2013-1021, -1022

United States Court of Appeals for the Federal Circuit

ORACLE AMERICA, INC.,

Plaintiff-Appellant,

v.

GOOGLE INC.,

Defendant-Cross Appellant.

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

BRIEF OF APPELLEE AND CROSS-APPELLANT GOOGLE INC.

RENNY HWANG
GOOGLE INC.
1600 Amphitheatre Parkway
Mountain View, CA 94043
Telephone: (650) 253-2551

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 94111-1809
Telephone: (415) 391-5400
Facsimile: (415) 397-7188

DARYL L. JOSEFFER
BRUCE W. BABER
KING & SPALDING LLP
1700 Pennsylvania Avenue, N.W.
Washington, D.C. 20006
Tel: (202) 737-0500
Fax: (202) 626-3737

IAN C. BALLON
HEATHER MEEKER
GREENBERG TRAURIG, LLP
1900 University Avenue, 5th Floor
East Palo Alto, CA 94303
Telephone: (650) 328-8500
Facsimile: (650) 328-8508

Counsel for Defendant-Cross-Appellant Google Inc.

May 23, 2013

Form 9

FORM 9. Certificate of Interest

UNITED STATES COURT OF APPEALS FOR THE FEDERAL CIRCUIT

Oracle America, Inc. v. Google Inc.

No. 13-1021, 1022

CERTIFICATE OF INTEREST

Counsel for the (petitioner) (appellant) (respondent) (appellee) (amicus) (name of party) Google Inc. certifies the following (use "None" if applicable; use extra sheets if necessary):

1. The full name of every party or amicus represented by me is:

Google Inc.

2. The name of the real party in interest (if the party named in the caption is not the real party in interest) represented by me is:

Google Inc.

3. All parent corporations and any publicly held companies that own 10 percent or more of the stock of the party or amicus curiae represented by me are:

NONE

4. ☑ The names of all law firms and the partners or associates that appeared for the party or amicus now represented by me in the trial court or agency or are expected to appear in this court are:

See Attachment 4

Nov. 1, 2011
Date

/s/ Robert Van Nest
Signature of counsel

Robert Van Nest
Printed name of counsel

Please Note: All questions must be answered

cc: See Attached

Oracle America, Inc. v. Google, Inc.
Case No. 13-1021, 1022
Attachment 4 to Form 9, Certificate of Interest

4. The name of all law firms and the partners or associates that appeared for the party or amicus now represented by me in the trial court or are expected to appear in this court are:

KEKER & VAN NEST LLP

Robert A. Van Nest, Christa M. Anderson, Steven A. Hirsch,

Michael S. Kwun, Daniel E. Purcell, Eugene M. Paige,

Matthias A. Kamber, Reid P. Mullen, Dan Jackson

KING & SPALDING LLP

Donald F. Zimmer, Jr., Cheryl A. Sabnis, Scott T. Weingaertner,

Brian C. Banner, Bruce W. Baber, Christopher C. Carnaval,

Geoffrey M. Ezgar, Mark H. Francis, Robert F. Perry,

Steven T. Snyder, Truman H. Fenton

GREENBERG TRAURIG, LLP

Ian C. Ballon, Heather J. Meeker, Dana K. Powers, Joseph R. Wetzel

Luis Villa, IV, Valerie W. Ho, Wendy M. Mantell

GOOGLE INC.

Renny F. Hwang

Catherine Lacavera

TABLE OF CONTENTS

Page

TABLE OF AUTHORITIES

Page(s)

Cases

Legislative History

Statutes

Miscellaneous

STATEMENT OF RELATED CASES

Counsel agrees with the statement of related cases provided by the appellant and is only aware of the Petition for a Writ of Mandamus filed in and denied by this Court in *In re Google Inc*., 462 F. App'x 975 (Fed. Cir. 2012) (No. 2012-M106).

I. INTRODUCTION

The Java Application Programming Interface ("API") is not a work of imaginative fiction. It is a command structure that programmers writing in the Java Programming Language must use to access the functionality of pre-written software in the Java class libraries. The Java API enables programmers to incorporate commonly used functionality into their programs without writing their own "implementations"—the source code that makes that functionality work on the computer. The API's commands are arranged hierarchically into packages, classes, and methods to make them convenient for programmers to find and use.

However creative and useful the Java API may be, it is fundamentally a functional, utilitarian work. It exists for the practical convenience of programmers. A work of imaginative fiction like *Harry Potter* serves no such utilitarian function. Its chapter headings and topic sentences exist entirely for communicative and aesthetic purposes—not to "bring about a certain result" when used in a computer.[1]

No court accepts Oracle's premise that functional works like the Java API obtain the same level of copyright protection as works of imaginative fiction. The "fundamental purpose of the Copyright Act" is to "encourage the production of original works by protecting the expressive elements of those works while leaving the ideas, facts, and functional concepts in the public domain for others to build

[1] § 101 (defining "computer program"). All statutory references are to 17 U.S.C.

on." *Sega Enters. Ltd. v. Accolade, Inc*., 977 F.2d 1510, 1527 (9th Cir. 1993). Therefore, if a work is "largely functional," like software, "it receives only weak protection. This result is neither unfair nor unfortunate. It is the means by which copyright advances the progress of science and art." *Id.* (citation and quotation marks omitted).

Copyright protection of functional works is said to be "thin" because section 102(b) of the Copyright Act filters out and denies protection to the functional elements within those works. The more functional the work is, the more there is to filter out. Section 102(b) provides that "[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." Congress explained that one reason for enacting section 102(b) was to "make clear" that "the 'writing' expressing [a programmer's] ideas"—his code—is "the copyrightable element in a computer program," while "the actual processes or methods embodied in the program are not within the scope of the copyright law."[2]

Ninth Circuit precedents control here—and that court has twice held that, even though a computer program may be copyrightable overall, section 102(b)

[2] H.R. Rep. No. 94-1476, 94th Cong., 2d Sess. at 56-57 (1976), *reprinted in* 1976 U.S.C.C.A.N 5659, 5670.

filters out and denies protection to elements ***within*** that program that are "dictated by the function to be performed, . . . or by external factors such as compatibility requirements and industry demands." *Sega*, 977 F.2d at 1524; *Sony Computer Ent'mt, Inc. v. Connectix Corp*., 203 F.3d 596, 603 (9th Cir. 2000). And an important First Circuit case likewise held that a program's hierarchically arranged command structure is an uncopyrightable "method of operation" under section 102(b), if it is essential to accessing, controlling, or using the program. *See Lotus Dev. Corp. v. Borland Int'l, Inc*., 49 F.3d 807, 815-18 (1st Cir. 1995), *aff'd by an equally divided court*, 516 U.S. 233 (1996).

Here, the district court correctly applied these principles to detailed factual findings that it announced after a 10-day bench trial featuring 24 witnesses and scores of documents.[3] Those findings compelled the conclusion that section 102(b) denies copyright protection to the sequence, structure, and organization ("SSO") of 37 Java API packages that Google reimplemented in its Android mobile platform. Among other things, the district court found that:

- The 37 packages' class and method names and declarations were dictated by an external factor—the need to achieve compatibility with existing computer programs written in the Java Programming

[3] The evidence was presented in the course of a parallel jury trial on copyright infringement.

Language. Under the rules of Java, the declarations in Java and Android "must be identical" to declare a method specifying the same functionality—even when the implementation (the code that makes a computer actually execute the method) is different.[4]

- Google used only what was needed to achieve compatibility, taking care to provide its own implementation of the 37 packages.[5]
- The Java API's overall scheme of file-name organization is a command structure for a system or method of operation of the API—"a long hierarchy of over six thousand commands to carry out pre-assigned functions."[6] Thus, the API elements that Google used are a "system" or "method of operation" excluded from copyright protection by the express terms of section 102(b).[7]

On appeal, Oracle and its amici demonstrate no error—"clear" or otherwise—in the court's dispositive factual findings. Instead, Oracle accuses the district court of practicing "software exceptionalism"—discriminating against software by granting it less protection than other works. But it is Oracle that seeks an exception here by asking the Court to excuse the Java API from the normal rule

[4] A136,141,167-68.

[5] A167.

[6] A166-67.

[7] A166-67.

of "thin" protection for functional works. Oracle argues that the Java API deserves special treatment because it is "original," "creative," "intuitive," "attractive," "appealing," "intricate," "efficient," and "user-friendly." Those characteristics are relevant to whether the Java API clears the low "originality" threshold set by section 102***(a)***. The district court found that it does, and Google does not dispute that finding. But it is beside the point, because section 102***(b)*** then filters out the work's functional elements—no matter how original they are. The district court found—correctly—that the SSO of the 37 packages could not survive filtration under section 102(b). Any protection for those elements must be sought in patent law.[8]

Oracle also takes Google to task for reimplementing the 37 packages in order to broaden Android's appeal to Java programmers. This criticism is mystifying. Adopting a technical industry standard like the Java API fosters interoperability and the creation of new works; indeed, the Ninth Circuit held in *Sega* and *Sony* that the defendants were free to copy the plaintiffs' code to the extent necessary to achieve interoperability with immensely popular, commercially successful gaming platforms.[9]

Ninth Circuit law also disposes of Oracle's remaining appellate arguments

[8] *See* Parts V.B.1.-4. & V.B.5.a.-b., below.

[9] *See* Part V.B.5.f., below.

that the district court misunderstood the "merger doctrine" (it didn't),[10] and that reimplementing the 37 packages' SSO did not foster interoperability (it did).[11]

Sensing that its SSO claim may be doomed, Oracle argues that—SSO aside—there is an "independent" ground for reversal based on the purported copyrightability of 7,000 lines of non-implementing code that make up the class and method names and declarations of the 37 packages.[12] But any asserted error regarding those 7,000 lines is both harmless and waived, because the district court gave the jurors instructions and a verdict form that ***barred*** them from considering whether Google infringed those lines, independent of their SSO. Oracle failed to challenge the instructions or the verdict form at trial, or in its opening brief. Thus, the copyrightability of the 7,000 lines (apart from the SSO) is not an issue in this case, and a reversal based on that theory could not alter the ultimate judgment.[13]

Accordingly, this Court should affirm the copyrightability judgment while granting Google's cross-appeal on two minor issues of literal infringement.[14] However, if the Court reverses the copyrightability judgment, it should direct the

[10] *See* Part V.B.5.c., below.

[11] *See* Parts V.B.5.d.-e., below.

[12] *See* Oracle's Opening Brief ("Br.") 31-32.

[13] *See* Part V.D., below.

[14] *See* Cross-Appellant's Brief, below.

district court on remand to retry Google's fair-use defense (as well as the inseparable issue of infringement).[15]

II. STATEMENT OF THE ISSUES

In Oracle's appeal

1. Based on an extensive trial record, the district court found as a factual matter that the command structure of 37 Java API packages is functionally necessary (1) to achieve compatibility with programs written in the Java Programming Language and (2) to access, control, and use the Java class libraries. The district court also found that the command structure is composed of short names and phrases. Those findings, viewed in light of the Ninth Circuit's *Sega* and *Sony* decisions and the First Circuit's *Lotus* decision, compelled the conclusion that section 102(b) excludes the command structure from copyright protection. Were those findings clearly erroneous?

2. The district court gave the jurors instructions and a verdict form that barred them from considering whether Google had infringed 7,000 lines of non-implementing code, independent of their SSO. Oracle did not object to the instructions or verdict form, either below or in its opening appellate brief. Can Oracle nevertheless assert that the district court's failure to find the 7,000 lines independently copyrightable provides an alternative ground for reversal?

[15] *See* Part V.F., below.

3. If the judgment is reversed and remanded, should Google be denied the right to try to the jury a fair-use defense that nine jurors found persuasive at the first trial and that would succeed on retrial if the new jury made some of the same factual findings found in the district court's copyrightability order?

In Google's cross-appeal

Was Google's use of eight decompiled test files and nine lines of rangeCheck code de minimis and thus non-infringing when compared to the 2.8 million lines of code in the class libraries of the registered Java 2 SE version 5.0 platform?

III. STATEMENT OF THE FACTS AND STATEMENT OF THE CASE

A. Sun develops Java as a *free* language and platform to bypass Microsoft's monopoly by fostering a global "Java Community."

Sun Microsystems, with other companies, developed the Java Programming Language ("JPL") and platform in the 1990s in an effort to bypass Microsoft's Windows-based monopoly.[16] Former Sun CEO Jonathan Schwartz explained that Sun's goal was to create a platform that would enable programmers to "'write once, run anywhere,' as opposed to 'write once, and write a check to Microsoft to run it.'"[17] Sun worked with other companies to create "the Java Community,"

[16] A22132-33,22138-39.

