

No. 2013-1021, -1022

---

IN THE  
**United States Court of Appeals  
for the Federal Circuit**

ORACLE AMERICA, INC.,

*Plaintiff-Appellant,*

v.

GOOGLE INC.,

*Defendant-Cross Appellant.*

---

ON APPEAL FROM THE UNITED STATES DISTRICT COURT FOR THE NORTHERN  
DISTRICT OF CALIFORNIA, CASE NO. 10-CV-3561, HON. WILLIAM H. ALSUP

---

**OPENING BRIEF AND ADDENDUM OF PLAINTIFF-  
APPELLANT**

---

Dorian E. Daley  
Deborah K. Miller  
Matthew Sarboraria  
Andrew C. Temkin  
ORACLE AMERICA, INC.  
500 Oracle Parkway  
Redwood Shores, CA 94065

E. Joshua Rosenkranz  
Mark S. Davies  
Andrew D. Silverman  
Kelly M. Daley  
ORRICK, HERRINGTON & SUTCLIFFE LLP  
51 West 52nd Street  
New York, NY 10019  
(212) 506-5000

Annette L. Hurst  
Gabriel M. Ramsey  
Elizabeth C. McBride  
ORRICK, HERRINGTON & SUTCLIFFE LLP  
405 Howard Street  
San Francisco, CA 94105-2669

Dale M. Cendali  
Diana M. Torres  
Sean B. Fernandes  
Joshua L. Simmons  
KIRKLAND & ELLIS LLP  
601 Lexington Avenue  
New York, NY 10022

---

*Attorneys for Plaintiff-Appellant*

---

## CERTIFICATE OF INTEREST

Counsel for appellant certifies the following:

1. We represent Oracle America, Inc.
2. That is the real name of the real party in interest.
3. Oracle America, Inc., is a division of Oracle Corporation.
4. The following law firms and partners or associates appeared

for Oracle America, Inc., in the Northern District of California or are expected to appear in this court:

ORRICK, HERRINGTON & SUTCLIFFE LLP:

E. Joshua Rosenkranz  
Mark S. Davies  
Annette L. Hurst  
Gabriel M. Ramsey  
Andrew D. Silverman  
Elizabeth C. McBride  
Kelly M. Daley

KIRKLAND & ELLIS LLP

Dale M. Cendali  
Susan Davies  
Diana M. Torres  
Sean Fernandes  
Joshua L. Simmons

MORRISON & FOERSTER LLP

Michael A. Jacobs  
Kenneth A. Kuwayti  
Marc David Peters  
Daniel P. Muino  
Mark E. Ungerman  
Richard S. Ballinger  
Roman A. Swoopes  
Ruchika Agrawal  
Rudolph Kim  
Yuka Teraguchi

BOIES, SCHILLER & FLEXNER LLP

David Boies  
Steven C. Holtzman  
Alanna Rutherford  
Beko Osiris Ra Reblitz-Richardson  
Meredith R. Dearborn  
William F. Norton, Jr.

Date: February 11, 2013

By: /s/ E. Joshua Rosenkranz  
*Attorney for Plaintiff-Appellant*

## TABLE OF CONTENTS

	Page
CERTIFICATE OF INTEREST .....	i
TABLE OF AUTHORITIES .....	vi
STATEMENT OF RELATED CASES.....	xi
INTRODUCTION .....	1
JURISDICTIONAL STATEMENT .....	5
STATEMENT OF THE ISSUES .....	6
STATEMENT OF THE CASE.....	6
STATEMENT OF FACTS .....	7
Sun Revolutionizes Computer Programming With Java And Its Packages Of Prewritten Programs.....	7
Writing A Java Package Is An Iterative And Creative Exercise.....	12
Sun Develops A Licensing Regime To Foster A Community And Ensure Compatibility .....	13
Google Is Desperate To Include Certain Java Packages In The Android Platform.....	16
Google Acknowledges It Needs A License But Wants To Defy “Write Once, Run Anywhere” .....	18
Google Copies Verbatim The Declaring Code In 37 Java Packages Into Android .....	20
Google’s Copying Fragments The Java Community And Marginalizes Sun/Oracle In The Smartphone Market.....	28
The Jury Finds Copyright Infringement, But The District Court Finds No Copyright Protection .....	29

SUMMARY OF ARGUMENT .....	31
STANDARD OF REVIEW .....	35
ARGUMENT .....	35
I. COPYRIGHT PROTECTS ORACLE’S SOFTWARE PACKAGES. ....	35
A. The Copyright Act Sets A Low Threshold For Protection And Applies The Same Standard To Software As Other Protectable Works. ....	36
B. The Declaring Source Code And Organization Of Each Package Is Protectable Expression. ....	38
1. Declaring source code is protectable because it is expressive. ....	40
2. The organization of each package is protectable as creative expression. ....	43
C. The District Court Erred In Concluding That Each Line Of Declaring Code Is Completely Unprotected. ....	48
1. The district court misapplied merger. ....	48
2. The district court misapplied “short phrases.” ....	53
D. The District Court Erred In Holding That The Organization Is Devoid Of Protection. ....	57
1. The district court erred in dismissing the organization as an unprotected “method of operation.” ....	59
2. The district court erred in invoking interoperability in support of “method of operation.” ....	63

II. GOOGLE CANNOT ESTABLISH THAT ITS COMMERCIALY MOTIVATED AND ILLICIT VERBATIM COPYING IS FAIR USE .....	68
A. Google’s Copying Was Commercially Motivated, Not Transformative, And Illicit. ....	69
B. Google Copied A Creative Work. ....	72
C. Google Verbatim Copied The Code And Structure That Matters To A Java Programmer. ....	73
D. Google’s Copying Damaged The Value Of The Java Platform In The Smartphone Market. ....	75
CONCLUSION .....	77

### ADDENDUM

Transcript of Proceedings, pages 3356-3439, Dated May 9, 2012 (Excerpt including Order regarding Google Inc.’s Motion for Judgment as a Matter of Law on Google Inc.’s Fair Use Defense).....	A44-128
Order on Motions for Judgment as a Matter of Law, Dated May 10, 2012 (Doc. 1119).....	A129
Order Re Copyrightability of Certain Replicated Elements of the Java Application Programming Interface, Dated May 31, 2012 (Doc. 1202).....	A130-70
Final Judgment, Dated June 20, 2012 (Doc. 1211).....	A171-72
Order Denying Motion for Judgment as a Matter of Law and New Trial, Dated July 13, 2012 (Doc. 1221).....	A173

**TABLE OF AUTHORITIES**

<b>FEDERAL CASES</b>	<b>Page(s)</b>
<i>Am. Dental Ass’n v. Delta Dental Plans Ass’n</i> , 126 F.3d 977 (7th Cir. 1997) .....	47, 61
<i>Apple Computer, Inc. v. Franklin Computer Corp.</i> , 714 F.2d 1240 (3d Cir. 1983).....	37, 60, 64, 65
<i>Apple Computer, Inc. v. Microsoft Corp.</i> , 35 F.3d 1435 (9th Cir. 1994) .....	59
<i>Applied Innovations, Inc. v. Regents of Univ. of Minn.</i> , 876 F.2d 626 (8th Cir. 1989) .....	55
<i>Atari Games Corp. v. Nintendo of America, Inc.</i> , 975 F.2d 832 (1992).....	passim
<i>Baker v. Selden</i> , 101 U.S. 99 (1880) .....	48
<i>Boisson v. Banian, Ltd.</i> , 273 F.3d 262 (2d Cir. 2001).....	50, 51
<i>Campbell v. Acuff-Rose Music, Inc.</i> , 510 U.S. 569 (1994) .....	69, 70
<i>CDN Inc. v. Kapes</i> , 197 F.3d 1256 (9th Cir. 1999) .....	37
<i>Computer Assocs., Inc. v. Altai, Inc.</i> , 982 F.2d 693 (2d Cir. 1992).....	37, 41, 44, 59, 60
<i>Elvis Presley Enters., Inc. v. Passport Video</i> , 349 F.3d 622 (9th Cir. 2003) .....	71
<i>Eng’s Dynamics, Inc. v. Structural Software, Inc.</i> , 26 F.3d 1335 (5th Cir. 1994) .....	32

*Ets-Hokin v. Skyy Spirits, Inc.*,  
 225 F.3d 1068 (9th Cir. 2000) ..... 35

*Feist Publ’ns, Inc. v. Rural Tel. Serv. Co.*,  
 499 U.S. 340 (1991) ..... 31, 36, 46

*Gates Rubber Co. v. Bando Chem. Indus., Ltd.*,  
 9 F.3d 823 (10th Cir. 1993) ..... 44

*Gen. Universal Sys., Inc. v. HAL, Inc.*,  
 379 F.3d 131 (5th Cir. 2004) ..... 41

*In re Google Inc.*,  
 462 F. App’x 975 (Fed. Cir. 2012) ..... xi

*Harper & Row Publishers, Inc. v. Nation Enters.*,  
 471 U.S. 539 (1985) ..... passim

*Hutchins v. Zoll Med. Corp.*,  
 492 F.3d 1377 (Fed. Cir. 2007) ..... 37, 55

*Johnson Controls Inc. v. Phoenix Control Syst., Inc.*,  
 886 F.2d 1173 (9th Cir.1989) ..... 45, 60

*JustMed, Inc. v. Byce*,  
 600 F.3d 1118 (9th Cir. 2009) ..... 41

*Key Publ’ns, Inc. v. Chinatown Today Publ’g Enters., Inc.*,  
 945 F.2d 509 (2d Cir. 1991)..... 36, 37, 47

*Kregos v. Associated Press*,  
 937 F.2d 700 (2d Cir. 1991)..... 37

*Lamps Plus, Inc. v. Seattle Lighting Fixture Co.*,  
 345 F.3d 1140 (9th Cir. 2003) ..... 46

*Leatherman Tool Gp., Inc. v. Cooper Indus., Inc.*,  
 199 F.3d 1009 (9th Cir. 1999) ..... 35

*Los Angeles News Serv. v. KCAL-TV Channel 9*,  
 108 F.3d 1119 (9th Cir. 1997) ..... 72



*Lotus Dev. Corp. v. Borland Int’l, Inc.*,  
 49 F.3d 807 (1st Cir. 1995).....62

*Lotus Dev. Corp. v. Borland Int’l, Inc.*,  
 516 U.S. 233 (1996) ..... 62, 63

*Mazer v. Stein*,  
 347 U.S. 201 (1954) ..... 5, 38, 67

*Mitel, Inc. v. Iqtel, Inc.*,  
 124 F.3d 1366 (10th Cir. 1997) ..... 61, 62, 63, 64

*Monge v. Maya Magazines, Inc.*,  
 688 F.3d 1164 (9th Cir. 2012) ..... 68, 76

*Nichols v. Universal Pictures Corp.*,  
 45 F.2d 119 (2d Cir. 1930).....44

*Oriental Art Printing, Inc. v. Goldstar Printing Corp.*,  
 175 F. Supp. 2d 542 (S.D.N.Y. 2001) .....37

*Peter Letterese & Assocs. v. World Inst. of Scientology Enters., Int’l*,  
 533 F.3d 1287, 1311 (11th Cir. 2008) ..... 71

*Practice Mgmt. Info. Corp. v. Am. Med. Ass’n*,  
 121 F.3d 516 (9th Cir. 1997) ..... 47, 53, 65

*Rogers v. Koons*,  
 960 F.2d 301 (2d Cir. 1992)..... 72

*Salinger v. Random House, Inc.*,  
 811 F.2d 90 (2d Cir. 1987).....55

*Satava v. Lowry*,  
 323 F.3d 805 (9th Cir. 2003) ..... 46, 49, 52

*Sega Enters. Ltd. v. Accolade, Inc.*,  
 977 F.2d 1510 (9th Cir. 1992) ..... 54, 64

*Sheldon v. Metro-Goldwyn Pictures Corp.*,  
 81 F.2d 49 (2d Cir. 1936).....42

*Sid & Marty Krofft Television Prods., Inc. v. McDonald’s Corp.*,  
562 F.2d 1157 (9th Cir. 1977) ..... 32

*Soc’y of the Holy Transfiguration Monastery, Inc. v. Gregory*,  
689 F.3d 29 (1st Cir. 2012)..... 56

*Softel, Inc. v. Dragon Med. & Sci. Commc’ns*,  
118 F.3d 955 (2d Cir. 1997)..... 51

*Sony Computer Entm’t, Inc. v. Connectix Corp.*,  
203 F.3d 596 (9th Cir. 2000) ..... 64

*Sparaco v. Lawler, Matusky & Skelly Eng’rs LLP*,  
303 F.3d 460 (2d Cir. 2002)..... 49

*Syrus v. Bennett*,  
455 F. App’x 806 (10th Cir. 2011)..... 56, 57

*Toro Co. v. R & R Prods. Co.*,  
787 F.2d 1208 (8th Cir. 1986) ..... 62

*Twin Peaks Prods., Inc. v. Publ’ns Int’l, Ltd.*,  
996 F.2d 1366 (2d Cir. 1993)..... 71, 72

*Wall Data Inc. v. Los Angeles County Sheriff’s Dep’t*,  
447 F.3d 769 (9th Cir. 2006) ..... 73

*Worldwide Church of God v. Phil. Church of God, Inc.*,  
227 F.3d 1110 (9th Cir. 2000) ..... 68

**FEDERAL CONSTITUTION, STATUTES, RULES AND REGULATIONS**

U.S. Const. article I, § 8..... 5, 67

17 U.S.C. § 101 ..... 37, 61

17 U.S.C. § 102 ..... passim

17 U.S.C. § 106 ..... 67, 75

17 U.S.C. § 107 ..... 69

17 U.S.C. § 302 ..... 52, 53, 64  
28 U.S.C. § 1295 .....6  
28 U.S.C. § 1331 .....5  
28 U.S.C. § 1338 .....5  
Fed. R. App. P. 4 .....6  
37 C.F.R. § 202.1.....53

**LEGISLATIVE HISTORY**

H.R. Rep. No. 94-1476 (1976),  
*reprinted in* 1976 U.S.C.C.A.N. 5659 ..... 59, 60

**MISCELLANEOUS**

Hon. Pierre Leval, *Toward a Fair Use Standard*,  
103 Harv. L. Rev. 1105, 1111 (1990) .....70  
Melville B. Nimmer & David Nimmer,  
Nimmer on Copyright (Matthew Bender, Rev. Ed.)..... 57, 62

## STATEMENT OF RELATED CASES

Defendant-Cross-Appellant Google, Inc., filed a writ of mandamus in this Court on November 8, 2011. Petition for a Writ of Mandamus, *In re Google Inc.*, 462 F. App'x 975 (Fed. Cir. 2012) (No. 2012-M106). The petition arose from the district court's denial of Google's assertion of privilege over an email that stated in relevant part: "What we've actually been asked to do (by [Google co-founders] Larry and Sergei [sic]) is to investigate what technical alternatives exist to Java for Android and Chrome. We've been over a bunch of these, and think they all suck. We conclude that we need to negotiate a license for Java under the terms we need." *Id.* at 976.

This Court (Judges Lourie, Prost, and Moore) agreed with the district court that the email was not privileged and denied the writ. *Id.* at 977-79.

No other appeal from this civil action was previously before this Court or any other appellate court. There is no case pending in this Court or any other court that will directly affect or be directly affected by the Court's decision here. There are no other cases related to this dispute.

## INTRODUCTION

Ann Droid wants to publish a bestseller. So she sits down with an advance copy of *Harry Potter and the Order of the Phoenix*—the fifth book—and proceeds to transcribe. She verbatim copies all the chapter titles—from Chapter 1 (“Dudley Demented”) to Chapter 38 (“The Second War Begins”). She copies verbatim the topic sentences of each paragraph, starting from the first (highly descriptive) one and continuing, in order, to the last, simple one (“Harry nodded.”). She then paraphrases the rest of each paragraph. She rushes the competing version to press before the original under the title: *Ann Droid’s Harry Potter 5.0*. The knockoff flies off the shelves.

J.K. Rowling sues for copyright infringement. Ann’s defenses: “But I wrote most of the words from scratch. Besides, this was fair use, because I copied only the portions necessary to tap into the Harry Potter fan base.”

Obviously, the defenses would fail.

Defendant Google Inc. has copied a blockbuster literary work just as surely, and as improperly, as Ann Droid—and has offered the same defenses. The work was authored by software developers at Sun

Microsystems, Inc., now Oracle America, Inc.<sup>1</sup> Sun’s developers spent years refining, writing, organizing, and promoting packages of computer source code to help outside application (“app”) programmers write new computer programs in the Java language faster and more efficiently by just incorporating the packages into their own Java programs. The packages were wildly popular, largely because they were written and organized in a way that made intuitive sense. A community of millions of application programmers coalesced around them. But everyone—IBM, Sony, Cisco, Red Hat, and others—understood that no one was allowed to use the packages without a license from Sun/Oracle.

Enter Google. Google was late to the smartphone market. Google’s top brass was desperate to develop its own mobile platform: Android. They concluded that the platform needed to include Sun’s popular Java packages. As one email to Google executives indelicately put it, the alternatives “all suck[ed].” A1168.

---

<sup>1</sup> In 2009, Oracle purchased Sun mainly for the value of the Java platform at issue here. A20,514, 20,689-90. Accordingly, this brief will occasionally refer to Sun/Oracle’s contributions and policies and, for simplicity, the argument section will refer to Oracle as if it had been the author throughout.

Google executives agreed that they “[m]ust take a license from Sun,” A1132, because “[S]un ... own[s] the brand and the IP,” A1200-01. But Google rejected the key terms that every other licensee accepted. Then, it just copied Sun’s work. Google admits that it literally copied into Android thousands of lines of original Sun/Oracle source code. The code Google copied embodied the elements and the detailed, complex structure and design of 37 packages of code, essentially the chapter and subchapter headings and the topic sentences from those packages—all copied verbatim. Google then paraphrased the rest.

The copying enabled Google to rush Android to market. It also made Android instantly familiar to the programming community Sun fostered. So millions of programmers who came of age on Sun’s Java source code, nomenclature, and organization eagerly wrote apps for Android. That helped make Android a phenomenal commercial success.

Had Google done exactly the same thing with a novel or a symphony, there would be no doubt that it copied protectable expression—both as to the large volume of work copied and the work’s organization. Copyright protects a short poem or even a Chinese menu or jingle. But the copied works here were vastly more original, creative,

and labor-intensive. Nevertheless, the district court stripped them of all copyright protection. The court saw this software as just different. It believed that each line of source code Google copied is an unprotectable “method of operation,” 17 U.S.C. § 102(b), because it is just a command to carry out a pre-assigned function.

This notion of software exceptionalism for any code is wrong. Congress chose to protect computer programs under copyright law. As this Court recognized in *Atari Games Corp. v. Nintendo of America, Inc.*, 975 F.2d 832 (1992), software code is protectable expression because authors select and arrange lines of source code in an original way. *Id.* at 838. Under *Atari*, copyright law recognizes no difference between original expression embodied in the topic sentences or organization of *Harry Potter* and original expression embodied in like features in software. This Court should reverse the district court’s basic legal error.

But this Court should go a step further and reject Google’s fair-use defense as a matter of law. A commercial competitor may not copy verbatim crucial features of another’s expression, depriving the original author of a potential market for the work. Google copied the source



code upon which programmers most rely, incorporated that code into a competing mobile platform, and competed directly with Oracle which was already profiting from licensing the packages for mobile devices. That is decidedly unfair.

If, as the Supreme Court explained, “the best way to advance public welfare” is to “encourage[]” authors to engage in “individual effort by” offering them “personal gain,” *Mazer v. Stein*, 347 U.S. 201, 219 (1954); see U.S. Const. art. I, § 8, the principle applies at least as much to software as to novels and Chinese menus. Oracle would never have revolutionized the computer world if it knew up front that a court would find that “the public w[as] and remain[s] free to” co-opt its work for financial gain. A165. J.K. Rowling does not write blockbusters for everyone to knock off. Neither will innovators like Oracle.

### **JURISDICTIONAL STATEMENT**

The district court had jurisdiction under 28 U.S.C. §§ 1331, 1338(a), and entered final judgment on June 20, 2012. A171-72. The parties moved for judgment as a matter of law or, alternatively, a new trial. A1077-111, 1112-18. The court denied the final post-trial motion on September 4, 2012. A1119. Oracle timely appealed on October 3,

2012, and Google cross-appealed. A1120-23; Fed. R. App. P. 4(a)(1)(A), 4(a)(4)(A)(i). Because this action included patent claims, *infra* at 29 n.3, this Court has jurisdiction under 28 U.S.C. § 1295(a)(1).

### **STATEMENT OF THE ISSUES**

1. Google copied thousands of lines of Oracle source code arranged in a manner that users find attractive. The district court found the code and the structure and organization it embodied original and creative.

Does the Copyright Act protect the expression that Google copied?

2. Google inserted the code it copied (and the corresponding structure) into a commercial product in a market where Oracle was already competing. Does Google's fair-use defense fail as a matter of law?

### **STATEMENT OF THE CASE**

Oracle sued Google in the U.S. District Court for the Northern District of California, alleging that Google's Android mobile operating system infringed Oracle's copyrights and patents. A336-47, 524-35. The jury found no patent infringement. A1069-70. But the jury found that Google infringed Oracle's copyright in the packages and a specific

computer routine (called “rangeCheck”). A41-43. The jury hung on Google’s fair-use defense. *Id.* Thereafter, the district court ruled that the infringed code and organization of the 37 packages were devoid of copyright protection. A163-70. It also denied Oracle’s motion for judgment as a matter of law on fair use. A129.

## STATEMENT OF FACTS

### **Sun Revolutionizes Computer Programming With Java And Its Packages Of Prewritten Programs**

For years, computer programmers had to pick one platform when writing new programs. A20,463, 20,529-31. Computer giants like Apple and Microsoft had developed their own versions of programming languages. Thus, “when you wrote a program for [a] Microsoft Windows computer, that program would not run on ... an Apple MacIntosh computer.” A20,463. “So if you want something that ran on Windows and ... Apple ... , you would have to write that program twice.” *Id.*; *accord* A20,529-31.

Sun developers changed all that with the Java platform. Released in 1996, a distinguishing feature of the Java platform is the “virtual

machine,” which allows programmers “to write programs that ... run on different types of computer hardware without having to rewrite them.”

A133. A programmer could now write a program once and it would run on any device with a Java virtual machine regardless of operating system. A20,549, 20,676-77. “Write once, run anywhere” became Java’s credo. A20,888-89, 20,463-64, 22,132.

Sun developers wrote a vast array of Java programs to perform often-needed functions and organized those programs into “packages.”

A134, 139. Each package is arranged in an intricate hierarchy (more on that below) and consists of numerous modules of “tried-and-true pre-packaged programs” comprising a vast arrangement of functions. A141. These packages were a huge benefit to programmers writing apps for all sorts of devices. Whatever the problem, the programmer does not have to “re-invent[] the wheel[]” to write a solution. *Id.* The parties and the district court often called these software packages “APIs.”

**TERMINOLOGY CLARIFICATION:** “*API*,” which stands for “application programming interface,” is confusing because it is a verbal chameleon. It can describe a trivial communication protocol to pass information between programs. Or it can describe sophisticated computer programs, like the ones Sun wrote. Even in this case, the parties and court confusingly applied the same label to describe both the entire set of Java packages, *see, e.g.*, A131-32, 134-36, and a single package of Java source code, *see, e.g.*, A131-32, 167-68. To avoid confusion, we refer more precisely to what we are describing: a “package” or “packages” of source code.

Also, for ease of reference, this brief uses the term “*developers*” for software engineers who write new code for the Java platform and its packages, and “*programmers*” for those who *use* the Java packages to write new apps.

By way of illustration, in one package, Sun developers wrote a program called “URLConnection” to establish an internet connection (e.g., to a bank or store website). A10,013-28; 20,753-54. As simple as that sounds, it is exceedingly complex. Developers needed special expertise to write the network protocol and cryptographic algorithms. Once they did their work, a programmer wishing to write a program that connects to, say, BankofAmerica.com, could either (1) reinvent the wheel, writing his own algorithms or (2) simply “declare”—i.e., type—“new URL(‘http://bankofamerica.com’).openConnection()” in the program, which calls on Sun’s prewritten code. If the programmer used

the feature frequently, he would not bother looking it up, since using the package would become second nature. A20,937-38. Apropos of the terminology clarification, `URLConnection` is a computer program, not some trivial communication protocol between programs.

Every package consists of two related types of Java source code. First, each component in a package begins with one or more lines of code including, among other things, a description of the function, such as “`public URLConnection openConnection() throws java.io.IOException.`” Also, this code reflects the component’s place in the package hierarchy. A133, 136-39. The similar code that the programmer declares in order to invoke the prewritten program is: “`URL(String spec).openConnection().`” These lines of code are called declarations, headers, signatures, or names (depending on various factors not relevant here). In the interest of brevity, we call them all “declaring code” or simply “declarations.” The second type of code tells the computer how to perform the declared function. A133, 137-39. In the illustration above, this second category encompasses the lines of code opening the internet connection. Consistent with the district court’s terminology, we call them “implementing code.” A163.

As is evident from this description, a programmer *does not have to* invoke these packages to write a program in the Java language, any more than an avid networker must use Hallmark’s prewritten greetings to write to friends on special occasions. A20,458-59. The packages are shortcuts. With only a few minor exceptions (portions of three packages), programmers can write in Java without using the packages. A140-41, 20,946-49. Indeed, programmers not satisfied with the existing packages can create their own packages in the Java language with similar—or completely different—functionality. A22,388-89. Sun specified how the code in the packages works in the *Java API Specification*, a massive-40,000 page manual that details the Java declaring code and its hierarchical arrangement. A20,710-11, 20,775, 21,422-24. The *Specification* also describes the structure and organization of each package, its elements and their relationships, and how each element and package works. A20,754-55, 20,710-102. It “exactly” mirrors the declaring code in the package, including its structure and organization. A20,775-77.

## **Writing A Java Package Is An Iterative And Creative Exercise**

The original Java Standard Edition Platform (“Java SE”) had “eight packages of pre-written programs.” A140. When Google began its copying in 2005, Java SE 5.0 had 166 packages. A141. It now contains 209. A20,766.

Writing any of these packages is an iterative and creative process. It took Sun’s “most senior[,] experienced and talented” developers years to write some of them. A20,459; *see* A20,791, 20,921. Much of the creativity lies in determining what to include in the packages and how to organize the declaring code in a way that programmers using the packages in their own apps will find appealing and intuitive. The process usually begins as a “high level exercise.” A20,790. Sun/Oracle developers identify areas of need in the Java programming community for new or different functions. A20,790, 21,410. Suggestions also come through the Java Executive Committee—Java’s governing body, composed of Oracle and its competitors like IBM and Google. A20,790-96, 20,471. Sun/Oracle developers then organize a “high level summary of ... a possible structure” for the package. A20,790-91; *see* A20,913. They wrestle with what functions to include in the package,



which to put in other packages, and which to omit entirely. *Id.* They send sketches “around to get comments from [their] colleagues,” and may “revise th[eir] design” based on “feedback.” A20,791. The developers work with a “clean slate,” A21,412; *accord* A20,913, so ex ante, their options are infinite.

Sun/Oracle invested hundreds of millions of dollars in these labors. A20,454, 20,557. Sun registered Java SE—including all the packages—with the Copyright Office. A1066, 2342-99, 2400-25, 2520-24, 5908-12; Supp. App’x Ex. 1076. It also invested millions in teaching a community of programmers how to use those packages. A20,557, 21,438-39. So, when Java programmers see “URL.openConnection(),” they instantly conjure “creating an internet connection,” just as surely as fans who see “Hermione Granger,” instantly think, “brainy, fearless Harry Potter sidekick.”

### **Sun Develops A Licensing Regime To Foster A Community And Ensure Compatibility**

Although Oracle owns the copyright on Java SE and the corresponding packages, Oracle encourages their use by others—both a vast community of programmers writing clever apps and businesses

developing proprietary and competing products. To accommodate all comers, Sun/Oracle offers three different licenses:

- (1) The General Public License (“GPL”) is free of charge, but subject to a strict—and legally binding—obligation: Licensees may use the packages (both declaring code and implementing code), but must “contribute back” the new work. It is called an “open source” license, not because it is open for all to use unconditionally, but because the licensee must expose his innovations publicly. A20,460, 20,537, 21,923-24.
- (2) The Specification License, unlike the GPL, does not permit the licensee to use the full Java source code. A20,469-70. Rather, the licensee can use only the *Specification*—which, as explained, recites the declaring code. So, a Specification Licensee may write packages using the familiar declaring code and organization of the Sun/Oracle packages but must write its own implementing code. A20,461-63.
- (3) The Commercial License is for businesses that want to use and customize the full Java code in their commercial products and keep their code secret. Oracle offers a Commercial License in return for royalties. A21,225-26, 21,242.

Both the Specification and Commercial Licenses require that licensees’ programs pass a series of tests that ensure compatibility with the Java platform. A21,242, 21,226, 22,230. This compatibility requirement enforces adherence to Java’s critical “write once, run anywhere” principle. A133, 167, 20,549.

Sun/Oracle's investments in user-friendly, efficient, and intuitive source-code packages, combined with flexible licensing options, attracted millions of programmers to Java's "write once, run anywhere" platform, A133, 167, 20,549, 20,688, and made Java "one of the world's most popular programming languages and platforms," A133.

Corporations like Sony, Cisco, and General Electric, all took Commercial Licenses for software packages, A20,550-51, and IBM, SAP, and Oracle (before purchasing Sun), all took Specification Licenses, A20,46-67—each maintaining compatibility with the Java platform.

Even before the smartphone market exploded, Sun was licensing and profiting from a derivative version of the Java platform for use on mobile devices, called Java Micro Edition ("Java ME"). A20,889. Oracle licensed Java ME for use on feature phones and smartphones. A20,468, 20,551-52, 21,273, 21,658, 22,097. "[J]ust about every smart phone carrier ... around the world" used Java ME, A22,237, including Research in Motion in Blackberries, Danger in Sidekicks, Cisco in several of its systems, and Nokia in Series 60 devices, A20,551-52, 21,273, 21,760, 22,097. The royalties from these licenses were "very lucrative" and "quite valuable." A22,237.

As is evident from those client lists, Sun/Oracle—a significant force in personal computers, servers, and web-based applications, A133, 141—was already a strong presence in mobile devices and poised to be a major force in smartphones, A22,237.

Until Google entered the picture.

### **Google Is Desperate To Include Certain Java Packages In The Android Platform**

The Google juggernaut rests on a grand bargain. A21,631. Google, famously, does not directly charge users. Instead, Google collects information about its users and makes money selling advertising targeted at them. Advertisers pay large sums for that targeting. A7898, 7902-04, 7916-18, 7922, 7979.

Since its founding in 1998, practically every penny of Google's enormous revenue was tied to personal computers. A7898. By 2005, however, Google faced a grave threat. Increasingly, consumers were searching the internet from mobile devices rather than personal computers. A7959. But Google's products were not optimized for mobile search. *Id.* Google was desperate to extend its market dominance to mobile search. As Google's 10-K reported: Without quick action, Google would "fail to capture significant share of an increasingly

important portion of the market for online services.” *Id.*; A20,657-58, 21,631, 22,018.

Google set out to “build[] an Open Source handset solution with built-in Google applications.” A2026. Its strategy was to work with wireless carriers and manufacturers to incorporate a Google mobile operating system into handset designs. A1128. So, in 2005, Google acquired Android, Inc., which was developing a mobile software platform. A134.

