ORACLE AMERICA, INC. v. GOOGLE INC.

United States Court of Appeals, Federal Circuit.

Decided: May 9, 2014.

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

17 **O'MALLEY, Circuit Judge.**

18 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.

19 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.

20 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. [...] 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. *[...]*

21 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. *[...]* 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. [...] 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.

22 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

25 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." *[...]* With Java, a software programmer could "write once, run anywhere."

26 The Java virtual machine ("JVM") plays a central role in the overall Java platform. The Java programming language itself—which includes words, symbols, and other units, together with syntax rules for using them to create instructions—is the language in which a Java programmer writes source code, the version of a program that is "in a human-readable language." *[...]* 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. *[...]* 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." *[...]*

27 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." *[...]* 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. *[...]*

28 The original Java Standard Edition Platform ("Java SE") included "eight packages of pre-written programs." *[...]* 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." *[...]* 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.

29 Every package consists of two types of source code— what the parties call (1)declaring code; and (2) implementing code. Declaring code is the expression that identifies the prewritten function and is sometimes referred to as the "declaration" or "header." As the district court explained, the "main point is that this header line of code introduces the method body and specifies very precisely the inputs, name and other functionality." *[...]* 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.

30 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. *[...]* 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.

31 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." [...] 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.

32 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

34 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." *[...]* 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." *[...]* 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.

35 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." *[...]* 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.

36 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." *[...]* 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." *[...]* 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.

37 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." *[...]*

38 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. [...] Android smartphones "rapidly grew in popularity and now

comprise a large share of the United States market." *[...]* 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." [...]

DISCUSSION

I. ORACLE'S APPEAL

49 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.

50 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.

51 "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." *[...]* Copyright issues are not exclusively assigned to the Federal Circuit. *[...]* 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." *[...]*[3]

52 We are mindful that the application of copyright law in the computer context is often a difficult task. *[...]* 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.

53 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

55 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." [...]

56 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." *[...]*

57 Copyright protection extends only to the expression of an idea—not to the underlying idea itself. *[...]* This distinction—commonly referred to as the "idea/expression dichotomy"—is codified in Section 102(b) of the Copyright Act, which provides:

58 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.[...]

60 The idea/expression dichotomy traces back to the Supreme Court's decision in *Baker v. Selden,* 101 U.S. 99, 101 (1879). [...]

62 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).

63 It is well established that copyright protection can extend to both literal and non-literal elements of a computer program. *[...]* The literal elements of a computer program are the source code and object code. *[...]* Courts have defined source code as "the spelled-out program commands that humans can read." *[...]* Object code refers to "the binary language comprised of zeros and ones through which the computer directly receives its instructions." *[...]* Both source and object code "are consistently held protected by a copyright on the program." *[...]* Google nowhere disputes that premise. *[...]*

64 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. *[...]* 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." *[...]*

65 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.

66 The distinction between literal and non-literal aspects of a computer program is separate from the distinction between literal and non-literal copying. *[...]* "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." [...] 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. [...] 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.

67 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. *[...]* The court found, however, that neither the declaring code nor the SSO was entitled to copyright protection under the Copyright Act.

68 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 [...]. "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." *[...]* Section 102(a) and 102(b) are to be considered collectively so that certain expressions are subject to greater scrutiny. *[...]* 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." *[...]*

69 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. *[...]* This test rejects the notion that anything that performs a function is necessarily uncopyrightable. *[...]* 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.

70 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." *[...]* 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]

71 In the second step, the court is first to assess whether the expression is original to the programmer or author. *[...]* 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. *[...]* These conclusions are to be informed by traditional copyright principles of originality, merger, and scenes a faire. *[...]*

72 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. *[...]* 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.

73 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. *[...]* 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." *[...]* It, nonetheless, has made clear that, in that circuit, these concepts are to be treated as defenses to infringement. *[...]*

74 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 abstractionfiltration-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." *[...]*

75 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

77 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.

78 We find that Oracle did not waive arguments based on Google's literal copying of the declaring code. [...]

80 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.'" *[...]* 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

82 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. *[...]* As noted, the Ninth Circuit treats this concept as an affirmative defense to infringement. *[...]* 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.

83 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. *[...]* 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[...]. 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." *[...]*

85 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." *[...]* In its analysis, the court identified the method declaration as the idea and found that the implementation is the expression. *[...]* 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." *[...]*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. *[...]* Accordingly, the court held there could be "no copyright violation in using the identical declarations." *[...]*

86 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." [...][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.

