NO. 14-___

In the

Supreme Court of the United States

GOOGLE INC.,

PETITIONER,

v.

ORACLE AMERICA, INC.,

RESPONDENT.

On Petition for a Writ of Certiorari to the United States Court of Appeals for the Federal Circuit

PETITION FOR A WRIT OF CERTIORARI

BRUCE W. BABER
KING & SPALDING LLP
1180 Peachtree Street, NE
Atlanta, GA 30309

ROBERT A. VAN NEST
STEVEN A. HIRSCH
CHRISTA M. ANDERSON
MICHAEL S. KWUN
DAN JACKSON
KEKER & VAN NEST LLP
633 Battery Street
San Francisco, CA 94111

DARYL L. JOSEFFER
Counsel of Record
ASHLEY C. PARRISH
ADAM M. CONRAD
ETHAN P. DAVIS
KING & SPALDING LLP
1700 Pennsylvania Ave., NW
Washington, DC 20006
(202) 737-0500
djoseffer@kslaw.com

RENNY HWANG
GOOGLE INC.
1600 Amphitheatre Parkway
Mountain View, CA 94043

Counsel for Petitioner

October 6, 2014

QUESTION PRESENTED

Congress specified that "original works of authorship" are generally eligible for copyright protection, 17 U.S.C. § 102(a), but "[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." *Id.* § 102(b).

In this case, the Federal Circuit held that Section 102(b) does *not* exclude systems or methods of operation from copyright protection and that all elements of an original work are "entitled to copyright protection as long as the author had multiple ways to express the underlying idea." App. 47.

The question is:

Whether copyright protection extends to all elements of an original work of computer software, including a system or method of operation, that an author could have written in more than one way.

PARTIES TO THE PROCEEDING AND RULE 29.6 STATEMENT

Petitioner in this Court, defendant-cross appellant below, is Google Inc. Respondent in this Court, plaintiff-appellant below, is Oracle America, Inc.

Google Inc. is a publicly traded company (NASDAQ: GOOG and GOOGL). No publicly held company owns 10 percent or more of Google Inc.'s stock.

TABLE OF CONTENTS

TABLE OF AUTHORITIES

PETITION FOR A WRIT OF CERTIORARI

In 1995, this Court granted certiorari in *Lotus Development Corp. v. Borland International, Inc.*, 516 U.S. 233 (1996), to resolve the question presented here. The First Circuit had held—consistent with the plain language of 17 U.S.C. § 102(b) but in conflict with other courts of appeals—that methods of operation embodied in computer programs are not entitled to copyright protection. This Court deadlocked, affirming by an equally divided court. Two decades later, this oft-acknowledged circuit split has deepened and the question presented has grown even more important as software has become a fixture of modern life.

This case directly implicates the unanswered question in *Lotus* because the Federal Circuit extended copyright protection to systems and methods of operation, including computer interfaces. That holding would obstruct an enormous amount of innovation in fast-moving, high-technology industries, in part because innovation depends on software developers' ability to build on what has come before. If the Federal Circuit's holding had been the law at the inception of the Internet age, early computer companies could have blocked vast amounts of technological development by claiming 95-year copyright monopolies over the basic building blocks of computer design and programming. By the time Google and countless other innovators even came onto the scene, others could have locked up the field for longer than most people will live.

Consider, for example, the well-known keyboard design known as QWERTY. After Remington developed that organization of letters and symbols decades ago, it became standard for typewriters and, later, for computer keyboards. People invested time and effort in learning the QWERTY design, and then expected all keyboards to use it. Later, companies like IBM and Apple added their own additional keys to the original QWERTY layout. If Remington had brought a copyright infringement lawsuit against a keyboard manufacturer for copying the QWERTY layout, it would have failed. That design was original and creative, but Remington was not entitled to appropriate the investments made by others in learning how to use it. Otherwise, Remington could have monopolized not only the sale of its patented typewriters for the length of a patent term, but also the sale of all keyboards for nearly a century.

This case raises the same basic issue. Individual computer programmers and third-party companies develop applications (the ubiquitous "apps") for mobile devices, such as smartphones, that use the Android platform. Because many computer programmers are familiar with the Java programming language, Google allowed programmers to write programs for Android using it, including the basic shorthand commands of the Java language. As relevant here, a person writing an Android application in the Java language may use shorthand commands to cause a computer to perform certain functions, such as choosing the larger of two numbers. Programmers have made significant investments in learning these commands; they are, in effect, the basic vocabulary words of the Java language. When programmers sit down to write applications, they expect to be able to use them.

The Federal Circuit nonetheless held that, although the Java language is concededly not entitled to copyright protection, the elements of the Java platform that enable the use of the shorthand commands are copyrightable. The court based that conclusion on its view that 17 U.S.C. § 102(b) does *not* exclude systems and methods of operation from copyright protection—even though the statute unambiguously does exactly that:

> 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).

By replacing that statutory directive with a different one—that copyright protection *does* extend to a system or method of operation so long as there was more than one way to write it—the Federal Circuit usurped Congress's role, deepened a circuit split that this Court previously granted certiorari to resolve, allowed Oracle to use copyright law to evade the limits on patent protection, and thereby blocked developers from building on what has come before. The court did so, moreover, in one of the most important cases of its kind, concerning the widely-used Java language and Android platform. This Court's review is needed now, before tomorrow's innovation falls victim to the decision below.

OPINIONS BELOW

The opinion of the court of appeals is reported at 750 F.3d 1339 and reproduced at App. 1. The district court's opinion is published at 872 F. Supp. 2d 974 and reproduced at App. 100.

JURISDICTION

The court of appeals rendered its decision on May 8, 2014. On July 10, 2014, the Chief Justice extended the time for filing a petition to and including October 6, 2014. This Court has jurisdiction under 28 U.S.C. § 1254(1).

STATUTORY PROVISION INVOLVED

Section 102 of the Copyright Act provides:

> (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. . . .
>
> . . . .
>
> (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. § 102.

STATEMENT OF THE CASE

A. Java and Android

1. Sun Microsystems released the Java programming language and software platform in 1996. By making the Java language free for all to use, Sun sought to "build the biggest tent and invite as many people as possible." C.A. App. 22141.