Google could have tried to develop its operating system from scratch as Apple, Microsoft, and so many other technology companies have. But there were two problems. First, this had to be done yesterday. Designing a new operating system from scratch with its own array of pre-written software packages would take years. Second, Google executives understood that Android would rise or fall on “build[ing] a community force around Google handset APIs and applications.” A1148; *see* A21,627-30. That was because consumers choose smartphones over conventional cell phones mainly for the apps. Google needed immediate access to a community of independent programmers to write for Android all the useful, quirky, and

entertaining apps that users craved, from Solitaire to mobile banking to social networking. A1148, 21,627-30. But a community of loyal supporters does not coalesce overnight.

The only way for Google to get the technical and community-building jobs done quickly was to hitch its wagon to the Java platform. A21,890. Google's internal documents detail why the prewritten Java packages were "[c]ritical" to Google's shortcut "strategy." A1187, 1200-01. If Google could "[l]everage Java for its existing base of [programmers]," it would not need to educate new programmers on a whole new body of declaring code. A2033. The familiar packages would expedite development of Android and its apps, *id.*; A2092, 21,627-28, 21,630-31, which, in turn, would "[d]ramatically accelerate[] [Google's] schedule," A1187; *see* 21,636.

### **Google Acknowledges It Needs A License But Wants To Defy "Write Once, Run Anywhere"**

Google was fully aware that no one could use Sun's software packages—or even just the declaring code that Specification Licensees use—without a license. Several former Sun engineers at Google were steeped in Sun's licensing regime and knew how much effort and creativity went into designing each package. They were aware of the

many companies that took Specification Licenses to use the declaring code and write their own implementing code. *See, e.g.*, A20,902-03, 20,906. Accordingly, Google never doubted it needed a license for Sun's packages. Andy Rubin, the head of Google's mobile efforts, conceded: Java "apis [sic] are copyrighted" and "I don't see how you can open [J]ava without [S]un, since they own the brand and the IP." A1200-01. Google "[m]ust take [a] license from Sun," the Android team insisted in a presentation to top Google executives. A1132. It was "critical." A1199. Even five years later, with litigation imminent, Google still knew it needed a license. Google cofounders ordered its engineers to investigate alternatives to using Sun's packages. A1168. A former Sun engineer, A21,009, tasked to investigate responded bluntly: The alternatives "all suck." A1168. "We need to negotiate a license for Java." *Id.*

In keeping with Google's understanding of the legalities, "[i]n late 2005, Google began discussing with Sun the possibility of ... a license." A135. At no point in the ensuing five years did Google so much as suggest that it did not need a license. Instead, Google proposed a "custom" deal, A2837, 21,779—a "co-development partnership" or

“license to use and to adapt the entire Java platform for mobile devices,” A135. The parties reached an impasse. *Id.* The sticking point was Google’s insistence on terms that were anathema to “write once, run anywhere” A2837, 21,779, terms that Sun/Oracle denied all other licensees. Google wanted to be the only company ever allowed to use the Java packages commercially without making its implementation compatible with the Java virtual machine and therefore interoperable with other Java programs. A20,561-63, 22,233-35.

This was a nonstarter for Oracle. A20,561-63. When the parties reached an impasse, Android’s chief advised Google executives: “If Sun doesn’t want to work with us, we have two options: 1) Abandon our work ... or 2) Do Java anyway and defend our decision, perhaps making enemies along the way.” A1166. Google elected option 2.

### **Google Copies Verbatim The Declaring Code In 37 Java Packages Into Android**

Google wrote some of its own packages for Android in the Java *language*. Nothing wrong with that. What is objectionable is Google’s copying of portions of Sun’s packages. Google cherry-picked “the good stuff from Java,” A5874—the 37 packages that it “believed Java [programmers] would want to find ... in the new Android system



callable by the same names as used in Java.” A135; *accord* A21,503, 21,957-59 (the significance of the stolen 37 packages is “huge”). As to those 37 packages, Google admits it copied verbatim virtually all of Sun’s declaring source code, thereby replicating the entire detailed structure of each package, and then paraphrased the implementing code. A136, 976-97, 22,771-73. Google did what other businesses that took a Specification License to do—but without the requisite compatibility. A20, 461-63.

***What Google copied.*** We illustrate pictorially what Google copied, below. Some technical jargon is necessary to appreciate the extent and significance of Google’s copying: Every Java package is arranged in a hierarchy and divided into related “classes” of defined source code files; classes contain numerous “methods”; each method performs a discrete function, A137-39, such as opening an internet connection, *supra* at 9-10.

Figure 1, below, represents a high-level schematic of a single package, java.io. A5730. Java.io generally manages system inputs and outputs. A10,029-33. Arrayed on the left are the numerous classes. Each relates to the overall input/output theme, ranging from

InputStreamReader (which translates data streams into human-readable text) to Writer (which enables devices to write streams of characters). *Id.* The classes are arranged in a hierarchy of their own. Classes can contain subclasses, which in turn can contain subclasses, and so on. Figure 1 indicates that hierarchy by indentation.

Google copied the declaring code in the java.io package, including the classes shown in Figure 1. Figure 1 does not, however, show all the methods—569 of them, distributed among approximately 50 classes.

*See* A1065. The InputStreamReader class, for example, has five methods, ranging from “read” (which reads a single character) to “ready” (which signals whether the stream is ready to read).

A10,034-38. Other classes in java.io encompass between three and 39 methods. Google copied verbatim declaring code for all those methods, too.

Figure 1. High-Level Schematic of a Package.

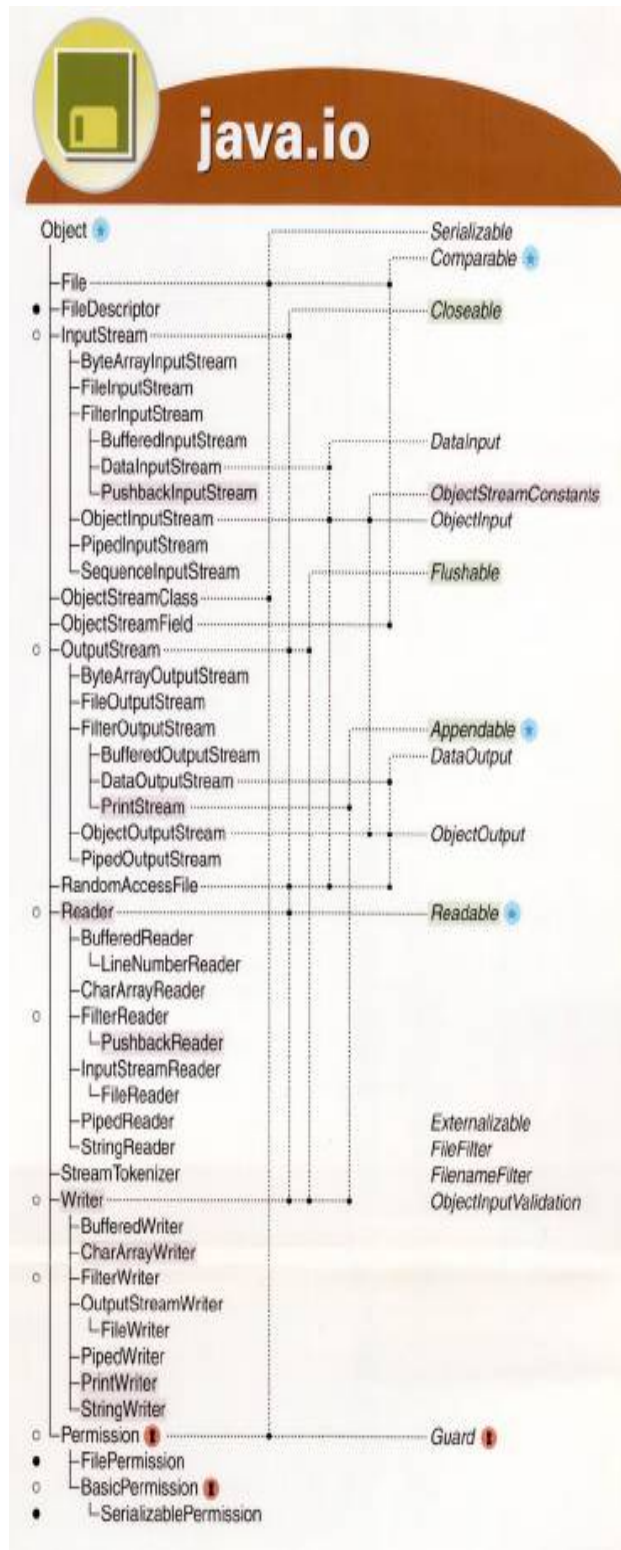
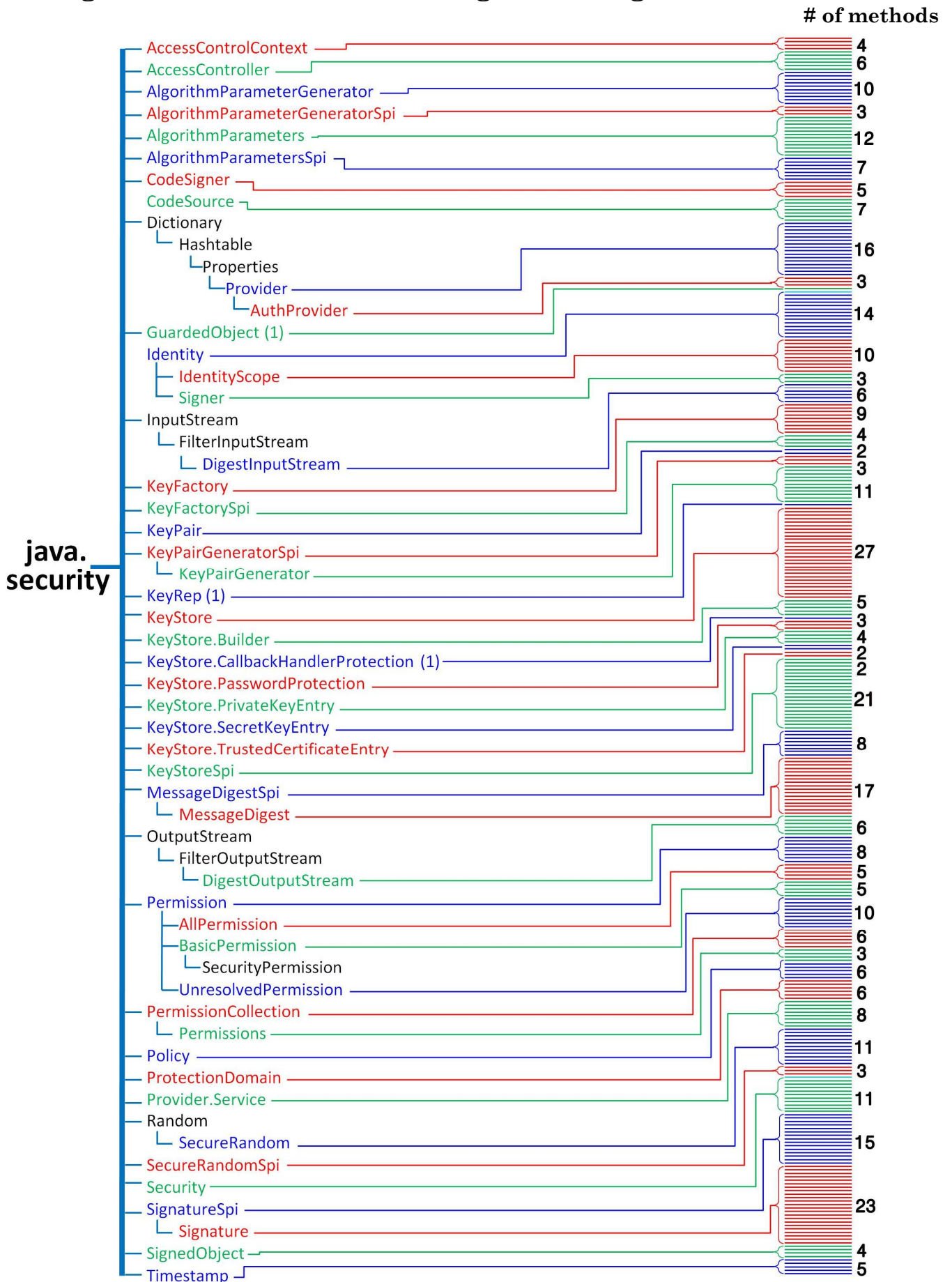


Figure 2, below, begins to portray the intricacy of the work Google copied at the method level. (We use a simpler package, `java.security`, because `java.io`'s 569 methods cannot fit on one page.) Arrayed on the left are the class names (with subclasses indented down to sub-sub-sub-subclass). On the right is a layout of the methods. Each of the almost imperceptibly thin lines represents a method—362 in all—each of which begins with one or more lines of declaring code. Google copied verbatim the declaring code for the `java.security` methods.

We say Figure 2 *begins* to portray the intricacy, because it leaves out important detail. The declaring code that defines a method or class includes more information than just its name and function. A137-38. Most include “parameters,” which define its operation (e.g., “5” directs `java.io.StringReader.skip` to skip the next five characters); “exceptions” (e.g., “Here’s the type of error you must be prepared to handle”); and “fields,” which hold data values (e.g., `pi`). A138, 140. Figures 1 and 2 do not convey this information.

**Figure 2. Schematic of a Package Including Methods**



Moreover, the structure of the packages is not simply linear or purely hierarchical. A true schematic would be three dimensional, with an intricate web of interconnections. Imagine each colored grouping of lines in Figure 2 as the floor plan of one story in a 50-story building. Imagine a web of chutes and ladders connecting some floors to others and even to other buildings, representing other packages. Those connections are “interfaces” (not to be confused with interfaces associated with “API,” *supra* at 9), which group classes sharing similar characteristics. Figure 1 indicates interfaces in italics on the right: the lines connecting one class to another, both in this package and in other packages. A5730 (illustrating the latter interfaces with an accompanying icon); *see* A139 (explaining the difference between classes and interfaces). There are many other types of relationships, some of which transcend package and class. A20,770, 21,391, 21,411-12.

So far, the description of Google’s copying (assisted by Figures 1 and 2) has focused on individual packages. Now multiply by 37. The 37

packages Google copied from Oracle contain 677 classes and 6508 methods—totaling *at least* 7000 lines of declaring code. A1065.<sup>2</sup>

Google admits that it copied all of this declaring source code verbatim—thousands of specific package, class, and method declarations; the definitions and parameters; and the exceptions. A136. Because declaring code identifies, specifies, and defines the components and their arrangement within the packages, when Google copied Java’s declaring code, it also copied the “sequence and organization” of the packages (i.e., the three-dimensional structure with all the chutes and ladders), A984-85, 22,367, 22,771-73. Google’s “Java guru,” A142, conceded (and the district court found, A140-41) that Google did not need to engage in this massive copying in order to design its own platform in the Java language. A20,946-49.

***What Google paraphrased.*** Once it had the declaring code, “do[ing] [the] implementation from scratch is a relatively easier job.”

---

<sup>2</sup> The district court estimated 7000, which, it believed, represented 3% of the lines of code in those 37 packages. *E.g.*, A136. We do not challenge those numbers on appeal but note that they are too low. The classes and methods, alone, number 7000, and declaring code for a single element can be several lines long. *See infra* at 57.

A22,405-06. Google merely had to “follow th[e] map” laid out in the 40,000-page *Specification* and “fill in the details” (i.e., implementing code). *Id.*

Google hired Noser—a foreign contractor that the Android Project director described as “super shady,” A2177, 21,161-62—to help write Android’s implementing code, A2101, 21,858. Google’s programmers admit that they and Noser pored over the *Specification* as they did their paraphrasing. A21,153-56; *see* A1724, 21,422-25, 21,925.

### **Google’s Copying Fragments The Java Community And Marginalizes Sun/Oracle In The Smartphone Market**

As planned, Google released its incompatible Java-based Android system for free. A135. And as expected, Google reaped billions in advertising revenue in connection with searches on mobile devices. A135, 5977. The move hampered Sun/Oracle’s “very lucrative revenue stream” that had attracted “just about every smart phone carrier” to Java. A22,237. As Oracle’s President put it, “It’s pretty hard to compete with free.” A22,498. For example, although Amazon had paid for a Java Commercial License for the Kindle, A20,468, 20,553, the new Amazon Kindle Fire, runs on Android—not Java, A21,192-93, 21,206.



Google’s copying caused the very fragmentation Sun/Oracle strove to prevent. Even while copying verbatim the declaring code from the 37 packages to attract loyal Java programmers, Google ultimately designed Android to be *incompatible* with the Java platform, so that apps written for one will not work on the other. *Infra* at 65-66; A2042-43, 21,503-04; *see* A21,192-93 (Kindle Fire not compatible). Google replaced “write once, run anywhere” with “write once, run only on Android.”

### **The Jury Finds Copyright Infringement, But The District Court Finds No Copyright Protection**

Oracle sued Google in 2010. Oracle alleged copyright infringement with respect to the 37 Java packages of source code. A526, 532-33.<sup>3</sup> The parties “agreed that the judge would decide ... copyrightability and Google’s equitable defenses and that the jury would decide infringement, fair use, and whether any copying was de minimis.” A131. The court reserved decision on copyrightability until after trial. *Id.*

---

<sup>3</sup> Oracle is not appealing from the jury’s verdict on patent infringement or copyright infringement by Google’s Android documentation.

“[T]he jury found that Google infringed” “as to the compilable code for the 37 Java API packages.” *Id.*; *accord* A41-43. But it deadlocked on fair use. A131. The jury and the court also found that Google infringed nine files, including one called “rangeCheck.” A132, 1058A-B, which is the subject of Google’s cross-appeal.

The district court declined to order a new trial on fair use, because it held that the code and structure Google copied were completely devoid of copyright protection. A170. As to declaring code, the court held that “no matter how creative or imaginative,” “there is only one way to write” it and thus “the merger doctrine bars anyone from claiming exclusive copyright ownership” of it. A164. “Therefore, there can be no copyright violation in using the identical declarations.” *Id.* It also held that declaring code consists of unprotectable “names and short phrases.” A165.

The court acknowledged that the “structure, sequence and organization” of each package was “creative” and “original.” A166. Nevertheless, it held the structure and organization unprotectable under 17 U.S.C. § 102(b), as “a command structure for a system or method of operation,” A166.

## SUMMARY OF ARGUMENT

**I. A.** Two axioms decide this case. **Axiom 1:** The Copyright Act’s threshold for copyright protection is very low. Any “creative spark” counts, “no matter how crude [or] humble.” *Feist Publ’ns, Inc. v. Rural Tel. Serv. Co.*, 499 U.S. 340, 345 (1991) (internal quotation marks omitted). **Axiom 2:** “[C]opyright protection extends to computer programs,” just as it does to any other literary work. *Atari Games Corp. v. Nintendo of Am., Inc.*, 975 F.2d 832, 838 (Fed. Cir. 1992). The district court’s approach of exempting some software from the standard copyright rules is an assault on both principles.

**B.** Applying these axioms, the software Google copied is protectable on two independent bases. First, declaring code is protectable because it is expressive—many orders of magnitude more expressive than necessary to overcome the threshold. Google confessed to literal copying—7000 times over. Under this Court’s *Atari* opinion, the declaring code that Google copied is as protectable as any other literary work because Oracle “exercised creativity in the selection and arrangement of its instruction lines.” 975 F.2d at 840.

Second, the original and creative structure and organization of each package is protectable. This is protected both through the declaring code that embodies it and, independently, through “its ‘nonliteral’ elements, such as the program architecture, ‘structure, sequence and organization’, [and] operational modules.” *Eng’g Dynamics, Inc. v. Structural Software, Inc.*, 26 F.3d 1335, 1341 (5th Cir. 1994) (as corrected). The package developers had infinite options for the structure and organization. They labored to create an organization for complex packages that programmers would find attractive—easy to learn and remember.

C. The district court erred in concluding that each individual line of declaring code is completely unprotected under the “merger” and “short phrases” doctrines. Merger holds that “[w]hen the ‘idea’ and its ‘expression’ are ... inseparable, copying the ‘expression’ will not be barred, since protecting the ‘expression’ in such circumstances would confer a monopoly of the ‘idea.’” *Sid & Marty Krofft Television Prods., Inc. v. McDonald’s Corp.*, 562 F.2d 1157, 1168 (9th Cir. 1977) (citation omitted). Merger cannot apply here because, as the court found, “the Android method and class names could have been different from the

names of their counterparts in Java and still have worked,” A132, and the selection and organization could have been different as well.

The “short phrases” regulation prevents an author from copyrighting “work” consisting of a naked bit of text. It does not wipe out the copyright on an assemblage of 7000 lines of code.

**D.** The district court also erred in holding that the organization of the packages is devoid of protection as “a command structure, a system or method of operation” under 17 U.S.C. § 102(b). A166. The district court was mistaken. Under § 102(b), copyright protection does not “extend to any idea, procedure, process, system, method of operation, concept, principle, or discovery.” That provision merely codifies the idea-expression dichotomy. The question under § 102(b) is whether “alternate expressions ... are available”—i.e., “a multitude of different ways to” perform a particular function. *Atari*, 975 F.2d at 840. The district court found: “[T]here were many ways to group the methods yet still duplicate the same range of functionality.” A133.

The court was wrong to hold that no line of declaring code “can[] receive copyright protection” because they are “commands to carry out pre-assigned functions.” A166-67. Since all software consists of

“commands to carry out pre-assigned functions,” that would mean that no software is protectable.

Equally mistaken was the court’s invocation of “[i]nteroperability.”

A167. Interoperability is irrelevant to copyrightability.

Copyrightability must be considered *ex ante*—from the perspective of the original developers of *Oracle’s* packages, considering what constrained *them*. What Google felt it needed to provide to programmers years later may bear on whether its *subsequent use* was fair, but not on copyrightability. In any event, Google’s copying was not about interoperability. Android is downright *incompatible* with the Java platform.

**II.** The district court should also have dismissed Google’s fair-use defense as a matter of law. Every factor cuts against fair use.

First, Google’s purpose was purely commercial. It was not transformative. The packages serve the identical purpose in Android as they do in Java. Second, the Java packages are the product of significant creative effort, years of labor, and hundreds of millions of dollars in research and development. Third, Google copied the *most important* portion of Oracle’s code—the parts that were both most

creative and most relevant to programmers writing programs. Fourth, Android competes directly with Java licensing business. Also, Android damaged Java by fragmenting the platform and undermining the central “write once, run anywhere” credo.

## STANDARD OF REVIEW

“To resolve issues of copyright law, this [C]ourt applies the law as interpreted by the regional circuits, in this case, ... the Ninth Circuit.” *Atari*, 975 F.2d at 837. Whether particular expression is protected by copyright law “is a mixed question of law and fact ... subject to de novo review.” *Ets-Hokin v. Skyy Spirits*, 225 F.3d 1068, 1073 (9th Cir. 2000). “[D]enial of a motion for judgment as a matter of law after a jury trial” is upheld only “if there is substantial evidence to support the verdict.” *Leatherman Tool Gp., Inc. v. Cooper Indus.*, 199 F.3d 1009, 1011 (9th Cir. 1999) (internal quotation marks omitted).

## ARGUMENT

### I. COPYRIGHT PROTECTS ORACLE’S SOFTWARE PACKAGES.

The Copyright Act sets a very low threshold for copyrightability, and the Act protects computer software as it does other “literary” work.

§ I.A. Applying these principles, what Google copied is creative and

protectable expression far exceeding this low threshold. § I.B. The district court's contrary holdings are dangerously erroneous. §§ I.C & I.D.

**A. The Copyright Act Sets A Low Threshold For Protection And Applies The Same Standard To Software As Other Protectable Works.**

There can be no dispute as to the axioms that control this case.

**Axiom 1:** The Copyright Act protects “original” expression.

17 U.S.C. § 102(a). A work is “original” if “it possesses at least some *minimal* degree of creativity.” *Feist*, 499 U.S. at 345 (emphasis added).

Oracle “obtained the benefit of a presumption” of copyrightability, *Atari*, 975 F.2d at 840, by registering the Java platform with the Copyright Office, A1066-68. But, even without the presumption, the packages are easily copyrightable, as “[t]he vast majority of works make the grade quite easily, as they possess some creative spark, *no matter how crude, humble or obvious it might be.*” *Feist*, 499 U.S. at 345 (quotation marks omitted; emphasis added).

Courts have no trouble finding the minimally sufficient “creative spark” in works that are “humble,” indeed. They include a Chinese yellow pages, *Key Publ'ns, Inc. v. Chinatown Today Publ'g Enters.*,



945 F.2d 509, 514 (2d Cir. 1991); estimates of coin values, *CDN Inc. v. Kapes*, 197 F.3d 1256, 1257-58, 1260-61 (9th Cir. 1999); and pitcher's statistics, *Kregos v. Associated Press*, 937 F.2d 700, 702, 704 (2d Cir. 1991). Even a Chinese menu. *Oriental Art Printing, Inc. v. Goldstar Printing Corp.*, 175 F. Supp. 2d 542, 548 (S.D.N.Y. 2001).

**Axiom 2:** As this Court explains, “copyright protection extends to computer programs,” just as it does to any other work. *Atari*, 975 F.2d at 838; *Computer Assocs., Inc. v. Altai, Inc.*, 982 F.2d 693, 701-03 (2d Cir. 1992). The Copyright Act protects as “[l]iterary works” “works ... expressed in words, numbers, or other verbal or numerical symbols or indicia.” 17 U.S.C. § 101. Computer programs meet that definition. *Atari*, 975 F.2d at 838. That is Ninth Circuit law (which, as in *Atari*, controls here). It is also the universal view of the circuits. *See, e.g., Hutchins v. Zoll Med. Corp.*, 492 F.3d 1377, 1385 (Fed. Cir. 2007); *Altai*, 982 F.2d at 702-03; *Apple Computer, Inc. v. Franklin Computer Corp.*, 714 F.2d 1240, 1249 (3d Cir. 1983).

The district court's decision was an assault on both axioms. The code Google copied indisputably exceeds by many orders of magnitude the minimal level of creativity necessary for copyright protection. Yet

the opinion is a manifesto of software exceptionalism—the notion that software (or perhaps some undefined category of software) deserves less copyright protection than any other work. The court’s premise was that software innovation is entitled to copyright protection *or* patent protection, never both. A144, 161-62, 164, 167. The Court preferred patent protection because “copyright exclusivity lasts 95 years.” A144. This either/or notion is, of course, incorrect. The Supreme Court has “h[e]ld that ... [n]either the Copyright Statute nor any other says that because a thing is patentable it may not be copyrighted.” *Mazer*, 347 U.S. at 217.

The district court countered with an idea from a law review article: “As software patents gain increasingly broad protection, whatever reasons there once were for broad copyright protection of computer programs disappear.” A162 (citation and internal quotation marks omitted). If this Court adopts the district court’s rationale, “copyright protection of computer programs” will indeed “disappear.”

**B. The Declaring Source Code And Organization Of Each Package Is Protectable Expression.**

Under these axioms, Oracle’s packages are protectable. There was nothing humble about the expression Google copied. It was the

programming equivalent of a magnum opus—an intensely creative endeavor involving thousands of subjective and intuitive, even artistic, judgments as to what sorts of elements, structures, and relationships a community of programmers would find intuitive, coherent, and aesthetically pleasing. As the chief architect of Oracle’s packages explained, his team strove for “something that we thought was coherent, would be easy to use and attractive to [programmers], but it could have ended up in many different ways and been just as good.” A20,766; *accord* A20,761, 21,949, 22,399. Even Google’s own “Java guru,” A142, admitted that designing the packages “is very much a creative process.” A20,917; *accord* A20,910-17, 20,920-22, 20,970.

Oracle’s chief architect also testified that the effort to reach that level of aesthetic appeal entailed “many, many design choices”—so many that “I wouldn’t know how to start counting them.” A20,798, 20,796-97; *see* A3003-49. The choices included:

[H]ow should classes be organized under other classes?  
How should interfaces be organized under other interfaces?  
How should classes and interfaces relate? Where should the methods be? What should the methods be named? What kinds of inputs do the methods take? What kind of outputs do the methods provide for the fields? How do ... they relate? Is the value in a field a color, or is it just a number, or is it a string, or is it something else?

A20,797-98. The structural options were infinite.

The district court acknowledged as much when it announced (albeit in something of an understatement): “Yes, it is creative. Yes, it is original. Yes, it resembles a taxonomy.” A166. Embedded in these findings are two distinct bases for protecting Oracle’s source code.

Either suffices: (1) the expressive declaring code; and (2) the creative arrangement of each package.

**1. Declaring source code is protectable because it is expressive.**

This is an uncommon copyright case. Usually, the accused infringer has created a work—whether a visual work, work of literature, or computer program—that bears *similarities* to the plaintiff’s work. And the court’s role is to determine whether the two are sufficiently similar to amount to plagiarism. Here, we have admitted plagiarism—at least 7000 times over—and the jury found infringement. A41-43, 136, 22,771-73.

The last time the Supreme Court addressed such a situation was *Harper & Row Publishers, Inc. v. Nation Enterprises*, 471 U.S. 539 (1985). *The Nation Magazine* obtained an advance copy of President Ford’s memoir. *Id.* at 543. “The Nation ... admitted to lifting verbatim

quotes ... totaling between 300 and 400 words,” *id.* at 548—probably less than 1% of the 655-page memoir, *id.* at 570. The Court took as granted that this “verbatim copying ... would constitute infringement unless excused as fair use.” *Id.* at 548.

The only way to reach a different conclusion here is to declare that different rules apply when the verbatim infringement is of source code. But, it is “well settled” that source code is protectable. *Altai*, 982 F.2d at 702; accord *JustMed, Inc. v. Byce*, 600 F.3d 1118, 1125 n.3 (9th Cir. 2009); *Gen. Universal Sys., Inc. v. HAL, Inc.*, 379 F.3d 131, 142 (5th Cir. 2004). In *Atari*, Nintendo’s video game program included the “10NES program,” a single program far less sophisticated than Oracle’s web of declaring code. That simple program merely made it impossible for a game to work without transmission of the correct coded message. 975 F.2d at 836. Atari obtained and copied Nintendo’s security source code. *Id.* at 836, 841. This Court held that the “literal expression” in Nintendo’s security program was “protectable.” *Id.* at 840. Relying on non-software-specific copyright authority, *Atari* reasoned that “Nintendo ... exercised creativity in the selection and arrangement of its instruction lines.” *Id.* at 840. Because there were “alternative” ways of

achieving the same result—a security program—there was “creativity in the [code’s] selection and arrangement.” *Id.*

If Nintendo’s relatively simple security code cleared the creativity threshold, then the declaring code here does so with ease. Oracle’s developers began, too, with a “clean slate” when they set out to write their original code—including the declaring code—for each package. A20,734, 20,788. Neither programming conventions nor the Java language dictated what the declaring code would be.

Google’s decision not to copy some code (implementing code) does not make the code it did copy (declaring code) unprotectable. Like the topic sentences in *Harry Potter*, the modest memoir excerpts in *Harper & Row*, and the 10NES in *Atari*, the various sentences of infringed code do not lose protection just because Google found other code insufficiently valuable to copy. “As Judge Learned Hand said, ‘No plagiarist can excuse the wrong by showing how much of his work he did not pirate.’” *Atari*, 975 F.2d at 845 (quoting *Sheldon v. Metro-Goldwyn Pictures Corp.*, 81 F.2d 49, 56 (2d Cir. 1936)).

Google’s effort to trivialize the proportion of the infringed code is especially misguided because what it copied—practically 100% of the

declaring code in 37 packages—was vastly more valuable to programmers than what Google paraphrased. Programmers do not know the implementing code. All they know, and all they need to know to program seamlessly, is the declaring code. Had Google selected different names for the thousands of packages, classes, and methods (e.g., “java.key” rather than “java.security”), the developer community Oracle fostered would not have been as immediately receptive to Android. A20,914-15. It would have been as unfamiliar, and possibly as unpopular, as an Ann Droid knock off that copied a *Harry Potter* plot, but substituted John Smith and Jane Doe for the protagonists and Duke Jones for the antagonist.