[17] A22132.

whose members "agree on the language and the specifications" for Java[18] so that, "when [programmers] at Oracle or SAP write an application, it can run on an IBM computer. . . . It can run on a Sun computer. It can run on any computer that runs Java. And that was our way of bypassing the monopoly."[19]

Sun and its collaborators, including Oracle, recognized that they could not accomplish these goals by creating another proprietary platform, which Microsoft would dwarf.[20] Instead, they made Java open and free for anyone to use, to create a larger and more competitive market.[21] Sun's strategy was to "build trust" with potential partners by declaring that all specifications would be "decided in the open" and that "[e]veryone [would] have equal access to them" so that they could then "go off and create [their] own products."[22] As part of that strategy, Sun "made a lot of noise about open APIs" so as to "bring in as many people as possible . . . to the Java Community. . . . We wanted to basically build the biggest tent and invite as many people as possible."[23]

---

[18] A22139.

[19] A22139.

[20] A22140.

[21] A22139-40.

[22] A22140.

[23] A22140-41.

Oracle admits that nobody owns the JPL; it is free for anyone to use without obtaining any license or paying any royalty.[24] Also free to use is the Java API that programmers use to access pre-written code in the Java class libraries (as discussed below).[25]

Sun's strategy enticed an entire generation of programmers into the Java Community and turned the programming conventions of the free and open Java language into a *de facto* industry standard. Sun "went across the world" to help universities create Java curricula and courseware so that Java could be taught in colleges and high schools, "because then [students] would graduate and . . . go to work for a big company that could become a customer, or they would go off and start a whole new company based on Java."[26] JavaOne, the annual Java developer conference, became the largest developer conference in the nation.[27] Soon after Java's release, thousands of programmers adopted it; eventually, millions did[28]—and Java became the world's most popular programming language.[29]

---

[24] A20475,8248(47:05-47:10),21133,21651,22100,22131-32,22137.

[25] A22137,22140-41,22180-83,21133,21651.

[26] A22133.

[27] A21438.

[28] A20682-83.

[29] A21438-39.

The Java platform includes the Java Virtual Machine ("JVM"), the JPL, and the Java class libraries. The Java API that is at the core of this case is the interface that programmers use to access pre-written code in the Java class libraries.[30]

1. The JVM

The JVM mediates between a Java program and a computer's operating system. Java source code is compiled into an intermediate "bytecode," which the JVM then translates into object code that the operating system can understand. Thus, a program written in Java can run on computers that use a variety of different operating systems: "write once, run anywhere."[31]

2. The JPL

The JPL specifies the rules for writing Java source code.[32] The JPL is an "object-oriented" programming language, like the popular C++ language on which it was modeled.[33] Object-oriented programming languages are based on "classes"—software constructs that define the data fields and methods of the "objects" in the classes.[34] For example, Java (like C++) has a String class whose objects are sequences of characters (*e.g.*, "Hello").[35] The methods of the String

[30] A21946:2-5,22326-45,22272:24-22274:2,22276:16-21,21738-39.
[31] A20463-64,20737-38,20530-31,21400.
[32] A45397-4779.
[33] A4568-69,3590,20868,21414.
[34] A21390,4568,4611.
[35] A3705-06.

class include "length," which returns the number of characters in a particular String object (*e.g.*, 5 for "Hello").[36]

3. The Java class libraries and the Java API.

Fundamental to object-oriented programming languages are "class libraries"—pre-written, ready-made classes, with associated methods, fields, and interfaces,[37] that programmers can (and in some cases, must) use when writing object-oriented programs.[38] A class library simplifies the programmer's work by allowing him or her to incorporate pre-written code by reference.[39] To access the pre-written classes, programmers use the API for the class library—the naming and calling conventions prescribed by the "declarations" (discussed below) for the classes and their associated methods, fields, and interfaces.[40]

The Java class libraries consist of "packages," each of which "groups classes and interfaces that have similar functionality."[41] A two-volume specification, available online and in book form, describes each class in the Java class libraries, including its name, fields, and methods, the names of those methods, what they do,

---

[36] A3714.

[37] Interfaces in this sense (which is distinct from "Application Program Interface," discussed below) link methods of different classes. A22364,22381-82,20758-91,21391-93,21411.

[38] A20946.

[39] A20753-54,20455,22348,22136-37.

[40] A22326-45;21946:2-5.

[41] A3594.

what kinds of inputs (or "arguments") they take, and what kinds of outputs they return.[42] The district court found that "the rules of Java dictate" the precise form of class and method names and declarations[43] and that "there is no bright line between the language and the API."[44]

Packages, classes, methods, and interfaces in the Java class libraries are identified by means of their "fully qualified" names, which specify "where you can find that code."[45] Java's fully qualified names do not merely ***reflect*** or ***reveal*** the hierarchical organization of the Java class libraries—they ***dictate and determine*** that organization.[46] Fully qualified method names follow this format:

java.package.Class.method()[47]

For example, the fully qualified name of the Java API method that finds the larger of two numbers is java.lang.Math.max(), which refers to the "max" method in the "Math" class of the "java.lang" package.[48] A proper Java implementation[49] must

---

[42] A3576-4104,20939,22134-36,22326-45.

[43] A132,136,139,165.

[44] A140-41.

[45] A20944; *see also* A4709.

[46] A20939-45.

[47] A166,20939-41,20944-45. Some of the API packages at issue start with "javax" instead of "java." A20785,21152,21176,22536,1971-72.

[48] A21944-45.

[49] An "implementation" of an API method is the code that makes a computer actually execute the method. A22137-38, 22225-26,22362. It is the functionality that lies "on the other side" of the API and that is invoked by a programmer's use of a proper method name. A21946:3-5.

implement the "max" method using its fully qualified name—exactly.[50] If an implementation altered the fully qualified class and method names, source code that conforms to JPL rules would not compile, and the result would be the opposite of "write once, run anywhere."[51] Thus, as the district court concluded, these names function as a command structure for Java packages and their subparts.[52]

A proper Java implementation also must follow a specific format for the "declarations" (also known as "signatures" and "headers") that introduce and identify particular packages, classes, or methods. For example, the declaration for the "max" method, as defined for integers, is:

public static int max(int arg1, int arg2)[53]

The terms "public" and "static" specify that the method is generally accessible and not specific to a particular object; the first "int" indicates that the method returns an integer; "max" is the name of the method; the final two uses of "int" indicate that the arguments—*i.e.*, inputs—are also integers; and "arg1" and "arg2" are variables referring to those arguments (in this instance, the two numbers being compared).[54]

---

[50] A21472-75,21960-62,22359-61,22362-63.

[51] A21472-75,21960-62,22359,22362-63.

[52] A133,141,166-67.

[53] A20955-67; *see also* A3666,3671,1968-70.

[54] A20955-67; *see also* A3744,3816.

The ***only*** parts of this declaration that can be altered are the names of the variables ("arg1" and "arg2"), which could be replaced by, *e.g.*, "x" and "y" or "a" and "b."[55] Otherwise, any implementation of Java must contain this declaration in this exact form to function properly. As the witnesses testified,[56] and as Oracle's trial counsel admitted,[57] there is no other way to do it. The district court therefore found that, "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."[58] "There is no choice in how to express it."[59] The same is true of package and class declarations.[60]

The "max" method is just one of many. The API for version 5.0 of Java 2 SE[61] has 166 packages,[62] hundreds of classes, and thousands of methods.[63] The JPL is practically useless without the API's packages, classes, and methods.[64] For example, one of the packages at issue here, java.lang, "provides the classes and

---

[55] A21962.

[56] A21962,22281,22341,21750.

[57] A22674:24-25.

[58] A136.

[59] A139.

[60] A165.

[61] A21509-10.

[62] A22421,22431,22477.

[63] A22465.

[64] A22266,20853,20877-78,21678,21720-21.

interfaces that are the core of the Java language,"[65] including the class Object.[66] As Java "guru" Joshua Bloch[67] testified, "You need that one. You can't have an object-oriented language without objects."[68] Likewise, "you need strings,"[69] and the String class also is part of the java.lang package.[70] Programmers also need classes and methods to communicate with the outside world by outputting data to a monitor or printer, and to do just about anything else they might want to do.[71] Even the "Hello World" program—the simplest program in any language—can't be written in JPL without using the Java API.[72]

Sun never sold or licensed the API packages separately.[73] Rather, Sun made them free and available along with the JPL as part of Sun's strategy of "build[ing] the biggest tent and invit[ing] as many people as possible."[74]

---

[65] A3610.

[66] A3676-83; *see also* A4103 (including java.lang, java.io, java.util, and java.net among libraries that are "the foundation of the Java language" and "fundamental to every Java program"); A5875 (accusing java.lang, java.io, java.util, and java.net, among other packages and files in Android).

[67] A20903.

[68] A20946; *see also* A21446.

[69] A20946.

[70] A3705-20.

[71] A20878,20899-20900,20952.

[72] A20952-53.

[73] A22137.

[74] A22141.

B. The Sun-Google business discussions fail after several months without Sun's ever having mentioned its copyrights.

Before "smartphones" were introduced in 2006, Sun dominated the market for software to run on "feature phones"—the simpler precursors to today's smartphones.[75] But Java was not fully integrated into those handsets, and the district court found that "Sun and Oracle never successfully developed [their] own smartphone platform using Java technology."[76] Oracle's contrary suggestions are false and not supported by the record.[77]

In late 2005, Google and Sun began discussing the idea of Google's taking a license to use and adapt the entire Java platform for smartphones. They also discussed a possible partnership under which Java technology would become an open-source part of the Android platform, adapted for smartphones. Google and Sun negotiated over several months, but were unable to reach a deal.[78] At no time during those discussions did Sun talk about its Java copyrights.[79]

---

[75] A22156,21759-60.

[76] A135,22082-87,22114.

[77] Br.6,16,28. *Cf.* A22082-87,22114,8259-61(Cizek depo 32:14-33:18),21246-47.

[78] A135.

[79] A21272; *cf.* A22241(patent infringement discussed in 2006).

After the Sun talks failed, Google continued with its ongoing independent implementation of the Java API packages needed for the Android platform.[80] Sun knew that Google was developing Android using Java.[81]

C. Google develops the Android smartphone platform.

The Android platform took over three years to build[82] and features about 15 million lines of code and thousands of files.[83] By the time version 1.0 was released in 2008, Google had 85 to 90 engineers working on the project.[84]

Google supported use of the JPL by Android developers because the JPL was widely taught in universities and was the best environment for software development.[85] The Android platform came to include 168 API packages.[86] Oracle's copyright infringement claim is based on 37 of those packages ("the 37 packages"). The 37 packages use the same package, class, and method names and declarations as packages described in the Java API specification.[87]

All 37 of the accused packages were required to meet industry

---

[80] A21793-94.

[81] A22155-56.

[82] A21858.

[83] A21862.

[84] A21861.

[85] A21674,21771.

[86] A22356.

[87] A5875,22360-61.

expectations—the expectations of a computer programmer using the JPL.[88] Three of the accused packages were "fundamental to being able to use the Java language at all."[89] The remaining 34 perform a variety of critical functions, such as allowing programmers to access databases, ensuring security, enabling cryptography, controlling the appearance of displayed text, and allowing the API classes to work together.[90] These packages enabled programmers to make practical use of the JPL.[91] The code enabling many of their functions was complicated, so an ordinary programmer would need to rely on pre-written Java packages to call that functionality.[92]

Many other Java API packages were not implemented in Android because they were not appropriate for smartphone applications; in addition, Google had to develop entirely new API packages for smartphones.[93]

With some trivial exceptions,[94] Google did not use any Sun source code to implement the specifications of the 37 packages.[95] Instead, Google's Android

---

[88] A22378-79,21956-57.

[89] A140-41.

[90] A22372-79.

[91] A22372-79,21956-57.

[92] A22375-76.

[93] A21956-59,22356-58.

[94] *See* A142-4; *see also* Cross-Appellant's Brief, below.

[95] A21959,21676-77.

team created new implementations relying on its own expertise, sometimes collaborating with a contractor and sometimes referencing open-source projects, but never relying on or using Sun implementations.[96]

D. Sun welcomes Android with open arms.

Google unveiled Android in October 2007 and released it as open source in October 2008. The first Android phone was launched the next month.[97]

Sun welcomed Android into the world with apparent jubilation. Sun CEO Jonathan Schwartz[98] posted a blog entry stating that he "just wanted to add [his] voice to the chorus of others from Sun in offering [his] heartfelt congratulations to Google on the announcement of their new Java/Linux phone platform, Android."[99] Schwartz added that "Google and the Open Handset Alliance [had] just strapped another set of rockets to the [Java] community's momentum"; and he promised that Sun would furnish "a complete developer environment around the platform[.]"[100]

Following the blog post, Sun continued to make supportive public comments about Android. Schwartz testified that Sun "didn't think [Google was] doing

---

[96] A21674-77,21959-60,21971-77,21155-56,21359-61,21865-73.

[97] A21876,21892-93.

[98] A22130,22141.

[99] A6541.