As the district court explained, the Java language is made up of "keywords and other symbols" as well as "a set of pre-written programs to carry out various commands." App. 106. In encouraging computer programmers to learn and use Java, Sun touted those pre-written programs. C.A. App. 22137. Sun succeeded in bringing an entire generation of programmers into the Java community. App. 105. Millions of programmers invested time and effort into learning Java, making it one of the world's most popular programming languages. App. 104.

Programmers access the set of pre-written programs through the Java application programming interface ("API")—a highly structured system with its own nomenclature. The application programming interface provides access to thousands of "methods," each of which performs a function such as choosing the higher of two numbers. The methods are grouped into "classes," which are further grouped into 166 "packages" of programs—much like members of the animal kingdom are grouped into species, genuses, and families. *See* App. 106–07.

The computer code for each method "consists of the method header and the method body." App 111. The method header, also known as a "declaration," "introduces the method body" and "specif[ies] the names, parameters and functionality of the methods and classes." App. 7, 29–30. "The method body is a block of code that then implements the method" by instructing a computer how to perform the relevant function; it is therefore known as "implementing code." App. 111.

To use the methods, programmers do not need to concern themselves with the methods' implementing code. Instead, programmers use a shorthand command that causes the implementing code to perform the desired function, such as choosing the greater of two numbers. App. 33. In this way, a programmer uses the shorthand commands to operate the methods, *i.e.*, the pre-written programs. By using a method's shorthand command, a programmer can write complex software efficiently, without having to write out implementing code for each individual routine task.

These shorthand commands take the specific format "java.package.Class.method(input)." App. 112–16. For example, "java.lang.Math.max(1,2)" refers to a particular method ("max") that returns the greater of two numbers (*i.e.*, 1 and 2) and is located in the "Math" class, which in turn is located in the "java.lang" package. App. 112. Each shorthand command is derived from the method's header, which, like the command, specifies the method's name, class, package, and inputs. App. 7, 29–30.

2. Google is the lead developer of Android, one of the most popular mobile device platforms in the world. In the second quarter of 2014, third-party manufacturers such as Samsung, HTC, LG, and

Lenovo sold more than 255 million smartphones that use the Android platform. *See* International Data Corporation, Worldwide Smartphone OS Market Share (2014), *available at* http://www.idc.com/prodserv/smartphone-os-market-share.jsp.

The Android platform includes 168 packages of methods. App. 109. For every one of those methods, Google wrote or acquired original implementing code. App. 101. As the district court explained, "[a]ll agree that Google was and remains free to use the Java language itself" and that the "method implementations by Google are free of copyright issues." App. 108. The parties' dispute centers on Google's use of the same headers for the methods found in 37 of the Android packages—methods that perform "functions . . . that [a]re key to mobile devices." App. 107.

Independent computer programmers create applications for use on Android devices. Because those programmers know and often prefer to use the Java programming language, Google concluded that 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." App. 9. For those shorthand commands to work on the Android platform, Google had to replicate the method headers precisely; any change to the headers would have prevented the shorthand commands from working properly. App. 109–10. As the district court found, therefore, "Android and Java *must be* identical when it comes to those particular lines of code." App. 109. Because Google replicated only the method headers, and the body of each method (the implementing code) was written from scratch, "only three percent of the lines of code are the same" in the 37 disputed packages. App. 109.

B. The District Court Proceedings

After Oracle acquired Sun in 2010, Oracle brought this action for patent and copyright infringement. The district court entered judgment in Google's favor on Oracle's patent claims, and Oracle has not appealed that determination. App. 170.

Oracle's copyright claims accused Google of copying the method headers and the so-called "structure, sequence, and organization" of the Java application programing interface. App. 3. Oracle premised its "structure, sequence, and organization" claim on the theory that the method headers "embody the structure" of the application programming interface by specifying the name, package, and class of each method. App. 21. All of Oracle's claims thus challenged the same thing: Google's replication of the method headers. App. 101. Google responded, in part, that Java's method headers are not entitled to copyright protection because, among other things, they constitute or embody a system or method of operation—specifically, a system or method of operating the pre-written programs.

The district court considered the copyrightability of the method headers at the same time the jury considered whether—*if* the district court held the method headers to be copyrightable—Google would be liable for infringement. Those two determinations proceeded on parallel tracks, with the district court instructing the jury to assume that Oracle was entitled to copyright protection and to consider only infringement and fair use. The jury found in Oracle's favor on infringement but hung on Google's fair-use defense. App. 12.

In an extensive published opinion, the district court held that the method headers are not copyrightable and that Google is therefore entitled to judgment as a matter of law. App. 100–65. The court emphasized that Google was entitled to write its own code implementing the same functions or methods that are found in the Java application programming interface. "[C]opyright law does not confer ownership over any and all ways to implement a function or specification, no matter how creative [it] may be." App. 154.

The district court then held that the method headers, including their names and organization, are a system or method of operation excluded from copyright protection under Section 102(b) of the Copyright Act. App. 159. Because the system of method headers is a "command structure" for operating the pre-written programs, the court concluded that it might receive "patent protection perhaps—but not copyright protection." *Id.*

The district court emphasized that compatibility "sheds further light on the character of the command structure as a system or method of operation." App. 159. By the time Android came into existence, programmers had written "millions of lines of code" in Java, which "necessarily used the java.package. Class.method() command format" and "called on all or some of the specific 37 packages at issue and necessarily used the command structure of names [used by Google].” *Id.* “In order for at least some of this code to run on Android, Google was required to [use] the same java.package.Class.method() command system using the same names with the same ‘taxonomy’ and with the same functional specifications.” App. 159–60. As a result, “Google replicated what was necessary to achieve a degree of interoperability—but no more.” App. 160.

The district court found further support for its holding in other principles of copyright law. First, “[u]nder 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.” App. 153. Second, “names and short phrases are not copyrightable.” *Id.* Third, citing this Court’s decision in *Feist Publications, Inc. v. Rural Telephone Service Co.*, 499 U.S. 340, 356 (1991), the court observed that “we should not yield to the temptation to find copyrightability merely to reward an investment made in a body of intellectual property.” App. 153.