**2. The organization of each package is protectable as creative expression.**

The packages are independently protectable on two related bases, both arising from Google’s concession (and the district court’s explicit finding) “that the structure, sequence and organization of the 37 accused API packages in Android is substantially the same as the structure, sequence and organization of the corresponding 37 API packages in Java.” A985, 22,771-72. That structure, sequence and

organization would be protectable, even if Google had not copied a single line of code, but all the more so because it copied 7000.

***Copying of non-literal elements.*** If Ann Droid had paraphrased in the same order every chapter title and topic sentence without copying a single word verbatim, the entire plot that she copied—the structure, sequence, and organization of the overall work—would be protected. *See Nichols v. Universal Pictures Corp.*, 45 F.2d 119 (2d Cir. 1930) (collecting cases). This principle applies equally to software. As with a novel, “[i]t is well-established ... that *non-literal* structures of computer programs are protected by copyright.” *Altai*, 982 F.2d at 702 (emphasis added); *accord Gates Rubber Co. v. Bando Chem. Indus., Ltd.*, 9 F.3d 823, 836 (10th Cir. 1993). The reason is that Oracle developers, like novelists, make creative organizational choices, such as which methods go where.

In fact, many precedents involving accusations of copied software do not involve verbatim copying of *any* literal code elements. *See, e.g., Altai*, 982 F.2d at 696, 702. In *Atari*, this Court addressed infringement “[e]ven in the absence of verbatim copying.” *Id.* at 844. Atari copied the “unique” and “creative organization and sequencing” of Nintendo’s



security function. 975 F.2d at 840. This Court determined that Nintendo’s developers had “a multitude of different” organizations available to perform the needed function, so they “exercised creativity” in their selection. *Id.* Thus, this Court held that, “[a]t a minimum,” Nintendo could “copyright the unique and creative *arrangement* of” its work. *Id.* (emphasis added). The Ninth Circuit reached the same conclusion, on the same logic, in *Johnson Controls Inc. v. Phoenix Control Systems, Inc.*, 886 F.2d 1173 (1989). It held—without regard to the program’s actual source code—that “the nonliteral components of a program, including the structure, sequence and organization ... may be protected by copyright where they constitute expression.” *Id.* at 1175-76. The key was that the developers had “some discretion and opportunity for creativity ... in the structure.” *Id.* at 1176.

Even if Google had not copied a single line of code, the organization of each package would be protected expression.

***Verbatim copying of literal elements.*** Of course, Google *did* verbatim copy thousands of lines of source code, which embody the structure of each package, just as the chapter titles and topic sentences represent the structure of a novel. The verbatim copying of literal

elements encapsulating the overall structure, sequence and organization makes this an even more compelling case for copyrightability.

This is true even if not a single code element, by itself, were copyrightable. As the district court understood, “a combination of unprotectable elements is eligible for copyright protection ... if those elements are numerous enough and their selection and arrangement original enough that their combination constitutes an original work of authorship.” A35 (quoting *Lamps Plus, Inc. v. Seattle Lighting Fixture Co.*, 345 F.3d 1140, 1147 (9th Cir. 2003)). That principle comes directly from Supreme Court precedent: Even if elements of a work “contain[] absolutely no protectable written expression,” the original “selection or arrangement” of those elements are protected so long as they “entail a minimal degree of creativity.” *Feist*, 499 U.S. at 348; see *Satava v. Lowry*, 323 F.3d 805, 811 (9th Cir. 2003) (“It is true, of course, that a *combination* of [even] unprotectable elements may qualify for copyright protection.”).

Applying this principle, a Chinese Yellow Pages is copyrightable because it arranged the entries under ordinary Yellow Pages categories

and categories of “interest” that are “not common to yellow pages, e.g., ‘BEAN CURD AND BEAN SPROUT SHOPS.’” *Key Publ’ns*, 945 F.2d at 514.

The Seventh Circuit likewise found copyright protection for a directory of dental procedures, in which the procedures were assigned numbers, classified into groups, and described in long and short form. *Am. Dental Ass’n v. Delta Dental Plans Ass’n*, 126 F.3d 977, 977-78 (7th Cir. 1997).

Judge Easterbrook’s opinion explained that the directory was copyrightable as a creative “taxonomy,” because “[c]lassification is a creative endeavor” and “each scheme of classification [in the directory] could be expressed in multiple ways.” *Id.* at 978-81. The Ninth Circuit reached the same conclusion for a taxonomy of medical procedures.

*Practice Mgmt. Info. Corp. v. Am. Med. Ass’n*, 121 F.3d 516, 517-20 (9th Cir. 1997) (as amended).

\* \* \*

If the works in *Atari* and *Johnson Controls* (copying of non-literal elements) and *Practice Management*, *American Dental*, and *Key Publications* (copying of literal elements embodying an arrangement) were protectable, the same is true many times over for Oracle’s works.

**C. The District Court Erred In Concluding That Each Line Of Declaring Code Is Completely Unprotected.**

The district court surveyed the caselaw generally at length, but its section entitled “application of controlling law to controlling facts” was just a few pages. A163-70. The court first discussed the verbatim copying of lines of code (addressed in this section) and then turned to the structure and organization of each package (addressed below, § I.D).

As to the copyrightability of declaring code, the district court “h[e]ld that, under the Copyright Act, no matter how creative or imaginative a Java method specification [i.e., declaring code] may be, the entire world is entitled to use the same method specification (inputs, outputs, and parameters).” A164. The court packed its rationale into 22 lines. A164-65. It held that the “merger” and “short phrases” doctrines barred copyright protection. Each holding is erroneous.

**1. The district court misapplied merger.**

The merger doctrine represents an application of *Baker v. Selden*, 101 U.S. 99 (1880): The Copyright Act grants no protection for “the author’s generalized ideas and concepts,” only for the author’s expression (i.e., the “more precisely detailed realization of those ideas”).

*Sparaco v. Lawler, Matusky & Skelly Eng'rs LLP*, 303 F.3d 460, 468 (2d Cir. 2002). As the district court acknowledged, A164, the merger doctrine holds that “courts will not protect a copyrighted work from infringement if the idea underlying the copyrighted work can be expressed in only one way.” *Satava*, 323 F.3d at 812. When that happens, the idea and the expression merge.

*Atari* illustrates the limits of merger for software. Nintendo could not prevent anyone from writing a security program, but since there were many ways to achieve the same security function, Nintendo *could* copyright its “creative[e] ... selection and arrangement” of “arbitrary programming instructions” to unlock the console. 975 F.2d at 840. Nintendo’s *specific choice of code* was protectable: It did “not merge with the process,” because “alternative expressions [we]re available.” *Id.*

By the same token, merger cannot bar copyright protection for any single line of declaring code—much less for all 7000—unless the original authors had available to them “only one way” to write them. *Satava*, 323 F.3d at 812 n.5. But the authors had many options as to

each individual line and unlimited options as to the selection and arrangement of the 7000 lines Google copied.

As to the individual lines, take the district court's go-to "simple" snippet: "java.lang.Math.max." A139. That name was not preordained. A20,970. The developers could have called it "Math.maximum," "Equations.compare," "Arith.bigger," "MeasuringStick," etc. The computer would have accepted "Rumpelstiltskin." A20,788. The district court observed that "the Android method and class names could have been different from the names of their counterparts in Java and still have worked." A132; *accord* A133, 140-41. That necessarily means that the idea and the expression did not merge.

Even more fundamentally, the court erred in ending its merger analysis at the individual line of declaring code without taking stock of the larger body of declaring code of which it is a part. In so doing, the court made the mistake that so many appellate courts warn against: dissecting a work down to the most minute level of abstraction. That will "result in almost nothing being copyrightable because original works broken down into their composite parts would usually be little more than basic unprotectable elements like letters, colors and

symbols.” *Boisson v. Banian, Ltd.*, 273 F.3d 262, 272 (2d Cir. 2001) (internal quotation marks omitted); see *Softel, Inc. v. Dragon Med. & Sci. Commc’ns*, 118 F.3d 955, 963 (2d Cir. 1997). The court must not miss the forest for the leaves.

The conclusion that merger does not defeat the copyrightability of any particular line of declaring code applies with even greater force to the thousands of methods and the overall structure described by the large body of declaring code. Here, too, Oracle had an infinite number of choices for which methods to include and how to arrange them. Different software platforms handle the same problems in very distinct ways, as demonstrated by alternative Java packages with quite different designs, written by other developers. A21,412-16. As the district court recognized, granting the original authors copyright protection would not prevent Android from providing even the exact same functions with different organizations:

This could have been done by re-arranging the various methods under different groupings among the various classes and packages (even if the same names had been used). In this sense, there were *many* ways to group the methods *yet still duplicate the same range of functionality*.

A132-33 (emphasis added); *accord* A165-66. Indeed, for over 100 packages in Android, Google wrote its own declaring code with its own internal structure and organization. But not as to each of the 37 packages at issue. A136. Because “alternative expressions are available,” merger does not apply. *Atari*, 975 F.2d at 840; *Satava*, 323 F.3d at 812.

The district court also misunderstood another basic point. Whether the focus is on the individual line of declaring code (where the court focused) or the entire body of declaring code, the proper focus in copyright is on the options available to the *original author*. It must be. After all, “[c]opyright in a work ... subsists from its creation and[] ... endures for [the copyright] term.” 17 U.S.C. § 302(a). Elevating “Java rules” over copyright law, the district court observed that “[t]o carry out any given function, the method specification as set forth in the declaration *must be identical*.” A164. To be sure, once the *original author* chooses “java.lang.Math.max,” programmers who want to use Oracle’s pre-packaged software have to call it by that name. And once J.K. Rowling turned her work into a blockbuster, a plagiarist who wants to tap into that Harry Potter fan base must copy her work. But



that does not mean that the original work never had copyright protection. What a plagiarist feels it must copy to benefit from the original work is irrelevant, because copyright “subsists from ... creation and ... endures.” 17 U.S.C. § 302(a); *see PMI*, 121 F.3d at 520 n.8 (rejecting theory that a work loses copyright protection when use became pervasive due to government agency requirement).

**2. The district court misapplied “short phrases.”**

The district court also disqualified every single line of declaring code in an isolated and unexplained sentence: “[N]or can there be any copyright violation due to the *name* given to the method ... , for under the law, names and short phrases cannot be copyrighted.” A165. The court invoked a Copyright Office regulation that lists “works not subject to copyright,” including: “[w]ords and short phrases such as names, titles, and slogans; familiar symbols or designs; [and] mere variations of typographic ornamentation, lettering or coloring.” 37 C.F.R. § 202.1(a).

The regulation prevents an author from copyrighting a “work” consisting of a naked bit of text. That is what trademark protection is for. The regulation continues: “The Copyright Office cannot register claims to exclusive rights in brief combinations of words.” *Id.* So,

Nicholas Sparks cannot register the title *Safe Haven*, thereby precluding anyone from ever using that phrase. A videogame manufacturer may not be able to copyright a naked line of code, “consist[ing] merely of 20 bytes of initialization code plus the letters S-E-G-A,” that “is of such de minimis length that it is probably unprotected under the words and short phrases doctrine.” *Sega Enters. Ltd. v. Accolade, Inc.*, 977 F.2d 1510, 1524 n.7 (9th Cir. 1992) (as amended) (cited by A144).

In contrast, “short phrases” does not eliminate all copyright protection for an assemblage of 7000 lines of code, which Google itself conceded was not de minimis. A976-97; accord A22,777. Imagine our hypothetical plagiarist decided to write a Broadway hit, *Ann Droid’s Glengarry Glen Ross*. She transcribes every short sentence (which is practically all Mamet ever writes) and paraphrases the rest. No one would say that what she copied was unprotected because each isolated sentence was a short phrase. That is because the “work” she stole was not isolated phrases, but many, many phrases carefully authored and assembled in a specific order and melded together into a coherent whole. “Copyright protection does not protect individual words and

‘fragmentary’ phrases when removed from their form of presentation and compilation,” but short phrases are “subject to copyright in the form in which [they are] presented.” *Hutchins v. Zoll Med. Corp.*, 492 F.3d 1377, 1385 (Fed. Cir. 2007); accord *Salinger v. Random House, Inc.*, 811 F.2d 90, 98 (2d Cir. 1987).

The Eighth Circuit illustrated the point in a case involving the copyrightability of a personality test with 550 concededly “short, simple, declarative sentences,” such as “I am a good mixer” and “No one seems to understand me.” *Applied Innovations, Inc. v. Regents of Univ. of Minn.*, 876 F.2d 626, 634-35 (8th Cir. 1989). The court had no trouble finding the questionnaire protectable.

Here, again, the district court reached the wrong conclusion because it dissected the work too minutely—fixating on the individual line of code rather than the larger arrangement of which it was a part. *Supra* at 50-51. Moreover, applying “short phrases” as the district court did would invalidate practically any computer program. Virtually every line of the typical program is a short phrase. If each were unprotectable without regard to its relation to the larger whole, there would be nothing left of the program. And nothing left of *Atari*.

Even if it were appropriate to apply this principle to isolated sentences in a larger work, that could not justify stripping every single line of Oracle's declaring code of protection. The First Circuit explained (in an opinion Justice (Ret.) Souter joined): Copyrightability "turns on the specific short phrases at issue, as not all short phrases will automatically be deemed uncopyrightable." *Soc'y of the Holy Transfiguration Monastery, Inc. v. Gregory*, 689 F.3d 29, 50-52 (1st Cir. 2012) (collecting authorities). Take, for example, the declaring code for a method in the java.security.cert package:

```
public abstract void verify (PublicKey key, String sigProvider)
    throws CertificateException, NoSuchAlgorithmException,
    InvalidKeyException, NoSuchProviderException
    SignatureException
```

A10,042. Or the declaring code for a class in java.nio.channels:

```
public abstract class DatagramChannel
    extends AbstractSelectableChannel
    implements ByteChannel, ScatteringByteChannel, GatheringByte
    Channel {
```

A10,044.

Even shorter declaring code in isolation is not necessarily unprotected, for "a short phrase may command copyright protection if it exhibits sufficient creativity." *Syrus v. Bennett*, 455 F. App'x 806, 809

(10th Cir. 2011) (quoting 1-2 Nimmer on Copyright § 2.01[B] at 2-17). Google’s “Java guru,” A142, admitted: There can be “creativity and artistry even in a single method declaration.” A20,970. Whatever might be said of “math.max,” the declaring code above—and even “java.beans,” A1065—are not devoid of creativity.

**D. The District Court Erred In Holding That The Organization Is Devoid Of Protection.**

The district court next turned to the independent basis for copyrightability—structure and organization. Its “main answer” was less than half a page. A166. The court acknowledged that “thousands of commands arranged in a creative taxonomy” can still be creative and original. *Id.* But it held that the structure and organization—no matter how “creative” and “original” or how much “it resembles a taxonomy”—was not protectable on the ground that the commands are a “command structure, a system or method of operation” under 17 U.S.C. § 102(b).

That analysis is wrong for reasons we address below. But as important, it is also not dispositive because it addresses only half the argument. As is explained above, there are two related reasons why the structure and organization are copyrightable: (1) the copying of non-

literal elements of the structure; and (2) verbatim copying of literal elements of the structure. *Supra* at 43-47. The court’s focus on the “commands” that “carry out pre-assigned functions,” A166-67, has no bearing on the first rationale, which has nothing to do with specific “commands.” Like paraphrasing Harry Potter without copying a single word, the non-literal claim relates only to the organization that enables a programmer to figure out where to find particular methods, without regard to what names are assigned. The district court gave *no reason* to deny copyright protection to that organization. The method-of-operation point is irrelevant to that theory of protection. After all, Oracle’s developers could have “put all of the classes [of the platform] into one giant package”—devoid of all structure—since the virtual machine does not care where the code resides. A20,788.

The irony of the court’s approach is that it found the entire assemblage devoid of protection *because of* copying of literal elements; i.e., because the “commands” or declaring code were not, in its view, protected. But, even if Google wrote different declaring code for every method and class, it would still have copied a creative organization that is protected. It is enough that Google organized the Android package

elements to map directly onto Oracle’s organization. *Altai*, 982 F.2d at 702. For that reason, alone, the judgment must be reversed.

In any event, even if the focus were just on structure and organization of specific commands, the court’s “method of operation” analysis was wrong—devastatingly so, for the software industry. So, too, was the “interoperability” concept it invoked to support its analysis.

**1. The district court erred in dismissing the organization as an unprotected “method of operation.”**

Copyright does not “extend to any idea, procedure, process, system, method of operation, concept, principle, or discovery.” 17 U.S.C. § 102(b). Section 102(b) codifies the traditional idea/expression dichotomy and the merger doctrine that flows from it. *Apple Computer, Inc. v. Microsoft Corp.*, 35 F.3d 1435, 1443 & n.11 (9th Cir. 1994). The § 102(b) argument fails for the same reason merger fails. *Supra* at 48-53.

Regarding software in particular, courts routinely invoke Congress’s explanation that § “102(b) is intended ... to make clear that the *expression* adopted by the programmer is the copyrightable element in a computer program,” but “that the actual processes or methods

embodied in the program are not.” H.R. Rep. No. 94-1476 (1976) (emphasis added) (quoted in *Altai*, 982 F.2d at 703, and *Apple Computer*, 714 F.2d at 1252-53).

This Court’s *Atari* opinion applied this consensus. It described merger and “method of operation” in the same breath and then conducted the inquiries together: “This court, in applying Ninth Circuit law, must determine whether each component of the 10NES program ‘qualifies as an expression of an idea, or an idea itself.’” 975 F.2d at 839-40 (citing *Johnson Controls*, 886 F.2d at 1175). The security code was not a “method of operation” for the same reason that it did not run afoul of the merger doctrine: there were “alternate expressions ... available.” *Id.* at 840.

So, too, here. Since, as we demonstrate above, the original authors of Oracle’s packages had limitless “alternate expressions ... available” for selecting and arranging the various programs in each package, the idea for each package does not merge into the expression, and § 102 is not implicated. *Supra* at 51-53.

Instead of following this Court’s and the Ninth Circuit’s definition of a “method of operation,” the district court substituted its own



definition. It reasoned that not a single line of declaring code “can[] receive copyright protection” because they are “commands to carry out pre-assigned functions.” A166-67. That is wrong. That definition is essentially the same as the Copyright Act’s definition of protectable “computer program”: “a set of statements or instructions to be used directly or indirectly in a computer *in order to bring about a certain result.*” 17 U.S.C. § 101 (emphasis added). If a computer program is unprotectable simply because it “carr[ies] out pre-assigned functions,” no computer program is protectable.

The software cases the district court discussed disprove its rationale. Those cases recognize the copyrightability of source code, even though it consists of “commands to carry out pre-assigned functions.” The Seventh Circuit, for example, rejected the district court’s theory, reasoning that “word-processing software” does not become an unprotectable “‘system’ just because it has a command structure for producing paragraphs.” *Am. Dental*, 126 F.3d at 980. The Tenth Circuit agreed: “[A]lthough an element of a work may be characterized as a method of operation, that element may nevertheless contain expression that is eligible for copyright protection.” *Mitel, Inc.*

*v. Iqtel, Inc.*, 124 F.3d 1366, 1372 (1997); see *Toro Co. v. R & R Prods. Co.*, 787 F.2d 1208, 1212 (8th Cir. 1986) (that a work is a “system” under § 102(b) does not preclude copyright protection for the “particular expression” of the system); 1-2 Nimmer on Copyright § 2.03[D].

The district court was evidently influenced by a line in *Lotus Development Corp. v. Borland International, Inc.*, 49 F.3d 807, 810 (1st Cir. 1995), *aff'd by equally divided court*, 516 U.S. 233 (1996), that has no bearing here. There, the defendant copied a relatively simple menu system and interface but “did not copy any ... underlying computer code.” *Id.* at 810. *Lotus*, therefore, does not address and cannot apply to verbatim copying of source code.

To be sure, *Lotus* defines “method of operation” very expansively: as “the means by which a person operates something.” *Id.* at 815. But that definition cannot be applied beyond the facts of that case, because it would strip all computer programs of copyright protection despite Congress’s decision to protect software.

No other circuit adopts the *Lotus* formulation and no opinion applies it to a case like this. At least one circuit explicitly “decline[d] to adopt the *Lotus* court’s approach to section 102(b).” *Mitel*, 124 F.3d at

1372. The breadth of the definition was undoubtedly part of the reason the Supreme Court granted certiorari in *Lotus*. That the case split the Court 4-4 confirms that the holding is highly in doubt and the definition on which it was based almost certainly doomed. 516 U.S. 233 (1996).

**2. The district court erred in invoking interoperability in support of “method of operation.”**

Equally mistaken was the court’s view that “[i]nteroperability sheds further light on the character of the command structure as a system or method of operation.” A167. The court adopted Google’s argument that declaring code in Oracle’s 37 packages lost all copyright protection—years after they were written—because they became wildly popular and Google wanted the Java community of programmers to flock to Android. It is like Ann Droid saying: “No one would buy my book unless I copied the key parts of a wildly popular series. Ergo, that material must not have any copyright protection.” Google’s argument is wrong on the law and the record.

Doctrinally, interoperability is irrelevant to whether the packages are copyrightable. The court’s interoperability rationale suffers from the same focus on the wrong author as its merger analysis. *Supra* at

52-53. Here, again, the copyrightability analysis must be conducted *ex ante*—from the perspective of the *original* authors of *Oracle's* packages, and what constrained *them*. Oracle's developers wanted a bestseller. For the reasons already explained, Oracle's developers were not in the least bit constrained by an imperative to cater to some competitor who might come along years later and want to copy all of their declaring code without a promise of compatibility.

If interoperability is relevant at all, it would be only to fair use (discussed *infra* at 68-77), as in the cases the district court invoked. A152-61, 168 (discussing *Sony Computer Entm't, Inc. v. Connectix Corp.*, 203 F.3d 596, 602-08 (9th Cir. 2000); *Sega*, 977 F.2d at 1522-23). But, in any event, for the reasons explained above, an interoperability defense has no bearing on whether the work was protectable—and specifically, on whether it was a method of operation. *See* 17 U.S.C. § 302(a).

*Apple Computer* rejected Google's exact argument. There, the defendant justified its infringement because there were only "a limited number of ways to arrange operating systems to enable a computer to run ... Apple-compatible software." 714 F.2d at 1253 (quotation marks

omitted). The court held: “[Defendant] may wish to achieve total compatibility with independently developed application programs[,] ... but that is a commercial and competitive objective which does not enter into the somewhat metaphysical issue of whether particular ideas and expressions have merged.” *Id.*

The Ninth Circuit agrees. In *PMI*, it rejected the argument that everyone could copy the plaintiff’s medical coding system because the system had become the “industry standard.” 121 F.3d at 520 n.8. Like Google here, competitors remained free to “develop comparative or better coding ... and lobby” for their adoption. *Id.* Copyright “prevents wholesale copying of an existing system.” *Id.*

Moreover, Google’s copying was not about interoperability. Interoperability means that programs written for Android would run on the Java platform and vice versa. Google wanted the opposite: to copy enough code to make Android familiar enough to “[l]everage” the millions of Java programmers who knew Oracle’s packages, A2033, but not copy all the code that would be required for interoperability. Accordingly, many programmers now write *Android-only* programs that do *not* work on the Java platform. In so doing, Google made Android

unusable for many of the “millions of lines of code [that] had been written in Java before Android arrived.” A167; *see* A22,397-98, 22,463.

Don’t take our word for it. Google’s internal “Android Press Q&A” answering the most important questions upon Android’s release said it all:

Q48. Does Android support existing Java apps?

A. No.

Q49. Is Android Java compatible?

A. No.

A2205. That was not some mistake by a benighted PR department. If one person at Google knew the definitive answer to the question, it would have been its Technical Program Manager for Android Compatibility. A21,172. He testified that “Android does not support Java applications” and “is not Java compatible.” A21,179; *see also* A21,503-04 (“[Y]ou don’t really have compatibility. You can’t ship code from one platform to another.”).

\* \* \*

The district court worried that “[t]o accept Oracle’s claim would be to allow anyone to copyright one version of code to carry out a system of

commands and thereby bar all others from writing their own different versions to carry out all or part of the same commands.” A170. Not so. The issue is not Google’s different implementing code. Rather, it is Google’s violation of the Copyright Act by co-opting thousands of the exact lines of code Oracle wrote and the exact creative organization Oracle designed that made its work so popular.

Congress, the courts, and the Founders determined that “the best way to advance public welfare” is to “encourage[]” the authors of such works to engage in “individual effort by” offering them “personal gain.” *Mazer*, 347 U.S. at 219; *accord* U.S. Const. art. I, § 8; 17 U.S.C. § 106. That wisdom applies as much to Oracle’s declaring code as to all other literary works. Oracle invested years and hundreds of millions of dollars to author software packages that it would license others. The surest way to guarantee that no company ever makes an investment like that again is to declare that “the public w[as] and remain[s] free to ... us[e] exactly the same [declaring code] ... and organiz[ation].” A165.

## II. GOOGLE CANNOT ESTABLISH THAT ITS COMMERCIALY MOTIVATED AND ILLICIT VERBATIM COPYING IS FAIR USE

This Court should not stop at finding that Google infringed Oracle's copyrighted work. A remand to decide fair use is pointless. This Court should rule, as a matter of law, that Google's commercial use of Oracle's work in a market where Oracle already competed was not fair use.

Fair use is "an affirmative defense"—a limited exception to the copyright holder's exclusive rights, where Google bears the burden of proof. *Harper & Row*, 471 U.S. at 561. The district court should have declared, as a matter of law, that Google's copying was not fair use, A129, as the Supreme Court and Ninth Circuit frequently have done. *Harper & Row*, 471 U.S. at 561; see *Monge v. Maya Magazines, Inc.*, 688 F.3d 1164, 1184 (9th Cir. 2012); *Worldwide Church of God v. Phil. Church of God, Inc.*, 227 F.3d 1110, 1121 (9th Cir. 2000).

The relevant facts are discussed at length above, so we focus on the law and reference (but do not repeat) prior factual discussions.

Congress framed the "fair use" inquiry around four nonexclusive factors:



(1) the purpose and character of the use, including whether such use is of a commercial nature or is for nonprofit educational purposes;

(2) the nature of the copyrighted work;

(3) the amount and substantiality of the portion used in relation to the copyrighted work as a whole; and

(4) the effect of the use upon the potential market for or value of the copyrighted work.”

17 U.S.C. § 107. “The central purpose of the investigation is to see ... whether the new work merely supersedes the objects of the original creation.” *Campbell v. Acuff-Rose Music, Inc.*, 510 U.S. 569, 579 (1994); *accord Harper & Row*, 471 U.S. at 550. “[F]air use ... always preclude[s] a use that supersedes”—i.e., replaces—“the use of the original.” *Harper & Row*, 471 U.S. at 550. This is what happened when Android preempted the burgeoning market for Java in smartphones. Every factor cuts against Google here.

**A. Google’s Copying Was Commercially Motivated, Not Transformative, And Illicit.**

Google’s use of Oracle’s code is purely commercial. *Id.* at 562.

Android is one of the most lucrative endeavors of the past decade.

Google’s counsel conceded that “the fact that it’s a commercial use is not in dispute .... The evidence is pretty clear that they created it to

provide a platform on which other Google product[s] could do better.” A21,591. The district court agreed: “Google’s internal documents show[] how many billions of dollars they expected to make off of [Android].... This was intended for commercial purposes with large amounts of money at stake and, therefore, it was not fair use. It was copying.” A21,594.

In considering the first factor, courts also inquire whether the copied material substitutes for the original or, instead, adds or changes the purpose of the original, thereby “transform[ing]” it in a meaningful way. *Campbell*, 510 U.S. at 579. Google’s use of Java declaring code was anything but transformative. Mere alteration is not transformation. A use of copied material is not transformative unless the material is used “in a different manner or for a different purpose from the original.” Hon. Pierre Leval, *Toward a Fair Use Standard*, 103 Harv. L. Rev. 1105, 1111 (1990); *see Campbell*, 510 U.S. at 579.

Google’s use of the declaring code was exactly the same as in Java: to call upon prewritten packages to perform the same functions. The packages also serve the identical purpose: solving the same complex problems and performing the same often-needed functions programmers

desire. While using the declaring code in exactly the same way as the original, Google deployed that purloined code in Android to compete directly with commercially licensed derivatives of Oracle's work. Oracle licenses Java in the mobile market and licensed it for smartphones specifically, including RIM's Blackberry, Nokia's Series 60 phones, and Danger's Sidekick/Hiptop. *Supra* at 15-16.

Such superseding use is “*always*” unfair. *Harper & Row*, 471 U.S. at 550 (emphasis added); *id.* at 569 (“[A] use that supplants any part of the normal market for a copyrighted work would ordinarily be considered an infringement.” (citation omitted)). Copying work to use for the same purpose simply cannot be fair use. *See Peter Letterese & Assocs. v. World Inst. of Scientology Enters., Int'l*, 533 F.3d 1287, 1311, 1318 (11th Cir. 2008) (book about sales techniques superseded the original even though it “adopt[ed] a different format, incorporate[d] pedagogical tools ... , and condense[d] the material in the [original] book”); *Elvis Presley Enters., Inc. v. Passport Video*, 349 F.3d 622, 628-30 (9th Cir. 2003) (documentary containing “significant portions” of video clips supersedes original and is not fair use); *Twin Peaks Prods., Inc. v. Publ'ns Int'l, Ltd.*, 996 F.2d 1366, 1375-77 (2d Cir. 1993) (holding

that a comprehensive guide to the characters and plot of a television show supersedes the show and is not fair use).

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

### **B. Google Copied A Creative Work.**

The nature of Oracle’s copyrighted work also precludes fair use. The Java platform is the product of “substantial creative effort.” *Rogers v. Koons*, 960 F.2d 301, 304 (2d Cir. 1992). Indeed, Google’s “Java guru,” A142, explained: Designing packages is “a noble and rewarding craft,” where “creative[ity]” and “aesthetic[s] matter.” A20,917, 20,920-22. Java developers’ creative efforts resulted in thousands of lines of

original declaring code, embodying a sophisticated design and organization that programmers find easy to use. “Of course” this was “creative,” the district court found. A164.

*Wall Data Inc. v. Los Angeles County Sheriff’s Dep’t*, 447 F.3d 769, 778 (9th Cir. 2006), is instructive. The defendant there also copied software. *Id.* at 778. Because designing the software took “several years” work and a “multi-million dollar” investment, “the nature of the copyrighted work weigh[ed] against a finding of fair use.” *Id.* at 780. Accordingly, the Ninth Circuit rejected fair use.

Here too. Oracle invested years of labor and hundreds of millions of dollars in researching and developing the packages. Java’s chief architect spent almost two years developing a single set of related packages. A22,403-04. Oracle’s substantial and ongoing development effort cost “hundreds of millions of dollars” a year just “on Java.” A20,557.

**C. Google Verbatim Copied The Code And Structure That Matters To A Java Programmer.**

Google also loses under “the amount and substantiality of the portion used” factor. *Harper & Row* is critical. The 300-400 words that *The Nation* copied verbatim were less than 1% of the original work. But

the Court held this factor favored the copyright owner because the infringing work was “structured around the quoted excerpts which serve as its dramatic focal points.” 471 U.S. at 566. Indeed, the infringer “quoted these passages precisely because they qualitatively embodied” the author’s “distinctive expression.” *Id.* at 565. The “expressive value of the excerpts and their key role in the infringing work” meant the use was not fair. *Id.* at 566.