[100] A6541. The Open Handset Alliance is a group of cell-phone manufacturers, wireless carriers, and technology providers that support Android. A21768-69.

anything wrong . . . . We didn't like it, but we weren't going to stop it by complaining about it."[101] Schwartz noted that, by adopting Java, Google had brought Sun into the Android community and had prolonged Java's relevance to computer-science educators.[102]

Instead of making legal threats, Sun proposed running its own Java technology on top of Android. Google responded that, because Android was open-source, Sun could do whatever it liked with it, subject to the license.[103] But Sun's proposals to build a "Java Phone" failed to obtain internal funding as Sun's financial picture worsened and the company laid off engineers.[104]

E. Oracle acquires Sun and then sues Google.

In February 2010, Oracle purchased Sun and all of its Java technology, hoping to expand Java into smartphones.[105] That March, Oracle CEO Larry Ellison told the JavaOne conference that the advent of Android devices incorporating Java elements was "very exciting" and "flatter[ing]."[106]

Internally, Ellison commissioned a study that examined alternatives for building a Java smartphone platform; but Oracle concluded that it lacked the

---

[101] A22166-67.

[102] A22167.

[103] A21694-97,8262,22168-69,21880-84,21691, Trial Exhibit("TX")3103(video).

[104] A20492-95,6165-66.

[105] A20487.

[106] A20480-82,TX2939.1(video).

necessary expertise and dropped the project.[107] Oracle also tried to partner with Google by offering its own virtual machine for use in Android.[108] Finally, after failing to build or partner with a smartphone platform, Oracle resorted to litigation threats.[109] But Oracle never mentioned the SSO of the 37 packages, let alone asserted any copyright in the SSO, prior to suing Google.

Oracle filed this lawsuit against Google in August 2010, alleging counts for patent and copyright infringement. Oracle claimed that it had suffered as much as $6.1 billion in patent damages;[110] but a series of adverse *Daubert* rulings eliminated the expert testimony for the vast bulk of that claim.[111] Only then did the copyright tail begin to wag the patent dog.

F. Judge Alsup tries the case and concludes that the reimplemented Java API elements are not copyrightable.

The district court trifurcated the case into a first phase covering copyrightability, copyright infringement, and equitable defenses; a second phase covering patent infringement; and a possible third phase covering damages and willfulness.[112] The second (patent) phase ended with a jury verdict of no

[107] 20494-507,6191.

[108] A20508-13,22088-90.

[109] A20559-63,22490-91.

[110] A24582.

[111] A24581-85,24589-92,24593-96.

[112] A24587-88,131.

infringement, from which no appeal was taken. Due to a combination of verdicts and stipulations, the third phase never occurred.

The first phase—the copyright trial—proceeded on two parallel tracks: by day, judge and jury viewed documents and heard testimony from 24 witnesses bearing on API copyrightability, infringement, fair use, and Google's statutory and equitable defenses. But outside the jury's presence, the attorneys grappled with scores of oral and written questions from the court and submitted numerous briefs concerning copyrightability, which the parties agreed the district court would decide.[113]

At trial's end, the court instructed the jury that, for purposes of its deliberations on infringement and fair use, it must assume that Oracle's copyrights covered the SSO of the compilable code in the 37 allegedly infringed Java API packages.[114]

1. The jury verdict and post-trial rulings.

On April 30, 2012, the jury returned a verdict that Google had infringed the overall SSO of the compilable code of 37 Java API packages and also had literally infringed nine lines of code comprising the rangeCheck method in TimSort.java

[113] A131. The parties also agreed that Judge Alsup could decide any subsidiary fact questions relating to his copyrightability determination. *See* A24602,24598-99.

[114] A22769.

and ComparableTimSort.java.[115] The jury deadlocked on Google's "fair use" defense.[116]

After trial, the district court denied Oracle's JMOL motion that Google's fair-use defense be rejected as a matter of law.[117] Oracle appeals from that ruling.

The court granted another Oracle JMOL motion, overturning the jury's finding that Google had not infringed the eight decompiled files and holding that no reasonable jury could have found the copying de minimis.[118] Google cross-appeals from that ruling.

On May 31, 2012—in the key ruling on appeal—the district court issued a 41-page order holding, on the specific facts of this case, that Google was free to use the contested Java elements because they are not protected by copyright. Oracle appeals from that ruling ("the Order").[119]

The district court later denied Oracle's JMOL/new-trial motion on issues of patent and copyright infringement.[120] Oracle appeals from the copyright aspect of that order.

---

[115] A42.

[116] A41.

[117] A129.

[118] A1058A-B.

[119] A130-70.

[120] A173.

After entering judgment, the district court denied Google's JMOL motion asking the court to rule that Google's copying of the rangeCheck method was de minimis when compared to the relevant work as a whole.[121] From that ruling, Google filed a separate appeal, now consolidated with the main appeal.

Before Oracle filed this appeal, the parties stipulated that Oracle had suffered no damages from the copying of rangeCheck and the eight decompiled files.[122]

2. The Order.

Based on extensive factual findings,[123] the district court concluded that the reimplemented SSO elements received no copyright protection. Three established rules compelled that result:

- ***Under Ninth Circuit precedents, the functional elements necessary to achieve compatibility with an existing computer program or platform are excluded from copyright protection by section 102(b) of the Copyright Act.*** The district court found that the Java API method names, class names, and declarations that Google used in its Android mobile-computing platform were functional elements necessary to achieve compatibility with a vast

[121] A1119.

[122] A24612-17.

[123] "All declarative fact statements set forth in the order are factual findings." A133n.3.

body of existing programs written in the JPL. To achieve interoperability with those programs, Google had to provide the same java.package.Class.method() command system in Android, using the same names, the same "taxonomy," and the same functional specifications.[124] But Google only replicated what was necessary to achieve a degree of interoperability with existing Java programs, taking care to provide its own implementation of the Java methods and packages.[125] Thus, the API elements that Google used were functional compatibility elements excluded from copyright protection by section 102(b).

- ***Names and short phrases are likewise excluded from protection.*** The district court found that the method names, class names, and declarations that Google used in Android were too short to qualify for copyright protection by themselves.[126]
- ***Command structures that users employ to operate a computer program are excluded from copyright protection as "method[s] of operation" under section 102(b).*** The district court found that the Java API's overall scheme of file-name organization is a command structure for a system or method of operation of the API—a long hierarchy of over six thousand commands to

[124] A167.

[125] A167.

[126] A132,143-44,164-65.

carry out pre-assigned functions.[127] Thus, the API elements that Google used were a system or method of operation excluded from copyright protection by section 102(b).

The court cautioned that its Order was limited to "the specific facts of this case" and "the particular elements replicated by Google."[128]

IV. SUMMARY OF ARGUMENT

The district court committed no clear error in finding that the command structure of 37 Java API packages is functionally necessary (1) to achieve compatibility with programs written in the Java Programming Language and (2) to access, control, and use the Java class libraries. The district court also correctly found that the command structure is composed of short names and phrases. Those findings, viewed in light of the Ninth Circuit's *Sega* and *Sony* decisions and the First Circuit's *Lotus* decision, compelled the conclusion that section 102(b) excludes the command structure from copyright protection. The judgment should be affirmed.

The Introduction (Part I, above) summarizes the Argument in more detail.

[127] A133,166-67.

[128] A170.

V. ARGUMENT

A. This Court applies Ninth Circuit copyright precedents under Ninth Circuit review standards.

This Court applies copyright law as interpreted by the regional circuits—in this case, the Ninth Circuit. *See Amini Innovation Corp. v. Anthony Cal., Inc.*, 439 F.3d 1365, 1368 (Fed. Cir. 2006). If the appeal raises any issues not yet resolved by the Ninth Circuit, this Court predicts how that Circuit would decide them. *See Litecubes, LLC v. N. Light Prods., Inc.*, 523 F.2d 1353, 1371 (Fed. Cir. 2008).

This Court also applies the regional circuit's standard of review. *See Atari Games Corp. v. Nintendo of Am. Inc.*, 975 F.3d 832, 837 (Fed Cir. 1992); *Amini Innovation*, 439 F.3d at 1368. In an appeal from a bench trial, the Ninth Circuit applies "clear error" review to factual findings, whether based on oral or documentary evidence. *Saltarelli v. Bob Baker Group Med. Trust*, 35 F.3d 382, 384 (9th Cir. 1994); *see also* FED. R. CIV. P. 52(a)(6). "The question whether a product feature is functional" and thus uncopyrightable under section 102(b) "is a question of fact" reviewed for clear error, while "[d]etermination of the correct legal standard to apply in evaluating functionality . . . is a question of law" reviewed de novo. *Sega*, 977 F.2d at 1530-31; *see also Atari*, 975 F.2d at 840.

B. Binding Ninth Circuit precedents compel affirmance of the judgment dismissing Oracle's SSO-infringement claim.

Two controlling Ninth Circuit precedents—*Sega*[129] and *Sony*[130]—hold that section 102(b) filters out and denies protection to computer-program elements that must be copied to achieve interoperability with that program. As applied to the district court's unchallenged factual findings, those cases compel the conclusion that copyright law does not protect the SSO of the 37 packages. *Sega* and *Sony* also dispose of Oracle's six main appellate arguments.

1. Section 102(a) imposes a minimal originality requirement for copyrightability, and section 102(b) then filters out and denies protection to the work's functional elements.

Sections 102(a) and (b) of the Copyright Act prescribe a two-stage copyrightability analysis.

First, the work must meet the test for copyright ***eligibility*** under section 102(a), which states that "[c]opyright protection subsists . . . in original works of authorship fixed in any tangible medium of expression" and extends to "literary works" (including software). Section 102(a) sets a low threshold for copyright eligibility: It requires "originality" only in the limited sense that the work was not copied from an earlier work and possesses a minimal degree of creativity. *See Feist Publ'ns, Inc. v. Rural Tel. Serv. Co., Inc.*, 499 U.S. 340, 345 (1991). The

[129] 977 F.2d 1510.

[130] 203 F.3d 596.

district court found, and Google does not contest, that the 37 packages' SSO met this low threshold.

Second, the court applies the copyright ***exclusions*** of section 102(b) to filter out and deny protection to the functional and factual aspects of an otherwise copyrightable work. Section 102(b) provides that "[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."

Thus, under section 102(b), "the mere fact that a work is copyrighted does not mean that every element of the work may be protected." *Feist*, 499 U.S. at 348. Rather, "[o]nce a work qualifies for copyright protection under § 102(a), § 102(b) informs its author and the rest of the world about certain aspects of the work that are not within the scope of copyright protection." Pamela Samuelson, *Why Copyright Law Excludes Systems and Processes from the Scope of Its Protection*, 85 TEX. L. REV. 1921, 1921 (2007) [hereinafter *Why Copyright*].

Congress gave the courts an important indication of how it wanted them to apply section 102(b) to computer programs. The legislative history states that Congress intended section 102(b) to "make clear" that "the '***writing***' expressing [a programmer's] ideas"—that is, the code—is "the copyrightable element in a

computer program," while "the ***actual processes or methods*** embodied in the program are ***not*** within the scope of the copyright law."[131]

Oracle's appellate arguments relate almost entirely to the first stage of the copyrightability inquiry (originality under section 102(a)); but *Sega*, *Sony*, and the Order focus on the second stage (section 102(b) filtration). The district court held that the SSO of the 37 packages could not survive that second stage and therefore was unprotected. As discussed below, the governing precedents compelled that conclusion.

2. Section 102(b) is grounded in *Baker v. Selden*'s teaching that the functional elements within a copyrightable work should be protected, if at all, by patent law.

Section 102(b) codifies the Supreme Court's landmark holding in *Baker v. Selden*, 101 U.S. 99 (1879). *Baker* held that extending copyright protection from a ***writing that explained*** a new bookkeeping system to ***the bookkeeping system itself*** would circumvent the demanding requirements for obtaining patent protection and thereby work "a surprise and a fraud upon the public." *Id*. at 102.

Plaintiff Selden wrote and obtained copyrights on several books that explained a new bookkeeping system. *Id*. at 99-100. The books contained an introductory essay explaining the system, followed by forms with lines and

[131] H.R. Rep. No. 94-1476, 94th Cong., 2d Sess. 56-57 (1976) (emphases added), *reprinted in* 1976 U.S.C.C.A.N 5659, 5670; S. Rep. No. 94-473, 94th Cong., 1st Sess. 54 (1975) (emphases added).

headings that illustrated the system and showed how it was to be used and carried out in practice. *Id*. at 100. Selden had sought, but apparently not obtained, a patent for his system.[132]

Selden sued Baker for publishing account books employing a similar plan. Selden asserted that no one could practice his system without using the forms in his copyrighted books, or substantially similar ones. *Id*. at 101. The district court agreed and enjoined Baker from publishing or selling his book. *Id*. at 100, 107; *Baker Story* at 166.

The Supreme Court reversed, drawing "a clear distinction between the book, as such, and the [practical] art which it is intended to illustrate." *Id*. at 102. Selden's book had obtained copyright protection "without regard to the novelty, or want of novelty, of its subject matter"; but granting him, in addition, "an exclusive property in ***the art described therein***, when no examination of its novelty ha[d] ever been officially made, would be a surprise and a fraud upon the public. That is the province of letters-patent, not of copyright." *Id*. (emphasis added).