C. The Court of Appeals Proceedings

The Federal Circuit reversed, opining that copyrightability presents “a low bar” that requires only that a work be original and expressive in the sense that “the author had multiple ways to express the underlying idea.” App. 17, 47. The court noted a three-way circuit split on whether to deny copyright protection to all systems or methods of operation, grant copyright protection to essentially all elements of an original and creative computer program (including systems and methods of operation), or apply a third test known as the abstraction/filtration/ comparison test. App. 23–24.

Applying Ninth Circuit law because this case arose within that circuit and copyright law does not fall within the Federal Circuit's exclusive jurisdiction, the Federal Circuit concluded that the Ninth Circuit has adopted the abstraction/filtration/ comparison test. App. 24. After identifying a circuit split on how to apply that test, the court of appeals explained that it would: "first break down the allegedly infringed [computer] program into its constituent . . . parts"; then "sift out all non-protectable material, including ideas and expression that is necessarily incidental to those ideas"; and finally "compare[] the remaining creative expression with the allegedly infringing program." App. 25 (internal quotation marks and citation omitted).

Using that framework, the court of appeals first held that the merger doctrine is inapplicable for two reasons: merger is "irrelevant" to copyrightability and Sun could have written the method headers in more than one way. App. 30–31. The court also rejected the district court's reliance on the names-and-short-phrases doctrine. App. 33–35.

The Federal Circuit then held that Section 102(b)—which provides that "[i]n no case does copyright protection for an original work of authorship extend to any . . . system [or] method of operation," 17 U.S.C. § 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." App. 23 (internal quotation marks omitted; emphasis added). In the Federal

Circuit's view, Section 102(b) serves only to codify the "idea/expression dichotomy"—the principle that "[c]opyright protection extends only to the expression of an idea—not to the underlying idea itself." App. 18. Because "Google . . . could have designed its own . . . [application programming interface] packages if it wanted to do so," and the method headers "could have been written and organized in any number of ways and still have achieved the same functions," the court held that "Section 102(b) does not bar the packages from copyright protection." App. 49. In the court of appeals' view, "Section 102(a) and 102(b) are to be considered collectively so that certain expressions are subject to greater scrutiny." App. 23.

The court of appeals also rejected the district court's consideration of compatibility, calling it "[i]rrelevant to [c]opyrightability." App. 50. According to the Federal Circuit, compatibility, and the fact that Java's method headers "had become the effective industry standard," are only factors to be balanced with others as part of a fair-use defense. App. 45–53, 57. The court remanded for a new trial on that defense. App. 53–62.[1]

[1] The court of appeals addressed several other issues that are not relevant to the question presented in this petition. For example, the court affirmed the district court's determination that Google copied "certain small snippets of code." App. 102. By stipulation of the parties, the district court awarded no damages for that copying, which it characterized as "minor" and "innocuous." App. 118, 120.

REASONS FOR GRANTING THE PETITION

The Federal Circuit's decision warrants review for three reasons. *First*, it presents a longstanding, widely-recognized split in the courts of appeals. *Second*, the Federal Circuit's holding is in conflict with decisions of this Court and contrary to the plain language of the Copyright Act. *Third*, whether copyright may be used to evade the limits on patent protection, in order to secure 95-year (or longer) monopolies, is an exceptionally important question. This Court already recognized the certworthiness of this question by granting review in *Lotus*. Since then, the circuit split has only deepened and the question has grown even more important as software has become ubiquitous in daily life.

I. The Courts Of Appeals Are In Disarray About The Application Of Section 102(b) To Software.

The Copyright Act provides that copyright protection subsists in "original works of authorship." 17 U.S.C. § 102(a). But that protection does not extend to all elements of an original work. Section 102(b) specifies that "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 [original] work." *Id.* § 102(b).

As the Federal Circuit and other courts of appeals have acknowledged, the circuits are deeply divided on how to construe Section 102(b). *See, e.g.*, App. 23–24; *Lotus Dev. Corp. v. Borland Int'l, Inc.*, 49 F.3d 807, 815 (1st Cir. 1995); *Computer Assocs. Int'l Inc. v. Altai, Inc.*, 982 F.2d 693, 705 (2d Cir. 1992); *Mitel, Inc. v. Iqtel, Inc.*, 124 F.3d 1366 (10th Cir. 1997). Some courts follow the statute's plain meaning, holding that Section 102(b) precludes copyright protection for all systems or methods of operation, including those in computer programs. *See, e.g.*, *Lotus*, 49 F.3d at 815. Like the Federal Circuit, however, other courts have rejected the statutory text and held that Section 102(b) is merely a reminder of the dichotomy between ideas (which are not copyrightable) and expressions of ideas (which generally are). *See, e.g.*, *Whelan Assocs., Inc. v. Jaslow Dental Lab., Inc.*, 797 F.2d 1222, 1234 (3d Cir. 1986). In those courts' view, a "method of operation" embodied in a computer program is copyrightable so long as its creator could have designed it in different ways. *See id.* at 1234 (internal quotation marks omitted).

1. *Lotus* exemplifies the plain meaning approach. That case concerned a spreadsheet program's menu command hierarchy, which organized commands such as "print," "copy," and "quit" into more than 50 menus and submenus accessible by users. 49 F.3d at 809. The First Circuit held that the hierarchy was a "method[] of operation," and was therefore excluded from copyright protection under Section 102(b)—regardless of whether the hierarchy (or the overall program) satisfied the originality requirement of Section 102(a) and regardless of whether there were other ways to write or structure the hierarchy. *Id.* at 815.

The First Circuit reasoned that a "'method of operation' . . . refers to the means by which a person *operates* something, whether it be a car, a food processor, or a computer." *Id.* (emphasis added). Because the "menu command hierarchy provides the means by which users control and operate" the Lotus 1-2-3 program, the hierarchy was a method of operation excluded from copyright protection. *Id.* For that reason, it was "immaterial" that "Lotus developers could have designed the Lotus menu command hierarchy differently." *Id.* at 816.

In determining whether an element of a computer program is a method of operation, the First Circuit also took into account compatibility (whether the element enables the program to interact with other software or hardware) and the lock-in effect (whether users have invested time and effort in learning how to use the method of operation). The First Circuit noted that the fact "[t]hat the Lotus menu command hierarchy is a 'method of operation' becomes clearer when one considers program compatibility." *Id.* at 817. The court rejected as "absurd" Lotus's theory that, "if a user uses several different programs, he or she must learn how to perform the same operation in a different way for each program used." *Id.* at 817–18.