The same is true here. Google copied the only code with any relevance to programmers: the declaring code. *Supra* at 20-21, 43-44. Google “quoted these passages precisely because they qualitatively embodied” Oracle’s “distinctive expression” and were familiar to programmers. *Harper & Row*, 471 U.S. at 565; A5874 (Google took “the good stuff from Java”); *accord* A21,503, 21,957-59. As in *Harper & Row*, the “expressive value of the excerpts and their key role in the infringing work” mean that this factor does not support fair use. 471 U.S. at 566.

Google’s defense that it created its own implementing code is beside the point. *The Nation* wrote 87% of its article. But, like *The Nation*, Google copied the “focal point[]” of the work, *id.*—the declaring

code. “[T]he amount and substantiality of the portion used” factor does not favor Google.

**D. Google’s Copying Damaged The Value Of The Java Platform In The Smartphone Market.**

Google also fails the final factor—“the effect of the use upon the potential market for or value of the copyrighted work.” To prevail, Google must establish that Android did “not materially impair the marketability of the work which is copied.” *Harper & Row*, 471 U.S. at 566. “This inquiry must take account ... harm to the original [and] ... derivative works,” *id.* at 568; *see* 17 U.S.C. § 106(2) (exclusive statutory right “to authorize ... derivative works based upon the copyrighted work”), and the effect on the potential market if the challenged use “become[s] widespread,” *Harper & Row*, 471 U.S. at 568. In two different ways Android materially impaired the actual and potential market for Oracle’s derivative works and superseded the original and its derivatives, thereby compelling a conclusion against Google on this “single most important element of fair use.” *Id.* at 566.

First, Android was designed to be *incompatible* with and thereby fragment the Java platform. As explained, Android undercuts the “write once, run anywhere” credo that is central to Java’s value

proposition and replaces it with “write once, run only on Android.” *Supra* at 66-67. Android thereby deprives Java of the value of compatibility with those additional applications.

Second, Android was designed to replace Java SE derivative works in the smartphone market. Android gave handset manufacturers, wireless carriers, and software vendors a Java-friendly programming environment without licensing fees. A21,631, 21,763. If similar unauthorized, commercial use of the declaring code by other infringers became “widespread,” *Harper & Row*, 471 U.S. at 568, that would decimate all Java commercial licensing opportunities of every kind.

When Google copied the Java packages and released Android, Oracle was licensing in the mobile and smartphone markets. *Supra* at 15-16. Where the plaintiff is “in the business of selling [the work] and [has] done so in the past” it “unequivocally demonstrates a market for the [work].” *Monge*, 688 F.3d at 1181. Android thus substantially harmed Oracle’s already “very lucrative,” commercial opportunities. A22,237. Indeed, Android “now comprise[s] a large share of the United States [smartphone] market,” A135, with 750,000 new devices activated



*every day*, A21,188. Google gives away Android for free, A135, and competes against Oracle's licensing of Java derivatives. As Oracle's President and CFO pointedly observed: "It's pretty hard to compete with free," A22,498, which Oracle learned the hard way with Amazon. *Supra* at 29.

Nothing about Google's use was fair.

### CONCLUSION

For the foregoing reasons, this Court should reverse the judgment.

Dated: February 11, 2013

Respectfully submitted,

By: /s/ E. Joshua Rosenkranz

E. Joshua Rosenkranz

Orrick, Herrington & Sutcliffe LLP

51 West 52nd Street

New York, NY 10019

(212) 506-5000

*Attorney for Plaintiff-Appellant*

# **ADDENDUM**

**EXCERPT OF TRANSCRIPT OF  
PROCEEDINGS  
(INCLUDING ORDER REGARDING  
GOOGLE INC.'S FAIR USE DEFENSE)**

**DATED MAY 9, 2012**

Volume 19

Pages 3162 - 3441

UNITED STATES DISTRICT COURT

NORTHERN DISTRICT OF CALIFORNIA

BEFORE THE HONORABLE WILLIAM H. ALSUP

ORACLE AMERICA, INC.,	)	
	)	
Plaintiff,	)	
	)	
VS.	)	No. C 10-3561 WHA
	)	
GOOGLE, INC.,	)	
	)	
Defendant.	)	San Francisco, California
	)	May 9, 2012

TRANSCRIPT OF JURY TRIAL PROCEEDINGS

APPEARANCES:

**For Plaintiff:**

MORRISON & FOERSTER  
755 Page Mill Road  
Palo Alto, California 94304

**BY: MICHAEL A. JACOBS, ESQUIRE**  
**KENNETH A. KUWAYTI, ESQUIRE**  
**MARC DAVID PETERS, ESQUIRE**  
**DANIEL P. MUINO, ESQUIRE**

BOIES, SCHILLER & FLEXNER  
333 Main Street  
Armonk, New York 10504

**BY: DAVID BOIES, ESQUIRE**  
**ALANNA RUTHERFORD, ESQUIRE**

(Appearances continued on next page)

**Reported By: Katherine Powell Sullivan, RPR, CRR, CSR #5812**  
**Debra L. Pas, RMR, CRR, CSR #11916**  
Official Reporters - U.S. District Court

1 "expression" as it's used in the computer language itself.

2 That would be a useful thing to add in here?

3           And I want you also to know that the -- the books are  
4 not consistent with the way some of you use the word  
5 "declaration." The book that's in evidence and which was  
6 written by the pros seems to use the word "declaration"  
7 slightly differently than was -- and maybe even the word  
8 "signature."

9           I would like for you to help me understand what the  
10 correct answer is so that we can send it up to the Court of  
11 Appeals with a -- you know, a deck of cards that has 52 cards.  
12 You each know what's in there and you make your arguments based  
13 on that and have a common body of description here that works.

14           So there we are. I will see you back here at 1:45.  
15 Thank you.

16           (Whereupon there was a recess in the proceedings  
17 from 1:04 p.m. until 1:45 p.m.)

18           **THE COURT:** Please be seated. Let's go back to work.

19           **MR. JACOBS:** We have a joint request.

20           **THE COURT:** Okay. Make sure we're hooked up. Are we  
21 ready to go, Katherine?

22           We're ready. Okay. Joint request.

23           **MR. JACOBS:** We're both working hard on the Java  
24 description, but would request to 5 o'clock to get it back to  
25 you.

1           **THE COURT:** That's fine. No problem.

2           I have a related request, but you tell me if this is  
3 too hard to do.

4           I think it would be useful to have a chart that had  
5 the 37 packages down column 1. Next column would be number of  
6 classes in package Java.

7           Next would be number of methods. And here I need  
8 your help. Methods, interfaces and fields, or methods alone  
9 would be, perhaps, enough. But methods and interfaces Java.  
10 And then the same two columns but for Android.

11           So it would be a 5-column chart, 37 rows with titles.  
12 And it would indicate the number of classes, the number of  
13 methods broken out by each of the 37.

14           Now, you ought to be able to reconstruct that just  
15 from the code itself. And the code itself is in evidence,  
16 right?

17           **MR. JACOBS:** Yes, Your Honor.

18           **THE COURT:** So would that be a doable project, or is  
19 that just too much for two gigantic companies with seven or  
20 eight lawyers at each table?

21           (Laughter)

22           **MR. JACOBS:** Very doable, Your Honor.

23           **MR. VAN NEST:** Is that a rhetorical question, Your  
24 Honor?

25           **MR. JACOBS:** It's very doable, Your Honor.

1           **MR. BABER:** It's not only doable, Your Honor, I think  
2 it's already been done. The expert reports -- I believe,  
3 Professor Astrachan's report, he has a chart very much like  
4 that.

5           **THE COURT:** Well, I would like to see it, but I -- if  
6 you two are going to start arguing over the -- I'd like for you  
7 to iron out your -- this is a matter of counting up the items.

8           So, anyway, if you could get me something like that  
9 by tomorrow, that would be good.

10          **MR. JACOBS:** Thank you, Your Honor. We will do that.

11          **MR. KWUN:** Your Honor, there is one technical point.

12 There's two different ways you could count the number of  
13 methods that are in a class. One is, you could count the  
14 number that are actually declared in that class. But as Your  
15 Honor knows, you can also inherit methods from a super class.

16           So to the outside world, it doesn't matter whether  
17 those methods were declared within that class or inherited from  
18 a super class. So you could either count the number of methods  
19 that are sort of available to that class, or you can count the  
20 numbers that are expressly declared.

21          **THE COURT:** I want the expressly declared.

22          **MR. KWUN:** Thank you, Your Honor.

23          **THE COURT:** And you can put in an asterisk on that  
24 point. But the -- kind of like counting up the number of lines  
25 that declare something.

1           Okay. We're here for Rule 50. And we have motions  
2 on both sides. Let's start with the motion for JMOL by Oracle  
3 on fair use.

4           **MR. JACOBS:** I'd like to start, Your Honor, by giving  
5 you a list of cases from 1992 to the present, that have found  
6 no fair use as a matter of law.

7           And we begin with *Triad*. These are Ninth Circuit  
8 cases. 64 F.3d 1330. That was a case involving an independent  
9 service operator copying code into RAM.

10           And the Court decided that that was  
11 nontransformative, and on factor four noted that:

12           "If independent service operators like  
13 Southeastern freely used *Triad's* copyrighted  
14 software on a widespread basis to compete  
15 with *Triad*, this would likely cause a  
16 significant adverse impact on *Triad's*  
17 licensing and service revenues, and lower  
18 returns on its copyrighted software  
19 investment."

20           Then there's the *Wall Data* case, in which the Court  
21 held that while software was not purely creative, it is  
22 protected under the Copyright Act. The plaintiff presented  
23 undisputed evidence that its software products were developed  
24 over several years, required a multi-million-dollar investment.

25           And, on factor four:



1 "Whenever a user puts copyrighted software to  
2 uses beyond the uses it bargained for, it  
3 affects the legitimate market for the  
4 product. And widespread unlicensed  
5 copying" -- which would be the effect of a  
6 fair use ruling in the defendant's favor in  
7 *Wall Data* -- "could affect the market for the  
8 plaintiff's software."

9 So that's 447 F.3d 769.

10 So, again, these are cases deciding against fair use  
11 as a matter of law.

12 Then there's the *Worldwide Church of God* case, which  
13 is interesting in that it falls into the category that I think  
14 we could box this case into. It's kind of an adjacent markets  
15 case. "A" has a copyrighted work. It is distributing it,  
16 licensing it, marketing it in a certain realm. "B" comes along  
17 and takes the heart of the copyrighted work and decides to  
18 market it in a new way, in a new form, to a new audience. It  
19 doesn't fundamentally change it.

20 And *Worldwide Church of God* was a case about books  
21 that had been appropriated for use in a different church. So  
22 an entirely different audience. And no fair use was found  
23 there.

24 The Court noted there that:

25 "If there are no genuine issues of material

1 fact, or if even after resolving all issues  
2 in favor of the opposing party, if a  
3 reasonable trier of fact can reach only one  
4 conclusion, a court may conclude as a matter  
5 of law whether the challenged use qualifies  
6 as a fair use."

7 And citing cases that did that as a matter of law.

8 *L.A. News* is another interesting case for the kind of  
9 adjacent markets theory that I was describing. 149 F.3d 987.

10 That's a case where the plaintiff owned the works.  
11 They produced videotapes and licensed them to a broadcasting  
12 company. When the broadcasting company aired the works, it  
13 simultaneously transmitted them to a television news agency with  
14 which the broadcasting company had an agreement. And defendant  
15 copied the works and transmitted them to paying subscribers.

16 Summary judgment in favor of plaintiff, finding no  
17 fair use, was commercial, not nonprofit, not very  
18 transformative. They took the heart of the work.

19 "Allowing such use would result in a  
20 substantially adverse impact on the potential  
21 market for the original work."

22 So the potential market.

23 I think there's an important point here. We do not  
24 have to prove lost profits to prove an adverse impact on the  
25 market for the Java software that is threatened by Android.

1 This is not that kind of causal nexus.

2 Courts frequently make matter of law predictions  
3 about the impact of a defendant's activities on the  
4 plaintiff's -- on the market for the plaintiff's work.

5 The *Leadsinger* case. I mentioned this earlier in our  
6 discussions. This is 512 F.3d 522. A 2008 Ninth Circuit case.

7 This is at the motion to dismiss -- this is affirming  
8 a motion to dismiss. The defendant sought declaratory judgment  
9 that it could copy song lyrics into karaoke. That's kind of  
10 interesting because the song lyrics are only a component of a  
11 song, and they are arguably being placed into a new -- again,  
12 an adjacent market.

13 But this case didn't even get beyond the pleading  
14 stage. There was no claim of transformative use. Karaoke does  
15 not add to or alter the copyrighted lyrics and is not  
16 transformative.

17 And then that brings us to the *Abend* case, which in  
18 some ways is the most interesting of all because *Abend* owned  
19 the copyright on the original story for *Rear Window*, the  
20 Hitchcock movie. And *Rear Window* was more or less out of  
21 distribution. And MCA did a re-release of the film in  
22 theaters, on TV, and on video cassette.

23 And the District Court granted summary judgment for  
24 the defendants based on fair use. And the Court of Appeals  
25 reverse.

1 "Commercial use of a fictional story that  
2 adversely affects the story owner's  
3 adaptations rights is a classic example of  
4 unfair use."

5 Now, I say that's interesting because you can imagine  
6 how the defendant there would argue analogously to Google's  
7 argument.

8 All we did was take -- remember this, the story right  
9 that's been infringed. So it's the structure, sequence and  
10 organization of the *Rear Window* movie. It's the plot outline,  
11 not the movie itself. The owners of the movie rights were not  
12 the plaintiff. It was the story owner.

13 And the defendant there says, this is great for the  
14 story. Look how many more people are going to see the story  
15 underlying *Rear Window* if this movie is out in these new media,  
16 in video cassettes or just re-released into theaters.

17 The court did not give that argument much weight.

18 Applied here, these cases allow for only one outcome  
19 on Google's fair use defense.

20 Going through the factors: Google's use is purely  
21 commercial. There is no nonprofit purpose. This is no  
22 educational purpose.

23 Of course, they haven't taken the APIs and subjected  
24 them to programmers' criticism. They have not done anything  
25 other than port the APIs and the structure, sequence and

1 organization of the code into Android, and deployed it into a  
2 market close to -- arguably already occupied by Java, in that  
3 Java is on smart phones -- so close to and a market in which  
4 Java could reasonably expect to be deployed.

5           How do we know that? We know that because there was  
6 licensing negotiation between Google and Sun, in which Google  
7 sought to take the Java Application Programming Interfaces, the  
8 SSO, et cetera, and deploy it into this adjacent market into a  
9 slightly -- and using a different licensing model.

10           So they sought a license. There's facts going both  
11 ways on what license they sought for what when. But the fact  
12 of the matter is, the Java technology was so relevant as is to  
13 Google's proposed market, that the parties spent a considerable  
14 period of time trying to negotiate a license for that use.

15           And Google went ahead and used the valuable APIs  
16 anyway, in their commercial product, never taking that license.

17           So we know exactly what market has been interfered  
18 with here. There's no hypothetical. There was a licensing  
19 discussion. And then, of course, there's lots of testimony  
20 undisputed about Sun, now Oracle's, licensing model. The  
21 specification license in particular being the license for  
22 independent implementations. The most applicable license for  
23 what Google did or wished to do.

24           If Google's use is fair use, that licensing market is  
25 destroyed. There is no copyright right to back up that kind of

1 openness, that kind of freedom to develop independent  
2 implementations.

3           There is no way, to use Mr. Schwartz's language, to  
4 force compliance with this form of openness through the  
5 assertion of copyright.

6           And then that brings us to the other kind of license  
7 that was most -- that is most seriously disrupted here, and  
8 that's the GPL. Because, of course, there's undisputed  
9 testimony that Java is available pursuant to an open source  
10 license.

11           It's an open source license that didn't suit Google's  
12 commercial needs. And so they took the 37 packages and  
13 deployed them in their own -- to their own end, using their own  
14 license.

15           Once again, if that is fair use, there is no  
16 copyright right to enforce the GPL -- to enforce GPL license  
17 compliance, at least against someone who chooses to take only  
18 the structure, sequence and organization. Which, as we've all  
19 heard, is the heart of the matter here. It's the most valuable  
20 part of the copyrighted work as a whole. It is the 37 packages  
21 that Google wanted because they were the most popular.

22           So this is another reason why this cannot be fair  
23 use. Popularity does not allow for infringement. Investment  
24 does not allow for infringement.

25           It can't be the case that the more popular your work

1 is, the more the defendant can take it and deploy it to its own  
2 commercial benefit. That just gets copyright law and the  
3 incentives in the Constitution backwards.

4           There is case after case, and it is valid, strong  
5 authority today that a commercial use is presumptively unfair.  
6 The cases that have gone the other way on that question have  
7 been categorical -- have been cases in categories.

8           Parody cases, in which the parody itself is  
9 commercial. But we like parodies, and we think they are  
10 valuable, and they don't really destroy the market for the  
11 original work. Because if you want to see the original work,  
12 you've got to see the original work not just the parody.

13           There are the reverse engineering cases that we've  
14 discussed at length in our brief. Again, a kind of a category  
15 in the law where the ultimate product is concededly  
16 non-infringing. And the only question is whether the  
17 intermediate copying is excused on fair use grounds. Again, a  
18 very narrow category.

19           But *Harper & Row*, *Passport Video*, the *Leadsinger* case  
20 as recently as a couple of years ago, all reinforce that a  
21 commercial use -- in the general case, a commercial use is  
22 presumptively -- gives rise to a presumption of harm to the  
23 market and, therefore, the other factors are going to have to  
24 tilt very heavily in the defendant's favor if we're ever going  
25 to find in favor of the defendant on fair use.

1 Looking more narrowly at the doctrine around  
2 transformative, Google's use was not. The transformative cases  
3 are trying to address a case in which the expression is recast  
4 or recapitulated in a form that changes its underlying message.

5 And probably Google's best case is a Google case, in  
6 which the plaintiff owned photographic rights. Google took  
7 thumbnails of them as part of the search engine. And so Google  
8 wins that a search engine use of thumbnails of the original  
9 photos is fair use. Why? Well, a search engine is an entirely  
10 different purpose, an entirely different use.

11 Now, frankly, I think that's a pretty close case.  
12 The idea that you could reproduce a photograph in thumbnails  
13 and display it to the world without getting a copyright, that's  
14 a close case under fair use. But, it did go Google's way.

15 But just compare the facts there with the facts here.  
16 A search engine as compared to a photographer. Android as  
17 compared to Java. A software platform as against a software  
18 platform. Couldn't be software platform to software platform  
19 can't be much closer. And photograph to search engine  
20 considerably more remote.

21 So Google did nothing to change the message of the  
22 APIs. The APIs were a popular, attractive, heavily-invested-in  
23 way to reach programmers and make the Java platform attractive  
24 as a programming environment. And looking in the other  
25 direction, a set of design materials for class library



1 creators, creating independent implementations under the  
2 specification license.

3           What did Google do? Took the 37 packages. They  
4 created very similar documentation. They put it out there for  
5 Java programmers to use in the same way that they use the 37  
6 packages in Java, and created class libraries. Independent  
7 implementations, they claim. We saw that wasn't true, but it  
8 doesn't really matter for present purposes.

9           They created supposed independent implementations in  
10 core libraries, just like the licensed or Sun developers do  
11 with that information.

12           So there is no recasting. There is no reforming.  
13 There is no new message. It's the very same message to the  
14 very same purposes -- purpose, for purposes of fair use  
15 analysis.

16           Creative versus functional. One can debate this.  
17 Obviously, software is not a symphony. Software is not a poem.  
18 But what was interesting about this trial was the undisputed  
19 evidence from both sides and both sides' experts about how  
20 creative the authoring process of APIs is.

21           This is not the creation of a functional work within  
22 the meaning of copyright law. Not when we heard from witness  
23 after witness how much flexibility there is in creating APIs,  
24 and how much creative labor went into the process of creating  
25 these 37 Java APIs.

1 Now, let me -- let me just do a parenthetical here,  
2 because I think it's important.

3 From what you have signaled to us so far, Your Honor,  
4 the decision you're writing is going to be heavily grounded in  
5 the facts. The facts of Java. The facts of the Java  
6 Application Programming Interfaces and these 37 packages.

7 Not all interfaces are created equal for purposes of  
8 copyright or for purposes of fair use analysis.

9 These core library package APIs are very closely  
10 drawn to the underlying code itself. We saw that in the method  
11 declarations showing up in the API documentation and in the  
12 code.

13 This is not merely a set of -- of numerical values  
14 that represent an interface to a PlayStation box or a Nintendo  
15 box. Thousands of pages of writing represent the 37 packages  
16 here.

17 So the word "interface" shouldn't be overused in  
18 understanding these fair use cases. We're talking about these  
19 37 packages and the creativity and authorship that went into  
20 them, not whether every computer program is creative or  
21 functional. Not whether every interface is creative or  
22 functional. These programming interfaces.

23 Undisputed evidence about the creativity that went  
24 into the authoring process.

25 Third factor, amount and substantiality of the

1 portion used.

2 *Harper & Row*. Great case. 300 words out of 200,000,  
3 and just a week before the book is coming out. So all sorts of  
4 arguments: We made a bigger market for the book. We have more  
5 readers for the book because we're a magazine and we're the  
6 nation and we're publishing an extract from *Harper & Row*.

7 But, no, you can't take 300 words out of 200,000 from  
8 an unpublished Presidential Memoir. The portions actually  
9 quoted were selected, according to the Supreme Court, as among  
10 the most powerful passages in those chapters.

11 And that's what Google did here. They took the 37  
12 packages. They took -- one of their witnesses said, We took  
13 the good stuff.

14 They took the 37 packages that they thought the  
15 programmers programming for Android would most want to see, and  
16 left the rest behind.

17 And then, finally, that brings me back to the harm to  
18 the market. And their only evidence, if you call it that, on  
19 Google's side is that Java ME is doing okay under Oracle.

20 Well, a couple of things about that. First of all,  
21 undisputedly, Java ME was not the target for Android. Java SE  
22 was.

23 Secondly, the undisputed evidence is that Java ME is  
24 not doing well in Android's markets. Java ME is doing well in  
25 places where Android has not yet penetrated.

1 And, by contrast, Java ME was licensed, for example,  
2 to Amazon for its Kindle, and is no longer licensed to Amazon  
3 for the Kindle Fire because Android has taken its place.

4 So even in the most kind of granular causation-based  
5 analysis, much more rigorous than the cases require for fair  
6 use, we have evidence of direct market supplantation.

7 But I think -- but, the evidence was undisputed,  
8 again, that the Java model is a comprehensive model of  
9 licensing of both underlying code and of the specifications;  
10 that that model has been threatened by Android because Google,  
11 notwithstanding the previous negotiations, took what it wanted  
12 for itself without a license.

13 The whole approach that Sun, now Oracle, takes to  
14 fragmentation is threatened by Android because Android took  
15 subset and superset it.

16 And while there was a lot of evidence going both ways  
17 on fragmentation -- on whether the word "fragmentation" can  
18 reasonably be applied to say the differences between ME and SE,  
19 there was undisputed evidence that within platforms Sun, now  
20 Oracle, took aggressive measures to try as best they could to  
21 control this kind of open source model in which there are  
22 independent implementations.

23 And, it's undisputed that Oracle, upon acquiring Sun,  
24 invested substantially more resources in Java. There was  
25 testimony from Mr. Reinhold about a near doubling of the number

1 of engineers working on Java. So the commitment is evident in  
2 the assignment of resources to the project.

3 And what Google is doing is looking backwards and  
4 saying, well, Sun was not a strong company. Sun had let  
5 fragmentation develop.

6 That's not the right analysis. They don't get to  
7 take -- this is not a case where -- this is not an area of law  
8 where the defendant gets to take advantage of the plaintiff's  
9 weaknesses at a particular moment in time and then say, see, we  
10 get away with it.

11 So on all four factors, Google loses. Loses  
12 strongly. And, again, on undisputed evidence.

13 But the underlying purpose of copyright law and the  
14 underlying purpose of fair use also needs to be looked at here.  
15 One of the basic reasons for fair use analysis is to deal with  
16 a situation in which, by its nature, the plaintiff would be  
17 unwilling to grant a license. That's the parody case. That's  
18 why we have a commercial use for -- commercial use is okay even  
19 if it's a parody because most authors don't like their works to  
20 be parodied.

21 But here there is a license. Google may not like it  
22 for business reasons. May have had a different business model  
23 in mind. That was Mr. Rubin's testimony. The negotiations  
24 broke down because Sun would not adopt Google's business model.  
25 But business model differences do not give rise to fair use

1 defense.

2 Thank you, Your Honor.

3 **THE COURT:** All right. Google's turn.

4 **MR. VAN NEST:** Good afternoon, Your Honor.

5 I think the cases make very clear that commercial use  
6 is not a disabling factor. It's simply one factor.

7 All of the leading fair use cases recently in our  
8 Circuit have dealt with situations where the defendant's use  
9 was a commercial use.

10 If you look at *Sony vs. Connectix*, the video  
11 equipment manufacturer engineered Sony's APIs and created a  
12 competing game platform to play video games.

13 *Sega vs. Accolade*, video game maker copied Sega's  
14 APIs and made video games that were compatible with the Sega  
15 system.

16 *Campbell*, recent Supreme Court case was a rap  
17 group --

18 **THE COURT:** Sorry, did the *Sega* case involve APIs? I  
19 didn't remember that part.

20 **MR. VAN NEST:** They had to reverse engineer the  
21 interfaces so they could make games that were compatible. They  
22 are not the same kind of APIs we're talking about here.

23 But what they ended up making was video games that  
24 competed with the video games Sega sold for the Sega platform.

25 **THE COURT:** Well, of course, I understood that part.

1 But you said they copied the APIs. I don't even remember that  
2 term being in the decision.

3 **MR. VAN NEST:** They --

4 **THE COURT:** Are you sure that term is in the  
5 decision?

6 **MR. VAN NEST:** I'm not sure, Your Honor. But what  
7 they did was reverse engineer the Gateway, if you will, to the  
8 Sega system so they could make games that were compatible with  
9 it.

10 My point is simply that all these cases -- including  
11 *Campbell* in the Supreme Court, which says there is no  
12 presumption against fair use just because there's a commercial  
13 use. *Campbell* was a rap group selling its music for profit.

14 So all of these cases confirm that, just as Your  
15 Honor's instruction says, the commercial use is one factor.  
16 And fair use determination involves weighing all the factors,  
17 and giving them the weight that the jury deems appropriate.

18 Similarly, on the other big point for Oracle, the  
19 mere fact that the defendant's use has some impact on the  
20 plaintiff's copyrighted work also doesn't disable anything.

21 Obviously, in Sony, the whole point was to make a  
22 competing platform. Connectix did that so that people could  
23 play games on a computer that they were currently only playing  
24 on Sony's PlayStation.

25 So the court there said, very clearly, that, yes, we

1 understand there's going to be an impact on Sony. But we  
2 recognize that this is a brand-new platform that's been  
3 created; that the whole point of the copyright law to protect  
4 expression, not ideas. If people are creating a new platform  
5 where others can express themselves and compete, that's --  
6 that's consistent with fair use, consistent with the purpose of  
7 the Copyright Act.

8           And, obviously, in Sony there was no question there  
9 would be an impact on the plaintiff, and their fair use was  
10 determined to be appropriate.

11           Same with Sega. In Sega there was no question that  
12 both companies were going to be in the business of selling  
13 games, and there would be competition and impact.

14           So there again the Court said, because it's a  
15 functional nature of -- it's a computer code here, we're going  
16 to -- we're going to allow fair use.

17           Now, I would say the cases they are relying on, most  
18 of the ones Mr. Jacobs referred to are not in their brief, but  
19 they are all cases about things like books, where the book is  
20 wholesale copied, or poems, or songs, or movies, or plays.

21           This is not such a thing. This is a situation where  
22 we're talking about computer software, which is purely  
23 functional.

24           When you get down to looking at the four factors,  
25 there is an overwhelming amount of evidence of fair use,



1 consistent with what we told the jury in the opening and the  
2 closing.

3 And I'm not going to claim that there's anything  
4 undisputed in this case. There was plenty of evidence adduced  
5 by both parties.

6 But with respect to the transformative nature of  
7 Android, there were two kind of principal points. One was, it  
8 was an open platform. The platform was not licensed or sold.  
9 It was open for anyone to use.

10 And Your Honor heard testimony from Rubin, from  
11 Schwartz, from Ellison, that they all attempted to take  
12 advantage of the Android platform.

13 It was out there for everybody. It has fostered  
14 increased expression in the form of more application  
15 developers, increased competition among the handset makers, the  
16 carriers, and the app developers. And both Sun and Oracle had  
17 an opportunity to compete.

18 This is exactly what the Ninth Circuit held was  
19 appropriate in *Sony vs. Connectix*. If you are opening up a new  
20 platform, that is consistent with fair use.

21 It was also, point two, a brand-new product.

22 Your Honor is aware that the testimony was the 37  
23 APIs were incorporated into a full stack. The folks at Sun had  
24 not been able to create such a full stack, although they are  
25 the experts on Java.

1           You heard testimony that the 37 APIs interact with  
2 other layers in the stack, particularly the application  
3 framework. That all the source code, all the implementing code  
4 in these 37 packages was brand-new and totally different, other  
5 than the nine lines we may talk about later this afternoon.

6           The platform supports all sorts of new functionality  
7 and, therefore, there was more than enough evidence to find a  
8 transformative nature. And the case law doesn't say that you  
9 have to be using the functional features like this in a  
10 completely new and different way.

11           Your instruction was quite right; what has been  
12 added, what has been changed, and that's exactly consistent  
13 with what we proved with respect to Android.

14           With respect to the nature of the copyrighted work,  
15 no question it's functional. No question that to some degree  
16 the words and names are necessary to run existing code. If you  
17 want to run existing Java language code, you have to use the  
18 same fully qualified names. You heard that from a number of  
19 witnesses.

20           These APIs are expected to be available for  
21 programmers in Java. Dr. Bloch testified at length about that.  
22 Dr. Astrachan, too.

23           There are some of them that are absolutely required  
24 just to run the language. There was a lot of testimony about  
25 that and admissions by Dr. Mitchell and Dr. Reinhold that some

1 of these APIs are necessary just to use the language itself.

2 And it's clear in the Ninth Circuit that computer  
3 systems and APIs are deemed functional and accorded less  
4 protection as a result. There really is no doubt. I think the  
5 testimony was that there was a lot of creative effort that went  
6 into these. That's all fine and well, but the factor with  
7 respect to the nature of the copyrighted work is addressed to  
8 what's the nature of the completed work? And the nature of it  
9 here is it's functional.

10 With respect to the amount of copyrighted material  
11 used, again 7,000 lines out of 2.8 million lines, that was the  
12 conclusion of both Dr. Astrachan, Dr. Mitchell.

13 Dr. Reinhold conceded that the SSO, which was all  
14 that was at issue at the end of the day, was 7,000 to 10,000  
15 lines of code out of 2.8 million.

16 The 37 API packages, what was used were the names and  
17 the declarations. The source code was completely new,  
18 completely different; 15 million lines of code, according to  
19 Dr. Astrachan and Mr. Bornstein.

20 With respect to harm, I think there was -- there was  
21 a plentiful amount of evidence here on both sides. Certainly,  
22 there was conceded evidence that Java language is still number  
23 one in the world. Dr. Mitchell testified to that. Java  
24 profits at Oracle are up 10 percent year over year. Mr. Risvi  
25 testified to that. According to Oracle, they are having

1 continued success with Java products and products like Rim and  
2 Nokia. Mr. Screven and Dr. Reinhold testified to that.

3 And there is -- there may be evidence of a threat of  
4 fragmentation, but no evidence of real fragmentation. Nobody  
5 is confused that Android is a Java Platform or part of a Java  
6 family.

