or other sounds, but not including the sounds accompanying a motion picture or other audiovisual work, regardless of the nature of the material objects, such as disks, tapes, or other phonorecords, in which they are embodied.

"State" includes the District of Columbia and the Commonwealth of Puerto Rico, and any territories to which this title is made applicable by an Act of Congress.

A "transfer of copyright ownership" is an assignment, mortgage, exclusive license, or any other conveyance, alienation, or hypothecation of a copyright or of any of the exclusive rights comprised in a copyright, whether or not it is limited in time or place of effect, but not including a nonexclusive license.

A "transmission program" is a body of material that, as an aggregate, has been produced for the sole purpose of transmission to the public in sequence and as a unit.

To "transmit" a performance or display is to communicate it by any device or process whereby images

or sounds are received beyond the place from which they are sent.

A "treaty party" is a country or intergovernmental organization other than the United States that is a party to an international agreement.

The "United States", when used in a geographical sense, comprises the several States, the District of Columbia and the Commonwealth of Puerto Rico, and the organized territories under the jurisdiction of the United States Government.

For purposes of section 411, a work is a "United States work" only if—

(1) in the case of a published work, the work is first published—

(A) in the United States;

(B) simultaneously in the United States and another treaty party or parties, whose law grants a term of copyright protection that is the same as or longer than the term provided in the United States;

(C) simultaneously in the United States and a foreign nation that is not a treaty party; or

(D) in a foreign nation that is not a treaty party, and all of the authors of the work are nationals, domiciliaries, or habitual residents of, or in the case of an audiovisual work legal entities with headquarters in, the United States;

(2) in the case of an unpublished work, all the authors of the work are nationals, domiciliaries, or habitual residents of the United States, or, in the case of an

unpublished audiovisual work, all the authors are legal entities with headquarters in the United States; or

(3) in the case of a pictorial, graphic, or sculptural work incorporated in a building or structure, the building or structure is located in the United States.

A "useful article" is an article having an intrinsic utilitarian function that is not merely to portray the appearance of the article or to convey information. An article that is normally a part of a useful article is considered a "useful article".

The author's "widow" or "widower" is the author's surviving spouse under the law of the author's domicile at the time of his or her death, whether or not the spouse has later remarried.

The "WIPO Copyright Treaty" is the WIPO Copyright Treaty concluded at Geneva, Switzerland, on December 20, 1996.

The "WIPO Performances and Phonograms Treaty" is the WIPO Performances and Phonograms Treaty concluded at Geneva, Switzerland, on December 20, 1996.

A "work of visual art" is—

(1) a painting, drawing, print, or sculpture, existing in a single copy, in a limited edition of 200 copies or fewer that are signed and consecutively numbered by the author, or, in the case of a sculpture, in multiple cast, carved, or fabricated sculptures of 200 or fewer that are consecutively numbered by the author and bear the signature or other identifying mark of the author; or

(2) a still photographic image produced for exhibition purposes only, existing in a single copy that is signed by the author, or in a limited edition of 200 copies or fewer

that are signed and consecutively numbered by the author.

A work of visual art does not include—

(A) (i) any poster, map, globe, chart, technical drawing, diagram, model, applied art, motion picture or other audiovisual work, book, magazine, newspaper, periodical, data base, electronic information service, electronic publication, or similar publication;

(ii) any merchandising item or advertising, promotional, descriptive, covering, or packaging material or container;

(iii) any portion or part of any item described in clause (i) or (ii);

(B) any work made for hire; or

(C) any work not subject to copyright protection under this title.

A "work of the United States Government" is a work prepared by an officer or employee of the United States Government as part of that person's official duties.

A "work made for hire" is—

(1) a work prepared by an employee within the scope of his or her employment; or

(2) a work specially ordered or commissioned for use as a contribution to a collective work, as a part of a motion picture or other audiovisual work, as a translation, as a supplementary work, as a compilation, as an instructional text, as a test, as answer material for a test, or as an atlas, if the parties expressly agree in a written instrument signed by them that the work shall be considered a work made for hire. For the purpose of the foregoing sentence,

a "supplementary work" is a work prepared for publication as a secondary adjunct to a work by another author for the purpose of introducing, concluding, illustrating, explaining, revising, commenting upon, or assisting in the use of the other work, such as forewords, afterwords, pictorial illustrations, maps, charts, tables, editorial notes, musical arrangements, answer material for tests, bibliographies, appendixes, and indexes, and an "instructional text" is a literary, pictorial, or graphic work prepared for publication and with the purpose of use in systematic instructional activities.

In determining whether any work is eligible to be considered a work made for hire under paragraph (2), neither the amendment contained in section 1011(d) of the Intellectual Property and Communications Omnibus Reform Act of 1999, as enacted by section 1000(a)(9) of Public Law 106–113, nor the deletion of the words added by that amendment—

(A) shall be considered or otherwise given any legal significance, or

(B) shall be interpreted to indicate congressional approval or disapproval of, or acquiescence in, any judicial determination,

by the courts or the Copyright Office. Paragraph (2) shall be interpreted as if both section 2(a)(1) of the Work Made For Hire and Copyright Corrections Act of 2000 and section 1011(d) of the Intellectual Property and Communications Omnibus Reform Act of 1999, as enacted by section 1000(a)(9) of Public Law 106–113, were never enacted, and without regard to any inaction or awareness by the Congress at any time of any judicial determinations.

The terms "WTO Agreement" and "WTO member country" have the meanings given those terms in paragraphs (9) and (10), respectively, of section 2 of the Uruguay Round Agreements Act.

17 U.S.C. § 102

Subject matter of copyright: In general

(a) Copyright protection subsists, in accordance with this title, in original works of authorship fixed in any tangible medium of expression, now known or later developed, from which they can be perceived, reproduced, or otherwise communicated, either directly or with the aid of a machine or device. Works of authorship include the following categories:

(1) literary works;

(2) musical works, including any accompanying words;

(3) dramatic works, including any accompanying music;

(4) pantomimes and choreographic works;

(5) pictorial, graphic, and sculptural works;

(6) motion pictures and other audiovisual works;

(7) sound recordings; and

(8) architectural works.

(b) In no case does copyright protection for an original work of authorship extend to any idea, procedure, process, system, method of operation, concept, principle, or discovery, regardless of the form in which it is described, explained, illustrated, or embodied in such work.

17 U.S.C. § 107

Limitations on exclusive rights: Fair Use

Notwithstanding the provisions of sections 106 and 106A, the fair use of a copyrighted work, including such use by reproduction in copies or phonorecords or by any other means specified by that section, for purposes such as criticism, comment, news reporting, teaching (including multiple copies for classroom use), scholarship, or research, is not an infringement of copyright. In determining whether the use made of a work in any particular case is a fair use the factors to be considered shall include—

(1) the purpose and character of the use, including whether such use is of a commercial nature or is for nonprofit educational purposes;

(2) the nature of the copyrighted work;

(3) the amount and substantiality of the portion used in relation to the copyrighted work as a whole; and

(4) the effect of the use upon the potential market for or value of the copyrighted work.

The fact that a work is unpublished shall not itself bar a finding of fair use if such finding is made upon consideration of all the above factors.