Baker thus established that "copyright protection does not extend to complex and detailed useful innovations, such as new bookkeeping systems and methods of

[132] Pamela Samuelson, *The Story of* Baker v. Selden*: Sharpening the Distinction Between Authorship and Invention*, in INTELLECTUAL PROPERTY STORIES 159, 174-175 (Jane C. Ginsburg & Rochelle Cooper Dreyfuss, eds., 2006) [hereinafter *Baker Story*].

operation, even when they are embodied in copyrighted works." *Why Copyright* at 1935. Had Selden's copyright claim succeeded, "Baker and his fellow bookkeepers would have been precluded from engaging in the kind of incremental innovation characteristic of practical fields such as bookkeeping," and Baker's customers would have had to pay "substantially higher fees to use a Selden-like system or [else] refrain from using a more efficient system to keep their accounts and balance their books." *Id*. at 1934.

Baker and its progeny thus "constitute the principal case law foundations for the system, method, and process exclusions embedded in § 102(b)." *Id.* at 1923.[133] Those exclusions describe subject matter that is beyond copyright's scope and is "more appropriately protected, if at all, by the patent system." *Id*. at 1952.

3. *Sega* and *Sony* held that section 102(b) filters out and denies copyright protection to interfaces—functional program elements necessary for compatibility.

Sega and *Sony* posed the question whether defendants could engage in "intermediate copying" of copyrighted computer programs in order to identify, analyze, and use the unprotected, functional program elements necessary to achieve compatibility with those programs. The answer was "yes." In both cases, the defendants' intermediate copying was found to be a "fair use" because it had

[133] *See also Baker Story* at 180.

the legitimate purpose of accessing functional elements that were excluded from copyright protection under section 102(b).

The "fundamental purpose of the Copyright Act," the *Sega* court explained, is to "encourage the production of original works by protecting the expressive elements of those works while leaving the ideas, facts, and functional concepts in the public domain for others to build on." *Sega*, 977 F.2d at 1526 (citing *Atari*, 975 F.2d at 842-43). Extending copyright protection to a program's compatibility elements would violate that purpose by giving the copyright owner "a de facto monopoly over the functional aspects of his work—aspects that were expressly denied copyright protection by Congress" when it enacted section 102(b). *Id.* at 1527 (citing § 102(b)); *see also Sony*, 203 F.3d at 607.

a. The facts and holding in *Sega*.

In *Sega*, plaintiff Sega manufactured the "Genesis" game console as well as Genesis-compatible game cartridges. Functional program elements necessary to achieve compatibility were hidden within each game cartridge (unlike here, where they were published and supposedly free for anyone to use). 977 F.2d at 1514.

Defendant Accolade, an independent game maker with no license from Sega, needed to discover those functional compatibility elements to make its own Genesis-compatible game cartridges. Accolade therefore decompiled the object code in three Sega game cartridges to identify the hidden "interface specifications"

that restricted compatibility between the console and the cartridges. 977 F.2d at 1515. Accolade then wrote a development manual containing "functional descriptions of the interface requirements," but none of Sega's code. Using that manual, Accolade created its own Genesis-compatible game cartridges. *Id.* at 1514-15. Accolade's end product—the cartridges—contained code implementing the Genesis interface specifications, but no other Sega code. *Id*. at 1515.

Sega then released a new Genesis III console featuring a more sophisticated "trademark security system" ("TMSS") hidden inside the platform. Accolade responded by decompiling a Genesis III-compatible game to discover the TMSS initialization code. *Id*. at 1515. Accolade added that code segment to its development manual in the form of a standard header file to be used in all games. Again, the code in the Accolade header file was the only portion of Sega's code that Accolade copied into its own game programs. *Id*. at 1515-16.

A district court preliminarily enjoined Accolade; but the Ninth Circuit reversed on the ground that copying in order to access, identify, and use an uncopyrightable functional element necessary for compatibility is a "fair use" under section 107. Citing section 102(b) and *Baker*, the court observed that "[t]he protection established by the Copyright Act for original works of authorship does not extend to the ideas underlying a work or to the functional or factual aspects of the work. To the extent that a work is functional or factual, it may be copied . . . ."

Id. at 1524 (citations omitted). "Here," the court concluded, "while the [overall] work may not [have been] largely functional, it incorporat[ed] functional elements"—the interfaces—that did "not merit protection." *Id*. at 1527. Those elements were deemed functional, and thus uncopyrightable, because they were "dictated by the function to be performed," or by "external factors such as compatibility requirements and industry demands." *Sega*, 977 F.2d at 1524.[134]

b. The facts and holding in *Sony*.

The facts and result in *Sony* were similar. Plaintiff Sony made and sold the PlayStation game console and owned the copyright on the console's basic input-output system ("BIOS")—the built-in "firmware" that operated the console. 203 F.3d at 598-99. Sony sold PlayStation games on CDs that users loaded into the top of the console. *Id*. at 599.

Defendant Connectix created a PlayStation "emulator" that enabled users to play Sony PlayStation games using the CD-ROM drive of a regular computer instead of a PlayStation console. Connectix repeatedly copied Sony's copyrighted BIOS from a chip on the PlayStation onto a Macintosh computer to observe the

[134] By contrast, the allegedly copied elements of the program in *Johnson Controls, Inc. v. Phoenix Control Systems, Inc*., 886 F.2d 1173 (9th Cir. 1989)—cited by Oracle—were not "dictated" by anything as they were "customized to the needs of [each] purchaser." *Id*. at 1176. Moreover, *Johnson Controls* parallels the Third Circuit's approach in *Whelan Associates v. Jaslow Dental Lab*, 797 F.2d 1222 (3d Cir. 1986), which has been "widely—and soundly—criticized as simplistic and overbroad." *Sega*, 977 F.2d at 1525-26; *see also* A156,161,169 (district court's discussion of *Johnson Controls*).

PlayStation's "system interface procedures"—the signals sent between the BIOS and the Mac-based hardware emulator that Connectix was developing. *Id*. at 600. Connectix also copied the PlayStation BIOS to obtain CD-ROM code for a Windows version of the emulator. *Id*. at 601. Connectix's product, called Virtual Game Station ("VGS"), reimplemented 137 of the Playstation BIOS's 242 functions[135] but did not contain Sony's copyrighted code. *Id*. at 604 n.7, 606-07.

A district court preliminarily enjoined Connectix. Relying on *Sega*, the Ninth Circuit reversed. The appeals court defined the issue as being the same one posed in *Sega*: "to apply the principles of copyright law to computers and their software, to determine what must be protected as expression and what must be made accessible to the public as function." *Id*. at 598.

In answering that question, the *Sony* court distinguished between copyright eligibility under section 102(a) and the copyright exclusions of section 102(b): "The object code of a program may be copyrighted as expression, 17 U.S.C. § 102(a), but it also contains ideas and performs functions that are not entitled to protection. *See* 17 U.S.C. § 102(b)." *Sony*, 203 F.3d at 602.

As to the second stage—section 102(b) filtration—the court explained that copying is a fair use "if it was 'necessary' to gain access to the functional elements of the software itself. We drew this distinction [in *Sega*] because the Copyright

[135] *See* A168.

Act protects expression only, not ideas or the functional aspects of a software program." *Id*. at 603. The court concluded that the intermediate copying necessary to reverse-engineer the Sony BIOS was "fair use for the purpose of gaining access to the unprotected elements of Sony's software." *Id*. at 602.

Echoing *Baker*, the court added: "If Sony wishes to obtain a lawful monopoly on the concepts in its software, it must satisfy the more stringent standards of the patent laws." *Sony*, 203 F.3d at 605.

4. Under *Sega* and *Sony*, the Ninth Circuit would affirm the district court and reject Oracle's SSO-infringement claim.

Sega and *Sony* dictate the result here. Judge Alsup found that Java's class and method names and declarations must be expressed precisely, with no variation, to achieve interoperability with applications using the JPL and the APIs at issue. Thus, Android's use of the Java API elements was "dictated by the function to be performed, . . . or by external factors such as compatibility requirements and industry demands." *Sega*, 977 F.2d at 1524. Under *Sega* and *Sony*, the Ninth Circuit would uphold the Order and rule that the class and method names and declarations and overall SSO are not copyrightable.[136]

If the Java API class and method names and declarations are not copyrightable under Ninth Circuit law, then the same conclusion applies *a fortiori*

[136] The SSO of the 37 packages is limited to the functional requirements for compatibility, so there is no need to reach the fair-use question addressed in *Sega* and *Sony*.

to the "overall SSO" of the 37 packages—*i.e.*, their class-and-package structure. As Oracle itself admits,[137] Java's fully qualified method names do not merely ***reflect*** or ***reveal*** the hierarchical organization of the Java class libraries—they ***dictate and determine*** that organization.[138] Thus, there is effectively no "daylight" between the unprotected class and method names and the "overall SSO" that Oracle seeks to protect—no additional structure that is not already implicit in the names. Indeed, the "overall SSO" is, if anything, one step further removed from copyrightable expression because it is even more of an abstract "idea" or "concept" than the names themselves.

Oracle barely mentions *Sega* and *Sony*, preferring to brush them aside as "fair use" cases; but their fair-use rulings were predicated and dependent upon fully considered holdings that the compatibility elements of computer programs ***are not copyrightable*** under section 102(b). In both cases, the challenged copying of the plaintiffs' works was found "fair" precisely because it was done in furtherance of the defendants' legitimate need to study, understand, and use the "unprotected," "functional" elements necessary to achieve compatibility between

[137] *See* Br.39 (copied code "identifies, specifies, and defines the components and their arrangement within the packages"); *id*. at 45 (copied lines "embody the structure of each package").

[138] A20939-45.

programs. *See Sega*, 977 F.2d at 1514, 1520, 1525, 1526; *Sony*, 203 F.3d at 602, 603.

These predicate holdings are not *dicta*. In *Sega*, the copyrightability issue was "germane to the eventual resolution of the case" and was resolved after briefing[139] and reasoned consideration in a published opinion. *Garcia v. Holder*, 621 F.3d 906, 911 (9th Cir. 2010). *Sony* accordingly treated *Sega*'s copyrightability teachings as binding Circuit law. *Sony*, 203 F.3d at 603. And the practical effect of those holdings extends beyond fair use to the more fundamental issue of copyrightability: "After *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." Pamela Samuelson, *Are Patents on Interfaces Impeding Interoperability*? 93 Minn. L. Rev. 1943, 1959 (2009).[140]

5. *Sega* and *Sony* dispose of Oracle's key arguments.

Sega and *Sony* also compel rejection of the six main arguments that Oracle and its amici proffer on appeal.

---

[139] *Compare* Sega's Opening Brief, 1992 WL 12011898 at 15-22, *with* Accolade's Appellee's Brief, 1992 WL 12011899 at 35-38.

[140] Congress later reinforced that message by creating a safe harbor in the Digital Millenium Copyright Act for those who circumvent anti-copying technology to identify and analyze program elements "necessary to achieve interoperability . . . with other programs . . . ." § 1201(f).

a. Oracle's argument for exempting software from the normal section 102(b) filtration process is wrong.

Oracle accuses the district court of practicing "software exceptionalism" by denying the 37 packages the copyright protection granted to works of imaginative fiction. But it is Oracle that seeks preferential treatment by asking this Court to exempt the Java API from the normal process of section 102(b) filtration.

Oracle's plea misapprehends the law. The Ninth Circuit (like all other courts) rejects Oracle's assertion that the Copyright Act protects computer programs and works of imaginative fiction equally. Rather, courts recognize "that some works are closer to the core of intended copyright protection than others," *Campbell v. Acuff-Rose Music, Inc.*, 510 U.S. 569, 586 (1994), and that "not all copyrighted works are entitled to the same level of protection." *Sega*, 977 F.2d at 1524.

While elements of some computer programs "may be highly creative and idiosyncratic," they are, "in essence, utilitarian articles—articles that accomplish tasks. As such, they contain many logical, structural, and visual display elements that are dictated by the function to be performed, by considerations of efficiency, or by external factors such as compatibility requirements and industry demands." *Sega*, 977 F.2d at 1524. The presence of such unprotected functional elements results in "a lower degree of protection" for computer programs and platforms than for "more traditional literary works." *Id.* at 1526; *accord Sony*, 203 F.3d at 603.

There is nothing unfair or discriminatory about this, because all works are subjected to the same rules of copyrightability. However, applying the same rules evenhandedly to different works naturally produces different levels of protection. "To the extent that a work is functional or factual, it may be copied, as may be those expressive elements of the work that 'must necessarily be used as incident to' expression of the underlying ideas, functional concepts, or facts." *Sega*, 977 F.2d at 1524 (citations omitted). As a result, "[w]orks of fiction"—like *Harry Potter*—"receive greater protection than works that have strong factual elements, such as historical or biographical works, or works that have strong functional elements, such as accounting textbooks. Works that are merely compilations of fact are copyrightable, but the copyright in such a work is 'thin.'" *Id*. (citations omitted). Likewise, if a work is "largely functional," like software, "it receives only weak protection. This result is neither unfair nor unfortunate. It is the means by which copyright advances the progress of science and art." *Sega*, 977 F.2d at 1527 (citation and quotation marks omitted).