7 And we heard a lot of evidence from Dr. Reinhold that  
8 fragmentation now means limited to one platform, not across all  
9 platforms. Well, Android is not the same platform at J2ME or  
10 J2SE. It's a different platform.

11 **THE COURT:** Does the Android literature in any way  
12 say that these programs previously written in Java will run on  
13 Android, or is that just left to the developer to figure out on  
14 their own?

15 **MR. VAN NEST:** I don't know, your Honor, if the  
16 literature says that. I'd have to check. Given that we don't  
17 have the Java brand, I doubt it, but I'm not sure. I don't  
18 know the answer to that.

19 But I think developers are aware, as Dr. Bloch  
20 testified and Mr. Bornstein, that the -- since you're writing  
21 in the Java language and since people do know that Android is  
22 written in the Java programming language, that that  
23 functionality would certainly be available.

24 I think the other thing on potential market harm, I  
25 do think here the testimony from Schwartz was also important.

1 He made a decision to welcome Android. They made a decision to  
2 support platforms that support Android. They felt it would  
3 have been a disaster if Android had used Microsoft language or  
4 some other language. And they were persuaded, after debating  
5 it internally, that Android could be a good thing for Java,  
6 would be a good thing for Java. And, hence, that testimony  
7 goes to a lot more than the equitable defenses. It also goes  
8 to the fact that the people running Sun at the time thought  
9 that Java -- that Android could place a set of rockets onto  
10 Java. And they said that publicly.

11 They've used their own products on top of Android.  
12 As your Honor heard, the JavaFx product was something they  
13 featured at the JavaOne developer conference in 2008.

14 **THE COURT:** Repeat that again.

15 **MR. VAN NEST:** The folks at Sun not only endorsed  
16 Android, but they built their own product to run on the Android  
17 Platform. That's the product that was developed in the video  
18 that we saw of the JavaOne conference in 2008. That's folks  
19 from Sun on stage showing off the use of a Javafx product on  
20 top of the Android Platform.

21 And there was discussion about that between Mr.  
22 Schmidt and Mr. Schwartz. There was discussion about that  
23 between Mr. Gupta and Mr. Rubin. There was a demonstration at  
24 JavaOne in 2008.

25 And, again, it's all consistent with what Schwartz

1 said, which was: We knew that there were choices. We knew  
2 that Android could use other languages. We felt it would have  
3 been far, far worse for Android to go off and use some other  
4 language than Java where we, Sun, wouldn't have any opportunity  
5 to participate. And so that's the way it went.

6 The final point, your Honor --

7 **THE COURT:** Let me ask a question on this, on that  
8 point.

9 Prior to the time that Oracle acquired Sun, what  
10 emails or internal materials are there in the Sun records? Are  
11 there some? It seems like there was something.

12 **MR. VAN NEST:** Yes.

13 **THE COURT:** No, but where they were saying it is  
14 harmful, that Android is harmful to Java. Is there something  
15 like that or am I thinking of the wrong thing?

16 **MR. VAN NEST:** I'm not --

17 **THE COURT:** Publicly you've got the rockets blog,  
18 okay. That works in your favor. But were there some internal  
19 documents that contradicted that?

20 **MR. VAN NEST:** I'm not aware of any documents that  
21 were sent to Google that contradict that.

22 **THE COURT:** No, no.

23 **MR. VAN NEST:** But I'm not sure what --

24 **THE COURT:** I'm talking about internal.

25 **MR. VAN NEST:** There may have been some. I'm not

1 aware, but I would say this.

2           The rockets is just the beginning of it. Remember,  
3 that at the same time as the blog, there was a personal email  
4 from Schwartz to Schmidt saying, "What can we do to support  
5 your announcement? We want to support your announcement."  
6 That happened around the time in November.

7           Then they have a meeting in roughly March or April of  
8 '08. Schmidt and Schwartz meet and there are emails around  
9 that. Schwartz asked Schmidt: Can we build a product on top  
10 of Android? We would like to do that. Can you show me your  
11 licensing? What's the open source license? The Apache  
12 license. So there is an email exchange between the two of them  
13 where Schmidt sends Schwartz an email around the time of their  
14 meeting reflecting they can do it.

15           Then there is another meeting between Rubin and Gupta  
16 where Gupta comes to congratulate Rubin on the launch of  
17 Android and says: We would like to explore building our own  
18 JavaFx product on top of Android. Can we do that?

19           Then there is the JavaOne conference in '08 where  
20 they demonstrate on stage, in the video we all saw, that they  
21 have a JavaFX product running on Android.

22           All of that is happening. And the point of it is  
23 simply that the folks running Sun at the time were trying to  
24 use Android, the platform. They saw benefit in it for them.  
25 They saw benefit in making positive statements about Android

1 and its usage of Java and they were attempting to participate  
2 in the platform.

3           The final point I want to make, your Honor, is that,  
4 I mean, the fair use law is pretty clear that -- and your  
5 instructions reflect it absolutely; that the jury's job is to  
6 look at all the factors. No one factor is determinative. They  
7 are to be weighed together. The jury is to give them the  
8 weight that they feel they deserve. And no one factor alone is  
9 determinative.

10           And based on that, given the amount of evidence that  
11 exists on each of the four factors, JMOL is simply not  
12 appropriate.

13           Mr. Baber wants to add a comment or two.

14           (Brief pause.)

15           **MR. VAN NEST:** He's pointing out that in the Sega  
16 case there is a reference to interface specifications at  
17 Page 1515.

18           The defendant decompiled object code to get interface  
19 specifications, then used the specifications and included  
20 functional descriptions of interface requirements in their own  
21 manual, but they wrote their own procedures to be compatible.

22           So it's a similar situation where in Android the  
23 interfaces may be the same, but the source code implementing  
24 those is different, original written by Android, written by  
25 Google developers.



1 So that's an additional point on the *Sega* case.

2 **THE COURT:** All right. Very brief rebuttal.

3 **MR. JACOBS:** It's important to emphasize who bears  
4 the burden of proof here. It's an affirmative defense. Google  
5 bears the burden. They didn't come close to meeting it on the  
6 various factors.

7 Analyzing the decisions, many defendants claim:  
8 We're doing you a favor. We're helping you out. You don't  
9 realize it. You don't like it. So you're suing us, but  
10 actually we're helping you. Court's don't give that much  
11 weight.

12 **THE COURT:** Here Mr. Schwartz said that rockets were  
13 being put on Java. I mean, that -- that's a very helpful  
14 document for Google here.

15 **MR. JACOBS:** One week later Rich Green, senior  
16 executive at Sun, is published in an article that Google said  
17 it saw in which he said, "We're very concerned about the  
18 fragmentation of Android." Rockets on Java is before the SDK  
19 is released. Rich Green's comment about fragmentation is after  
20 the SDK is released.

21 Jonathan Schwartz, no friend --

22 **THE COURT:** Why is there more fragment -- didn't  
23 Google have the right to -- let's say that Google had written  
24 using the Java language it's own set of APIs -- it didn't even  
25 use the same words, names, whatever -- from the ground up, but

1 it used the Java language. So it's out there.

2 And, surely, you would admit that they had the right  
3 to do that; correct?

4 **MR. JACOBS:** Separating out the APIs, just  
5 implementing the --

6 **THE COURT:** Oh, no, no, no. They have their own  
7 APIs. They don't even use the same names. They have got them  
8 organized differently. There is no SSO problem. It's a  
9 completely different SSO, completely different set.

10 It has the same functionality spread around just  
11 like -- you can't possibly claim you have the right to ask a  
12 method to tell you the cosign of an angle.

13 **MR. JACOBS:** That's the Spring case came in, the  
14 evidence on Spring.

15 Spring is a company, as the testimony undisputed  
16 revealed, that has implemented an entirely different set of  
17 APIs, but supports the Java programming language. We know how  
18 Google supports the Java programming language --

19 **THE COURT:** All right. Let's say Spring -- are you  
20 saying that Spring did something wrong?

21 **MR. JACOBS:** No. Spring did not implement the API  
22 specifications owned by Oracle.

23 **THE COURT:** If Google could have done what Spring  
24 did, why is there any greater -- there there would have been  
25 immense fragmentation. It would have been completely

1 different. It would have been a Java-based platform that had  
2 nothing to do with your version of Java, but instead they did  
3 one that was part -- 37 were consistent. The others were not  
4 consistent. And why -- if they could do the greater, why  
5 couldn't they do the lesser?

6 **MR. JACOBS:** Because the lesser has its own  
7 commercial burdens. They would have had to make their own  
8 investment. They would have had to make their own investment  
9 in training developers. They would have had to make their own  
10 investment in creating the APIs in the first place.

11 I mean, that's the nature of intellectual property  
12 protection, your Honor.

13 **THE COURT:** I understand possibly that point, but I'm  
14 talking about in terms of fragmentation.

15 **MR. JACOBS:** Fragmentation is defined in the Java  
16 environment as implement -- subsetting or supersetting the API  
17 specifications. That is what the Java Community understands  
18 fragmentation to be. Again, that was undisputed.

19 And so if they want to do something that creates a  
20 whole new nother world, we can't stop that and the Java  
21 Community wouldn't understand that as fragmentation either.  
22 They would understand that as Google using the Java programming  
23 language --

24 **THE COURT:** There is no decision anywhere that says  
25 copyright prohibits somebody from supersetting, let's say, one

1 method -- one class.

2 **MR. JACOBS:** Of course not, your Honor.

3 **THE COURT:** All right. So that's just -- that's just  
4 the business model.

5 All right. So I guess your argument comes down to  
6 saying: We didn't want anyone to superset our classes and that  
7 hurts us in some way. And you call that fragmentation.

8 **MR. JACOBS:** There are internal emails at Google  
9 acknowledging fragmentation concern. And you heard a lot of  
10 testimony about how Google enforces anti-fragmentation in its  
11 world through anti-fragmentation provisions, through testing  
12 suites. It's an entirely analogous model. And you heard now  
13 in Phase two they have invested tremendously in  
14 anti-fragmentation.

15 So, fragmentation is a concern for any platform  
16 developer who is trying to create an ecosystem and doesn't have  
17 a closed model like -- say, like Apple. If you want to  
18 discourage that kind of model, if you want to discourage open  
19 models in which independent implementations are allowed, but  
20 there's a set of licensing restrictions and agreements that  
21 create consistency between those platforms, then go with Google  
22 on fair use because it would be devastating to the Java model  
23 if people can pick and choose at will and fragment to their  
24 heart's content. Because if Google can do it, so can the next  
25 guy. And it may not be so --

1           **THE COURT:** But aren't you the one that wanted the  
2 jury in this case? Didn't I hear Mr. Van Nest at one point say  
3 they would waive a jury, and you said no, you wanted a jury.  
4 So now we have the jury's work on this.

5           **MR. JACOBS:** We do, your Honor.

6           **THE COURT:** But now you don't like what the jury came  
7 out with. You want the judge to make a ruling.

8           **MR. JACOBS:** You're right.

9           **THE COURT:** Well, I'm not sure you're entitled to it.

10          **MR. JACOBS:** That's our burden to show you that we  
11 are, your Honor.

12          **THE COURT:** I have a different question I want to ask  
13 both of you to address. No one has addressed this and maybe  
14 it's because it's just completely off base.

15                 It's going to take a minute to develop this, because  
16 we have three, three of -- at least three and maybe four of  
17 these packages are referred to as core.

18                 The original -- when the language first came out, the  
19 book was all -- the book on the language, and it included the  
20 three. I think it was IO -- java.lang, java.io and  
21 java.something else.

22          **MR. KWUN:** Java.util.

23          **THE COURT:** Util, yes. And this had things that in  
24 other languages are just part of the language, like return the  
25 cosign. Return the tangent. Return the greater of two things.

1 Return the absolute number. Those are things that in other  
2 languages are just one of the normal parts of the language.  
3 And these were all lumped together as -- or print, print  
4 function. Now, at that time no sharp distinction was made. No  
5 distinction was really made between the packages and the rest  
6 of the language.

7 As time went on, the programming people who liked  
8 Java could see that it was a handy way to do pre-packaged  
9 programs that would do things a lot harder than return the  
10 cosign or the tangent, and this -- this group of 37, and now  
11 166, packages grew up that had many, many, many functions. 37  
12 of those were duplicated in some sense in the Android.

13 Now, one of the reasons I had broken out in that form  
14 of -- special verdict form that got rejected by both sides was  
15 that I thought it was plausible that a jury could say those  
16 first three -- util, IO, and lang -- it was fair use to use  
17 those three because the language wasn't any good without it.  
18 But because of the ownership issue, Oracle withdrew a  
19 one-by-one package analysis and went as a group, okay? So I  
20 said, Fine, we'll go with it the way you want to go with it.

21 But it concerns me that you would be asking for a  
22 home run on a question we put to the jury on where, to my mind,  
23 at least on those three there is a very strong argument for  
24 fair use, which in and of itself would deny a global win for  
25 Oracle on this point. And Oracle is the one that chose to put

1 it to the jury in that way for reasons that have to do with all  
2 of that ownership stuff that we spent many afternoons debating.

3 I don't know how to -- you know, this is -- this is a  
4 complication that I need some guidance on from you, but it  
5 weighs on my mind. I see some of -- I see some of the strength  
6 of the arguments made by Oracle on these factors, but it would  
7 be to my mind wrong to allow those three packages to be -- I  
8 think the fair use argument there is very strong. Certainly, a  
9 jury could say that it was fair use to use those three and  
10 based on that alone would have been entitled to say, no -- I'm  
11 sorry, reject Oracle's view.

12 Now, that's the one on which, though, the burden of  
13 proof was on Google. So even if they could prove three, do  
14 they have to prove -- do they have to prove all 37? Do they  
15 just have to prove one?

16 We didn't get into that level of detail and it didn't  
17 occur to me until after the verdict that that was lurking  
18 there. And maybe you all saw that and just let it go, you  
19 didn't want to raise it. But that's where we are now.

20 So I'd like to get your views on the fair use issue  
21 as it applies to those three. Maybe there's something I don't  
22 see.

23 And here is another complication. You didn't make on  
24 the Google -- you did not make a JMOL on that issue. No Rule  
25 50 by Google on these three packages. And maybe you played the

1 hand that way for precisely the flip side reason that you were  
2 going for the home run yourself.

3 So I don't know where this -- you know, I have been  
4 thinking about this case, you know, in my own way. I have been  
5 trying to work my way through it. And I see those three as  
6 possibly very different than the rest of this case, and so I'd  
7 like to hear your views on that subject.

8 Mr. Jacobs.

9 **MR. JACOBS:** I think there are two questions lurking  
10 there. One is the fair use merits of those three packages, and  
11 the second is how the possibility that there might be a  
12 dividing line between those three and the rest analytically  
13 might effect the JMOL motion.

14 On the first, we have to be clear what we're talking  
15 about. We're talking about packages that were referred to in  
16 the language specification, but not fully specified, and  
17 certainly not fully specified in their current form.

18 So if the idea was Oracle/Sun made some declarative  
19 statement about the programming language being free for all, we  
20 tried to figure out what that declarative statement means and  
21 its practical impact on packages. And so we go back to 1996  
22 and we look at what the programming language declaration might  
23 have said then.

24 Now, keep in mind the evidence on this is not very  
25 clear in terms of what was actually said and what was actually



1 made available for use by all, but let's stay with the  
2 hypothetical anyway.

3           There's a declaration made in 1996. The programming  
4 language is free for all. There is a book that you can look  
5 at. And that which is -- so, therefore, one thinks that that  
6 which is specified in the book that goes beyond the formal  
7 definition of the programming language is available for all.  
8 That is fragments. That is fragments of what we're talking  
9 about today with the 37 packages.

10           So whatever the fair use merits -- and I think I  
11 would like to get to the second question first.

12           If somebody takes a copyrighted work and copies it  
13 and it turns out that during the course of the litigation some  
14 component of the copying is not justiciable or is not  
15 probative, maybe it's held to be uncopyrightable, that doesn't  
16 mean that the infringing -- that you divided up the judgment.  
17 Absent some proper motion by the defendant, that would result  
18 in -- would result in that.

19           So take the *Abden* case again. There were plot  
20 elements in *Rear Window* that were common place, scenes a faire,  
21 et cetera. You could show that all you want, but the replay of  
22 *Rear Window* in *Abden* is a copyright violation.

23           So I don't think the possibility, kind of  
24 unadjudicated and not fully litigated possibility of fair use  
25 analysis on these packages should interfere with JMOL,

1 particularly as we settled on the definition of the work as a  
2 whole. Because the work as a whole here, I guess, on the  
3 giving side, if you will, on the copyright holder side is the  
4 166 packages, and the accused packages are the 37 Android  
5 packages taken as a group.

6 I don't think they get off the hook if eency-weency  
7 bits of those packages would be held to be non-infringing under  
8 any theory.

9 Thank you.

10 **THE COURT:** Thank you.

11 Mr. Baber.

12 **MR. BABER:** Your Honor, just a few comments on the  
13 issue of the core packages, as opposed to the others.

14 Obviously, our -- we also have the issue of whether  
15 or not those are copyrightable, the whole issue that's out  
16 there. That clearly --

17 **THE COURT:** That's a separate point. If you were to  
18 win on that, then this will all be moot. But we're assuming  
19 right now that you lose on that.

20 **MR. BABER:** That's right.

21 The reason why we didn't urge a separate jury verdict  
22 question just on some or all of those packages is because  
23 Oracle had withdrawn the claim for findings of infringement as  
24 to those packages. And the way the verdict was set up, you had  
25 to first find infringement and only if you found the

1 infringement as to something, whether it was the SSO, whether  
2 it was the documentation, only then would you reach a fair use  
3 issue.

4 So that -- I think in terms of the process and  
5 procedure and how we got there, that's why nothing got broken  
6 out package-by-package, whether it was their claim of  
7 infringement or our fair use defense.

8 But, in fact, your Honor, the record contains  
9 evidence. There's two parts to the evidence as to how these 37  
10 packages relate to the language.

11 You had testimony from Dr. Bloch and Dr. Reinhold of  
12 Oracle, who both agreed that there are some 60 or 61 classes  
13 within the 37 packages that are necessary to use the language.  
14 Doctor --

15 **THE COURT:** Those are only in those three packages I  
16 mentioned?

17 **MR. BABER:** I believe they are scattered in those  
18 three packages that you mentioned, your Honor.

19 Dr. Bloch then gave a second level of analysis and he  
20 said, but in order to fully implement those 61, you need things  
21 from a bunch of other classes. And it wound up being on the  
22 order of 2,000 different methods and fields, et cetera.

23 **THE COURT:** Are those all still within those three  
24 packages?

25 **MR. BABER:** No. Those then expand to about 10 of the

1 37 packages, as I recall.

2 But then you had testimony at the very end --

3 **THE COURT:** Is that really in the record? What are  
4 the names of those 10?

5 **MR. BABER:** They are not in the record, your Honor,  
6 and we didn't do it one-by-one, just like they didn't do their  
7 packages one-by-one, because I think what wraps it all together  
8 is at the end of our case, Professor Astrachan was on the  
9 stand, and Professor Astrachan first said, yes, I agree with  
10 Dr. Bloch's analysis, both the first level and the second  
11 level, and I agree with Dr. Reinhold's analysis.

12 And Dr. Reinhold also did analysis, you may recall,  
13 of which classes were necessary, had to be known to the  
14 compiler. So he came up with a different number, I think 40 or  
15 so.

16 But then Professor Astrachan went a step further, and  
17 he said, Well, there are clearly these parts of these packages  
18 that are necessary to practice the language, but all 37 of the  
19 packages that are in Android are necessary as a practical  
20 matter.

21 **THE COURT:** That's a different point. That's  
22 different. I mean, it's one thing to say the tentacles of  
23 the -- what was it -- 61 classes reach out to another 100  
24 classes and they are spread over 8 or 10 packages, and so you  
25 take those and you -- somehow you put that in group one.

1 But let's say it's 10. That still leaves 27 that  
2 are -- the tentacles don't even touch. So that -- that only --  
3 you have to fall back on your -- the developers are expecting  
4 these to be there as a practical matter.

5 **MR. BABER:** That's correct, your Honor. That's why I  
6 distinguished --

7 **THE COURT:** That's a different kettle of fish.

8 **MR. BABER:** Correct. And, again, a lot of that,  
9 frankly, that was in the record, your Honor, was for you on  
10 your issues of copyrightability. And --

11 **THE COURT:** So, all right. All right.

12 **MR. BABER:** And then I would also observe, just to  
13 the point that Mr. Jacobs just made, which is, well, if there's  
14 parts that are maybe scenes a fair, et cetera, that goes back  
15 to the whole methodology for determining infringement, which is  
16 anything that's not copyrightable, whether it's because it's a  
17 Section 102(b) method of operation, whether it's because it's a  
18 functional requirement for compatibility, whether because it's  
19 a scenes a faire, those are supposed to be removed from  
20 consideration before the analysis for copyright infringement.

21 You go through the process of what the Ninth Circuit  
22 calls analytic dissection or what the Second Circuit calls, you  
23 know, abstraction filtration comparison.

24 At the end of the day, all you're supposed to compare  
25 are the parts that are copyrightable. And given your Honor's

1 instruction to the jury, you should assume, you should take it  
2 for granted, that this is copyrightable. Getting down to this  
3 level of granularity as to whether one class or another class  
4 in java.io was necessary for the language, frankly, would have  
5 been an impossible task for the jury given the assumption that  
6 these are copyrightable for purposes of their analysis.

7 **THE COURT:** All right. We need to move on.

8 And now we'll go to a -- we'll go to your motions, so  
9 we'll take one from each side. So we've done one from the  
10 Oracle side, and we'll do one on the JMOL side for Google.

11 And the one that I think is the one that I would like  
12 to hear the most about is your one that -- and only one part of  
13 it, and that is that your view that declarations are not  
14 copyrightable. I guess as a matter of law you're saying that.

15 So I would like -- and I would like as you make this  
16 argument, for you to be very precise on what you mean by  
17 "declaration."

18 All right. Go ahead.

19 **MR. BABER:** Yes, your Honor.

20 What we mean by the declarations in our JMOL motion  
21 is what's been referred to at trial as the method signatures  
22 for the methods and the fully qualified names, Dr. Bloch talked  
23 about.

24 **THE COURT:** All right. Would you -- do we still have  
25 that chart?

1 And you show me what you mean because the books that  
2 you put in evidence use the word "declaration," I think,  
3 differently.

4 Can you bring it a little closer? That's good right  
5 there. Thank you.

6 (Demonstrative displayed)

7 **MR. BABER:** On your last point, your Honor, I believe  
8 that's correct and I believe Professor Astrachan mentioned that  
9 in his testimony; that in the language book when it uses the  
10 phrase "declaration," it would include all the implementing  
11 code as well.

12 **THE COURT:** Correct.

13 **MR. BABER:** Okay. But the way all the witnesses  
14 testified at trial, I believe -- and it's on Dr. Bloch's  
15 chart -- this is what everyone has referred to as the  
16 declaration (indicating).

17 **THE COURT:** Certainly, he did. And maybe all of them  
18 did, but I can't tell you whether all of them did.

19 But you mean the part that's in green.

20 **MR. BABER:** Well, it's inside the black box.

21 **THE COURT:** Is that black or green?

22 **MR. BABER:** It's black.

23 **THE COURT:** Black, okay.

24 **MR. BABER:** The fully qualified name for this is  
25 what's in the green boxes. `Java.lang.Math.max`, that's the

1 fully qualified name, but this is the declaration of the  
2 methods.

3 **THE COURT:** Just so it's clear for the record, it  
4 says "public static" -- I will do it exactly.

5 Public space static space int space max. No space.  
6 Paren. Space -- no, no. Would there be a space there?

7 **MR. BABER:** No, it would be inside the paren.

8 **THE COURT:** So int space arg1 comma space int space  
9 arg2 close paren, end of declaration.

10 **MR. BABER:** Correct.

11 **THE COURT:** All right. And what is your argument  
12 there?

13 **MR. BABER:** The reason we put this in our JMOL  
14 motion, your Honor, is you already ruled on summary judgment  
15 that the package names, java.lang, the class names,  
16 java.lang.Math, and method name java.lang.Math.max are  
17 unprotectable as words and short phrases under copyright law.

18 We believe the same is true of the declarations.  
19 They are short phrases and we believe they are not separately  
20 and individually copyrightable.

21 If you look -- it's just a question of clarification  
22 as to how far the ruling you already made on short names and  
23 phrases goes. We think the length of these method signatures  
24 or declarations are similar in length to the short bit --

25 **THE COURT:** Let me give you a different argument that



1 works in your favor on that very point. And the reason I want  
2 to do this is because I'm thinking about it and I want to give  
3 Mr. Jacobs a chance to shoot it down.

4 I'm not saying this is what I'm going to rule. I'm  
5 just saying this is what I'm thinking.

6 Now, I want to take my cosign example -- well let's  
7 take this example. Let's take this example. This is just as  
8 good.

9 Java has an API. Within the API are 37-plus  
10 packages. One of those packages is java.lang. It has a class  
11 called Math. Within Math there is a method. Actually, several  
12 methods that get the maximum of two numbers. This one that we  
13 have on the board is -- uses integers, if I understand it  
14 right. So it's only for the case where you have whole numbers  
15 as opposed to fractions.

16 So here is -- that's background.

17 Now, that is a concept or function that -- there  
18 would be many ways to write the implementation. We saw four  
19 ways during the trial. You could probably come up with other  
20 ways to write it.

21 I think Oracle would concede that it cannot claim a  
22 copyright over the idea of a method that would take two numbers  
23 and return the bigger of the two numbers. Just like it could  
24 not possibly claim to have a copyright over any and all ways to  
25 take a number, an angle, and return the co-sign merely because

1 it has a copyright on its way to do that.

2           So the public, the universe, is free to come up with  
3 its own method for comparing two numbers and returning the  
4 larger, so long as it does not use the specific code developed  
5 by Java.

6           Now, that line, though, is going to have to be the  
7 same, the declaration. If you want to have that function  
8 carried out, then under the rules that the programming language  
9 imposes on the user, you've got to use the word "public" if you  
10 want that function. There's the word "private." There's like  
11 a -- you know, there are several choices on each one of those  
12 words. There's "public," "private." And then on static  
13 there's several possibilities there. I think the word "void"  
14 is one, maybe. Integer you can have "double" instead of  
15 integer.

16           The word "max" is a name. I've already ruled that  
17 the word "max" is not protectable.

18           You can vary the arg1. You can put an "x" and "y."  
19 That part can vary, right?

20           **MR. BABER:** Your Honor, absolutely correct, Your  
21 Honor.

22           **THE COURT:** All right. So the proposition I put --  
23 and this is really a question to Oracle -- isn't that one line  
24 controlled by the merger doctrine?

25           There's only one way -- if you want to have that

1 function -- and everyone has the god-given right to have that  
2 function. Oracle does not have a monopoly on it.

3           If you want to have that function, that is the only  
4 way to write it. Therefore, merger would protect the right of  
5 the public to use that form of declaration.

6           Now, I want to pause here and say, that would only  
7 help Google insofar as at the method level. That does not --  
8 that would not be an answer to, Why did it happen to be that  
9 your methods got put into the same classes?

10           You could have had exactly that same method and put  
11 it under the IO. You could have had it under the IO. You  
12 could have had it under any of those other classes or packages.  
13 But you mimicked exactly the same -- but just take the  
14 declaration level for a method. I put to you all the  
15 proposition that the merger doctrine would protect that.

16           So why don't you have a seat and let me hear what  
17 Mr. Jacobs has to say, because I think this is something I did  
18 not understand going into the trial, but I will say this is the  
19 way I'm leaning.

20           You can kind of tell from the way I'm talking I've  
21 thought about it, and this is the way I'm leaning on that one  
22 line. For every single method, this is going to be the same  
23 analysis.

24           Go ahead.

25           **MR. JACOBS:** Well, our case is not about any single

1 method or any group of methods.

2           And the wisdom in the Court's instruction was to link  
3 the protectability of names or, in this case, method  
4 declarations, as we're now using the terminology, to the  
5 structure, sequence, and organization of the software.

6           And this is the key distinction. In any copyright  
7 case, you could get very granular and you could say, well,  
8 that's an unprotectable idea. That's an unprotectable idea.

9           But, of course, the plaintiff isn't suing for copying  
10 this idea or that idea. In any copyright case of any gravity,  
11 the plaintiff is suing for some combination of elements, any  
12 one of which could be characterized at the idea level. But  
13 because of the way they are combined, they represent original  
14 expression.

15           In a music case, no note is protectable. And  
16 probably no diad of notes. But you get to five notes, six  
17 notes, seven notes, all the sudden you have protectable  
18 expression.

19           So is this case about the max method? Max is a  
20 trivial method, so it's probably an unfair example --

21           **THE COURT:** But this principle is throughout because  
22 every class has many methods. And every package has many  
23 classes.

24           So the method thing is going to be there, I'm  
25 guessing, several hundred times in the overall problem we have

1 before us.

2 And you know those cases do say that what I'm  
3 supposed to do is -- is wade through this, in excruciating  
4 pain, to find the part that is protectable and the part that's  
5 not protectable. And then with the part that is protectable,  
6 to then that's the part you ask about fair use on.

7 So, you know, now that I've heard all this evidence,  
8 that's what I'm trying to do, is to -- so I did say something  
9 close to what you just said, which is, okay, even if this is  
10 right, does that then explain away the -- does the merger  
11 doctrine explain away why it happens to be that all of those  
12 methods are lockstep found in the Google version of the math  
13 class, for example? And it does not. It does not.

14 Now, maybe something else would. For all of you this  
15 may have been already obvious. But for me it didn't start to  
16 dawn on me until I tried to understand what this -- all these  
17 words mean.

18 And I think I understand the declaration level. All  
19 right. So maybe you're agreeing with me up to the point of  
20 the -- sounds like you're agreeing with me up to the level of  
21 the method declaration.

22 **MR. JACOBS:** Well, I'm not -- I think I'm agreeing  
23 that -- what I'm not trying to do is argue over max.

24 We had in our brief some examples of method  
25 declarations that were several lines long and are not as

1 trivial as max. So my proposition to you is that max was a  
2 useful teaching device in part because it was so simple we  
3 could get it all on one page, including what Google  
4 characterizes as the implementing code below.

5 But if you examine other method declarations --

6 **THE COURT:** No, I've been looking at some of these.  
7 I agree with you. They could be many, many, many lines long.

8 But isn't this still true? Every single word in a  
9 declaration serves a functional purpose.

10 **MR. JACOBS:** Every single line of code in a --  
11 probably not. But it doesn't matter. Every single line of  
12 code, of executable code in a computer program, serves a  
13 functional purpose.

14 **THE COURT:** Yes. I didn't say it quite right.

15 You don't have the right, the ownership, of every  
16 single way to do every single method.

17 Anybody has the right to mimic -- let's say you came  
18 up with a great way -- I'll use this example. This is not so  
19 trivial. Let's say you wanted to have a method that would take  
20 the month and the day, and maybe even the time, and figure out  
21 what the declination of the earth was to the sun.

22 So you wrote a method that would have to be more than  
23 one line long. It might be, I don't know, 20 lines long. And  
24 let's say that you chose all those publics and the statistics  
25 and whichever way you wanted to do it.

1           You wouldn't have the right to say: Okay, we've now  
2 discovered how we can do this. We now own this under copyright  
3 law. No one else can come along and do the exact same  
4 specification.

5           I don't think you have the right to say that.

6           **MR. JACOBS:** But it's really -- with respect, Your  
7 Honor, it's a false hypothetical for our case.

8           **THE COURT:** Well, it helps me to -- why is that?

9           **MR. JACOBS:** Because what we have here is a case of  
10 the comprehensive taking of the entire structure, sequence and  
11 organization.