The Sixth Circuit has similarly held that, "even if a work is in some sense 'original' under § 102(a), it still may not be copyrightable because [of] § 102(b)," which excludes original methods of operation from copyright protection. *Lexmark Int'l, Inc. v. Static Control Components, Inc.*, 387 F.3d 522, 534 (6th Cir. 2004). That court explained that, although systems and 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." *ATC Distribution Grp., Inc. v. Whatever It Takes Transmissions & Parts, Inc.*, 402 F.3d 700, 707 (6th Cir. 2005) (internal quotation marks omitted).

The Sixth Circuit also held that the merger doctrine precludes copyright protection for elements of a computer program that are necessary for compatibility. *Lexmark*, 387 F.3d at 536. The court explained that, if there is only one practical way to express an idea, that expression is not entitled to copyright protection. *Id.* at 535. "Program 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." *Id.* at 536 (internal quotation marks omitted).

2. Like the Federal Circuit in this case, other courts of appeals have disagreed with the First and Sixth Circuits in a number of respects. The Third Circuit, for example, insists that all elements of a computer program, including its structural elements, are copyrightable so long as the program could have been written differently and still served the same high-level purpose, such as "to aid in the business operations of a dental laboratory." *Whelan*, 797 F.2d at 1238. In that court's view, Section 102(b) "was not intended to enlarge or contract the scope of copyright protection," only to reinforce the "somewhat metaphysical" dichotomy between idea and expression, with "idea" referring to a program's general purpose. *Apple Computer, Inc. v. Franklin*

Computer Corp., 714 F.2d 1240, 1252, 1253 (3d Cir. 1983).

The Second Circuit has plowed a third path: the so-called "abstraction/filtration/comparison" test. Under that test, a court should first "dissect the allegedly copied program's structure and isolate each level of abstraction contained within it." *Altai*, 982 F.2d at 707. Then, the court should "filter[] . . . protectable expression from non-protectable material." *Id.* 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 has distinguished its approach from the Third Circuit's "inadequate . . . formulation that a program's overall purpose equates with the program's idea." *Id.* at 705. The First Circuit, in turn, rejected the Second Circuit's test, finding it "misleading" because "abstracting menu command hierarchies down to their individual word and menu levels and then filtering idea from expression at that stage . . . obscures the more fundamental question of whether a menu command hierarchy can be copyrighted at all." *Lotus*, 49 F.3d at 815.

Like the Second Circuit, the Fifth and Tenth Circuits employ the abstraction/filtration/comparison test. *See Eng'g Dynamics, Inc. v. Structural Software, Inc.*, 26 F.3d 1335 (5th Cir. 1994); *Eng'g Dynamics, Inc. v. Structural Software, Inc.*, 46 F.3d 408 (5th Cir. 1995) (supplemental opinion); *Gates Rubber Co. v. Bando Chem. Indus.*, 9 F.3d 823 (10th Cir. 1993). In adopting that test, the Tenth Circuit expressly disagreed with *Lotus*, holding 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." *Mitel*, 124 F.3d at 1372. The court opined that Section 102(b), despite its plain text, does not withdraw copyright protection from methods of operation. Instead, "sections 102(a) & (b) interact to secure ideas for [the] public domain and to set apart an author's particular expression for further scrutiny." *Id.* That court thus "declin[ed] to adopt the *Lotus* court's approach to section 102(b), and continue[d] to adhere to [its] abstraction-filtration-comparison approach." *Id.*

3. In addition to disagreeing about whether to replace Section 102(b)'s plain language with one of the court-created standards discussed above, the courts of appeals have divided on related issues, including the relevance of compatibility to copyrightability. As noted above, the First and Sixth Circuits treat compatibility and lock-in as important if not dispositive considerations. The Second Circuit agrees with those circuits that "compatibility requirements of other programs with which a program is designed to operate" are relevant to copyrightability, as part of the "filtration" step of its abstraction/filtration/comparison test. *Altai*, 982 F.2d at 709–10. In contrast, the Third Circuit held that "compatibility with independently developed application programs . . . is a commercial and competitive objective which does not enter into the somewhat metaphysical issue of whether particular ideas and expressions have merged." *Apple Computer*, 714 F.2d at 1253.

The courts of appeals are similarly divided on the merger doctrine. As noted above, the Sixth Circuit has split from other courts of appeals by holding that the merger doctrine precludes copyright protection for elements of a computer program necessary for interoperability. *See Lexmark*, 387 F.3d at 536. Other courts of appeals do not even agree that the merger doctrine limits copyrightability (in any way), holding that it is only an affirmative defense to infringement after copyrightability has been established—greatly diminishing its practical importance. *See, e.g.*, *Kregos v. Associated Press*, 937 F.2d 700, 705 (2d Cir. 1991); *see also* pp. 28–29, *infra*.[2]

4. The decision below recognizes and deepens the circuit split. The Federal Circuit held that, under Ninth Circuit precedent: Section 102(b) does not exclude systems or methods of operation from copyright protection; a judicially-created abstraction/filtration/comparison test governs instead; "[i]nteroperability [a]rguments are [i]rrelevant to [c]opyrightability"; the merger doctrine does not restrict copyright protection for computer code necessary for interoperability so long as the original

[2] As the Federal Circuit recognized, the circuit courts' disarray is so complete that they do not even agree on the correct standard of appellate review. App. 16 n.3. *Compare Matthew Bender & Co. v. West Publ'g Co.*, 158 F.3d 674, 681 (2d Cir. 1998) *and North Coast Indus. v. Jason Maxwell, Inc.*, 972 F.2d 1031, 1035 (9th Cir. 1992) (clear-error standard) *with Yankee Candle Co. v. Bridgewater Candle Co.*, 259 F.3d 25, 34 n.5 (1st Cir. 2001) *and Publications Int'l, Ltd. v. Meredith Corp.*, 88 F.3d 473, 478 (7th Cir. 1996) (*de novo* standard).

author could have written the code in more than one way; and merger plays no role in the copyrightability analysis in any event. *See* App. 23, 24, 50.