As applied to software, "thin" or "weak" protection means that section 102(b) filtration typically leaves behind two elements for copyright protection:

(1) Literal elements (source and object code), except insofar as specific sequences in that code were "dictated by the function to be

performed" or must be copied to meet "compatibility requirements" or "industry standards." *Sega*, 977 F.2d at 1524.

(2) Non-literal elements that are ***not*** necessary to the program's function. *See Computer Assocs. Int'l, Inc. v. Altai, Inc.*, 982 F.2d 693, 704-05 (2d Cir. 1992). By contrast, courts have identified many different functional aspects of software that "may not be entitled to copyright protection when they are the subject of nonliteral copying." 1 Ian C. Ballon, E-COMMERCE AND INTERNET LAW § 4.07[6] (2012-2013 update) (collecting examples).[141]

Whether a particular program element gets filtered out depends on the "factual nuances" of the technology. *See Bateman v. Mnemonics, Inc.*, 79 F.3d 1532, 1548 & n.33 (11th Cir. 1996). In *Atari*, for example, this Court (applying Ninth Circuit law) held that the district court did not clearly err in finding that the particular programming code used to unlock a game console was ***non-***functional and thus protectable, because the plaintiff "chose arbitrary programming

[141] In the software context, courts apply various versions of the "analytic dissection" method to filter out unprotectable program elements. But the method is complicated and courts only apply it to the extent that it is "helpful," *Apple Computer, Inc. v. Microsoft Corp.*, 35 F.3d 1435, 1443 (9th Cir. 1994), recognizing that "[n]ot every case requires an extensive abstraction-filtration-comparison analysis." *Mitel, Inc. v. Iqtel, Inc.*, 124 F.3d 1366, 1372 (10th Cir. 1997). "It is essential that one keep in mind that the approaches adopted by the circuits merely are a means to a very important end: filtering out all unprotectable material." *Bateman v. Mnemonics, Inc.*, 79 F.3d 1532, 1545 n.27 (11th Cir. 1996).

instructions and arranged them in a unique sequence to create a purely arbitrary data stream" so that there were "a multitude of different ways to generate a data stream which unlock[ed]" the console. 975 F.2d at 840.

By contrast, *Sega* applied the same rules to a seemingly similar situation concerning the codes required for game-platform compatibility—yet reached a different conclusion due to differences in the technology. *Sega* noted that *Atari* is "consistent both with our analysis and the result we reach," but distinguished *Atari*'s facts because "Sega's key appears to be functional" and there was "no showing that there is a multitude of different ways to unlock [Sega's] Genesis III console." *Sega*, 977 F.2d at 1514n.1, 1524 n.7. Accordingly, the TMSS initialization code in *Sega* fell into the category of "unprotected ideas and functional concepts." *Id.* at 1520, 1522. Likewise, the Ninth Circuit found in *Sony* that, "to develop a PlayStation emulator, Connectix ***needed to emulate*** both the PlayStation hardware and the firmware (the Sony BIOS)." 203 F.3d at 601 (emphasis added). As in *Sega*, there was no showing of multiple ways to achieve compatibility. The PlayStation's "system interface procedures" therefore were unprotected ideas and functional concepts filtered out by section 102(b). *Id.* at 600, 605.

Following these precedents, Judge Alsup examined the Java technology in detail and found, as a fact, that the SSO was not copyrightable, because it was

essential for invoking the functions of the 37 packages and for achieving interoperability. Oracle demonstrates no clear error in those findings.

Exempting the Java API from the normal section 102(b) filtration process as Oracle requests would upset the careful balance that copyright law strikes between "providing exclusive rights to expression" and "'encourag[ing] others to build freely upon the ideas and information conveyed by a work.'" *Atari*, 975 F.2d at 842 (citation omitted). Like other utilitarian works, software constantly is being refined by incremental, functional improvements. Granting monopolies on such improvements imperils progress by denying "those skilled in the relevant art" the chance to "try their hands at incremental improvement of functional works." Dennis S. Karjala, *Distinguishing Patent and Copyright Subject Matter*, 35 CONN. L. REV. 439, 454 (2003) [hereinafter *Distinguishing*]. The Court should reject Oracle's bid for special treatment.

b. Oracle's "creativity" argument is wrong.

Oracle and its amici miss the point when they harp on the theme that writing a Java API package is creative and challenging.[142] Originality and creativity are relevant primarily to the threshold section 102***(a)*** copyright-eligibility inquiry, not to the section 102***(b)*** filtration stage that concerned the district court here.

[142] *See* Br.3 6,12,18,30,31-32.

Under *Sega* and *Sony*, once a program element is found to be necessary for compatibility, section 102(b) takes over and the focus shifts from originality to functionality. Indeed, both cases denied copyright protection to functional elements that ***were*** original, complex, and creative. In *Sony*, the inter- and intra-chip signals sent and received by the Sony BIOS were both complex and integral to the working of the PlayStation platform—not just simple, numerical compatibility codes or "trivial communication protocols."[143] Yet those signals went unprotected.

Likewise, the TMSS code in *Sega* was not just a simple system for restricting compatibility; it also doubled as a clever device for imposing trademark and unfair-competition liability on anyone who managed to achieve compatibility through reverse engineering. *See Sega*, 977 F.2d at 1515. Despite this ingenious dual function, however, the Ninth Circuit held that the TMSS code was unprotected by copyright because it was required for compatibility. Nor did it matter that Sega had invested "considerable time, effort, and money" in developing its product, for the Supreme Court had "unequivocally rejected the 'sweat of the brow' rationale for copyright protection." *Id.* at 1527 (quoting *Feist*, 499 U.S. at 359-60).

[143] Br.9,10.

Thus, the fact that designing the Java API was creative and "labor-intensive" is irrelevant to whether the SSO of the 37 packages is necessary for compatibility and thus uncopyrightable under section 102(b).

c. Oracle's merger arguments are wrong.

The district court found that there is only one way to write Java method declarations; and "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 [method] declarations."[144]

Oracle argues that the district court erred in applying an "*ex post*" merger analysis that examined the expressive choices available to the alleged infringer (Google) when it created the allegedly infringing work (Android). Instead, argues Oracle, the court should have applied an "*ex ante*" merger analysis that considers only the expressive choices available to the plaintiff/author (Sun) when it created the allegedly infringed work (Java). Under that view, no merger occurred because Sun exercised creativity in choosing from among a multitude of ways that it could have organized the Java API.

Like its "creativity" argument, Oracle's *ex ante* view of merger improperly conflates the section 102(a) copyright-eligibility analysis with the section 102(b)

[144] A164.

copyright exclusions. When applying section 102(a), the issue is whether the allegedly infringed work is sufficiently original to be eligible for copyright protection. In that context, it may make sense to look at the options available to the author when she created the work (*ex ante*).

But when applying section 102(b), the issue (under *Sega* and *Sony*) is whether some aspect of the work has become an industry standard or is necessary for compatibility. In that context, it only makes sense to look at the options available to the alleged infringer (*ex post*).

Indeed, *Sega* and *Sony* specifically endorse the district court's *ex post* view of merger. *Sega* acknowledged that, "[t]o the extent that there are many possible ways of accomplishing a given task or fulfilling a particular market demand, the [original] programmer's choice of program structure and design may be highly creative and idiosyncratic." 977 F.2d at 1524. "In some circumstances," however, "even the exact set of commands used by the programmer is deemed functional rather than creative for purposes of copyright. '[W]hen 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 infringement.'" *Id.* (emphases added) (citation omitted). Significantly, *Sega* observed that there had been "no showing that there [was] a multitude of different ways" for Accolade—the defendant—to "unlock the Genesis III console" so that it could make Genesis-

compatible game cartridges. *Id.* at 1524 n.7. *Ex ante*, of course, plaintiff Sega had chosen from among a "multitude of different ways" to unlock its console; but Sega's ability to make original and creative choices *ex ante* did not render the choices that it made copyrightable.

Thus, *Sega* demonstrates that, "in the Ninth Circuit, . . . it is the range of expressive choice that existed at the time the ***competing*** product was created—not the range of expression that existed at the time the ***copyrighted*** work was created—that controls" the merger analysis. 1 Paul Goldstein, GOLDSTEIN ON COPYRIGHT § 2.3.2 & n.46, at 2:39-2:40 (3d ed. 2013) [hereinafter GOLDSTEIN] (emphases added); *cf. Lotus*, 49 F.3d at 816 (holding that "[t]he fact that Lotus developers could have designed the Lotus menu command hierarchy differently" was "immaterial" under section 102(b)).

Here, *Sega*'s *ex post* analysis required the district court to consider whether using 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 JPL. The district court found that it was, and Oracle demonstrates no clear error in that finding.[145]

[145] Oracle and an amicus cite *Practice Management Information Corp. v. AMA*, 121 F.3d 516 (9th Cir. 1997), as authority that industry standards have no bearing on copyrightability. But that case held that a defendant could not engage in "wholesale copying" of ***a book describing*** a code system. *Id*. at 520 n.8. The book identified "more than six thousand medical procedures" and provided "a five-

One amicus objects, however, that the Ninth Circuit's *ex post* view of merger could result in a program element losing copyright protection over time as it becomes an industry standard with which programmers seek to achieve compatibility.[146] At most, this would occur on the level of individual compatibility elements, not entire programs, because that's how section 102(b) works. Moreover, the argument fails for two reasons.

First, *Sega*'s *ex post* analysis cites not only "industry demands" but also "compatibility requirements" as constraints on the defendant's choices. A compatibility requirement restricts choice just as much on the program's release date as it does a decade later, and is therefore uncopyrightable from the start under *Sega*.

Second, to the extent that merger occurs because a program element becomes an industry standard, the loss of intellectual-property protection over time is nothing new. The same issue arises when a formerly distinctive trademark like "Aspirin" or "Thermos" eventually becomes a descriptive, generic designation that competitors "must be free to use if they are to be able to enter the market."

---

digit code ***and brief description for each***." *Id*. at 517 (emphasis added). Copying the entire book was as if Baker had copied the entire bookkeeping treatise in *Baker v. Selden*, including Selden's copyright-protected explanatory essay. Here, by contrast, Google did ***not*** copy any explanatory work. *See* A42(finding that Google did not copy Java documentation).

[146] Doc.46,p.24.

1 GOLDSTEIN § 2.3.2 at 2:41-2:42 (footnote omitted). Likewise, the s*cènes à faire* doctrine—which was "originally developed to recognize that certain plot structures are to be expected from works exploring certain literary or dramatic themes"—has been "adapted, especially in the software copyright case law, to recognize that expressive choices of subsequent authors may ***become constrained over time*** by the emergence of industry standards." Pamela Samuelson, *Questioning Copyrights in Standards*, 48 B.C. L. REV. 193, 215 (2007) (footnote omitted) (emphasis added). This loss of protection for works that become industry standards reflects the fact that, "over time," a work's "importance may come to reside more in the investment that has been made by ***users*** in learning" and using it. *Lotus*, 49 F.3d at 819-20 (Boudin, J., concurring) (emphasis in original).

d. Oracle's interoperability arguments are wrong.

The district court held that "interoperability is at the heart of the command structure" formed by Java's package-and-class scheme and "sheds further light on the character of the command structure as a system or method of operation."[147] But Oracle insists that interoperability (*i.e.*, compatibility) is irrelevant and had no place in the district court's analysis.[148]

[147] A167.

[148] Br.64.

Sega and *Sony* show that Oracle is wrong, as they held that copyright does not protect program elements "dictated by . . . external factors such as compatibility requirements and industry demands." *Sega*, 977 F.2d at 1524; *accord Sony*, 203 F.3d at 603. Indeed, courts have found a wide variety of interoperability issues pertinent to software protection, including "hardware standards and mechanical specifications, software standards and compatibility requirements, computer manufacture design standards, target industry practices and demands, and computer industry programming practices." *Gates Rubber Co. v. Bando Chem. Indus., Ltd.*, 9 F.3d 823, 838 (10th Cir. 1993) (collecting cases) (citations omitted). Most courts now agree that copyright does not protect interfaces required for interoperability. *See* Jonathan Band & Masanobu Katoh, INTERFACES ON TRIAL 2.0 at 36 (2011) [hereinafter INTERFACES].

Oracle and its amici also argue that interoperability is "all or nothing": to achieve it, Google must implement all 166 Java packages and Android must run all existing Java programs. But the district court correctly held that "'full compatibility' is not relevant to the Section 102(b) analysis."[149] The court noted that, "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

[149] A168.

tested. This parallels Google's decision to implement some but not all of the Java API packages in Android."[150]

Notably, the IEEE defines interoperability as "the ability of two or more systems or elements to exchange information and to use the information that has been exchanged." THE IEEE STANDARD DICTIONARY OF ELECTRICAL AND ELECTRONICS TERMS 548 (6th ed. 1997). Nothing in that definition requires that the two systems be identical. Moreover, there is no reason to consider the interoperability of non-accused program elements. In *Sony*, the Ninth Circuit limited its interoperability analysis to the 137 BIOS functions reimplemented by the defendant. Here, too, what matters is the interoperability of the 37 accused Android packages; and those packages meet the IEEE definition of interoperability with respect to existing Java programs that use the same packages. Although amicus BSA asserts that this interoperability is merely "hypothetical," it cites no facts to support that contention, which is both wrong and contrary to the district court's many careful findings on interoperability.

e. Oracle's "fragmentation" arguments are wrong.