12           **THE COURT:** I can see that's a different issue.

13           **MR. JACOBS:** And what we see in the copyright  
14 cases -- I mean, look, this is copyright, and it's being  
15 applied to computer software so we're struggling with it.

16           But, nonetheless, copyright law is pretty clear on  
17 this point. If you get over-granular and say, you know, this  
18 name in the phone directory, you can't be the only one to have  
19 a phone directory over that name. That name is not  
20 protectable. But, you know, creating a business directory of  
21 phone listings is protectable.

22           This is blurring into an originality issue as opposed  
23 to a copyright --

24           **THE COURT:** I'm not saying originality. No, no, no.

25           I'm saying if you want -- it's the merger doctrine.

1 If you want to have a way to specify, this is what we're going  
2 to put in, this is what we're going to put out, and you want it  
3 to be public or private, or whatever, that is a function. And  
4 you say -- and you -- once you decide how you want it to  
5 unfold, then there is a precise way to use the declaration to  
6 say it. And the fact is, there's only one way --

7 **MR. JACOBS:** No.

8 **THE COURT:** -- it can possibly be said.

9 **MR. JACOBS:** And that's factually wrong. And it's  
10 testable.

11 **THE COURT:** I don't believe that. Okay. Explain why  
12 I'm wrong.

13 **MR. JACOBS:** It's testable. Let me start with it's  
14 testable.

15 **THE COURT:** Okay. Tell me why I'm wrong.

16 **MR. JACOBS:** Because Google could have looked at  
17 Application Programming Interfaces and method declarations.  
18 They could have said, look, there is no claim that we copied  
19 the SSO of the non-37 packages. But, look, lo and behold, when  
20 we were doing the same method purpose as was being done in  
21 Java, in our own independently-created, you know, 38th, 39th  
22 API, lo and behold, we came up with the same method  
23 declaration.

24 And there's no claim of copying there. Proof: Two  
25 programmers doing the same task would come up with the same



1 thing.

2 They never introduced any evidence like that.

3 **THE COURT:** They don't have to. I'm willing to  
4 assume they copied that part. And I'm saying to you that the  
5 law would protect them, that you don't have the right to  
6 monopolize that method.

7 **MR. JACOBS:** We don't have a right to monopolize a  
8 collection -- we don't have a right to monopolize the ability  
9 to carry out this function by monopolizing the words associated  
10 with that function. I think --

11 **THE COURT:** But those words in that box can only be  
12 said that one way.

13 **MR. JACOBS:** We already saw that the variables could  
14 be different.

15 **THE COURT:** And, in fact, they are different, aren't  
16 they, in our case? They didn't copy those; did they?

17 **MR. JACOBS:** No, no. We have lots of evidence of  
18 direct copying of variables that could be different. I think  
19 hundreds and hundreds of them.

20 **MR. KUWAYTI:** Two-thirds --

21 **MR. JACOBS:** Two-thirds of them, Your Honor. They  
22 were counted. And they are in the record.

23 But, again, this is trivial. The relevant question  
24 is that if you gave a programmer an assignment, carry out this  
25 purpose, would there be only one way to write a declaration to

1 do that? Or very few ways, such that they really all look  
2 similar?

3 That's the merger doctrine. That's experimentally  
4 provable, and Google --

5 **THE COURT:** That's true. I'm saying there's only one  
6 way to do it.

7 **MR. JACOBS:** And this --

8 **THE COURT:** That's why they have the right to have --  
9 they have the right to have their own implementation. And  
10 merely because it is -- the declaration has to be written in  
11 one way to get there, that doesn't block this like -- you're  
12 saying you've got a monopoly over all ways to do it.

13 **MR. JACOBS:** Again, all I'm saying is that's an  
14 empirical or a fact-driven question as to declarations in  
15 question, and that you'd have to have proof on the topic, not  
16 just surmise based on examination. And we have only this  
17 example (indicating) --

18 **THE COURT:** No.

19 **MR. JACOBS:** And Google never argued that this was  
20 representative.

21 **THE COURT:** I've looked at those rules and those  
22 books you gave me, and put in -- the word "public" has very  
23 precise meaning. The word "static" has a very precise meaning.  
24 The word "int" has a very precise meaning. All of it.

25 And the word max: is a name. And that's not

1 protectable. They can borrow that name all they want.

2 **MR. JACOBS:** So, then, look at the declaration at the  
3 bottom of page 12 of our brief: Public abstract void verified  
4 public key key string sig provider throw certificate exception  
5 no such algorithm exception invalid key exception no such  
6 provider exception signature exception.

7 That's a method declaration. There are probably a  
8 lot of ways to write that method declaration. If it were true  
9 that for even the majority of the method declarations in Java  
10 there were only one way to do it, Google could have proven  
11 that.

12 **THE COURT:** I think it's in the rules. I think it's  
13 in the rules of the language that if you want to have an  
14 overall method that does what that declaration specifies, that  
15 is the exact and only way to do it, with the exception of you  
16 could have said X and Y instead of arg1 and 2.

17 **MR. JACOBS:** So in the hypothetical that I gave, in  
18 the hypothetical, the real one -- this is in  
19 java.security.cert.certificate -- "public" is defined by the  
20 language. I think. I'll check. "Abstract" is defined by  
21 the --

22 **THE COURT:** That's another word that comes in the  
23 language. I've seen that in reading. Abstract is one of the  
24 keywords.

25 **MR. JACOBS:** And I believe "void" is also a language

1 construct, and maybe even verified. But --

2 **THE COURT:** It means that no -- there's no return.  
3 "Void" means there is no return.

4 **MR. JACOBS:** But then the rest of that method  
5 declaration, public key key string sig provider throw  
6 certificate, et cetera, is subject of many different ways to  
7 write it.

8 We heard a lot from Mr. Bloch about the creativity in  
9 selecting and choosing and writing and authoring the right  
10 names of these Application Program Interface constructs.

11 **THE COURT:** I've already said names -- you know, the  
12 Federal Circuit may reverse me on this, but, in my judgment,  
13 names are not protectable.

14 **MR. JACOBS:** And we --

15 **THE COURT:** No matter how long they are, they are not  
16 protectable. They can use those names all they want.

17 **MR. JACOBS:** And we are not challenging, in this  
18 argument, that conclusion. We are agreeing with your -- with  
19 your instruction that while individual names are not  
20 protectable on a standalone basis, names must necessarily be  
21 used as part of the structure, sequence and organization, and  
22 are to that extent protected by copyright. This is not a case  
23 of --

24 **THE COURT:** Well, that's what I told the jury, and  
25 that -- that's correct. That's what I told the jury.

1 But this whole thing of giving it to the jury was on  
2 the assumption -- what I was trying to do was to avoid having  
3 to retry the case, so we could just have one trial.

4 But that -- I'm not even sure that what I told the  
5 jury was actually correct as a matter of law on -- that as part  
6 of the SSO they are somehow not protected.

7 **MR. JACOBS:** So let me just recapitulate what we  
8 think the strongest argument to you on this point is.

9 Our case is not about the taking of any individual or  
10 even any small set of method declarations.

11 Our case is about the comprehensive taking of -- to  
12 use the language of the instruction -- the structure, sequence  
13 and organization of the computer programs as defined by the  
14 Application Programming Interface specifications.

15 That structure, sequence and organization includes  
16 method declarations at the appropriate level. It is like the  
17 sub sub subchapter in the outline structure.

18 The code down here (indicating), if you're a  
19 Microsoft Word person, this is body text. And this is in the  
20 outline.

21 And what we are seeking to protect is our very  
22 complex outline. It would not be a relevant question, if you  
23 were protecting a particular taxonomy, whether any particular  
24 element of the taxonomy -- whether plants from Bulgaria is  
25 protectable, the relevant question would be: Did the defendant

1 take the entire outline structure of a book on plants from  
2 around the world, in which plants from Bulgaria was one tiny  
3 fragment of what was taken?

4 No one would bother to ask the question whether  
5 plants from Bulgaria was taken because that's not our claim.

6 **THE COURT:** All right. Let me ask Mr. Baber, to  
7 respond.

8 Let's assume that the Court is right on what I said a  
9 moment ago. That still does not answer most of what Mr. Jacobs  
10 just said, in a way.

11 In other words, even if the method has to be written  
12 in the way it's put there, that method could have shown up  
13 under the input/output. It doesn't have to be even in the  
14 package or the class.

15 You could have put it anywhere you wanted, and still  
16 had the same functionality. And the problem would have been  
17 that the developer community would not have -- would not have  
18 liked that. They would have said, Why did you put max in the  
19 wrong place? That's what they might have said.

20 So the fact -- that's just a business thing. That's  
21 not required by the language itself. There is more than one  
22 way to organize the SSO in the broader -- in the package and  
23 class level.

24 So even if it's true that this declaration is --  
25 that's the only way to write the declaration, so merger

1 protects it, that doesn't answer the whole SSO problem.

2 **MR. BABER:** One step at a time, Your Honor.

3 **THE COURT:** Mr. Jacobs didn't even like me taking  
4 that step. But I'm thinking about that step.

5 But having thought about that step, I see it as just  
6 like one step on a five- to six-step process. And you still  
7 have a long way to go.

8 **MR. BABER:** Well, Your Honor, take it one step at a  
9 time. First step is, I think you're absolutely correct on that  
10 issue that the form of a method declaration is the epitome of  
11 the merger doctrine.

12 **THE COURT:** You didn't have to use arg1 and 2.

13 **MR. BABER:** No.

14 **THE COURT:** Is it true that two-thirds of the time  
15 you copied even that?

16 **MR. BABER:** I don't recall candidly, Your Honor, who  
17 testified about that or what they said. But there are some  
18 common variables that are generally used, X and Y --

19 **THE COURT:** X and Y, A and B, I and J. You know, X1,  
20 X2. But there's more than one way to write it.

21 **MR. BABER:** But just to get more granular, if what  
22 you want to have is a public method, something developers can  
23 access, that they can call on and invoke it, it's a static  
24 method that returns an integer --

25 **THE COURT:** Remind me what "static" means.

1           **MR. BABER:** I'm going to defer --

2           **THE COURT:** Mr. Baber, you're shocking me.

3           (Laughter)

4           **MR. BABER:** I'm sorry, Your Honor. I'm still  
5 learning this stuff as I go. One of our folks will have to  
6 explain exactly what "static" means, as opposed to "void" or  
7 the other words that can go in this space.

8           But if you want to have a public method that is  
9 static, that returns an integer, and it takes as its input two  
10 integers, this is the only way you can write that declaration  
11 consistent with the language specification.

12           **THE COURT:** I think I agree with that. I think I  
13 agree with that. And the only parts that you would have any  
14 flexibility on are the name and what to call two variables. I  
15 think, otherwise, it's dictated by the rules of the program.

16           **MR. BABER:** And I think that same --

17           **THE COURT:** Rules of the language.

18           **MR. BABER:** I agree, Your Honor.

19           And I think that's true no matter how simple or  
20 complicated the method is. The example Mr. Jacobs gave, again,  
21 if you want to have a public method that returns --

22           **THE COURT:** All right. All right. So how do you  
23 address the more fundamental point? Okay. Let's say you win  
24 on step one. How do you get all the way to the package level?

25           Because you do have exactly the same lineup and



1 outline and taxonomy as the -- so how do you explain that part?

2 **MR. BABER:** Your Honor, that is driven a hundred  
3 percent by the language requirement for fully qualified names.

4 You've already ruled we have the right to use the  
5 names at each level. Package name. Class name. Method name.

6 And you asked a question this morning --

7 **THE COURT:** Let's assume that's right. You could  
8 have put this -- you could have put max not under the Math  
9 class. You could have put it under a different class.

10 **MR. BABER:** You could have. But then you get -- you  
11 move from merger to a different copyrightability issue, which  
12 is functional requirements for compatibility. Which is,  
13 someone who's used to these API methods, who's used to calling  
14 max all the time, they know it's java.lang.math.max. And in  
15 order for their code that they've written in the past to work,  
16 in order for them to continue to use the API methods they've  
17 memorized for compatibility reasons -- Professor Astrachan  
18 talked about this both on the part of developers as well as on  
19 the part of the part of industry --

20 **THE COURT:** Is this a fair use argument or  
21 copyrightability issue?

22 **MR. BABER:** This is a copyrightability issue, Your  
23 Honor.

24 **THE COURT:** Where does it say that in the law, that  
25 protectability turns on this compatibility idea?

1           **MR. BABER:** *Sega vs. Accolade*. The Ninth Circuit  
2 said that, quote, functional requirements for compatibility are  
3 not protected under Section 102(b). It's an idea method system  
4 point.

5           **THE COURT:** That's -- okay. That is the -- you know,  
6 that is the big, big issue in the case, is 102(b). So you fall  
7 back on the atomic bomb, the nuclear --

8           **MR. BABER:** No, I've got some other bombs, too.

9           **THE COURT:** 102(b) is a nuclear option. That's the  
10 big issue.

11           Maybe you're right about that, but I'm searching for,  
12 is there a way to get there without getting to 102(b).

13           **MR. BABER:** Yes, you can get there on merger as to  
14 the class level as well.

15           **THE COURT:** No, not at the class level. Because you  
16 could have put that -- in more than one class, you could have  
17 put that max method in.

18           **MR. BABER:** We could have, Your Honor. But we  
19 believe under your prior rulings we have the right to use the  
20 fully qualified, name `java.lang.math.max`, which puts it in that  
21 class.

22           **THE COURT:** If that's what I ruled -- I don't think I  
23 did. I thought I held off on that.

24           But if that were true, yes, you're right, because of  
25 the rules of the -- you would have had the right to -- that's

1 the only -- that's right. I don't think I said that. I think  
2 I -- I thought I said for multi-word names we were going to  
3 hold off.

4 **MR. BABER:** Just a second, Your Honor.

5 **THE COURT:** Well --

6 **MR. BABER:** Sorry. I don't have it in this. I had  
7 it in the brief --

8 **THE COURT:** My memory could be wrong, but I thought I  
9 said for the longer names I wasn't sure. I wouldn't say you  
10 were wrong on that. I just said I wanted to have the trial  
11 first.

12 **MR. BABER:** In your summary judgment order, Your  
13 Honor, you say you find that the names of the various items  
14 appearing in the disputed API package specifications are not  
15 protected by copyright.

16 I was just trying to see --

17 **THE COURT:** I'll go back and read it and see. But  
18 the implication of your argument to say that then  
19 java.lang.math.max is protected, that that -- that destroys the  
20 SSO argument right there.

21 **MR. BABER:** I found it, Your Honor.

22 **THE COURT:** All right.

23 **MR. BABER:** Summary judgment order, page 7. You talk  
24 about the API package specifications. You say, quote: Words  
25 and short phrases such as names, titles, and slogans, unquote,

1 are, quote, not subject to copyright, unquote.

2 You cite regulation 202.1. You cite the *Planesi*  
3 Ninth Circuit opinion.

4 "Google argues that, quote, the names of the  
5 Java Language API files, packages, classes,  
6 and methods are not protectable as a matter  
7 of law." Closed quote. Cite to our brief.

8 "This order agrees."

9 Because names and others -- sorry. Lost the page.

10 "Because names and other source phrases are  
11 not subject to copyright. The names of the  
12 various items appearing in the disputed API  
13 package specifications are not protected."

14 **THE COURT:** Well, what was the part that I said --  
15 there was more to it than that. There was something I said I  
16 was going to wait until the trial was over before I decided.

17 **MR. BABER:** Yes, your Honor. That's was where you  
18 said, Well, you know, Oracle also was arguing that, well, maybe  
19 the selection and arrangement of all these names taken together  
20 could have some copyright protection. And you said, you know,  
21 that's an issue for later, but for now each of the names, the  
22 class names, method names, the package names are not protected.

23 **THE COURT:** Did I say -- well, all right.

24 **MR. JACOBS:** May I read, your Honor?

25 **THE COURT:** Yes.

1           **MR. JACOBS:** The right of completeness here, I think,  
2 applies.

3           "In finding that the names of the various  
4 items appearing in the disputed API package  
5 specifications are not protected by  
6 copyright, this order does not foreclose the  
7 possibility that the selection or arrangement  
8 of those names is subject to copyright  
9 protection. See *Lamps Plus*."

10           The parenthetical on *Lamps Plus*:

11           "A combination of unprotectable elements --  
12 italicizing unprotectable elements -- "is  
13 eligible for copyright protection only if  
14 those elements are numerous enough and their  
15 selection and arrangement original enough  
16 that the combination constitutes an original  
17 work of authorship."

18           So we are -- we were back in originality land in  
19 those days. I think we passed originality in this case by  
20 stipulation. And there is no question about whether the  
21 selection and arrangement of those names is an original work of  
22 authorship. And now we're into, okay, infringement and  
23 protectability under copyrightability doctrines.

24           **THE COURT:** Okay. We've got to bring it to a close  
25 here. It's now 3:20.

1 Is there anything on any of your other motions  
2 that -- going either way, that either side has got to have an  
3 oral argument on? If so, we will do it, but I -- some of this  
4 can be submitted on the papers.

5 **MR. BABER:** Just one, your Honor, that I'll mention  
6 very briefly, but I think we discussed it many times, so it  
7 doesn't need to be reargued.

8 On rangeCheck, where the jury found infringement  
9 based on rangeCheck and you instructed the jury that for  
10 purposes of that claim, the work as a whole was just the  
11 arrays.java file in which rangeCheck appeared.

12 We believe that the proper test for infringement  
13 always has to be the work as a whole as its registered. And if  
14 so, then the nine lines of rangeCheck code is, as a matter of  
15 law, diminimus.

16 **THE COURT:** Well, if you were right about that, yes,  
17 but I don't think you're right about -- your position is that  
18 the work is registered.

19 Now, that would be a -- I've read the cases. The  
20 cases specifically reject that proposition and say that I am  
21 supposed to identify what the work as a whole is, and it can  
22 vary from work to work. So there's policy reasons that might  
23 support your argument, but I don't think that's the law in the  
24 Ninth Circuit, so.

25 All right. Is there any other one that anyone wants

1 to really argue?

2 **MR. JACOBS:** Your Honor, could I substitute just a  
3 brief follow-up on an earlier discussion in response to the --  
4 in response?

5 **THE COURT:** Okay.

6 **MR. JACOBS:** There is one issue that I wanted to  
7 return to, just because I think the record wasn't accurately  
8 reported to you, and that's on the question -- back on the  
9 interesting question of fair use for some of the classes that  
10 were part of the language specification.

11 **THE COURT:** All right.

12 **MR. JACOBS:** So there is -- there's a bit of outlier  
13 testimony from Josh Bloch. He did the downstream packages or  
14 downstream classes that you just elicited from Mr. Baber, that  
15 that was in 10 packages. But every other witness said it was  
16 60 or 61 classes, including Dr. Astrachan, who specifically and  
17 several times agreed during cross-examination with Dr.  
18 Mitchell's examination.

19 So we're talking again about fragments in terms of  
20 the overall issue here.

21 **MR. VAN NEST:** Your Honor, I wouldn't want to put too  
22 much reliance on that. Remember, we went to the jury on 37  
23 packages as a whole. That's what we all agreed to do and  
24 that's how it went in.

25 And our evidence on fair use is certainly by no means

1 limited to the point you raised. The point you raise is a good  
2 one, and it may mean that for some of those packages the fair  
3 use argument is even better; but it doesn't mean that there is  
4 no fair use argument for the rest of them.

5 Our whole point is that when you look at all of the  
6 factors taken together, we tried the case with the 37 packages  
7 as a whole and Oracle at the end of the day withdrew any  
8 request that they go package-by-package, and that's how they  
9 went to the jury.

10 So I think the fair use case needs to be evaluated  
11 on, you know, the merits of all the evidence on all the  
12 factors, which go far, far beyond just the fact that some  
13 number of these, whatever that number is, are absolutely  
14 required just to use the language.

15 So, again, your point is a good one. It's correct.  
16 But in terms of a JMOL, what we're looking at is the verdict  
17 that the jury rendered and the question the jury answered, or  
18 didn't, which affects all the packages taken as a whole. It's  
19 the SSO of the 37 API packages, not just a few.

20 And that's -- that's the main point that I want to  
21 make on that issue on JMOL.

22 **MR. BABER:** I have one tiny clarification and a  
23 question. I promise.

24 **THE COURT:** What is this a tag team?

25 (Laughter.)



1           **MR. BABER:** No, no.

2           First, just to clarify what I just said about the  
3 rangeCheck, that issue. We also believe that even accepting  
4 your Honor's decision that the file, the individual file is the  
5 appropriate work as a whole for rangeCheck, it's nine lines out  
6 of 3,000.

7           **THE COURT:** But it gets booted up 20,000 times a  
8 second.

9           **MR. BABER:** Understand, your Honor, but that's our  
10 second level on rangeCheck.

11           The second is I just don't know whether your Honor  
12 wanted -- you told us this morning you were curious about this  
13 issue of in the source code, how I read --

14           **THE COURT:** Yes.

15           **MR. BABER:** Okay. The answer to that is, your Honor,  
16 you have exhibits in evidence and testimony from Dr. Astrachan.  
17 The answer is, no. They are not in the same order.

18           And if I can hand up --

19           **THE COURT:** Is there -- what have you got there?

20           **MR. BABER:** Well, what we have in the record, your  
21 Honor, is we have several things.

22           First, we have from Java 5.0, we have Trial Exhibit  
23 623.101 and what it is is a printout of all the source code in  
24 the Math class, java.lang.Math.

25           **THE COURT:** I think I've got that right here.

1           **MR. BABER:** And we also have Trial Exhibit 47.101,  
2 which is the same thing from Android, java.lang.Math.

3           **THE COURT:** This is just the Math one.

4           **MR. BABER:** It's just Math, your Honor, just one  
5 package.

6           **THE COURT:** How many differences are there going to  
7 be when I look at this?

8           **MR. BABER:** You're going to find a lot of  
9 differences.

10          **THE COURT:** How about in the sequence?

11          **MR. BABER:** Well, I have a chart here, your Honor,  
12 just to hand out. It's demonstrative. I will give one to  
13 Mr. Jacobs as well.

14                   (Whereupon, document was tendered  
15 to the Court and counsel.)

16          **MR. BABER:** We just printed out the names as they  
17 appear in order in the classes. And you'll see they are they  
18 are very, very different. And we can use our favorite example  
19 "max" and I can show you how that plays out.

20          **THE COURT:** So this is the sequence in which they  
21 appear?

22          **MR. BABER:** Yes, sir. It's just -- it just takes in  
23 order what's in the larger exhibit, just in order.

24          **THE COURT:** Okay. I -- all right. Go ahead and make  
25 your point.

1           **MR. BABER:** Okay. So in Java, which is exhibit  
2 623.101, Java.language.Math method appears on Page 15 of the  
3 exhibit beginning at Line 782. Where it says what we have  
4 there: Public static int max open paren int, et cetera.  
5 That's the declaration of the method. Then there's the  
6 documentation. Then there is the implementing code.

7           If we go to the Android file, 47.101, we find the max  
8 method declared on Page 11 beginning at Line 555 of the code.  
9 You'll see exactly the same thing: Public space static space,  
10 et cetera. They are just in different places within the file,  
11 although they, obviously, are the same, implementations of the  
12 same method.

13           **THE COURT:** Where you have the arguments, are these  
14 faithful to the way it appears in the code?

15           **MR. BABER:** Yes, your Honor. I think if you just  
16 line it up with -- you have the code right there with you. For  
17 example, if you look at Exhibit 623.101, you will see --

18           **THE COURT:** Where can I find double ABS in yours?

19           **MR. BABER:** I'm sorry. Where can you find what, your  
20 Honor?

21           **THE COURT:** Where can I find double ABS. You see the  
22 first one under Java is double ABS. So where is double ABS?

23           **MR. BABER:** In Java it's going to be --

24           **THE COURT:** I'm sorry. In Android.

25           **MR. BABER:** First one in Android is double ABS.

1 Double ABS is in the middle of the second page, about  
2 two-thirds of the way down in the Java version.

3 **THE COURT:** All right. So that -- that's an example  
4 where you have a "D" and they've got an "A."

5 **MR. BABER:** Exactly.

6 **THE COURT:** All right. What is -- this is just one,  
7 one class out of many.

8 How many classes there are all together? 600?

9 **MR. BABER:** 6,000 I believe -- no, I'm sorry.

10 **THE COURT:** Classes in the 37.

11 **MR. BABER:** It's thousands.

12 **THE COURT:** 6,000?

13 **MR. BABER:** I believe it is 6,000. Because that, I  
14 believe, was the testimony that in order to replicate the SSO,  
15 you would need --

16 **MR. VAN NEST:** It's 600 or 700 classes.

17 **MR. BABER:** 600 or 700 classes, 6,000 or 7,000  
18 methods sounds about right.

19 **THE COURT:** All right. Well, you were the one  
20 that -- I don't want you to do bad math here.

21 **MR. BABER:** I've done that before, your Honor. I  
22 don't want to do it again.

23 **THE COURT:** It's less than one-tenth of one percent  
24 thing.

25 **MR. BABER:** And I think what this shows, your

1 Honor -- and your question this morning shows that the  
2 hierarchical structure that we see in documentation and things  
3 like that, that's simply so humans can find things. All the  
4 computer cares about, is it in the right file. Because then  
5 the computer knows if it's in the java.lang.Math file, it can  
6 be in any order whatsoever and it can find it.

7 **THE COURT:** Let me ask you a different question.  
8 You use -- which one of these is the Java language?

9 **MR. BABER:** The Java Platform, your Honor?

10 **THE COURT:** The platform.

11 **MR. BABER:** 623.

12 **THE COURT:** All right. 623, all right.

13 Using the 623, give me an example of an interface as  
14 opposed to a method. I would like to see what one looks like  
15 in the flesh.

16 **MR. BABER:** I don't know that there are any in  
17 java.lang --

18 **THE COURT:** It's okay if there aren't any in here.

19 **MR. VAN NEST:** I have my file cabinet here, your  
20 Honor.

21 (Laughter.)

22 **MR. VAN NEST:** I know where that is in the file  
23 cabinet.

24 **THE COURT:** All right, in the file cabinet.

25 All right. How about a field then? You know, the

1 classes have fields. They have got methods. And they have  
2 interfaces, and there is another one I'm leaving out. I know  
3 what a method is. I can recognize a method.

4 I would like to be able to recognize a field when  
5 it's called out in the -- can you show me one of those?

6 **MR. BABER:** Let me back up a minute because I can  
7 tell you how to recognize, I think, an interface when it's  
8 there if you're looking at code --

9 **THE COURT:** Don't do that.

10 **MR. KWUN:** Your Honor, right in the beginning of both  
11 of these, actually, they define "Pi" and they define "E."

12 Well, so if you look in 623.101 the first page is  
13 basically documentation of the class. And then you see the  
14 author information, and then on Line 81 you see public static  
15 final double e.

16 **THE COURT:** Right.

17 **MR. KWUN:** That's defining a field, which in this  
18 case is a constant, which is used for natural logarithms, of  
19 course, 2.718 and so on.

20 And then on Line 88 you see a definition of another  
21 constant, which is --

22 **THE COURT:** So those are regarded as fields?

23 **MR. KWUN:** Yes, your Honor.

24 **THE COURT:** It's a field of one, really. That's --  
25 is that what that means?

1           **MR. KWUN:** It's a field that has the name "E," and --

2           **THE COURT:** The value --

3           **MR. KWUN:** I'm not sure which of these words, but  
4 either "static" or "final," I believe, means this cannot be  
5 changed. Once I declare it, you cannot change that field,  
6 which makes sense for a constant.

7           **THE COURT:** Okay, I've got it. Okay.

8           Now, "E" I know what that is. Natural logarithm. So  
9 if you wanted to have the field that had like an array that  
10 had, say, five numbers in it, could that work here, too? Would  
11 that be the place you would define it?

12           **MR. KWUN:** Frankly, your Honor, I don't know exactly  
13 how you would define the array, but you could do it there.

14           **MR. HWANG:** Yes, your Honor.

15           **MR. KWUN:** And, your Honor, you may remember from  
16 trial there was some testimony from Dr. Reinhold about fields  
17 and how you could have, for example, a field for a -- he would  
18 find something called -- I think for car. He said you could  
19 have a field of whether or not it was painted and what color it  
20 was. It might have been an example like that. You could have  
21 a field like that for an object, which is an object would be  
22 something you create out of a class.

23           **THE COURT:** All right. If you go further down, the  
24 very last line 104. It says: "Return StrictMath.sign." What  
25 is StrictMath?

1           **MR. KWUN:** StrictMath, your Honor, is another class.

2 So this is saying we're returning --

3           **THE COURT:** Where would we find StrictMath?

4           **MR. KWUN:** StrictMath is defined, I believe, in  
5 another file. It's not defined in here. But StrictMath --  
6 what this is saying is you're not actually returning  
7 StrictMath. You're returning the sign of a --

8           **THE COURT:** Right. I got that part. But I want to  
9 try to find StrictMath.

10           **MR. KWUN:** It's not in here.

11           **THE COURT:** It's somewhere else.

12           **MR. KWUN:** Yes, your Honor. So what this is  
13 saying --

14           **THE COURT:** So where would I find it?

15           **MR. KWUN:** You would need to look in a separate  
16 class, and since it doesn't have --

17           **THE COURT:** And that's called StrictMath?

18           **MR. KWUN:** Pardon me?

19           **THE COURT:** The class is called StrictMath?

20           **MR. KWUN:** Yes, your Honor. Generally when you see  
21 things that start with capital letters, the convention is  
22 that's a class.

23           **THE COURT:** And if it's lower -- what if it's lower  
24 case?

25           **MR. KWUN:** So the StrictMath period sign, what that's



1 saying is that the sign method that is inside the StrictMath  
2 class is being used here.

3 **THE COURT:** If you look in the StrictMath, what  
4 language is that written in?

5 **MR. KWUN:** Well, we would have to look at it to see.  
6 It could have been written in native code, but as a general  
7 proposition there would be something that would be in Java.  
8 When you went there, it might say look somewhere else yet  
9 again, which could be in another language.

10 **THE COURT:** I think if you look at it, you'll find  
11 it's in native language.

12 **MR. KWUN:** Some of these are in native code and  
13 you'll see before the method the modifier "native."

14 **THE COURT:** All right. Now you've helped me  
15 understand what a field would be. That's the "E" and the "pi."

16 So find an example of an interface that's defined  
17 here that is not a method.

18 **MR. KWUN:** Your Honor, I don't think the Math class  
19 defines any interfaces. I can give your Honor an example of  
20 what an interface is.

21 **THE COURT:** All right. Give me a simple example.

22 **MR. KWUN:** So we had this discussion now many weeks  
23 ago, but you can have the interface of compare to. And the  
24 basic --

25 **THE COURT:** Say again? Compare what?

1           **MR. KWUN:** Compare to.

2           **THE COURT:** T-o?

3           **MR. KWUN:** Yes. And the basic idea is that you, in a  
4 variety of different classes, are going to have sometimes the  
5 desire to compare two members or two -- excuse me, two objects  
6 created out of that class.

7           So there's something called an interface that says if  
8 you are going to be a comparable class, a class where you can  
9 compare two objects, what that means is that you must have  
10 within your class a method called compare to. So the interface  
11 is called comparable. And when you declare in your class that  
12 you implement comparable, what that is is that is a promise  
13 that inside your class you will have a method called compare  
14 to.

15           **THE COURT:** Why would you ever do that as opposed to  
16 just using a method?

17           **MR. KWUN:** Well, two things. When you have an  
18 interface, you still must have a method that implements that.  
19 And I -- I don't actually know what the reason is of why you  
20 have the interfaces. I just know what they are.

21           **THE COURT:** Is there another good -- I have reviewed  
22 the Math one.