If the Federal Circuit's view of Ninth Circuit precedent is correct, that circuit is in conflict with other circuits on all of those important points of law. If the Federal Circuit's understanding of Ninth Circuit law is wrong, the Ninth Circuit is still in conflict with the courts on the other sides of the circuit splits. Either way, the longstanding division in lower court authority persists and requires this Court's resolution.

II. The Federal Circuit's Decision Runs Afoul Of The Statute, This Court's Controlling Precedents, And The Distinction Between Patent And Copyright.

As explained above, the Federal Circuit held that the method headers are copyrightable even if they constitute, or embody, systems or methods of operation. App. 23; pp. 11–12, *supra*. That holding is wrong. It is contrary to the text of the Copyright Act, and it erases a fundamental boundary between patent and copyright law.

A. The statute codifies this Court's exclusion of systems and methods of operation from copyright protection.

Under Section 102(a), an "original work of authorship" is generally copyrightable. Section 102(b) goes on to specify, however, that "in no case does copyright protection *for an original work of authorship* extend to any . . . system [or] 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) (emphasis added). There is nothing unclear or ambiguous about that provision. Though an original work of authorship is generally entitled to copyright protection, the protection afforded to that work does not extend to any systems or methods of operation included or embodied in the work. The statutory exclusion is explicit and absolute, governing "regardless of the form in which [a system or method of operation] is described, explained, illustrated, or embodied in such work." *Id.*

The Federal Circuit opined, however, that "components of a program that can be characterized as a 'method of operation' may nevertheless be copyrightable." App. 44. To reach that result, the court had to revise the statute, and it did: "Section 102(a) and 102(b) are to be considered collectively so that certain expressions are subject to greater scrutiny." App. 23. The court did not explain whence this "greater scrutiny" test comes—it certainly does not come from the statutory text. The court did not explain what "greater scrutiny" means or how to apply it. Nor did the court even appear to apply greater scrutiny; it simply held that because Sun could have written the method headers in different ways, they were copyrightable. *See* App. 47.

The Federal Circuit's error is especially stark because this Court determined more than twenty years ago that Section 102(b) "identifies specifically those elements of a work for which copyright is not available." *Feist*, 499 U.S. at 356. The Court said nothing in *Feist* about replacing that specific, statutory identification with a vague "greater scrutiny" test.

Ignoring this Court's interpretation of Section 102(b), the court of appeals looked instead to the legislative history. App. 23. Legislative history can never displace clear statutory text. *See Milner v. Dep't of Navy*, 131 S. Ct. 1259, 1267 (2011). And here, the legislative history specifically confirms that Section 102(b) means what it says: "processes or methods embodied in [a computer] program are not within the scope of the copyright law." H.R. REP. NO. 94-1476, at 57 (1976).

The Federal Court pointed to a different passage in the legislative history that indicates, as this Court has explained, that Section 102(b) did not change preexisting law, "but merely clarified it." *Feist*, 499 U.S. at 356; *see also* H.R. REP. NO. 94-1476, at 57; S. REP. NO. 94-473, at 54 (1975). That observation is fully consistent with the clear statutory text and the on-point legislative history quoted above. This Court had held, many decades before the 1976 Copyright Act, that systems and methods of operation (along with specific elements of expression that are "necessary incidents" to them) are not copyrightable. *Baker v. Selden*, 101 (11 Otto) U.S. 99, 103 (1880).

In *Baker*, Selden developed an accounting system and wrote a book explaining it. *Id.* at 100. He included in the book "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." *Id.* Selden contended that "the ruled lines and headings, given to illustrate the system, are a part of the book, and, as such, are secured by the copyright." *Id.* at 101.

This Court rejected Selden's argument; the forms were not copyrightable. The Court explained that "there is a clear distinction between the book, as such, and the art which it is intended to illustrate." *Id.* at 102. "The copyright of a work," in other words, "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." *Id.* at 103 (emphasis added).

In light of that holding, the Federal Circuit's decision runs headlong into not one, but two controlling decisions of this Court—*Feist* and *Baker*. The Federal Circuit attempted to distinguish *Baker* on the ground that it merely stands for a dichotomy between unprotectable ideas and protectable expression. App. 19. But nothing in *Baker* supports that interpretation. The case never even discusses that dichotomy. In any event, Section 102(b) codified *Baker* by unambiguously excluding systems and methods of operation from copyright protection, not by adopting a vague "greater scrutiny" test.

B. Systems and methods of operation are governed by patent, not copyright, law.

The Federal Circuit's error is confirmed by the extent to which it would eliminate a fundamental distinction between patent and copyright law—and thus allow copyright to be used as an end-run around the limits on patent protection, including this Court's recent decisions on patent-eligibility.

1. The *Baker* Court determined that the Patent Act, rather than the Copyright Act, governs the protectability of methods and systems. "The description of the art in a book, though entitled to the benefit of copyright, lays no foundation for an exclusive claim to the art itself." *Baker*, 101 U.S. at 105. "The object of the one is explanation; the object of the other is use. The former may be secured by copyright. The latter can only be secured, if it can be secured at all, by letters-patent." *Id.*

Thus, under *Baker* and Section 102(b), copyright cannot be used to secure a monopoly on a system or method of operating something. "[T]he rules and methods of useful art have their final end in application and use; and this application and use are what the public derive from the publication of a book which teaches them." *Id.* at 104. In the absence of a patent, "any person may practise and use the art itself." *Id.*

For this reason as well, the Federal Circuit's focus on whether there is more than one way to structure a system of method headers misses the point. There are, for example, many possible ways to design a keyboard, shorthand system, or accounting system. But under Section 102(b), *no* system or method of operation is protected by copyright.

2. Dismantling that boundary between patent and copyright protection would wreak havoc in the field of intellectual property by granting unwarranted, 95-year (or longer) monopolies on the basic building blocks of innovation. Unlike a claim to a copyright, "[t]he 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 Patent Act imposes strict limits on patentability to ensure that a government-granted monopoly on *use* of an invention will serve its purpose of encouraging inventions and discoveries. *See, e.g.*, 35 U.S.C. §§ 101, 102, 103; *KSR Int'l Co. v. Teleflex Inc.*, 550 U.S. 398, 427 (2007).