Oracle also asserts that Google's failure to include the full Java API in Android "fragments" the platform, thereby impeding rather than fostering interoperability. The district court detected the irony in that argument, noting that

[150] A168.

it "almost leav[es] the impression that if only Google had replicated ***all*** 166 Java API packages, Oracle would not have sued."[151] And Oracle's complaint that fragmentation compromises the functionality of the Java platform only underscores "the character of the command structure as a functional system or method of operation" that cannot be copyrighted.[152]

In any event, Oracle's fragmentation arguments are an irrelevant and hypocritical diversion. No statute or case law prohibits fragmenting. Oracle claims that its Java Specification License imposes anti-fragmentation obligations contractually; but the district court recognized that that argument "begs the question whether or not a license was required in the first place to replicate some or all of the command structure."[153] The answer is "no," because Google had no obligation to license a command structure that lacked intellectual-property protection.

Ironically, ***Sun itself*** fragmented Java into multiple incompatible versions.[154] The Sun executive in charge of Java knew that Sun was fragmenting Java and that "this was destructive to the overall [write]-once-run-anywhere value proposition"—but he "didn't care because it was the only way he could see to

[151] A167 (emphasis in original).

[152] A167.

[153] A167.

[154] A20704-10,21612.

make money."[155] Java's self-fragmentation made the platform "ever less stable," leading some Sun employees to propose a "OneJava" project aimed at "[c]ommonizing Java for Java ME and SE and EE."[156] But Sun never pursued OneJava.[157] Under Oracle's "all-or-nothing" criterion for interoperability, therefore, Java isn't even "interoperable" with itself.

f. Oracle's "commercial expediency" argument is wrong.

Oracle and its amici also contend that Google focused on commercial expediency rather than technical interoperability. In other words, Google was just trying to make Android more attractive to programmers who know the Java API conventions. But it is both lawful and desirable to adopt an industry standard that allows programmers to express themselves using familiar methods and interfaces and to reuse code that is "clearly [their] own work product" on multiple platforms. *Lotus*, 49 F.3d at 818. Indeed, the district court found that using the Java API method names, declarations, and SSO gave Android "a degree of interoperability" with existing Java programs that used the same names, declarations, and SSO.[158] Oracle demonstrates no clear error in that finding.

[155] A6234,22103-04,22108-10.

[156] A22093-96.

[157] A22096.

[158] A135,167.

Oracle's "commercial expediency" argument also fails in light of the facts in *Sega* and *Sony*. In those cases, defendants Accolade and Connectix surely were motivated by commercial expediency when they sought to achieve interoperability with platforms that gamers around the world already knew and loved. But the Ninth Circuit still held that the platforms' functional compatibility elements were unprotected. The same approach applies here. "[I]ndustry programming practices" constrain developers' expressive choices just as much as technical compatibility. *Gates Rubber*, 9 F.3d at 838. The question is "not whether any alternatives theoretically exist; it is whether other options practically exist under the circumstances" and are "feasible within real-world constraints." *Lexmark Int'l, Inc. v. Static Control Components, Inc*., 387 F.3d 522, 536 (6th Cir. 2004). Programmer expectations and capabilities are real-world feasibility constraints; and courts likewise acknowledge that program elements are dictated by external factors if selected in response to "customer demand." *Mitel, Inc. v. Iqtel, Inc*., 124 F.3d 1366, 1375 (10th Cir. 1997).

Oracle's "commercial expediency" argument also fails to grasp that programmers are attracted by an API's ***functional*** characteristics—not by some aesthetic quality that is independent of, and separable from, those characteristics. *See Distinguishing* at 448-51. An API's SSO attracts programmers if it makes it convenient for them to find the methods they need and thus to write their

applications more quickly and efficiently—an inherently functional objective that denotes a lack of copyrightability under section 102(b). *Cf. Apple Computer, Inc. v. Microsoft Corp.*, 799 F. Supp. 1006, 1023 (N.D. Cal. 1992) (finding Apple's user interface uncopyrightable because its "collection of visual displays and user commands" was "designed to render use of the computer . . . more 'utilitarian'"), *clarified*, 27 U.S.P.Q.2d 1081 (N.D. Cal. Apr. 14, 1993), *aff'd*, 35 F.3d 1435 (9th Cir. 1994).

C. The *Lotus* decision supports the district court's finding that the SSO is a method of operation excluded from copyright protection under section 102(b).

The district court held that the SSO of the 37 packages was a "command structure"—Oracle CEO Larry Ellison's phrase[159]—that functioned as an uncopyrightable "system" or "method of operation" under section 102(b). As discussed below, that holding finds direct support in *Lotus*, 49 F.3d 807, which likewise held that a spreadsheet program's "menu command hierarchy" was a "method of operation" excluded from copyright protection under section 102(b).

1. *Lotus* held that a program element is an uncopyrightable "method of operation" if it is essential to accessing, controlling, or using the program.

Plaintiff Lotus owned the copyright in a spreadsheet program called Lotus 1-2-3 that performed accounting functions. 49 F.3d at 809. Users manipulated and

[159] A20457-58.

controlled the program via a series of menu commands, such as "Copy," "Print," and "Quit," which they selected either by highlighting them on the screen or by typing their first letter. Lotus 1-2-3 had 469 commands arranged into more than 50 menus and submenus. *Id*. The program also allowed users to write customized "macros" that enabled them to execute a series of commands automatically by typing in a single pre-programmed macro keystroke. Macros shortened the time needed to set up and operate the program. *Id*. at 809-10. In the business world, a macro could have thousands of steps and represent a significant investment by the user. INTERFACES at 26.

Defendant Borland released its competing "Quattro" spreadsheet program after three years of development. Quattro included a "Lotus Emulation Interface" that allowed users to control the program through a virtually identical copy of the entire Lotus 1-2-3 menu tree, as an alternative to using Quattro's own command system. But Borland didn't copy Lotus's underlying computer code; it copied only the words and structure of Lotus's menu command hierarchy so that spreadsheet users familiar with Lotus 1-2-3 could switch to Borland's programs without learning new commands or rewriting their Lotus macros. *Id*. at 810.

Borland later removed the Lotus Emulation Interface from its products, but retained a "Key Reader" that recognized and interpreted Lotus 1-2-3 macros and thus allowed users to keep using their old macros. *Id*. at 811-12. The Key Reader

file included a virtually identical copy of the Lotus menu tree structure, but used only the first letters of Lotus command terms. *Id*. at 812.

The district court found that the Lotus Emulation Interface and the Key Reader infringed Lotus's copyrights, and it permanently enjoined Borland. *Id*. But the Court of Appeals reversed on the ground that the Lotus menu command hierarchy was a "method of operation" excluded from copyright protection under section 102(b).

The appeals court explained that the term "method of operation" refers to "the means by which a person operates something, whether it be a car, a food processor, or a computer." 49 F.3d at 815. "If specific words are essential to operating something, then they are part of a 'method of operation' and, as such, are unprotectable." *Id*. at 816. The Lotus menu command hierarchy met that definition because it "provid[ed] the means by which users control and operate Lotus 1-2-3 . . . . Without the menu command hierarchy, users would not be able to access and control, or indeed make use of, Lotus 1-2-3's functional capabilities." *Id*. at 815.

In the software context, therefore, *Lotus* held that a program element is an uncopyrightable "method of operation" under section 102(b) if it is essential to accessing, controlling, or using the program. Critically, it does not matter whether that method is original, creative, or expressive. That Lotus's developers could

have designed the menu command hierarchy differently was "immaterial," because "the 'expressive' choices of what to name the command terms and how to arrange them [did] not magically change the uncopyrightable menu command hierarchy into copyrightable subject matter." 49 F.3d at 816; *see also Apple*, 799 F. Supp. at 1023.

Equally central to *Lotus* was the fact that "users employ[ed] the Lotus menu command hierarchy in writing macros." 49 F.3d at 818. This fact confirmed that the menu command hierarchy was a method of operation, because users employed it to write macros that "perform[ed] an operation automatically." *Id.*

Macros also illustrated the problem of "user lock-in": Allowing a method of operation to be copyrighted harms competition by making it too costly for users who have invested in the old product to switch to a new one. Accepting Lotus's copyright claim would have forced a Lotus 1-2-3 user who wished to use Quattro "to rewrite his or her macro using [another] program's menu command hierarchy . . . despite the fact that the macro is clearly the user's own work product." *Id.* "[F]orcing the user to cause the computer to perform the same operation in a different way ignores Congress's direction in § 102(b) that 'methods of operation' are not copyrightable." *Id.*[160]

[160] Judge Boudin's concurrence also discussed "user lock-in," observing that "[r]equests for the protection of computer menus present the concern with fencing off access to the commons in an acute form. . . . Better typewriter keyboard layouts

The appellate court's reversal of the injunction in *Lotus* "lifted the cloud of uncertainty that had been hanging over developers of interoperable software products and their many customers." INTERFACES at 36. Like the Ninth Circuit in *Sega* and *Sony*, the First Circuit "ruled unambiguously that program elements necessary to achieve interoperability—to attach as well as to compete—were, by definition, methods of operation not protected by copyright." *Id*. at 36-37.

2. Under *Lotus*, the Java API SSO is an uncopyrightable method of operation because it is essential to accessing, controlling, or using the packages.

Lotus's reasoning affirms that the SSO of the 37 packages is an uncopyrightable method of operation under section 102(b).

The district court found that the method names in the 37 packages "are more than just names—they are symbols in a command structure"[161] that is "a system or method of operation of the application programming interface. The commands are (and must be) in the form Java.package.Class.method() and each calls into action a pre-assigned function."[162] Thus, the overall class-and-package scheme is "a

---

may exist, but the familiar QWERTY keyboard dominates the market because that is what everyone has learned to use." *Lotus*, 49 F.3d at 819-20 (Boudin, J., concurring).

[161] A133.

[162] A166.

system or method of operation—a long hierarchy of over six thousand commands to carry out pre-assigned functions."[163]

These findings align perfectly with *Lotus*'s holding that a program element is an uncopyrightable "method of operation" if it is essential to accessing, controlling, or using the program. Here, the relevant "user" is the programmer who uses the JPL to write new applications. For that user, the Java class and method names, declarations, and command structure (or SSO) are essential to accessing, controlling, and using the Java API packages. Accordingly, those elements constitute an uncopyrightable "method of operating" the Java class libraries.

Lotus's compatibility teachings are equally on-point. The user-created macros in *Lotus* needed the Key Reader to interoperate with Quattro. Similarly, existing Java applications that use the 37 packages need the Java class and method names, declarations, and overall SSO to interoperate with Android. And just like the menu command hierarchy in *Lotus*, the SSO is familiar to users around the world who have invested in that system by learning it and programming in it. Reversing the district court's judgment would "forc[e those users] to cause [their programs] to perform the same operation in a different way"—a result that "ignores Congress's direction in § 102(b) that 'methods of operation' are not

[163] A166-67.

copyrightable." 49 F.3d at 818.

Lotus fully supports the district court's ruling—which is why Oracle and its amici mount a series of misguided attacks on it.

3. Oracle fails to distinguish or discredit *Lotus*.

Oracle first tries to distinguish *Lotus* on the ground that Borland did not use Lotus's "underlying computer code"; but the same is true here—unless one counts the "7,000 lines of declaring code" (less than 3% of the lines in the 37 packages) that the district court found Google ***had*** to use to achieve interoperability.[164] As demonstrated at Part V.D., below, there is no claim in this case for the infringement of 7,000 lines of code, considered apart from the packages' SSO. Moreover, the declarations that Google used are closely analogous to what Borland used: just as the Lotus menus were the means by which users invoked Lotus functions, the Java API is the means by which users invoke Java class-library functions.

Oracle also criticizes *Lotus* for defining "method of operation" so broadly as to render all computer programs uncopyrightable. Not so. *Lotus* held that the exclusion applies to means that a user must employ to access, control, and use a product. Users operated the Lotus 1-2-3 spreadsheet through its menu command hierarchy, not through the countless other elements that were not captured by

[164] Br.62,A136,163,165,169.

Lotus's "method of operation" definition. Indeed, the decision states that the screen displays (apart from the command structure) as well as the code ***implementing*** the Lotus Emulation Interface and the Key Reader still could receive copyright protection. 49 F.3d at 815-16. Thus, applying the *Lotus* "method of operation" definition to computer programs will not strip all programs of copyright protection as Oracle predicts—indeed, it didn't even strip protection from most elements of Lotus 1-2-3.