23           Is there another good example like this, a  
24 side-by-side comparison that I could look at that -- in a  
25 different context that would -- I don't care what it is. Just

1 one that's about this thick (indicating) that I could look at  
2 each version to get a better feel for what's being contested?

3 **MR. JACOBS:** Well, we'd like you to look at Java.nio,  
4 your Honor, and we can make sure that you have that, those  
5 exhibits.

6 **THE COURT:** You have it right here? I will take it  
7 right now.

8 **MR. JACOBS:** No. But that would be --

9 **THE COURT:** You have it?

10 **MR. BABER:** No. I have a different one for you  
11 though.

12 **THE COURT:** What is that?

13 **MR. BABER:** I've got arrays.java, which is the  
14 class from which -- I'll just show you --

15 **THE COURT:** You gave me this one. You gave me Math.  
16 I want Mr. Jacobs to give me one he wants me to read.

17 **MR. JACOBS:** We will get it to you right away, your  
18 Honor.

19 **THE COURT:** Is it about this thick?

20 **MR. JACOBS:** I think it's might be thicker.

21 **THE COURT:** Don't give me a big thick one.

22 **MR. JACOBS:** I know. But I think we're being -- the  
23 simplicity of max and Math is distorting the analysis.

24 **THE COURT:** All right. You give my the one you want,  
25 but thicker it is, the less I can read.

1           **MR. JACOBS:** Understood, your Honor.

2           **THE COURT:** Let me just say this. I'm going to deny  
3 the motion for JMOL on fair use. And I'm not suggesting how I  
4 would come out on it if I were deciding this as the trier of  
5 fact, but I think there was enough on the way the jury was  
6 instructed that it could come out the way it did.

7           And I think if we have to have another trial on it,  
8 probably the way to do it is to figure out which pieces are  
9 protectable, which pieces are not, and then have an analysis on  
10 fair use that is limited to the parts that are protected.

11           So I hate to even contemplate the idea of another  
12 trial, but if we get there, that's the way it will have to be.

13           But I don't think it would be right to grant a Rule  
14 50 on fair use in favor of Oracle.

15           On the one about declarations are not copyrightable,  
16 I don't have to rule on that now. I think that's part of a  
17 harder project on the whole SSO project that I am working very  
18 hard on, but I don't have an answer for you.

19           On rangeCheck and whether it's diminimus, I'm not  
20 going to set the jury's verdict aside. They said it was  
21 infringing and I think the records can be construed to support  
22 that.

23           There is a couple more of these that I'm prepared to  
24 rule on. The jury said that the documentation -- there was not  
25 infringement on the documentation. I think the record supports

1 that verdict, so no Rule 50 there.

2 I want to think about the eight decompiled files. No  
3 ruling on that yet.

4 Equitable defenses, no ruling on that yet.

5 Improper registration and no ownership, I'm going to  
6 think about that as well, but I'll just say -- no ruling on  
7 that yet.

8 I think that's -- that's all the items that were on  
9 your motions.

10 **MR. VAN NEST:** Thank you, your Honor.

11 **THE COURT:** We are -- we're getting pretty close --  
12 do we have that -- do we have the jury instructions ready to  
13 give to counsel?

14 **LAW CLERK:** Yes, we have a draft, yes.

15 **THE COURT:** Right now?

16 **LAW CLERK:** It doesn't have the read-back part in it.

17 **THE COURT:** Oh, oh. I want you two to think about  
18 the read-back part.

19 Do you want me to give an instruction on the jury can  
20 ask for read-backs? Generally judges don't like read-backs,  
21 but the Ninth Circuit has a recently new pronouncement that --  
22 I, of course, salute when the Ninth Circuit speaks. I don't  
23 ask questions. I just do what they say.

24 But here is what they say, is that the ordinary rule  
25 is that you get a -- if the jury asks for a read-back, you read

1 back every word of what the witness says. So if you had a  
2 witness on the stand for two days and they wanted to hear about  
3 part of it, you do the entire two days.

4 Now, does that make sense? Of course -- I won't say  
5 that. I would say, does that make sense? You can -- a good  
6 argument is, no. It would be too long and then it would defeat  
7 the purpose and that the judge ought to have more discretion to  
8 isolate the part that really is responsive to what the jury  
9 wants. And we always have to remember, a lot of these get  
10 handed down in the context of a criminal case and there are  
11 special considerations there.

12 But the way it's always worked for many years, as far  
13 as I can tell, is that the lawyers are pretty good about  
14 agreeing on what should be read back; but the problem is if you  
15 don't agree, then we get into the problem of having to read  
16 back the entire thing.

17 I want you to think about whether we suggest -- not  
18 suggest, but we say to the jury that if they would like a  
19 read-back, they can have it, but it may take some time and so  
20 forth.

21 I need your -- I'd like to have your guidance on  
22 that. So think, be thinking. I have been thinking about it  
23 and if you can agree on language, then, of course, I would put  
24 that in.

25 But except for that, I think we have a set of jury

1 instructions ready to give you. And probably on Friday we  
2 should have the charging conference so that you can -- you can  
3 be ready to argue this on Monday.

4 **MR. VAN NEST:** We will give it some thought, your  
5 Honor. Absolutely. Thank you.

6 **THE COURT:** So how much longer do we have with the  
7 witness on the stand?

8 **MR. JACOBS:** About 20 minutes, your Honor.

9 **THE COURT:** All right. And then your cross. Then  
10 will that other missing witness be here tomorrow so that we  
11 can --

12 **MR. VAN NEST:** He will be. Mr. Bornstein is  
13 available tomorrow. I think unless we need to do it, I would  
14 just as soon finish up with Dr. Mitchell, but I need to confer  
15 with counsel on that. And then put Bornstein on and then our  
16 case.

17 **THE COURT:** What does your case look like?

18 **MR. VAN NEST:** Looks good.

19 (Laughter.)

20 **THE COURT:** How long is it going to be? How many  
21 witnesses?

22 **MR. VAN NEST:** Well, we have probably got six or  
23 seven witnesses and, but I still think what you told the jurors  
24 and then I did, too, about finishing the evidence this week is  
25 right. Assuming that we get through Dr. Mitchell and Mr.

1 Bornstein tomorrow, I think we'll get a significant part of our  
2 case in as well.

3 We have Mr. McFadden we'll be calling, a couple of  
4 Oracle folks. We have Dr. August on the '104 and Dr. Parr on  
5 the '520. And, you know, nobody is long.

6 **THE COURT:** I don't know, that sounds like we might  
7 not finish this week.

8 **MR. VAN NEST:** No, I think we will. I'm going to  
9 make every effort to do that. I would love to be able to get  
10 the evidence in this week and then do the charging conference  
11 and argue it on Monday. That is a good plan. And I think  
12 we'll try to trim our case down to accommodate it, too.

13 **THE COURT:** All right. There we go. So you're going  
14 to give me the IO, is that it, Mr. Jacobs? The IO version of  
15 these?

16 **MR. JACOBS:** Nio or something that's manageable, your  
17 Honor.

18 **THE COURT:** Great. I look forward to it. Okay.

19 **MR. VAN NEST:** Thank you, your Honor.

20 **THE COURT:** See you.

21 (Whereupon at 3:44 p.m. further proceedings  
22 in the above-entitled cause was adjourned  
23 until Thursday, May 10, 2012 at 7:30 a.m.)

24 - - - -

25



**ORDER ON MOTIONS FOR  
JUDGMENT AS A MATTER OF LAW**

**DATED MAY 10, 2012**

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25  
26  
27  
28

United States District Court  
For the Northern District of California

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

ORACLE AMERICA, INC.,

No. C 10-03561 WHA

Plaintiff,

v.

**ORDER ON MOTIONS  
FOR JUDGMENT AS A  
MATTER OF LAW**

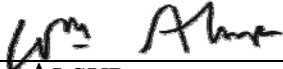
GOOGLE INC.,

Defendant.

For the reasons stated at the May 9 hearing, Oracle’s motion for judgment as a matter of law regarding fair use, API documentation, and comment-copied files is **DENIED**; Google’s motion for judgment as a matter of law regarding rangeCheck is **DENIED**.

**IT IS SO ORDERED.**

Dated: May 10, 2012.

  
\_\_\_\_\_  
WILLIAM ALSUP  
UNITED STATES DISTRICT JUDGE

**ORDER RE COPYRIGHTABILITY OF  
CERTAIN REPLICATED ELEMENTS  
OF THE JAVA APPLICATION  
PROGRAMMING INTERFACE**

**DATED MAY 31, 2012**

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25  
26  
27  
28

United States District Court  
For the Northern District of California

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

ORACLE AMERICA, INC.,

Plaintiff,

v.

GOOGLE INC.,

Defendant.

No. C 10-03561 WHA

**ORDER RE COPYRIGHTABILITY  
OF CERTAIN REPLICATED  
ELEMENTS OF THE  
JAVA APPLICATION  
PROGRAMMING INTERFACE**

**INTRODUCTION**

This action was the first of the so-called “smartphone war” cases tried to a jury. This order includes the findings of fact and conclusions of law on a central question tried simultaneously to the judge, namely the extent to which, if at all, certain replicated elements of the structure, sequence and organization of the Java application programming interface are protected by copyright.

**PROCEDURAL HISTORY**

In 2007, Google Inc., announced its Android software platform for mobile devices. In 2010, Oracle Corporation acquired Sun Microsystems, Inc., and thus acquired Sun’s interest in the popular programming language known as Java, a language used in Android. Sun was renamed Oracle America, Inc. Shortly thereafter, Oracle America (hereinafter simply “Oracle”) sued defendant Google and accused its Android platform as infringing Oracle’s Java-related copyrights and patents.

1 Both Java and Android are complex platforms. Both include “virtual machines,”  
2 development and testing kits, and application programming interfaces, also known as APIs.  
3 Oracle’s copyright claim involves 37 packages in the Java API. Copyrightability of the elements  
4 replicated is the only issue addressed by this order.

5 Due to complexity, the Court decided that the jury (and the judge) would best understand  
6 the issues if the trial was conducted in phases. The first phase covered copyrightability  
7 and copyright infringement as well as equitable defenses. The second phase covered patent  
8 infringement. The third phase would have dealt with damages but was obviated by stipulation  
9 and verdicts.

10 For the first phase, it was agreed that the judge would decide issues of copyrightability  
11 and Google’s equitable defenses and that the jury would decide infringement, fair use, and  
12 whether any copying was de minimis. Significantly, all agreed that Google had not literally  
13 copied the software but had instead come up with its own implementations of the 37 API  
14 packages. Oracle’s central claim, rather, was that Google had replicated the structure, sequence  
15 and organization of the overall code for the 37 API packages.

16 For their task of determining infringement and fair use, the jury was told it should take  
17 for granted that the structure, sequence and organization of the 37 API packages as a whole  
18 *was* copyrightable. This, however, was not a final definitive legal ruling. One reason for this  
19 instruction was so that if the judge ultimately ruled, after hearing the phase one evidence, that  
20 the structure, sequence and organization in question was not protectable but was later reversed  
21 in this regard, the court of appeals might simply reinstate the jury verdict. In this way, the court  
22 of appeals would have a wider range of alternatives without having to worry about an expensive  
23 retrial. Counsel were so informed but not the jury.

24 Each side was given seventeen hours of “air time” for phase one evidence (not counting  
25 openings, closings or motion practice). In phase one, as stated, the parties presented evidence  
26 on copyrightability, infringement, fair use, and the equitable defenses. As to the compilable  
27 code for the 37 Java API packages, the jury found that Google infringed but deadlocked on the  
28 follow-on question of whether the use was protected by fair use. As to the documentation for

1 the 37 Java API packages, the jury found no infringement. As to certain small snippets of code,  
2 the jury found only one was infringing, namely, the nine lines of code called “rangeCheck.”  
3 In phase two, the jury found no patent infringement across the board. (Those patents, it should  
4 be noted, had nothing to do with the subject addressed by this order.) The entire jury portion of  
5 the trial lasted six weeks.<sup>1</sup>

6 This order addresses and resolves the core premise of the main copyright claims, namely,  
7 whether the elements replicated by Google from the Java system were protectable by copyright  
8 in the first place. No law is directly on point. This order relies on general principles of  
9 copyright law announced by Congress, the Supreme Court and the Ninth Circuit.

10 \* \* \*

11 Counsel on both sides have supplied excellent briefing and the Court wishes to recognize  
12 their extraordinary effort and to thank counsel, including those behind the scenes burning  
13 midnight oil in law libraries, for their assistance.

14 **SUMMARY OF RULING**

15 So long as the specific code used to implement a method is different, anyone is free  
16 under the Copyright Act to write his or her own code to carry out exactly the same function  
17 or specification of any methods used in the Java API. It does not matter that the declaration or  
18 method header lines are identical. Under the rules of Java, they *must be identical* to declare a  
19 method specifying the *same* functionality — even when the implementation is different.  
20 When there is only one way to express an idea or function, then everyone is free to do so and  
21 no one can monopolize that expression. And, while the Android method and class names could  
22 have been different from the names of their counterparts in Java and still have worked, copyright  
23 protection never extends to names or short phrases as a matter of law.

24 It is true that the very same functionality could have been offered in Android  
25 without duplicating the exact command structure used in Java. This could have been done

26 \_\_\_\_\_  
27 <sup>1</sup> After the jury verdict, the Court granted Oracle’s Rule 50 motion for judgment as a matter of law of  
28 infringement of eight decompiled computer files, which were literally copied. Google admitted to copying eight  
computer files by decompiling the bytecode from eight Java files into source code and then copying the source  
code. These files were not proven to have ever been part of Android.

1 by re-arranging the various methods under different groupings among the various classes and  
 2 packages (even if the same names had been used). In this sense, there were many ways to group  
 3 the methods yet still duplicate the same range of functionality.

4 But the names are more than just names — they are symbols in a command structure  
 5 wherein the commands take the form

6 `java.package.Class.method()`

7 Each command calls into action a pre-assigned function. The overall name tree, of course, has  
 8 creative elements but it is also a precise command structure — a utilitarian and functional set  
 9 of symbols, each to carry out a pre-assigned function. This command structure is a system or  
 10 method of operation under Section 102(b) of the Copyright Act and, therefore, cannot be  
 11 copyrighted. Duplication of the command structure is necessary for interoperability.

## 12 STATEMENT OF FINDINGS

### 13 1. JAVA AND ANDROID.

14 Java was developed by Sun, first released in 1996, and has become one of the world’s  
 15 most popular programming languages and platforms.<sup>2</sup> The Java platform, through the use of a  
 16 virtual machine, enables software developers to write programs that are able to run on different  
 17 types of computer hardware without having to rewrite them for each different type. Programs  
 18 that run on the Java platform are written in the Java language. Java was developed to run on  
 19 desktop computers and enterprise servers.<sup>3</sup>

20 The Java language, like C and C++, is a human-readable language. Code written in  
 21 a human-readable language — “source code” — is not readable by computer hardware.

---

22  
 23 <sup>2</sup> For purposes of this order, the term “Java” means the Java platform, sometimes abbreviated to  
 24 “J2SE,” which includes the Java development kit (JDK), javac compiler, tools and utilities, runtime programs,  
 class libraries (API packages), and the Java virtual machine.

25 <sup>3</sup> Rather than merely vet each and every finding and conclusion proposed by the parties, this order has  
 26 navigated its own course through the evidence and arguments, although many of the proposals have found their  
 27 way into this order. Any proposal that has been expressly agreed to by the opposing side, however, shall be  
 28 deemed adopted (to the extent agreed upon) even if not expressly adopted herein. It is unnecessary for this  
 order to cite the record for all of the findings herein. In the findings, the phrase “this order finds . . .” is  
 occasionally used to emphasize a point. The absence of this phrase, however, does not mean (and should not be  
 construed to mean) that a statement is not a finding. All declarative fact statements set forth in the order are  
 factual findings.

1 Only “object code,” which is not human-readable, can be used by computers. Most object code  
2 is in a binary language, meaning it consists entirely of 0s and 1s. Thus, a computer program  
3 has to be converted, that is, compiled, from source code into object code before it can run, or  
4 “execute.” In the Java system, source code is first converted into “bytecode,” an intermediate  
5 form, before it is then converted into binary machine code by the Java virtual machine.

6 The Java language itself is composed of keywords and other symbols and a set of  
7 pre-written programs to carry out various commands, such as printing something on the screen  
8 or retrieving the cosine of an angle. The set of pre-written programs is called the application  
9 programming interface or simply API (also known as class libraries).

10 In 2008, the Java API had 166 “packages,” broken into more than six hundred “classes,”  
11 all broken into over six thousand “methods.” This is very close to saying the Java API had  
12 166 “folders” (packages), all including over six hundred pre-written programs (classes) to carry  
13 out a total of over six thousand subroutines (methods). Google replicated the exact names and  
14 exact functions of virtually all of these 37 packages but, as stated, took care to use different code  
15 to implement the six thousand-plus subroutines (methods) and six-hundred-plus classes.

16 An API is like a library. Each package is like a bookshelf in the library. Each class is  
17 like a book on the shelf. Each method is like a how-to-do-it chapter in a book. Go to the right  
18 shelf, select the right book, and open it to the chapter that covers the work you need. As to the  
19 37 packages, the Java and Android libraries are organized in the same basic way but all of the  
20 chapters in Android have been written with implementations different from Java but solving the  
21 same problems and providing the same functions. Every method and class is specified to carry  
22 out precise desired functions and, thus, the “declaration” (or “header”) line of code stating the  
23 specifications must be identical to carry out the given function.<sup>4</sup>

24 The accused product is Android, a software platform developed by Google for  
25 mobile devices. In August 2005, Google acquired Android, Inc., as part of a plan to develop  
26 a smartphone platform. Google decided to use the Java language for the Android platform.

---

27  
28 <sup>4</sup> The term “declaration” was used throughout trial to describe the headers (non-implementing code)  
for methods and classes. While “header” is the more technically accurate term, this order will remain consistent  
with the trial record and use “declaration” and “header” interchangeably.



1 In late 2005, Google began discussing with Sun the possibility of taking a license to use  
2 and to adapt the entire Java platform for mobile devices. They also discussed a possible  
3 co-development partnership deal with Sun under which Java technology would become  
4 an open-source part of the Android platform, adapted for mobile devices. Google and Sun  
5 negotiated over several months, but they were unable to reach a deal.

6 In light of its inability to reach agreement with Sun, Google decided to use the  
7 Java language to design its own virtual machine via its own software and to write its  
8 own implementations for the functions in the Java API that were key to mobile devices.  
9 Specifically, Google wrote or acquired its own source code to implement virtually all  
10 the functions of the 37 API packages in question. Significantly, all agree that these  
11 implementations — which account for 97 percent of the lines of code in the 37 API packages —  
12 are different from the Java implementations. In its final form, the Android platform also had its  
13 own virtual machine (the so-called Dalvik virtual machine), built with software code different  
14 from the code for the Java virtual machine.

15 As to the 37 packages at issue, Google believed Java application programmers would  
16 want to find the same 37 sets of functionalities in the new Android system callable by the same  
17 names as used in Java. Code already written in the Java language would, to this extent, run on  
18 Android and thus achieve a degree of interoperability.

19 The Android platform was released in 2007. The first Android phones went on sale  
20 the following year. Android-based mobile devices rapidly grew in popularity and now comprise  
21 a large share of the United States market. The Android platform is provided free of charge  
22 to smartphone manufacturers. Google receives revenue through advertisement whenever a  
23 consumer uses particular functions on an Android smartphone. For its part, Sun and Oracle  
24 never successfully developed its own smartphone platform using Java technology.

25 All agree that Google was and remains free to use the Java language itself. All agree  
26 that Google's virtual machine is free of any copyright issues. All agree that the  
27 six-thousand-plus method implementations by Google are free of copyright issues.  
28 The copyright issue, rather, is whether Google was and remains free to replicate the names,

1 organization of those names, and functionality of 37 out of 166 packages in the Java API, which  
2 has sometimes been referred to in this litigation as the “structure, sequence and organization” of  
3 the 37 packages.

4 The Android platform has its own API. It has 168 packages, 37 of which are in  
5 contention. Comparing the 37 Java and Android packages side by side, only three percent  
6 of the lines of code are the same. The identical lines are those lines that specify the names,  
7 parameters and functionality of the methods and classes, lines called “declarations” or “headers.”  
8 In particular, the Android platform replicated the same package, method and class names,  
9 definitions and parameters of the 37 Java API packages from the Java 2SE 5.0 platform.  
10 This three percent is the heart of our main copyright issue.

11 A side-by-side comparison of the 37 packages in the J2SE 5.0 version of Java versus in  
12 the Froyo version of Android shows that the former has a total of 677 classes (plus interfaces)  
13 and 6508 methods wherein the latter has 616 and 6088, respectively. Twenty-one of the  
14 packages have the same number of classes, interfaces and methods, although, as stated, the  
15 method implementations differ.

16 The three percent of source code at issue includes “declarations.” Significantly, the rules  
17 of Java dictate the precise form of certain necessary lines of code called declarations, whose  
18 precise and necessary form explains why Android and Java *must be* identical when it comes to  
19 those particular lines of code. That is, since there is only one way to declare a given method  
20 functionality, everyone using that function must write that specific line of code in the same way.  
21 The same is true for the “calls,” the commands that invoke the methods. To see why this is so,  
22 this order will now review some of the key rules for Java programming. This explanation will  
23 start at the bottom and work its way upward.

## 24 2. THE JAVA LANGUAGE AND ITS API — IMPORTANT DETAILS.

25 Java syntax includes *separators* (e.g., {, }, ;), *operators* (e.g., +, -, \*, /, <, >), *literal*  
26 *values* (e.g., 123, ‘x’, “Foo”), and *keywords* (e.g., if, else, while, return). These elements  
27 carry precise predefined meanings. Java syntax also includes *identifiers* (e.g., String,  
28

1 java.lang.Object), which are used to name specific values, fields, methods, and classes  
2 as described below.

3         These syntax elements are used to form statements, each statement being a single  
4 command executed by the Java compiler to take some action. Statements are run in the sequence  
5 written. Statements are commands that tell the computer to do work.

6         A method is like a subroutine. Once declared, it can be invoked or “called on” elsewhere  
7 in the program. When a method is called on elsewhere in the program or in an application,  
8 “arguments” are usually passed to the method as inputs. The output from the method is known  
9 as the “return.” An example is a method that receives two numbers as inputs and returns the  
10 greater of the two as an output. Another example is a method that receives an angle expressed  
11 in degrees and returns the cosine of that angle. Methods can be much more complicated.  
12 A method, for example, could receive the month and day and return the Earth’s declination to  
13 the sun for that month and day.

14         A method consists of the method header and the method body. A method header contains  
15 the name of the method; the number, order, type and name of the parameters used by the method;  
16 the type of value returned by the method; the checked exceptions that the method can throw;  
17 and various method modifiers that provide additional information about the method. At the trial,  
18 witnesses frequently referred to the method header as the “declaration.” This discrepancy has no  
19 impact on the ultimate analysis. The main point is that this header line of code introduces the  
20 method body and specifies very precisely its inputs, name and other functionality. Anyone who  
21 wishes to supply a method with the same functionality must write this line of code in the same  
22 way and must do so no matter how different the implementation may be from someone else’s  
23 implementation.

24         The method body is a block of code that then implements the method. If a method is  
25 declared to have a return type, then the method body must have a statement and the statement  
26 must include the expression to be returned when that line of code is reached. During trial, many  
27 witnesses referred to the method body as the “implementation.” It is the method body that does  
28 the heavy lifting, namely the actual work of taking the inputs, crunching them, and returning an

1 answer. The method body can be short or long. Google came up with its own implementations  
2 for the method bodies and this accounts for 97 percent of the code for the 37 packages.

3       Once the method is written, tested and in place, it can be called on to do its work.  
4 A method call is a line of code *somewhere else*, such as in a different program that calls on  
5 (or invokes) the method and specifies the arguments to be passed to the method for crunching.  
6 The method would be called on using the command format “java.package.Class.method()”  
7 where () indicates the inputs passed to the method. For example,  
8 a = java.package.Class.method() would set the field “a” to equal the return of the method called.  
9 (The words “java.package.Class.method” would in a real program be other names like  
10 “java.lang.Math.max”; “java.package.Class.method” is used here simply to explain the format.)

11       After a method, the next higher level of syntax is the class. A class usually includes  
12 fields that hold values (such as pi = 3.141592) and methods that operate on those values.  
13 Classes are a fundamental structural element in the Java language. A Java program is written as  
14 one or more classes. More than one method can be in a class and more than one class can be in a  
15 package. All code in a Java program must be placed in a class. A class declaration (or header) is  
16 a line that includes the name of the class and other information that define the class. The body of  
17 the class includes fields and methods, and other parameters.

18       Classes can have subclasses that “inherit” the functionality of the class itself. When a  
19 new subclass is defined, the declaration line uses the word “extends” to alert the compiler that  
20 the fields and methods of the parent class are inherited automatically into the new subclass so  
21 that only additional fields or methods for the subclass need to be declared.

22       The Java language does not allow a class to extend (be a subclass of) more than one  
23 parent class. This restrictiveness may be problematic when one class needs to inherit fields  
24 and methods from two different non-related classes. The Java programming language alleviates  
25 this dilemma through the use of “interfaces,” which refers to something different from the word  
26 “interface” in the API acronym. An interface is similar to a class. It can also contain methods.  
27 It is also in its own source code file. It can also be inherited by classes. The distinction is that a  
28

1 class may inherit from more than one interface whereas, as mentioned, a class can only inherit  
2 from one other class.

3 For convenience, classes and interfaces are grouped into “packages” in the same way we  
4 all group files into folders on our computers. There is no inheritance function within packages;  
5 inheritance occurs only at the class and interface level.

6 Here is a simple example of source code that illustrates methods, classes and packages.  
7 The italicized comments on the right are merely explanatory and are not compiled:

```
8  
9 package java.lang; // Declares package java.lang  
10 public class Math { // Declares class Math  
11     public static int max (int x, int y) { // Declares method max  
12         if (x > y) return x ; // Implementation, returns x or  
13         else return y ; // Implementation, returns y  
14     } // Closes method  
15 } // Closes class
```

16  
17 To invoke this method from another program (or class), the following call could be included in  
18 the program:

```
19 int a = java.lang.Math.max (2, 3);
```

20 Upon reaching this statement, the computer would go and find the max method under the Math  
21 class in the java.lang package, input “2” and “3” as arguments, and then return a “3,” which  
22 would then be set as the value of “a.”

23 The above example illustrates a point critical to our first main copyright issue, namely  
24 that the declaration line beginning “public static” is entirely dictated by the rules of the language.  
25 In order to declare a particular *functionality*, the language *demand*s that the method declaration  
26 take a particular form. There is no choice in how to express it. To be specific, that line reads:

```
27 public static int max (int x, int y) {  
28
```

1 The word “public” means that other programs can call on it. (If this instead says “private,”  
2 then it can only be accessed by other methods inside the same class.) The word “static” means  
3 that the method can be invoked without creating an instance of the class. (If this instead is an  
4 instance method, then it would always be invoked with respect to an object.) The word “int”  
5 means that an integer is returned by the method. (Other alternatives are “boolean,” “char,”  
6 and “String” which respectively mean “true/false,” “single character,” and “character string.”)  
7 Each of these three parameters is drawn from a short menu of possibilities, each possibility  
8 corresponding to a very specific functionality. The word “max” is a name and while any name  
9 (other than a reserved word) could have been used, names themselves cannot be copyrighted, as  
10 will be shown. The phrase “(int x, int y)” identifies the arguments that must be passed into the  
11 method, stating that they will be in integer form. The “x” and the “y” could be “a” and “b” or  
12 “arg1” and “arg2,” so there is a degree of creativity in naming the arguments. Again, names  
13 cannot be copyrighted. (Android did not copy all of the particular argument names used in Java  
14 but did so as to some arguments.) Finally, “{” is the beginning marker that tells the compiler  
15 that the method body is about to follow. The marker is mandatory. The foregoing description  
16 concerns the rules for the language itself. Again, each parameter choice other than the names  
17 has a precise functional choice. If someone wants to implement a particular function, the  
18 declaration specification can only be written in one way.

19 Part of the declaration of a method can list any exceptions. When a program violates  
20 the semantic constraints of the Java language, the Java virtual machine will signal this error to  
21 the program as an exception for special handling. These are specified via “throw” statements  
22 appended at the end of a declaration. Android and Java are not identical in their throw  
23 designations but they are very similar as to the 37 packages at issue.

24 A Java program must have at least one class. A typical program would have more  
25 than one method in a class. Packages are convenient folders to organize the classes.

26 This brings us to the application programming interface. When Java was first introduced  
27 in 1996, the API included eight packages of pre-written programs. At least three of these  
28 packages were “core” packages, according to Sun, fundamental to being able to use the Java

1 language at all. These packages were java.lang, java.io, and java.util. As a practical matter,  
2 anyone free to use the language itself (as Oracle concedes all are), must also use the three core  
3 packages in order to make any worthwhile use of the language. Contrary to Oracle, there is no  
4 bright line between the language and the API.

5 Each package was broken into classes and those in turn broken into methods.  
6 For example, java.lang (a package) included Math (a class) which in turn included max  
7 (a method) to return the greater of two inputs, which was (and remains) callable as  
8 java.lang.Math.max with appropriate arguments (inputs) in the precise form required  
9 (see the example above).

10 After Java's introduction in 1996, Sun and the Java Community Process, a mechanism  
11 for developing a standard specifications for Java classes and methods, wrote hundreds more  
12 programs to carry out various nifty functions and they were organized into coherent packages  
13 by Sun to become the Java application programming interface. In 2008, as stated, the Java API  
14 had grown from the original eight to 166 packages with over six hundred classes with  
15 over six thousand methods. All of it was downloadable from Sun's (now Oracle's) website  
16 and usable by anyone, including Java application developers, upon agreement to certain license  
17 restrictions. Java was particularly useful for writing programs for use via the Internet and  
18 desktop computers.

19 Although the declarations must be the same to achieve the same functionality, the names  
20 of the methods and the way in which the methods are grouped do not have to be the same.  
21 Put differently, many different API organizations could supply the same overall range of  
22 functionality. They would not, however, be interoperable. Specifically, code written for one  
23 API would not run on an API organized differently, for the name structure itself dictates the  
24 precise form of command to call up any given method.

25 To write a fresh program, a programmer names a new class and adds fields and methods.  
26 These methods can call upon the pre-written functions in the API. Instead of re-inventing the  
27 wheels in the API from scratch, programmers can call on the tried-and-true pre-packaged  
28 programs in the API. These are ready-made to perform a vast menu of functions. This is the

1 whole point of the API. For example, a student in high school can write a program that can call  
2 upon `java.lang.Math.max` to return the greater of two numbers, or to find the cosine of an angle,  
3 as one step in a larger homework assignment. Users and developers can supplement the API  
4 with their own specialized methods and classes.

5 The foregoing completes the facts necessary to decide the copyrightability issue but since  
6 Oracle has made much of two small items copied by Google, this order will now make findings  
7 thereon so that there will be proper context for the court of appeals.

### 8 **3. RANGE CHECK AND THE DE-COMPILED TEST FILES.**

9 Oracle has made much of nine lines of code that crept into both Android and Java.  
10 This circumstance is so innocuous and overblown by Oracle that the actual facts, as found  
11 herein by the judge, will be set forth below for the benefit of the court of appeals.

12 Dr. Joshua Bloch worked at Sun from August 1996 through July 2004, eventually  
13 holding the title of distinguished engineer. While working at Sun, Dr. Bloch wrote a nine-line  
14 code for a function called “rangeCheck,” which was put into a larger file, “Arrays.java,” which  
15 was part of the class library for the 37 API packages at issue. The function of rangeCheck was  
16 to check the range of a list of values before sorting the list. This was a very simple function.