87 On appeal, Oracle argues that the district court: (1) misapplied the merger doctrine; and (2) failed to focus its analysis on the options available to the original author. We agree with Oracle on both points. First, we agree that merger cannot bar copyright protection for any lines of declaring source code unless Sun/Oracle had only one way, or a limited number of ways, to write them. *[...]* The evidence showed that Oracle had "unlimited options as to the selection and arrangement of the 7000 lines Google copied." [...] 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." *[...]* Because "alternative expressions [we]re available," there is no merger. *[...]*

88 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. *[...]* 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]

89 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.

b. Short Phrases

91 The district court also found that Oracle's declaring code consists of uncopyrightable short phrases. [...]

92 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. *[...]*

93 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.

94 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." [...] 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. *[...]* 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

96 The scenes a faire doctrine, which is related to the merger doctrine, operates to bar certain otherwise creative expression from copyright protection. *[...]* 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." *[...]* 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." *[...]* 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.'" *[...]*

97 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." *[...]* 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*." *[...]*

98 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. *S[...]*

99 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. *[...]*

100 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. [...]

101 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. *[...]* 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. *[...]* It is this showing the trial court found Google failed to make, and Google cites to nothing in the record which indicates otherwise.

102 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

104 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). *[...]* 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]

105 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." [...] 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. *[...]* Although the defendant did not copy any Lotus source code, it copied the menu command hierarchy into its rival program. The question before the court was "whether a computer menu command hierarchy is copyrightable subject matter." *[...]*

106 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). [...]

107 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.

108 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. *[...]* And, while the court in *Lotus* held "that expression that is part of a `method of operation' cannot be copyrighted," [...] this court—applying Ninth Circuit law—reached the exact opposite conclusion, finding that copyright protects "the expression of [a] process or method," *[...]*

109 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-filtrationcomparison 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." *[...]* 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." *[...]*

110 Other courts agree that components of a program that can be characterized as a "method of operation" may nevertheless be copyrightable. [...]

112 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." *[...]*[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." *[...]* If we were to accept the district court's suggestion that a computer program is uncopyrightable simply because it "carr[ies] out pre-assigned functions," no computer program is protectable. That result contradicts Congress's express intent to provide copyright protection to computer programs, as well as binding Ninth Circuit case law finding computer programs copyrightable, despite their utilitarian or functional purpose. Though the trial court did add the caveat that it "does not hold that the structure, sequence and organization of all computer programs may be stolen," *[...]* it is hard to see how its method of operation analysis could lead to any other conclusion.

113 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. *[...]* 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." [...] 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.

114 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." [...] 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.

115 As the district court acknowledged, Google could have structured Android differently and could have chosen different ways to express and implement the functionality that it copied.[14] Specifically, the court found that "the very same functionality could have been offered in Android without duplicating the exact command structure used in Java." *[...]* 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." *[...]* 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.

116 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.

3. Google's Interoperability Arguments are Irrelevant to Copyrightability

118 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.

119 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." *[...]* 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." *[...]* 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." *[...]* 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).

120 Both *Sega* and *Sony* are fair use cases in which copyrightability was addressed only tangentially. [...]

122 The district court characterized *Sony* and *Sega* as "close analogies" to this case. *[...]* 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)." *[...]* The district court's reliance on *Sega* and *Sony* in the copyrightability context is misplaced, however.

123 As noted, both cases were focused on fair use, not copyrightability. [...]

124 We disagree with Google's suggestion that *Sony* and *Sega* created an "interoperability exception" to copyrightability. *[...]* 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. *[...]* And it would ignore the fact that the Ninth Circuit endorsed the abstractionfiltration-comparison inquiry in *Sega* itself.

125 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. *[...]* 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." *[...]*

126 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." *[...]* 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[...]. 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." *[...]*

127 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. *[...]* 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. *[...]* 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." *[...]*

128 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.

129 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]." [...] Indeed, given the record evidence that Google designed Android so that it would *not* be compatible with the Java platform, or the JVM specifically, we find Google's interoperability argument confusing. While Google repeatedly cites to the district court's finding that Google had to copy the packages so that an app written in Java could run on Android, it cites to no evidence in the record that any such app exists and points to no Java apps that either pre-dated or post-dated Android that could run on the Android platform.[15] The compatibility Google sought to foster was not with Oracle's Java platform or with the JVM central to that platform. Instead, Google wanted to capitalize on the fact that software developers were already trained and experienced in using the Java API packages at issue. The district court agreed, finding that, as to the 37 Java API packages, "Google believed Java application programmers would want to find the same 37 sets of functionalities in the new Android system callable by the same names as used in Java." *[...]* Google's interest was in accelerating its development process by "leverag[ing] Java for its existing base of developers." [...] 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.