Oracle infers from the fact that the Supreme Court affirmed *Lotus* without opinion by a 4-4 vote in 1996 that *Lotus* is doomed to eventual overruling. But Oracle can only speculate why a Supreme Court with three different members voted as it did 17 years ago, or how the current Supreme Court would vote now in a hypothetical case presenting the exact same facts. If anything, "the flow of the [*Lotus*] oral argument suggests that none of the justices was troubled by the First Circuit's refusal to extend copyright protection to program elements necessary for software interoperability." INTERFACES at 36.

Finally, Oracle asserts that the Tenth Circuit has explicitly declined to adopt *Lotus*'s approach to section 102(b). But that court simply found a different route to the same result by holding that a literally copied system of computer codes was unoriginal and also was uncopyrightable under the *scènes à faire* doctrine. *See Mitel*, 124 F.3d at 1373-76. Although the court preferred the "abstraction-

filtration-comparison method" over *Lotus*'s "method of operation" approach (*id*. at 1371-72), it declined to apply the preferred method because it was needlessly complex for the case at hand. *Id*. at 1373. If anything, *Mitel* shows that courts have "selected different legal theories" to reach the common conclusion that "developers should be permitted to make copies of . . . programs to the extent necessary to achieve interoperability." INTERFACES at 50.

D. Oracle's SSO claim is its only claim; it has no "independent" argument based on the copyrightability of 7,000 lines of non-implementing code.

Sensing that its SSO claim may not be salvageable, Oracle argues that—SSO aside—there is an "independent" ground for reversal based purely on the copyrightability of the 7,000 lines of non-implementing code[165] comprising the class and method names and declarations of the 37 packages.[166]

Oracle is wrong for two reasons.

First, there is only one way to write the names and declarations that make up those lines; and "when there is only one way to write something, the merger doctrine bars anyone from claiming exclusive copyright ownership of that expression."[167]

---

[165] We avoid Oracle's phrase "declaring code" because it obscures the fact that Java names and declarations are ***not*** implementing code. A22362,22390-91,134n.4. While implementing code may be copyrightable, it is undisputed that Google implemented the packages independently by writing its own implementing code or acquiring it from open source.

[166] *See* Br.31-32.

[167] A164.

Second, both here and below, Oracle did not challenge jury instructions and a verdict form that ***barred*** the jury from considering whether the 7,000 lines were infringed on a "standalone" basis. Accordingly, that theory is not in the case, any argument on Oracle's "7,000-lines theory" is waived, and any alleged error in rejecting that theory would be harmless.

The district court instructed the jury that Oracle's copyrights do not cover the names of files, packages, classes, or methods, except to the extent that they "must necessarily be used as part of" the SSO.[168] In closing, Oracle's counsel explained that, under that instruction, "[a]n individual name is not protectable, but the names ***as part of the [SSO]*** are protectable by copyright."[169] The first and most important interrogatory on the verdict form asked only about ***non-literal*** infringement of the work's SSO—***not*** about literal infringement of 7,000 lines of non-implementing code:

> As to the compilable code for the 37 Java API packages in question taken as a group . . . [h]as Oracle proven that Google has infringed ***the overall structure, sequence and organization of*** the copyrighted works?[170]

---

[168] A22772.

[169] A22661 (emphasis added).

[170] A41(emphasis added). Although the definition of "compilable code" included "declarations," A22770, the jurors were not asked to determine whether any declarations were literally infringed.

The only interrogatories that ***did*** ask about literal copying were limited to the two tiny categories of code at issue in Google's cross-appeal.[171] As to that code ***only***, the court instructed that SSO was "irrelevant."[172]

Oracle waived any challenge to this aspect of the instructions and verdict form by failing to object below[173] or to brief the issue here. *See Smith v. Marsh*, 194 F.3d 1045, 1052 (9th Cir. 1999). Yet Oracle now claims that its 7,000-lines theory forms an independent basis for reversal.

Not so. A reversal based on copyrightability of the 7,000 lines viewed apart from SSO ***could not alter the judgment*** because it would not relate to the jury's verdict that ***the SSO*** was infringed. The claimed error is therefore harmless, because it was "inconsequential to the ultimate . . . determination." *Molina v. Astrue*, 674 F.3d 1104, 1122 (9th Cir. 2012) (citation omitted); *see* 28 U.S.C. § 2111. The 7,000-lines theory also is waived due to Oracle's failure to object to instructions and a verdict form that effectively eliminated that theory from the case. *See Yeti by Molly, Ltd. v. Deckers Outdoor Corp.*, 259 F.3d 1101, 1110-11 (9th Cir. 2001).

Thus, Oracle does not get a second bite at the apple if its SSO claim fails.

---

[171] A42,22773.

[172] A22773.

[173] A22613-14,22617-18.

E. Oracle's challenge to the court's words-and-short-phrases ruling fails.

Oracle's sole objection to the district court's application of the words-and-short-phrases doctrine[174] is that the doctrine cannot apply to "the copyright on an assemblage of 7000 lines of code."[175] As discussed above, however, there is no independent claim here based on literal infringement of "7000 lines of code."

F. If the Court reverses and remands, it should direct the district court to retry Google's fair-use defense.

Although the jury hung on Google's fair-use defense, nine presumably reasonable jurors reportedly found that Google had proved that defense;[176] and many of Judge Alsup's copyrightability findings confirm that a new jury could reach a unanimous decision in Google's favor on remand. Oracle's argument that Google should be precluded from retrying that defense on remand is therefore a clear case of overreaching and should be rejected.

1. A jury could find that Google's use was transformative, and thus fair.

In deciding the first statutory fair-use factor—purpose and character of the use[177]—the key question is whether the use is "transformative." *Campbell,* 510 U.S. at 579. A transformative use "'adds something new, with a further purpose or

---

[174] A164-65.

[175] Br.54.

[176] *See* A131,24622-24.

[177] *See* § 107 (listing fair-use factors).

different character, altering the first with new expression, meaning, or message.'" *Id.* at 579 (citation omitted); *see also Cariou v. Prince*, No. 11-1197-CV, 2013 WL 1760521, at *5-6 (2d Cir. Apr. 25, 2013).

Android is transformative, as shown by the district court's unchallenged factual findings. Google's independent implementation of the 37 packages "accounts for 97 percent of the lines of code in [those] packages" and is "different from the Java implementations."[178] Moreover, Android has "its own virtual machine" (the DVM), built with different code and operating in a different way than the JVM.[179]

Android also is transformative because it incorporates the 37 packages into an entirely new smartphone platform that accommodates existing works while making new creative works possible. Indeed, the asserted copyright is in Java SE—which was designed for desktops, not mobile phones. Android has fostered a world of new smartphone applications. A defendant's use of copyrighted material to create a new platform that is compatible with existing programs is "a legitimate one under the first factor of the fair use analysis." *Sony*, 203 F.3d at 607. Here, as in *Sega* and *Sony*, Google's use fosters the "dissemination of other creative works" through interoperability. *Sega*, 977 F.2d 1523.

[178] A135.

[179] A135-36.

Oracle and its amici argue that the fact that "Google considered, negotiated, and ultimately rejected" a partnership arrangement that might have included some form of IP license indicates bad faith and "weighs heavily" against a finding of fair use.[180] But *Campbell* rejected the argument that the defendant's request for permission to use the original "should be weighed against a finding of fair use," noting that the request "may simply have been made in a good-faith effort to avoid this litigation." 510 U.S. at 585 n.18. *Sega* likewise found Accolade's use fair as a matter of law even though it, like Google, "explored the possibility of entering into a licensing agreement with Sega, but abandoned the effort because" it did not like Sega's licensing terms. 977 F.2d at 1514.

Thus, a jury could find that the first statutory factor favors Google.

2. A jury could find that the "nature" of the work favors Google.

The second fair-use factor—the nature of the copyrighted work—"calls for recognition that some works are closer to the core of intended copyright protection than others." *Campbell*, 510 U.S. at 586. As *Sega* recognized, computer programs are "essentially utilitarian" in nature, and, under the Copyright Act, "if a work is largely functional, it receives only weak protection." 977 F.2d at 1527. "'[W]hen specific instructions, even though previously copyrighted, are the only and

[180] Br.72.

essential means of accomplishing a given task, their later use by another will not amount to infringement.'" *Id.* at 1524 (citation omitted).

Here, a jury could find—as the district court did—that the "command structure" of the 37 packages is the essential means for accomplishing interoperability with "third-party source code relying on [those] packages."[181] A jury also could find that the Java API is functional, and therefore entitled only to weak protection. Thus, a jury could find that the second factor favors Google.

3. A jury could find that Google only used what was necessary for interoperability—a small fraction of the overall code.

The third factor—the amount and substantiality of the use—asks whether the defendant used more of the copyrighted work than it needed to in light of "the purpose and character of the use." *Campbell*, 510 U.S. at 586-87. "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).

A jury could find—as the district court did—that "[c]omparing the 37 Java and Android packages side by side, only three percent of the lines of code are the same."[182] These "***must be*** identical" because "there is only one way to declare a given method"; and the "same is true for the 'calls,' the commands that invoke the

[181] A166-68.

[182] A136.

methods."[183] And "Google replicated what was necessary to achieve a degree of interoperability—but no more, taking care, as said before, to provide its own implementations."[184]

Thus, a jury could find that the third factor favors Google.

4. A jury could find that the market-effect factor favors Google.

The fourth factor—the use's effect on the market for or value of the copyrighted work—also favors Google because a transformative work like Android is "less likely" to cause a substantially adverse impact on the original's market than "a work that merely supplants or supersedes" the original. *Sony*, 203 F.3d at 607.

Android did not "supplant" Oracle in the market because, as the district court found, "Oracle never successfully developed its own smartphone platform using Java technology."[185] Oracle is therefore limited to arguing about the "market for Oracle's derivative works," by which it apparently means any smartphone platform using the Java APIs.[186] But the district court ruled that Google's

[183] A136 (emphasis in original).

[184] A167.

[185] A135.

[186] Br.75-77.

implementations of the Java APIs "are not derivative works. They are independent works that simply start with the idea of the specification."[187]

In any event, *Sega* defeats Oracle's argument that Google's use was unfair because it supposedly interfered with Oracle's licensing opportunities. By "facilitating the entry of a new competitor, the first lawful one that is not a Sega licensee," Accolade "undoubtedly 'affected' the market for Genesis-compatible games in an indirect fashion." 977 F.2d at 1523. But the fourth statutory factor still favored Accolade, because a contrary holding would allow copyright holders "to monopolize the market by making it impossible for others to compete," which "runs counter to the statutory purpose of promoting creative expression and cannot constitute a strong equitable basis for resisting the invocation of the fair use doctrine." *Id.* at 1523-24.

Thus, a jury could find that the fourth factor favors Google.

Accordingly, if this Court reverses the district court's judgment, it should allow Google to retry its fair-use defense on remand. As discussed in briefing filed below,[188] retrying fair use requires retrying infringement as well, because (1) that is the only way to guarantee a unanimous verdict by one jury on both liability and fair use, *see United States v. Southwell*, 432 F.3d 1050, 1055 (9th Cir. 2005), and

[187] A22612.

[188] A24604-11. The district court did not reach this issue because the copyrightability determination rendered it moot.

(2) the issues are too intertwined to be decided by separate juries. *See Witco Chem. Corp. v. Peachtree Doors, Inc*., 787 F.2d 1545, 1549 (Fed. Cir. 1986).

CROSS-APPELLANT'S BRIEF

I. JURISDICTIONAL STATEMENT

Google concurs in the Jurisdictional Statement on pages 5-6 of Oracle's Opening Brief. Google timely cross-appealed from a final order on October 4, 2012.[189] The cross-appeal is proper because Google "seeks to enlarge its own rights under the judgment or to lessen the rights of its adversary under the judgment." *Aventis Pharma S.A. v. Hospira*, Inc., 637 F.3d 1341, 1343 (Fed. Cir. 2011).

II. ARGUMENT

Google appeals from two erroneous decisions on the parties' post-trial JMOL motions.

The district court erred in granting Oracle's JMOL motion, thereby overruling the jury's verdict that Google did not infringe Oracle's copyright by duplicating eight "decompiled files"—a verdict supported by substantial evidence.

The district court also erred in denying Google's JMOL motion, which showed that the jury lacked a legally sufficient basis to conclude that Google committed copyright infringement when it used the rangeCheck function in the

[189] A24620-21,1122-23.

Arrays.java file of the Java 2 Standard Edition ("J2SE") platform. The district court's rulings on these two motions therefore should be reversed.

A. The "work as a whole" is the entire J2SE version 5.0 platform.

Over Google's objection,[190] the district court instructed the jury that, for purposes of verdict Question 3 (concerning infringement of the rangeCheck method), "the work as a whole' is the compilable code for the individual file . . . ."[191]

This was error. Registration fixes the scope of copyright protection. *See Express, LLC v. Fetish Group, Inc*., 424 F. Supp. 2d 1211, 1218 (C.D. Cal. 2006). Sun registered two "works" with the Copyright Office, and proceeded to trial on just one of them—Java 2 Standard Edition, Version 5.0.[192] Accordingly, the relevant "work as a whole" is the entire J2SE Version 5.0 platform.[193]

---

[190] A22592-93,22595.