Just last Term, this Court confirmed that, while some software-related patent claims may be eligible for patent protection under 35 U.S.C. § 101, many are not. *Alice Corp. Pty. Ltd. v. CLS Bank Int'l*, 134 S. Ct. 2347, 2358–59 (2014). Like Section 102(b) of the Copyright Act, Section 101 of the Patent Act protects future innovation by preventing anyone from "'inhibit[ing] further discovery by improperly tying up the future use of' the[] building blocks of human ingenuity." *Id.* at 2354 (quoting *Mayo Collaborative Servs. v. Prometheus Labs., Inc.*, 132 S. Ct. 1289, 1301 (2012)).

Extending copyright protection to methods and systems of operation would undermine the limits on patent protection. While the requirements for patentability are strict, Section 102(b) is the only requirement for copyrightability that does not present a very "low bar." App. 17. Under Section 102(a), copyright protection is generally available for original works. The "originality requirement is not particularly stringent," requiring "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." *Feist*, 499 U.S. at 345, 358.

The threshold eligibility bar of Section 102(a) is so low as to be essentially non-existent for computer software, as confirmed by the Federal Circuit's focus on whether Sun could have written the method headers in different ways. If one disregards the need to be compatible with other systems or programs, as the Federal Circuit did, there will nearly *always* be more than one way to write software code to accomplish a particular function (such as choosing the greater of two numbers), just as this sentence could have been written a dozen different ways without changing its import. Thus, virtually every element of every computer programming system or language would qualify for copyright protection under the court of appeals' approach.

As *Baker* concluded, "[t]o give to the author of the [work] 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." 101 U.S. at 102. And a long-lasting fraud at that. Compared to the 20-year patent term, a copyright confers monopoly rights that can last for well over a century—for the remaining life of the author plus 70 years, for 95 years after first publication, or for 120 years after creation. 17 U.S.C. § 302. Permitting such an end-run around the carefully crafted limits on patent protection would stifle competition and innovation in the software industry—the very competition and innovation this Court has sought to protect by enforcing the comparable limits on patentability. *See, e.g.*, *Alice*, 134 S. Ct. at 2354.

That does not, of course, mean that all computer software is unprotected by copyright. There is no dispute, for example, that the implementing code that instructs a computer how to perform a method may be subject to copyright protection. *See* 17 U.S.C. § 101 (defining "computer program[s]" that may qualify as protectable works). But whether the method headers are entitled to protection is exclusively a question for patent law because the headers constitute, or embody, a system or method of operating the pre-written programs.

3. The Federal Circuit's error is all the more glaring because it is essentially the same error for which this Court has repeatedly reversed the Federal Circuit in patent cases. The court of appeals criticized the district court for confusing "the threshold question of what is copyrightable—which presents a low bar—and the scope of conduct that constitutes infringing activity." App. 17. It then transformed Section 102(b)'s limits on copyright eligibility into just one of several factors to be considered as part of a fair-use defense. *See* App. 50–56.

The Federal Circuit had similarly held that the limits on patent eligibility are minimal and that other requirements of the Patent Act do the real work in limiting monopoly protections. *See, e.g.*, *Research Corp. Techs., Inc. v. Microsoft Corp.*, 627 F.3d 859, 869 (Fed. Cir. 2010) (referring to Section 101 of the Patent Act as a "coarse eligibility filter"). This Court has repeatedly corrected that misperception in recent years, stressing the importance of enforcing Section 101's limits on patentable subject matter—including for software-related patents. *See, e.g.*, *Alice*, 134 S. Ct. 2347; *Ass'n for Molecular Pathology v. Myriad Genetics, Inc.*, 133 S. Ct. 2107 (2013); *Mayo*, 132 S. Ct. 1289; *Bilski v. Kappos*, 130 S. Ct. 3218 (2010). But the Federal Circuit would now eviscerate the analogous limitation on copyright eligibility for some of the same types of works.

The Federal Circuit's error carries even more dire consequences in the copyright context than it did in the patent arena. There was at least a non-frivolous argument that the limits on patent eligibility were not exceptionally important because other limits on patentability could do some of the same work. *See, e.g.*, *Mayo*, 132 S. Ct. at 1303–04 (rejecting the United States' argument to that effect). Here, such an argument would not even be colorable.

As discussed above, Section 102(b) places "any idea, procedure, process, system, method of operation, concept, principle, or discovery" in the public domain, as a matter of law, by excluding it from the scope of copyright protection. In contrast, the fair-use defense applies to materials that are within the scope of copyright protection, but blesses unauthorized uses that satisfy a multi-factor balancing test. *See* 17 U.S.C. § 107; App. 58–60. The Federal Circuit underscored the difference between the two by indicating that compatibility and lock-in are, in its view, not even the most important factors for a jury to consider as part of the fair-use inquiry. *See* App. 68.

In *Lotus*, the district court concluded, based on the facts of that case, that the defendant's use of the menu command hierarchy was not a fair use. *Lotus Dev. Corp. v. Borland Int'l, Inc.*, 831 F. Supp. 223, 240–45 (D. Mass. 1993), *rev'd*, 49 F.3d 807 (1st Cir. 1995). As the First Circuit recognized, however, Section 102(b)'s exclusion of the hierarchy from copyright protection controlled the outcome, making consideration of fair use unnecessary. 49 F.3d at 819.

C. The Java method headers are a system or method of operation.

This case illustrates the importance of applying Section 102(b) as written. The Java method headers, which enable programmers to use the familiar shorthand commands based on them, are certainly a system or method of operating the pre-written programs of the Java language and platform.

"All agree that Google was and remains free to use the Java language itself." App. 108. That language is made up of "keywords and other symbols" as well as methods, "a set of pre-written programs to carry out various commands." App. 106. As discussed above, programmers call the pre-written methods with shorthand commands that work only in software platforms that use the Java method headers. *See* pp. 6–8, *supra*.

The Second Circuit (including Learned Hand) long ago recognized that there is no "exclusive right to the use of a published system of shorthand." *Brief English Sys., Inc. v. Owen*, 48 F.2d 555, 556 (2d Cir. 1931). Under *Baker*, a "system of condensing written words into less than the number of letters usually used to spell them out" could be protected, if at all, only "by letters patent and not by copyright." *Id.* (Under the Federal Circuit's approach, presumably that case would have come out differently because there is more than one imaginable system of English shorthand; that absurd result illustrates how far from *Baker* the Federal Circuit strayed.)