130 Finally, to the extent Google suggests that it was entitled to copy the Java API packages because they had become the effective industry standard, we are unpersuaded. Google cites no authority for its suggestion that copyrighted works lose protection when they become popular, and we have found none.[16] In fact, the Ninth Circuit has rejected

the argument that a work that later becomes the industry standard is uncopyrightable. *See Practice Mgmt. Info. Corp. v. Am. Med. Ass'n,* 121 F.3d 516, 520 n.8 (9th Cir. 1997) (noting that the district court found plaintiff's medical coding system entitled to copyright protection, and that, although the system had become the industry standard, plaintiff's copyright did not prevent competitors "from developing comparative or better coding systems and lobbying the federal government and private actors to adopt them. It simply prevents wholesale copying of an existing system."). 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

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

133 Fair use is an affirmative defense to copyright infringement and is codified in Section 107 of the Copyright Act. *[...]*

134 "Section 107 requires a case-by-case determination whether a particular use is fair, and the statute notes four nonexclusive factors to be considered." *[...]* 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." *[...]*

135 "Fair use is a mixed question of law and fact." *[...]* 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. *[...]* 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." *[...]*

136 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. *[...]* 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.

137 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," *[...]*; and (2) whether the use serves a commercial purpose.

138 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." *[...]* The critical question is "whether the new work merely supersede[s] the objects of the original creation . . . or instead adds something new." *[...]* 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." *[...]* "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." *[...]*

139 Courts have described new works as "transformative" when "the works use copy-righted material for purposes distinct from the purpose of the original material." *E[...]* "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) [...].

140 A work is not transformative where the user "makes no alteration to the *expressive content or message* of the original work." *[...]* Where the use "is for the same intrinsic purpose as [the copyright holder's] . . . such use seriously weakens a claimed fair use." *[...]*

141 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." *[...]* "[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." *[...]*

142 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." *[...]* This factor "turns on whether the work is informational or creative." *[...]* Creative expression "falls within the core of the copyright's protective purposes." *[...]* 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." *[...]* 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.

143 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." *[...]* "As Judge Learned Hand cogently remarked, `no plagiarist can excuse the wrong by showing how much of his work he did not pirate.'" *[...]* 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." *[...]* 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." *[...]* "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." *[...]* 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." *[...]*

144 The fourth and final factor focuses on "the effect of the use upon the potential market for or value of the copyrighted work." *[...]* 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." *[...]* The Supreme Court has said that this factor is "undoubtedly the single most important element of fair use." *[...]* 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." *[...]* "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." *[...]*

145 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.

146 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.

147 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. [...] 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.

148 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. *[...]* 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.

149 Finally, as to market impact, the district court found that "Sun and Oracle never successfully developed its own smartphone platform using Java technology." *[...]* 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.

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

II. GOOGLE'S CROSS-APPEAL

152 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. [...] 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.

153 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. *[...]* The Ninth Circuit reviews a district court's JMOL decision de novo, applying the same standard as the district court. *[...]* 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." *[...]*

154 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." [...] 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." [...]

155 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." *[...]* As such, the court granted Oracle's motion for JMOL of infringement as to the decompiled security files.

156 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." [...]

157 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.

[...][18]

158 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.

159 First, the unrebutted testimony at trial revealed that rangeCheck and the decompiled security files were significant to both Oracle and Google. [...]

160 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." [...] 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. *[...]* 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.[...]

III. GOOGLE'S POLICY-BASED ARGUMENTS

163 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. *[...]* 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." [...] 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." [...] 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." [...]

164 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-intellectualproperty-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.").

165 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." *[...]* 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)[...]. 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

167 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.

168 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.

Notes:

173 [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. *[...]*

174 [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." [...]

175 [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").[...]

183 [14] Amici McNealy and Sutphin explain that "a quick examination of other programming environments shows that creators of other development platforms provide the same functions with wholly different creative choices." Br. of McNealy and Sutphin 17. For example, in Java, a developer setting the time zone would call the "setTime-Zone" method within the "DateFormat" class of the java.text package. *Id.* Apple's iOS platform, on the other hand, "devotes an entire class to set the time zone in an application—the `NSTimeZone' class" which is in the "Foundation framework." *Id.* at 17-18 (noting that a "framework is Apple's terminology for a structure conceptually similar to Java's `package'"). Microsoft provides similar functionality with "an entirely different structure, naming scheme, and selection." *Id.* at 18 ("In its Windows Phone development platform, Microsoft stores its time zone programs in the `TimeZoneInfo' class in its `Systems' namespace (Microsoft's version of a `package' or `framework')."). Again, this is consistent with the evidence presented at trial.[...]

185 [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." *[...]* 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. *[...]*