[191] A989-93,22779.

[192] A524-705,675-84.

[193] Oracle cannot plausibly contend that the Arrays.java file, or any one of the eight decompiled files, constitutes the "essence" of the J2SE platform; indeed, it is not clear from the record what function any of these files performs. *Cf. Hustler Magazine Inc. v. Moral Majority Inc.*, 796 F.2d 1148, 1154-55 (9th Cir. 1986). And Oracle introduced no evidence that these files can "stand totally alone" or that they are "stored separately" from the rest of the J2SE platform. *Id*.; *see also Am. Geophysical Union v. Texaco Inc*., 60 F. 3d 913, 926 (2d Cir. 1994).

Accordingly, the jury should have been instructed to evaluate the alleged literal infringement of rangeCheck and the eight decompiled files in light of the fact that the entire J2SE platform is the copyrighted "work" at issue.

B. The district court should have denied Oracle's JMOL motion because substantial evidence supported the jury's verdict that Google's use of eight decompiled test files was de minimis.

The jury concluded that Google's use of the eight decompiled files was de minimis and did not infringe any copyright. Although the district court found that those files were "minor items," it erroneously granted Oracle's JMOL on this issue and overturned the jury's verdict.

"A jury's verdict must be upheld if it is supported by substantial evidence, which is evidence adequate to support the jury's conclusion, even if it is also possible to draw a contrary conclusion." *Pavao v. Pagay*, 307 F.3d 915, 918 (9th Cir. 2002). When ruling on a JMOL, a court should "disregard evidence favorable to the moving party that the jury is not required to believe." *Id*. In this case, Oracle, as the accuser, bore the burden of proof at trial. *See Granite Music Corp. v. United Artists Corp*., 532 F.2d 718, 723 (9th Cir. 1976).

For an "unauthorized use of copyright work to be actionable, the use must be significant enough to constitute infringement." *Newton v. Diamond*, 388 F.3d 1189, 1192-93 (9th Cir. 2004). No legal consequences attach to the copying of a copyrighted work "unless the copying is substantial." *Id*. at 1193.

"Substantiality is measured by considering the qualitative and quantitative significance of the copied portion in relation to the plaintiff's ***work as a whole***." *Id*. at 1195 (emphasis added). The copied portion of the copyrighted work is de minimis and not actionable unless it is qualitatively or quantitatively significant to the plaintiff's work. *See Newton*, 388 F.3d at 1193; *Ringgold v. Black Ent'mt Television, Inc.*, 126 F.3d 70, 74 (2d Cir. 1997).

The jury had ample evidence that Oracle failed to prove that Google's use of the eight decompiled files was significant enough to constitute infringement.

First, Oracle offered no evidence that the decompiled files were qualitatively significant to the J2SE platform or to the Android platform. Oracle's expert, Dr. Mitchell, testified that the files are "characterized as the default implementation for the security functions."[194] But Oracle offered no evidence or explanation of what that meant, or how it made the files qualitatively significant. Notably, there is no evidence that the eight files contributed a single byte to the millions of lines of code in the Android platform or ever appeared on an Android smartphone. At most, they were used to test the platform.[195] Dr. Mitchell vaguely suggested that testing tools "may" be significant; but he offered no specifics about whether these specific eight files were, in fact, significant to the J2SE platform or

[194] A21502.

[195] A21490-92,21502,21982-84,5875.

to the Android platform, or whether they were even used for testing in either platform.[196] Indeed, Oracle did not notice the copying of the eight files until it conducted an extensive forensic analysis of Android's source code in connection with this litigation.[197]

Second, Oracle offered no evidence of the eight files' quantitative significance to the Java platform. The trial testimony revealed nothing about the size of the eight files or the number of lines of source code that comprise them; nor did Oracle offer any testimony as to how the size of those files compared to the Java platform as a whole.

Thus, the trial record substantially supports the jury's verdict that Oracle failed to meet its burden as to Google's use of the eight decompiled files. The district court erred in granting Oracle's JMOL on this issue.

C. The district court should have granted Google's JMOL motion on Oracle's copyright claim because Google's use of the rangeCheck code was de minimis.

The trial evidence revealed that the nine lines of rangeCheck code were both quantitatively and qualitatively insignificant in relation to the J2SE platform.[198] In fact, those nine lines represented an infinitesimal percentage of the 2.8 million

[196] A21470-90.

[197] A21480-21485.

[198] *See* A142-43.

lines of code in the 166 Java packages[199]—let alone the millions of lines of code in the entire J2SE platform.[200] Not surprisingly, the district court found that Google's use of the rangeCheck function "was an innocent and inconsequential instance of copying in the context of a massive number of lines of code,"[201] and was "innocuous and overblown by Oracle."[202]

Google's use of those nine lines was therefore de minimis and could not constitute copyright infringement. Accordingly, the district court should have granted Google's JMOL on this issue.

[199] A22361,22368-22369.

[200] A22367-70.

[201] A143.

[202] A142.

CONCLUSION

The Court should affirm the copyrightability judgment while granting Google's cross-appeal on two minor issues of literal infringement. However, if the Court reverses the copyrightability judgment, it should direct the district court on remand to retry Google's fair-use defense (as well as the inseparable issue of infringement).

Dated: May 23, 2013

Respectfully submitted,

/s/ Robert A. Van Nest
ROBERT A. VAN NEST
STEVEN A. HIRSCH
KEKER & VAN NEST LLP
633 Battery Street
San Francisco, CA 94111-1809
Telephone: 415 391 5400
Facsimile: 415 397 7188
Attorneys for Defendant, Appellee, and Cross-Appellant GOOGLE INC.

United States Court of Appeals
for the Federal Circuit

ORACLE AMERICA, INC. *v.* GOOGLE INC., 2013-1021, -1022

CERTIFICATE OF SERVICE

I, Robyn Cocho, being duly sworn according to law and being over the age of 18, upon my oath depose and say that:

Counsel Press was retained by KEKER & VAN NEST LLP, Attorneys for Defendant-Cross-Appellant to print this document. I am an employee of Counsel Press.

On **May 23, 2013**, Counsel for Cross-Appellant has authorized me to electronically file the foregoing **Brief of Appellee and Cross-Appellant** with the Clerk of Court using the CM/ECF System, which will serve via e-mail notice of such filing to any of the following counsel registered as CM/ECF users:

Counsel for Appellant Oracle America, Inc.

E. Joshua Rosenkranz
Counsel of Record
drew D. Silverman
Orrick, Herrington & Sutcliffe LLP
51 West 52nd Street
New York, NY 10019s
212-506-5380
jrosenkranz@orrick.com
asilverman@orrick.com

Dale M. Cendali
Joshua L. Simmons
Kirkland & Ellis LLP
601 Lexington Avenue
Citigroup Center
New York, NY 10022
212-446-4800
dale.cendali@kirkland.com
joshua.simmons@kirkland.com

Mark S. Davies
Orrick, Herrington & Sutcliffe LLP
Columbia Center
1152 15th Street, N.W.
Washington, DC 20005
202-339-8631
mark.davies@orrick.com

Susan M. Davies
Kirkland & Ellis LLP
655 15th Street, N.W.
Washington, DC 20005
202-879-5000
susan.davies@kirkland.com

Annette Louise Hurst
Orrick, Herrington & Sutcliffe LLP
405 Howard Street
San Francisco, CA 94105
415-773-4585
ahurst@orrick.com
Elizabeth Cincotta McBride
Gabriel Morgan Ramsey
Orrick, Herrington & Sutcliffe LLP
1000 Marsh Road
Menlo Park, CA 94025
650-614-7377
emcbride@orrick.com
gramsey@orrick.com

Dorian Estelle Daley
Deborah Kay Miller
Matthew Sarboraria
Andrew C. Temkin
Oracle America, Inc.
Oracle Legal Department
500 Oracle Parkway
Redwood Shores, CA 94065
650-506-5200
dorian.daley@oracle.com
deborah.miller@oracle.com
matthew.sarboraria@oracle.com
andrew.temkin@oracle.com

Kenneth Alexander Kuwayti
Morrison & Foerster LLP
755 Page Mill Road
Palo Alto, CA 94304-1018
650-813-5600
KKuwayti@mofo.com

Sean Fernandes
Kirkland & Ellis LLP
3330 Hillview Ave.
Palo Alto, CA 94304
650-859-7014
sean.fernandes@kirkland.com

Michael Allen Jacobs
Morrison & Foerster LLP
Firm: 415-268-7178
425 Market Street
San Francisco, CA 94105
415-268-7455
mjacobs@mofo.com

Diana M. Torres
Kirkland & Ellis LLP
333 S. Hope Street
Los Angeles, CA 90071
213-680-8400
diana.torres@kirkland.com

Counsel for Amici Curiae

Jared Bobrow
Aaron Y. Huang
Weil, Gotshal & Manges LLP
201 Redwood Shores Parkway
Redwood Shores, CA 94065
650-802-3000
jared.bobrow@weil.com
aaron.huang@weil.com
Counsel for Amici Curiae
Eugene H. Spafford, et al.

Matthew S. Hellman
Paul March Smith,
Jenner & Block LLP
1099 New York Avenue, NW,
Suite 900
Washington, DC 20001
202-639-6861
mhellman@jenner.com
mith@jenner.com
Counsel for Amicus Curiae
BSA - The Software Alliance

William A. Rudy
Lathrop & Gage, LC
2345 Grand Boulevard
Kansas City, MO 64108
816-292-2000
wrudy@lathropgage.com
Counsel for Amici Curiae
Picture Archive Council of America, Inc., et al.

Brianna E. Dahlberg
Carole E. Handler
Lathrop & Gage LLP
1888 Century Park East
Suite 1000
Los Angeles, CA 90067
310-789-4600
chandler@lathropgage.com
bdahlberg@lathropgage.com
Counsel for Amici Curiae
Picture Archive Council of America, Inc., et al.

Gregory G. Garre
Lori Alvino McGill
Latham & Watkins LLP
Suite 1000
555 Eleventh Street, NW
Washington, DC 20004
202-637-2200
gregory.garre@lw.com
lori.alvino.mcgill@lw.com
Counsel for Amicus Curiae
Microsoft Corporation

Marcia Beth Paul
Deborah A. Adler
Lacy H. Koonce
Davis Wright Tremaine, LLP
1633 Broadway, 27th Floor
New York, NY 10019
212-489-8230
marciapaul@dwt.com
rahadler@dwt.com
koonce@dwt.com
Counsel for Amicus Curiae
Ralph Oman

Douglas B. Luftman
NetApp, Inc.
495 East Java Drive
Sunnyvale, CA 94089
408-822-6521
Douglas.luftman@netapp.com
Counsel for Amicus Curiae
NetApp, Inc.

Paul T. Dacier
Krishnendu Gupta
EMC Corporation
176 South Street
Hopkington, MA 01748
paul.t.dacier@emc.com
krish.gupta@emc.com
Counsel for Amicus Curiae
EMC Corporation

Steven Thomas Cottreau
Clifford Chance Rogers & Wells LLP
The William Rogers Building
2001 K Street, N.W.
Washington, DC 20005-1001
202-912-5000
steve.cottreau@cliffordchance.com
Counsel for Amici Curiae
Scott McNealy, et al.

Paper copies will also be mailed to principal counsel of record for Appellant at the time paper copies are sent to the Court.

Upon acceptance by the Court of the e-filed document, six paper copies will be filed with the Court, via Federal Express, within the time provided in the Court's rules.

May 23, 2013

/s/ Robyn Cocho
Robyn Cocho
Counsel Press

CERTIFICATE OF COMPLIANCE WITH TYPE-VOLUME LIMITATION, TYPEFACE REQUIREMENTS AND TYPE STYLE REQUIREMENTS

1. This brief complies with the type-volume limitation of Federal Rule of Appellate Procedure 32(a)(7)(B).

_X_ The brief contains __16,498__ words, excluding the parts of the brief exempted by Federal Rule of Appellate Procedure 32(a)(7)(B)(iii),or

____ The brief uses a monospaced typeface and contains _____ lines of text, excluding the parts of the brief exempted by Federal Rule of Appellate Procedure 32(a)(7)(B)(iii).

2. This brief complies with the typeface requirements of Federal Rule of Appellate Procedure 32(a)(5) and the type style requirements of Federal Rule of Appellate Procedure 32(a)(6).

_X_ The brief has been prepared in a proportionally spaced typeface using MS Word 2002 in a 14 point Times New Roman font or

____ The brief has been prepared in a monospaced typeface using MS Word 2002 in a ___ characters per inch_____ font.

Dated: May 23, 2013

/s/ Robert A. Van Nest
ROBERT A. VAN NEST
Attorneys for Defendant, Appellee, and Cross-Appellant GOOGLE INC.