As Oracle's then-CEO Larry Ellison testified, moreover, "[t]he [Java] API's are a command structure." C.A. App. 20457. If Google had not replicated the method headers exactly, code that used the shorthand commands based on those headers would not have run on Android. *See* pp. 9–10, *supra.* Google took pains to replicate only the elements necessary to allow programmers to use the shorthand commands (*i.e.*, it copied only the method headers)—not the code that actually implements or performs the methods. App. 109. Computer programmers' investment of time and resources in learning the shorthand commands confirms that the corresponding method headers, from which the shorthand commands are derived, constitute or embody the system or method of operating the pre-written programs of the underlying platform.

Compatibility and lock-in concerns confirm the applicability of both Section 102(b) and, in the alternative, the merger doctrine. If one must use specific computer code in order to operate computer programs such as the pre-written programs at issue here, that means, almost by definition, that the copied code is part of a system or method of operating the programs. *See, e.g.*, *Lotus*, 49 F.3d at 817–18. As discussed above, Google replicated the method headers so that computer programmers could operate the pre-written programs using the familiar shorthand commands derived from the headers. If Google changed the headers, the commands would not successfully operate the methods.

Copyright cannot lock up this system or method of operation any more than it could lock up the QWERTY keyboard. Pressing a key on a QWERTY keyboard sends a command that causes a computer to perform a specific function, such as drawing a "Q" on the screen. QWERTY is thus both a keyboard design and a command structure for causing computers of all kinds to produce letters and symbols—just as the method headers are the command structure for using the pre-written programs in the Java and Android platforms.

Oracle and the Federal Circuit have emphasized that, because Google replicated the method headers from only 37 of the Java packages, programs written in Java for the Java platform will not necessarily run as intended on the Android platform. App. 56–57. As the district court observed, however, "imperfect interoperability, and Oracle's angst over it," only prove the point by "illustrat[ing] the character of the command structure as a functional system or method of operation." App. 160.

There is no dispute that Google replicated the method headers that were most important for mobile devices precisely because of the lock-in effect: like computer users who are familiar with the QWERTY keyboard layout, programmers were already accustomed to using the Java shorthand commands based on the headers. App. 58. Google's decision *not* to use more than it needed for a mobile-device platform certainly does not *expand* the scope of Oracle's copyright protection, any more than a decision to omit the number keys on a keyboard would make a copyright claim for QWERTY more plausible.

Indeed, this case is a prime example of the importance of compatibility and lock-in. Programmers have invested significant time and effort in learning the Java language, including the shorthand commands. *See* p. 2, *supra*. But now, long after Sun lured computer programmers into the Java community and after any patent protection likely would have expired, Sun's successor Oracle is attempting to build a wall around use of Java's method headers. That would work precisely the "surprise and . . . fraud" on the public that *Baker* sought to prevent. *See* 101 U.S. at 102.

III. This Case Presents a Recurring Question of Exceptional Importance.

This case is an ideal vehicle for considering the question this Court tried to resolve in *Lotus*. As discussed above, this case alone is exceptionally important, as it involves both a ubiquitous interface (the method headers of the Java programming language) and a product relied on by many millions of people daily (the Android platform).

Moreover, the district court's detailed factual findings and the Federal Circuit's legal analysis cleanly present the question presented. Although the Federal Circuit remanded for a retrial on fair use, the court of appeals definitively resolved the threshold legal question presented in this petition. There is no need to await a second trial on fair use before considering that question—especially considering the pressing need for this Court's resolution.

The decision below is casting a pall over computer hardware and software development. *See, e.g.*, Van Lindberg, *The Copyrightability Of APIs In The Land Of OpenStack* (2014), *available at* http://www.rackspace.com/blog/the-copyrightability-of-apis-in-the-land-of-openstack/. As history has shown, the ability to build on existing interfaces in creating new products and services is a critical driver of innovation in the computer and software fields.

When IBM created the personal computer, for example, it developed an interface called the Basic Input/Output System. Competitors like Compaq and Phoenix re-implemented that system to create their own IBM-compatible computers, increasing the number of choices available to consumers. *See* Charles H. Ferguson & Charles R. Morris, *Computer Wars: The Post-IBM World* 53–55 (1994). Later, Apple used the pre-existing UNIX application programming interface in its computers' operating system, allowing programmers familiar with UNIX to write software that could run on Apple's innovative computers. *See* Joe Wilcox, *Will OS X's Unix Roots Help Apple Grow?*, CNET.com (May 21, 2001). Oracle built upon the Linux operating system in much the same way. *See generally* Oracle Corp., *Frequently Asked Questions, Oracle Linux* (2014), *available at* http://www.oracle.com/us/technologies/

027617.pdf. And in order to compete in the word-processing field, Microsoft re-implemented WordPerfect's interface so that Microsoft Word, a competing product, could open documents created in WordPerfect. Br. of *Amici Curiae* Rackspace US, Inc. *et al.*, at 12–13, *Oracle Am., Inc. v. Google Inc.*, Dkt. No. 116, Nos. 13-1021, *et al.* (Fed. Cir. May 30, 2013). As these examples show, innovation depends on software developers' ability to achieve compatibility with, and build on, what has come before as they create new products and services.

The need to use existing interfaces without fear of copyright liability is even more essential in today's interconnected world. Cloud computing, for example, allows users to access virtual storage facilities and processing power from anywhere in the world via the Internet. *See* Amazon Web Services, *AWS Products & Solutions* (2014), *available at* http://aws.amazon.com/. Because the major cloud computing providers (Amazon, Eucalyptus, and CloudStack) use compatible interfaces, consumers are able to switch platforms and services seamlessly regardless of which browser or operating system they use. Steven J. Vaughan-Nichols, *OpenStack vs. CloudStack: The Beginning of the Open-Source Cloud Wars*, ZDNet (Apr. 12, 2012), *available at* http://www.zdnet.com/blog/open-source/openstack-vs-cloudstack-the-beginning-of-the-open-source-cloud-wars/10763. Those services compete with each other to provide the best implementations of the cloud-services interface; none should be entitled to an exclusive right to use the method of operation itself.

To take another example, millions of people use a computer program called Wine to make Microsoft Windows programs run on different operating systems. Wine works by re-implementing the Windows interface so that Windows programs will run on other operating systems. WineHQ, *About Wine*, *available at* http://www.winehq.org/about/. If Microsoft could threaten Wine with copyright liability, Wine could be shut down, depriving its customers of the ability to run Windows-based software on their computers.

Domestic and international laws also reflect the importance of protecting the public's right to use interfaces freely, without risking copyright liability. Congress has authorized "reverse engineering" for the "purpose of identifying and analyzing those elements of the program that are necessary to achieve interoperability of an independently created computer program with other programs." 17 U.S.C. § 1201(f). The European Union's Software Directive similarly provides a broad exception from liability for "black box reverse engineering." Council of Ministers Directive 91/250/EEC of 14 May 1991 on the Legal Protection of Computer Programs, Art. 5(3), 1991 O.J. (L 122).

Those laws make sense because, after identifying and analyzing the computer code that is necessary to achieve interoperability, developers are free to use it, as Google did here. Indeed, the European Union's highest court recently held that "neither the functionality of a computer program nor the programming language and the format of data files used in a computer program in order to exploit certain of its functions constitute a form of expression of that program and, as such, are not protected by copyright." *SAS Institute Inc. v. World Programming Ltd.* Case C-406/10 ¶ 71, 2012 E.C.L.I 259, [2012] 3 C.M.L.R. 4. A contrary conclusion would "amount to making it possible to monopolise ideas, to the detriment of technological progress and industrial development." *Id.* ¶ 40.

As these real-world examples and laws reflect, the developer community has long understood that interfaces are free for everyone to use. That understanding has enabled all of the innovation described above, and much more. The Federal Circuit's decision turns this understanding on its head, balkanizing computer languages and interfaces, requiring programmers to build from the ground up, precluding interoperability, and depriving consumers of the benefits of compatibility. At a bare minimum, that would make innovation much costlier and raise severe barriers to entry.

The decision below also inflicts particular and immediate hardship on smaller companies and start-ups—major sources of jobs and innovation. *See* Tim Kane, Ewing Marion Kauffman Foundation, *The Importance of Startups in Job Creation and Job Destruction* 3 (2010). These start-ups (the ranks of which Google, Sun, and Oracle once were members) are characterized by extraordinary creativity. They are innovating all the time, building on existing technology to bring products and services to market. To attract customers, these new market entrants must build on what has come before.

Consider how difficult it would have been for Tesla to build an electric car if the familiar arrangement and functions of a steering wheel, accelerator, and brake pedal were protected. The Java method headers and shorthand commands derived from them are to today's software programmers as those standard controls are to today's drivers—crucial methods for operating a complex system.

Delay in resolving this issue would magnify the harm caused by the decision below by impairing important innovation now in the fast-moving, high-technology sector. Just last Term, this Court granted review of an important copyright case even though there was no circuit split, and barely any percolation in the courts of appeals, because of the need for a timely ruling. *See Am. Broad. Cos. v. Aereo, Inc.*, 134 S. Ct. 2498 (2014). This case is no less important, as confirmed by the filing of eleven *amicus* briefs by dozens of *amici* (on both sides) in the court of appeals. Especially considering the clear and well-recognized circuit split on this issue, and the fact that this Court has already recognized the issue's certworthiness by granting review in *Lotus*, the Court should resolve this important and pressing issue now.

CONCLUSION

The petition for a writ of certiorari should be granted.

Respectfully submitted.

DARYL L. JOSEFFER
Counsel of Record
ASHLEY C. PARRISH
ADAM M. CONRAD
ETHAN P. DAVIS
KING & SPALDING LLP
1700 Pennsylvania Ave., NW
Washington, DC 20006
(202) 737-0500
djoseffer@kslaw.com

RENNY HWANG
GOOGLE INC.
1600 Amphitheatre Parkway
MOUNTAIN VIEW, CA 94043

BRUCE W. BABER
KING & SPALDING LLP
1180 Peachtree Street, NE
Atlanta, GA 30309

ROBERT A. VAN NEST
STEVEN A. HIRSCH
CHRISTA M. ANDERSON
MICHAEL S. KWUN
DAN JACKSON
KEKER & VAN NEST LLP
633 Battery Street
San Francisco, CA 94111

Counsel for Petitioner

APPENDIX

TABLE OF APPENDICES

Appendix A

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

[1] Oracle acquired Sun in 2010.

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 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

[2] The 37 API packages involved in this appeal are: java.awt.font, java.beans, java.io, java.lang, java.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.

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 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

[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.

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 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, 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

[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.

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] 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

[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—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.

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, "the Android method and class names could have been different from the names of their counterparts in Java and still have worked." *Copyrightability*

[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, 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").

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 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]

[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

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 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.

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

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 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

[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.

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 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.*

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

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 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*

[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").

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 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. § 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

[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.

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 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

[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.

[14] Amici McNealy and Sutphin explain that "a quick examination of other programming environments shows that creators of other development platforms provide the same

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 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.

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.

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

[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.

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 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

[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

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."). 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

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).

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]

[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

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 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

---

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).

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 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

[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).

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 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 September 15, 2011

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

[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.

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." 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 particular category of elements they contain, are unprotectable under the *scenes a faire* or merger doctrines.

[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.

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 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 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 D

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 May 31, 2012

ORDER REGARDING 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]

[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

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 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

the source code. These files were not proven to have ever been part of Android.

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 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

[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.

[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.

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 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 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 —

[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.

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     // Declares method max
  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

[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."

dichotomy 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

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

[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).

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* 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]

* * *

In view of the foregoing, this order concludes that our immediate case is controlled by these principles of copyright law:

[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 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).

- 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. 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]

[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).

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 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 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

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()

---

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*

[11] The parentheses indicate that inputs/arguments may be included in the command.

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 — 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 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 May 31, 2012

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 F

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

ORACLE AMERICA, INC., Plaintiff,	No. C 10-03561 WHA
v.	
GOOGLE INC., Defendant.	June 20, 2012

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 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 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 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

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 I

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) ofPublic 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) ofPublic 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.