17 In 2004, Dr. Bloch left Sun to work at Google, where he came to be the “chief Java  
18 architect” and “Java guru.” Around 2007, Dr. Bloch wrote the files, “Timsort.java” and  
19 “ComparableTimsort,” both of which included the same rangeCheck function he wrote while  
20 at Sun. He wrote the Timsort files in his own spare time and not as part of any Google project.  
21 He planned to contribute Timsort and ComparableTimsort back to the Java community by  
22 submitting his code to an open implementation of the Java platform, OpenJDK, which was  
23 controlled by Sun. Dr. Bloch did, in fact, contribute his Timsort file to OpenJDK and Sun  
24 included Timsort as part of its Java J2SE 5.0 release.

25 In 2009, Dr. Bloch worked on Google’s Android project for approximately one year.  
26 While working on the Android team, Dr. Bloch also contributed Timsort and  
27 ComparableTimsort to the Android platform. Thus, the nine-line rangeCheck function  
28 was copied into Google’s Android. This was how the infringement happened to occur.



1 When discovered, the rangeCheck lines were taken out of the then-current version of Android  
2 over a year ago. The rangeCheck block of code appeared in a class containing 3,179 lines of  
3 code. This was an innocent and inconsequential instance of copying in the context of a massive  
4 number of lines of code.

5 Since the remainder of this order addresses only the issue concerning structure, sequence  
6 and organization, and since rangeCheck has nothing to do with that issue, rangeCheck will not  
7 be mentioned again, but the reader will please remember that it has been readily conceded that  
8 these nine lines of code found their way into an early version of Android.

9 Google also copied eight computer files by decompiling the bytecode from eight Java  
10 files back into source code and then using the source code. These files were merely used as test  
11 files and never found their way into Android or any handset. These eight files have been treated  
12 at trial as a single unit.

13 Line by line, Oracle tested all fifteen million lines of code in Android (and all files used  
14 to test along the way leading up to the final Android) and these minor items were the only items  
15 copied, save and except for the declarations and calls which, as stated, can only be written in one  
16 way to achieve the specified functionality.

## 17 ANALYSIS AND CONCLUSIONS OF LAW

### 18 1. NAMES AND SHORT PHRASES.

19 To start with a clear-cut rule, names, titles and short phrases are not copyrightable,  
20 according to the United States Copyright Office, whose rule thereon states as follows:

21 Copyright law does not protect names, titles, or short phrases or  
22 expressions. Even if a name, title, or short phrase is novel or  
23 distinctive or lends itself to a play on words, it cannot be protected  
by copyright. The Copyright Office cannot register claims to  
exclusive rights in brief combinations of words such as:

- 24 • Names of products or services.
- 25 • Names of business organizations, or groups (including the  
26 names of performing groups).
- 27 • Pseudonyms of individuals (including pen or stage names).
- 28 • Titles of works.

- 1 • Catchwords, catchphrases, mottoes, slogans, or short  
2 advertising expressions.
- 3 • Listings of ingredients, as in recipes, labels, or formulas.  
4 When a recipe or formula is accompanied by an  
5 explanation or directions, the text directions may be  
6 copyrightable, but the recipe or formula itself remains  
7 uncopyrightable.

8 U.S. Copyright Office, Circular 34; *see* 37 C.F.R. 202.1(a).

9 This rule is followed in the Ninth Circuit. *Sega Enters., Ltd. v. Accolade, Inc.*, 977 F.2d  
10 1510, 1524 n.7 (9th Cir. 1992). This has relevance to Oracle’s claim of copyright ownership  
11 over names of methods, classes and packages.

## 12 2. THE DEVELOPMENT OF LAW ON THE COPYRIGHTABILITY 13 OF COMPUTER PROGRAMS AND THEIR STRUCTURE, 14 SEQUENCE AND ORGANIZATION.

15 Turning now to the more difficult question, this trial showcases a distinction between  
16 copyright protection and patent protection. It is an important distinction, for copyright  
17 exclusivity lasts 95 years whereas patent exclusivity lasts twenty years. And, the Patent and  
18 Trademark Office examines applications for anticipation and obviousness before allowance  
19 whereas the Copyright Office does not. This distinction looms large where, as here, the vast  
20 majority of the code was *not* copied and the copyright owner must resort to alleging that the  
21 accused stole the “structure, sequence and organization” of the work. This phrase — structure,  
22 sequence and organization — does not appear in the Act or its legislative history. It is a phrase  
23 that crept into use to describe a residual property right where literal copying was absent.  
24 A question then arises whether the copyright holder is more appropriately asserting an exclusive  
25 right to a functional system, process, or method of operation that belongs in the realm of patents,  
26 not copyrights.

### 27 A. *Baker v. Seldon.*

28 The general question predates computers. In the Supreme Court’s decision in *Baker v.*  
*Seldon*, 101 U.S. 99 (1879), the work at issue was a book on a new system of double-entry  
bookkeeping. It included blank forms, consisting of ruled lines, and headings, illustrating the

1 system. The accused infringer copied the method of bookkeeping but used different forms.

2 The Supreme Court framed the issue as follows:

3 The evidence of the complainant is principally directed to the  
4 object of showing that Baker uses the same system as that which is  
5 explained and illustrated in Selden's books. It becomes important,  
6 therefore, to determine whether, in obtaining the copyright of his  
7 books, he secured the exclusive right to the use of the system or  
8 method of book-keeping which the said books are intended to  
9 illustrate and explain.

7 *Id.* at 101. *Baker* held that using the same accounting system would not constitute copyright  
8 infringement. The Supreme Court explained that only patent law can give an exclusive right to  
9 a method:

10 To give to the author of the book an exclusive property in the art  
11 described therein, when no examination of its novelty has ever  
12 been officially made, would be a surprise and a fraud upon the  
13 public. That is the province of letters-patent, not of copyright.  
14 The claim to an invention or discovery of an art or manufacture  
15 must be subjected to the examination of the Patent Office before  
16 an exclusive right therein can be obtained; and it can only be  
17 secured by a patent from the government.

14 *Id.* at 102. The Supreme Court went on to explain that protecting the method under copyright  
15 law would frustrate the very purpose of publication:

17 The copyright of a work on mathematical science cannot give to  
18 the author an exclusive right to the methods of operation which he  
19 propounds, or to the diagrams which he employs to explain them,  
20 so as to prevent an engineer from using them whenever occasion  
21 requires. The very object of publishing a book on science or the  
22 useful arts is to communicate to the world the useful knowledge  
23 which it contains. But this object would be frustrated if the  
24 knowledge could not be used without incurring the guilt of piracy  
25 of the book.

21 *Id.* at 103. *Baker* also established the “merger” doctrine for systems and methods intermingled  
22 with the texts or diagrams illustrating them:

23 And where the art it teaches cannot be used without employing the  
24 methods and diagrams used to illustrate the book, or such as are  
25 similar to them, such methods and diagrams are to be considered  
26 as necessary incidents to the art, and given therewith to the public;  
27 not given for the purpose of publication in other works explanatory  
28 of the art, but for the purpose of practical application.

27 *Ibid.* It is true that *Baker* is aged but it is not passé. To the contrary, even in our modern era,  
28 *Baker* continues to be followed in the appellate courts, as will be seen below.

**B. The Computer Age and Section 102(b) of the 1976 Act.**

Almost a century later, Congress revamped the Copyright Act in 1976. By then, software for computers was just emerging as a copyright issue. Congress decided in the 1976 Act that computer programs would be copyrightable as “literary works.” *See* H.R. REP. NO. 94-1476, at 54 (1976). There was, however, no express definition of a computer program until an amendment in 1980.

The 1976 Act also codified a *Baker*-like limitation on the scope of copyright protection in Section 102(b). *See Apple Computer, Inc. v. Microsoft Corp.*, 35 F.3d 1435, 1443 n.11 (9th Cir. 1994). Section 102(b) stated (and still states):

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

The House Report that accompanied Section 102(b) of the Copyright Act explained:

Copyright does not preclude others from using the ideas or information revealed by the author’s work. It pertains to the literary, musical, graphic, or artistic form in which the author expressed intellectual concepts. Section 102(b) makes clear that copyright protection does not extend to any idea, procedure, process, system, method of operation, concept, principle, or discovery, regardless of the form in which it is described, explained, illustrated, or embodied in such work.

*Some concern has been expressed lest copyright in computer programs should extend protection to the methodology or processes adopted by the programmer, rather than merely to the ‘writing’ expressing his ideas. Section 102(b) is intended, among other things, to make clear that the expression adopted by the programmer is the copyrightable element in a computer program, and that the actual processes or methods embodied in the program are not within the scope of the copyright law.*

Section 102(b) in no way enlarges or contracts the scope of copyright protection under the present law. Its purpose is to restate, in the context of the new single Federal system of copyright, that the basic dichotomy between expression and idea remains unchanged.

1 H.R. REP. NO. 94-1476, at 56–57 (1976) (emphasis added).<sup>5</sup>

2 Recognizing that computer programs posed novel copyright issues, Congress established  
3 the National Commission on New Technological Uses of Copyrighted Works (referred to as  
4 CONTU) to recommend the extent of copyright protection for software. The Commission  
5 consisted of twelve members with Judge Stanley Fuld as chairman and Professor Melville  
6 Nimmer as vice-chairman.

7 The Commission recommended that a definition of “computer program” be added to the  
8 copyright statutes. This definition was adopted in 1980 and remains in the current statute:

9 A “computer program” is a set of statements or instructions to be  
10 used directly or indirectly in a computer in order to bring about a  
certain result.

11 17 U.S.C. 101. Moreover, the CONTU report stated that Section 102(b)’s preclusion of  
12 copyright protection for “procedure, process, system, method of operation” was reconcilable  
13 with the new definition of “computer program.” The Commission explained the dichotomy  
14 between copyrightability and non-copyrightability as follows:

15 Copyright, therefore, protects the program so long as it remains  
16 fixed in a tangible medium of expression but does not protect the  
17 electromechanical functioning of a machine. The way copyright  
18 affects games and game-playing is closely analogous: one may not  
adopt and republish or redistribute copyrighted game rules, but the  
copyright owner has no power to prevent others from playing the  
game.

19 *Thus, one is always free to make a machine perform any*  
20 *conceivable process (in the absence of a patent), but one is not free*  
*to take another’s program.*

21 NAT’L COMM’N ON NEW TECHNOLOGICAL USES OF COPYRIGHTED WORKS, FINAL REPORT 20  
22 (1979) (emphasis added). The Commission also recognized the “merger” doctrine, a rule of  
23 importance a few pages below in this order (emphasis added):

24 The “idea-expression identity” exception provides that copyrighted  
25 language may be copied without infringing when there is but a  
26 limited number of ways to express a given idea. This rule is the  
logical extension of the fundamental principle that copyright  
cannot protect ideas. *In the computer context this means that when*

27  
28 <sup>5</sup> The Court has reviewed the entire legislative history. The quoted material above is the only passage  
of relevance. This order includes a summary of the CONTU report but it came after-the-fact and had little  
impact on the Act other than to include a definition of “computer program.”

1           *specific instructions, even though previously copyrighted, are the*  
 2           *only and essential means of accomplishing a given task, their later*  
 3           *use by another will not amount to an infringement . . . .*

4           [C]opyright protection for programs does not threaten to block the  
 5           use of ideas or program language previously developed by others  
 6           when that use is necessary to achieve a certain result. When other  
 7           language is available, programmers are free to read copyrighted  
 8           programs and use the ideas embodied in them in preparing their  
 9           own works.

10          *Ibid.* The Commission realized that differentiating between the copyrightable form of a program  
 11          and the uncopyrightable process was difficult, and expressly decided to leave the line drawing to  
 12          federal courts:

13                 [T]he many ways in which programs are now used and the new  
 14                 applications which advancing technology will supply may make  
 15                 drawing the line of demarcation more and more difficult.  
 16                 To attempt to establish such a line in this report written in 1978  
 17                 would be futile. . . . Should a line need to be drawn to exclude  
 18                 certain manifestations of programs from copyright, that line should  
 19                 be drawn on a case-by-case basis by the institution designed to  
 20                 make fine distinctions — the federal judiciary.

21          *Id.* at 22–23.

22                 Congress prepared no legislative reports discussing the CONTU comments regarding  
 23                 Section 102(b). *See* H.R. REP. NO. 96-1307, at 23–24 (1980). Nevertheless, Congress followed  
 24                 CONTU’s recommendations by adding the definition of computer programs to the statute and  
 25                 amending a section of the Act not relevant to this order. *See Apple Computer, Inc. v. Formula*  
 26                 *Intern. Inc.*, 725 F.2d 521, 522–25 (9th Cir. 1984).

27                 Everyone agrees that no one can copy line-for-line someone else’s copyrighted computer  
 28                 program. When the line-by-line listings are different, however, some copyright owners have  
 29                 nonetheless accused others of stealing the “structure, sequence and organization” of the  
 30                 copyrighted work. That is the claim here.

### 31                 **C. Decisions Outside the Ninth Circuit.**

32                 No court of appeals has addressed the copyrightability of APIs, much less their  
 33                 structure, sequence and organization. Nor has any district court. Nevertheless, a review of the  
 34                 case law regarding non-literal copying of software provides guidance. Circuit decisions outside  
 35                 the Ninth Circuit will be considered first.

1 The Third Circuit led off in *Whelan Associates, Inc. v. Jaslow Dental Laboratory, Inc.*,  
 2 797 F.2d 1222 (3d Cir. 1986). In that case, the claimant owned a program, Dentalab, that  
 3 handled the administrative and bookkeeping tasks of dental prosthetics businesses. The accused  
 4 infringer developed another program, Dentcom, using a different programming language.  
 5 The Dentcom program handled the same tasks as the Dentalab program and had the following  
 6 similarities:

7 The programs were similar in three significant respects . . . most  
 8 of the file structures, and the screen outputs, of the programs  
 9 were virtually identical . . . five particularly important  
 10 “subroutines” within both programs — order entry, invoicing,  
 11 accounts receivable, end of day procedure, and end of month  
 12 procedure — performed almost identically in both programs.

13 *Id.* at 1228. On these facts, the district court had found, after a bench trial, that the accused  
 14 infringer copied the claimant’s software program. *Id.* at 1228–29.

15 On appeal, the accused infringer argued that the structure of the claimant’s program was  
 16 not protectable under copyright. In rejecting this argument, the court of appeals created the  
 17 following framework to deal with non-literal copying of software:

18 [T]he line between idea and expression may be drawn with  
 19 reference to the end sought to be achieved by the work in question.  
 20 In other words, *the purpose or function of a utilitarian work would  
 21 be the work’s idea, and everything that is not necessary to that  
 22 purpose or function would be part of the expression of the idea.*

23 *Id.* at 1236 (emphasis in original). Applying this test, *Whelan* found that the structure of  
 24 Dentalab was copyrightable because there were many different ways to structure a program that  
 25 managed a dental laboratory:

26 [T]he idea of the Dentalab program was the efficient management  
 27 of a dental laboratory (which presumably has significantly  
 28 different requirements from those of other businesses). Because  
 that idea could be accomplished in a number of different ways with  
 a number of different structures, the structure of the Dentalab  
 program is part of the program’s expression, not its idea.

*Id.* at 1236 n.28. The phrase “structure, sequence and organization” originated in a passage in  
*Whelan* explaining that the opinion used those words interchangeably and that, although not  
 themselves part of the Act, they were intended to capture the thought that “sequence and order  
 could be parts of the expression, not the idea, of a work.” *Id.* at 1239, 1248.

1 To summarize, in affirming the district court's final judgment of infringement, *Whelan*  
2 held that the *structure* of the Dentalab program was copyrightable because there were many  
3 other ways to perform the same function of handling the administrative and bookkeeping tasks  
4 of dental prosthetics businesses with different structures and designs. *Id.* at 1238. Others were  
5 free to come up with their own version but could not appropriate the Dentalab structure.  
6 This decision plainly seems to have been the high-water mark of copyright protection for the  
7 structure, sequence and organization of computer programs. It was also the only appellate  
8 decision found by the undersigned judge that affirmed (or directed) a final judgment of  
9 copyrightability on a structure, sequence and organization theory.

10 Perhaps because it was the first appellate decision to wade into this problem, *Whelan*  
11 has since been criticized by subsequent treatises, articles, and courts, including our own court  
12 of appeals. *See Sega Enters., Ltd. v. Accolade, Inc.*, 977 F.2d 1510, 1524–25 (9th Cir. 1992).  
13 Instead, most circuits, including ours, have adopted some variation of an approach taken later  
14 by the Second Circuit. *See Apple Computer, Inc. v. Microsoft Corp.*, 35 F.3d 1435, 1445  
15 (9th Cir. 1994).

16 In *Computer Associates International, Inc. v. Altai*, 982 F.2d 693 (2d Cir. 1992),  
17 the claimant owned a program designed to translate the language of another program into  
18 the particular language that the computer's operating system would be able to understand.  
19 The accused infringer developed its own program with substantially similar structure but  
20 different source code (using the same programming language). The Second Circuit criticized  
21 *Whelan* for taking too narrow a view of the "idea" of a program. The Second Circuit adopted  
22 instead an "abstract-filtration-comparison" test. The test first dissected the copyrighted program  
23 into its structural components:

24 In ascertaining substantial similarity under [the  
25 abstract-filtration-comparison test], a court would first break down  
26 the allegedly infringed program into its constituent structural parts.  
27 Then, by examining each of these parts for such things as  
28 incorporated ideas, expression that is necessarily incidental to  
those ideas, and elements that are taken from the public domain, a  
court would then be able to sift out all non-protectable material.

*Id.* at 706.



1 Then, the test filtered out structures that were not copyrightable. For this filtration step,  
2 the court of appeals relied on the premise that programmers fashioned structures “to maximize  
3 the program’s speed, efficiency, as well as simplicity for user operation, while taking into  
4 consideration certain externalities such as the memory constraints of the computer upon which  
5 the program will be run.” *Id.* at 698. Because these were “practical considerations,” the court  
6 held that structures based on these considerations were not copyrightable expressions.

7 Thus, for the filtration step, the court of appeals outlined three types of structures that  
8 should be precluded from copyright protection. *First*, copyright protection did not extend to  
9 structures dictated by efficiency. A court must inquire

10 whether the use of *this particular set* of modules [is] necessary  
11 efficiently to implement that part of the program’s process being  
12 implemented. If the answer is yes, then the expression represented  
13 by the programmer’s choice of a specific module or group of  
14 modules has merged with their underlying idea and is unprotected.

15 *Id.* at 708 (emphasis in original). Paradoxically, this meant that non-efficient structures might be  
16 copyrightable while efficient structures may not be. Nevertheless, the Second Circuit explained  
17 its reasoning as follows:

18 In the context of computer program design, the concept of  
19 efficiency is akin to deriving the most concise logical proof or  
20 formulating the most succinct mathematical computation.  
21 Thus, the more efficient a set of modules are, the more closely  
22 they approximate the idea or process embodied in that particular  
23 aspect of the program’s structure.

24 While, hypothetically, there might be a myriad of ways in  
25 which a programmer may effectuate certain functions within  
26 a program — *i.e.*, express the idea embodied in a given  
27 subroutine — efficiency concerns may so narrow the practical  
28 range of choice as to make only one or two forms of expression  
workable options.

*Ibid.* Efficiency also encompassed user simplicity and ease of use. *Id.* at 708–09.

*Second*, copyright protection did not extend to structures dictated by external factors.

The court explained this as follows:

[I]n many instances it is virtually impossible to write a program  
to perform particular functions in a specific computing  
environment without employing standard techniques. This is a  
result of the fact that a programmer’s freedom of design choice  
is often circumscribed by extrinsic considerations such as (1) the  
mechanical specifications of the computer on which a particular

1 program is intended to run; (2) compatibility requirements of  
2 other programs with which a program is designed to operate in  
3 conjunction; (3) computer manufacturers' design standards;  
(4) demands of the industry being serviced; and (5) widely  
accepted programming practices within the computer industry.

4 *Id.* at 709–10.

5 *Third*, copyright protection did not extend to structures already found in the public  
6 domain. The court reasoned that materials in the public domain, such as elements of a computer  
7 program that have been freely accessible, cannot be appropriated. *Ibid.* Ultimately, in the case  
8 before it, the Second Circuit held that after removing unprotectable elements using the criteria  
9 discussed above, only a few lists and macros in accused product were similar to the copied  
10 product, and their impact on the program was not large enough to declare copyright  
11 infringement. *Id.* at 714–15. The copyright claim, in short, failed.

12 The Tenth Circuit elaborated on the abstract-filtration-comparison test in *Gates Rubber*  
13 *Co. v. Bando Chemical Industries, Ltd.*, 9 F.3d 823 (10th Cir. 1993). There, the claimant  
14 developed a computer program that determined the proper rubber belt for a particular machine  
15 by performing complicated calculations involving numerous variables. The program used  
16 published formulas in conjunction with certain mathematical constants developed by the  
17 claimant to determine belt size. The Tenth Circuit offered the following description of a  
18 software program's structure:

19 The program's architecture or structure is a description of how  
20 the program operates in terms of its various functions, which are  
performed by discrete modules, and how each of these modules  
21 interact with each other.

22 *Id.* at 835. As had the Second Circuit, the Tenth Circuit held that filtration should eliminate the  
23 unprotectable elements of processes, facts, public domain information, merger material, *scenes a*  
24 *faire* material, and other unprotectable elements suggested by the particular facts of the program  
25 under examination. For Section 102(b) processes, the court gave the following description:

26 Returning then to our levels of abstraction framework, we note  
27 that processes can be found at any level, except perhaps the main  
28 purpose level of abstraction. Most commonly, processes will be  
found as part of the system architecture, as operations within  
modules, or as algorithms.

1 *Id.* at 837. The court described the *scenes a faire* doctrine for computer programs as follows:

2 The *scenes a faire* doctrine also excludes from protection those  
 3 elements of a program that have been dictated by external factors.  
 4 In the area of computer programs these external factors may  
 5 include: hardware standards and mechanical specifications,  
 6 software standards and compatibility requirements, *Sega*  
*Enterprises Ltd. v. Accolade, Inc.*, 977 F.2d 1510, 1525–27  
 (9th Cir. 1993), computer manufacturer design standards, target  
 industry practices and demands, and computer industry  
 programming practices.

7 \* \* \*

8 We recognize that the *scenes a faire* doctrine may implicate the  
 9 protectability of interfacing and that this topic is very sensitive and  
 10 has the potential to effect [sic] widely the law of computer  
 11 copyright. This appeal does not require us to determine the scope  
 of the *scenes a faire* doctrine as it relates to interfacing and  
 accordingly we refrain from discussing the issue.

12 *Id.* at 838 & n.14 (all citations omitted except *Sega*). Like the Second Circuit, the Tenth Circuit  
 13 also listed many external considerations — such as compatibility, computer industry  
 14 programming practices, and target industry practices and demands — that would exclude  
 15 elements from copyright protection under the *scenes a faire* doctrine. Ultimately, the  
 16 Tenth Circuit remanded because the district court had failed to make specific findings  
 17 that fit this framework.

18 The First Circuit weighed in with its 1995 decision *Lotus Development Corp. v. Borland*  
 19 *International, Inc.*, 49 F.3d 807 (1st Cir. 1995). In *Lotus*, the claimant owned the Lotus 1-2-3  
 20 spreadsheet program that enabled users to perform accounting functions electronically on  
 21 a computer. Users manipulated and controlled the program via a series of menu commands,  
 22 such as “Copy,” “Print,” and “Quit.” In all, Lotus 1-2-3 had 469 commands arranged into more  
 23 than 50 menus and submenus. Lotus 1-2-3 also allowed users to write “macros,” whereby a user  
 24 could designate a series of command choices (sequence of menus and submenus) with a single  
 25 macro keystroke. Then, to execute that series of commands, the user only needed to type the  
 26 single pre-programmed macro keystroke, causing the program to recall and perform the  
 27 designated series of commands automatically. *Id.* at 809–10.

28 The accused infringer Borland developed a competing spreadsheet program.  
 Borland included the Lotus menu command hierarchy in its program to make it compatible

1 with Lotus 1-2-3 so that spreadsheet users who were already familiar with Lotus 1-2-3 would  
2 be able to switch to the Borland program without having to learn new commands or rewrite  
3 their Lotus macros. In so doing, Borland did not copy any of Lotus's underlying source or  
4 object code. (The opinion did not say whether the programs were written in the same language.)

5 The district court had ruled that the Lotus 1-2-3 menu command hierarchy was a  
6 copyrightable expression because there were many ways to construct a spreadsheet menu tree.  
7 Thus, the district court had concluded that the Lotus developers' choice and arrangement of  
8 command terms, reflected in the Lotus menu command hierarchy, constituted copyrightable  
9 expression. *Id.* at 810–11.

10 The First Circuit, however, held that the Lotus menu command hierarchy was not  
11 copyrightable because it was a method of operation under Section 102(b). The court explained:

12 We think that “method of operation,” as that term is used in  
13 § 102(b), refers to the means by which a person operates  
14 something, whether it be a car, a food processor, or a computer.  
15 Thus a text describing how to operate something would not extend  
16 copyright protection to the method of operation itself; other people  
17 would be free to employ that method and to describe it in their  
18 own words. Similarly, if a new method of operation is used rather  
19 than described, other people would still be free to employ or  
20 describe that method.

21 *Id.* at 815.

22 The court reasoned that because the menu command hierarchy was essential to make use  
23 of the program's functional capabilities, it should be properly categorized as a “method of  
24 operation” under Section 102(b). The court explained:

25 The Lotus menu command hierarchy does not merely explain and  
26 present Lotus 1-2-3's functional capabilities to the user; it also  
27 serves as the method by which the program is operated and  
28 controlled . . . . In other words, to offer the same capabilities as  
29 Lotus 1-2-3, Borland did not have to copy Lotus's underlying code  
30 (and indeed it did not); to allow users to operate its programs in  
31 substantially the same way, however, Borland had to copy the  
32 Lotus menu command hierarchy. Thus the Lotus 1-2-3 code is not  
33 a uncopyrightable “method of operation.”

34 *Ibid.* Thus, the court reasoned that although Lotus had made “expressive” choices of what  
35 to name the command terms and how to structure their hierarchy, it was nevertheless an

1 uncopyrightable “method of operation.” The *Lotus* decision was affirmed by an evenly divided  
2 Supreme Court (four to four).

3 The Federal Circuit had the opportunity to apply *Lotus* in an appeal originating from  
4 the District of Massachusetts in *Hutchins v. Zoll Medical Corp.*, 492 F.3d 1377 (Fed. Cir. 2007)  
5 (affirming summary judgment against copyright owner). In *Hutchins*, the claimant owned a  
6 program for performing CPR and argued that his copyright covered the “system of logic  
7 whereby CPR instructions are provided by computerized display, and [] the unique logic  
8 contained in [his] software program.” *Id.* at 1384. The claimant argued that the accused  
9 program was similar because it “perform[ed] the same task in the same way, that is, by  
10 measuring heart activity and signaling the quantity and timing of CPR compressions to be  
11 performed by the rescuer.” *Ibid.* The court of appeals rejected this argument, holding that  
12 copyright did not protect the “technologic method of treating victims by using CPR and  
13 instructing how to use CPR.” *Ibid.* (citing *Lotus*).

#### 14 **D. Decisions in the Supreme Court and in our Circuit.**

15 Our case is governed by the law in the Ninth Circuit and, of course, the Supreme Court.  
16 The Supreme Court missed the opportunity to address these issues in *Lotus* due to the  
17 four-to-four affirmance and has, thus, never reached the general question. Nonetheless, *Baker*,  
18 which is still good law, provides guidance and informs how we should read Section 102(b).

19 Another Supreme Court decision, *Feist Publications, Inc. v. Rural Telephone Services*  
20 *Co., Inc.*, 499 U.S. 340 (1991), which dealt primarily with the copyrightability of purely factual  
21 compilations, provided some general principles. In *Feist*, the Supreme Court considered the  
22 copyrightability of a telephone directory comprised of names, addresses, and phone numbers  
23 organized in alphabetical order. The Supreme Court rejected the notion that copyright law was  
24 meant to reward authors for the “sweat of the brow.” This meant that we should not yield to the  
25 temptation to award copyright protection merely because a lot of sweat went into the work.  
26 The Supreme Court concluded that protection only extended to the original components of an  
27 author’s work. *Id.* at 353. The Supreme Court concluded:

28 This inevitably means that the copyright in a factual compilation  
is thin. Notwithstanding a valid copyright, a subsequent compiler

1 remains free to use the facts contained in another's publication to  
2 aid in preparing a competing work, so long as the competing work  
does not feature the same selection and arrangement.

3 *Id.* at 349.

4 Turning to our own Ninth Circuit, our court of appeals has recognized that non-literal  
5 components of a program, including the structure, sequence and organization and user interface,  
6 can be protectable under copyright depending on whether the structure, sequence and  
7 organization in question qualifies as an expression of an idea rather than an idea itself.

8 *Johnson Controls, Inc. v. Phoenix Control Sys., Inc.*, 886 F.2d 1173, 1175 (9th Cir. 1989).

9 This decision arrived between the Third Circuit's *Whelan* decision and the Second Circuit's  
10 *Computer Associates* decision. *Johnson Controls* is one of Oracle's mainstays herein.

11 In *Johnson Controls*, the claimant developed a system of computer programs to  
12 control wastewater treatment plants. The district court found that the structure, sequence and  
13 organization of the program was expression and granted a preliminary injunction even though  
14 the accused product did not have similar source or object code. *Id.* at 1174. Therefore, the  
15 standard of review on appeal was limited to abuse of discretion and clear error. Our court  
16 of appeals affirmed the preliminary injunction, stating that the claimant's program was very  
17 sophisticated and each individual application was customized to the needs of the purchaser,  
18 indicating there may have been room for individualized expression in the accomplishment  
19 of common functions. Since there was some discretion and opportunity for creativity in the  
20 structure, the structure of the program was expression rather than an idea. *Id.* at 1175.

21 *Johnson Controls*, however, did not elaborate on which particular structures deserved copyright  
22 protection.

23 In *Brown Bag Software v. Symantec Corp.*, 960 F.2d 1465 (9th Cir. 1992), our court  
24 of appeals outlined a two-part test for determining similarity between computer programs:  
25 the extrinsic and intrinsic tests. This pertained to infringement, not copyrightability.  
26 The claimant, who owned a computer program for outlining, alleged that an accused infringer  
27 copied his program's non-literal features. *Id.* at 1472. The claimant alleged that seventeen  
28

1 specific features in the programs were similar. On summary judgment, the district court had  
2 found that each feature was either not protectable or not similar as a matter of law:

3 The district court ruled that one group of features represented a  
4 claim of copyright in “concepts . . . fundamental to a host of  
5 computer programs” such as “the need to access existing files,  
6 edit the work, and print the work.” As such, these features, which  
7 took the form of four options in the programs’ opening menus,  
8 were held to be unprotectable under copyright.

9 A second group of features involved “nine functions listed in  
10 the menu bar” and the fact that “virtually all of the functions of  
11 the PC-Outline program [ ] can be performed by Grandview.”  
12 The district court declared that “these functions constitute the idea  
13 of the outlining program” and, furthermore, “[t]he expression of  
14 the ideas inherent in the features are . . . distinct.” The court also  
15 held that “the similarity of using the main editing screen to enter  
16 and edit data . . . is essential to the very idea of a computer  
17 outlining program.”

18 The third group of features common to PC-Outline and Grandview  
19 concerned “the use of pull-down windows.” Regarding these  
20 features, the district court made three separate rulings. The court  
21 first found that “[p]laintiffs may not claim copyright protection of  
22 an . . . expression that is, if not standard, then commonplace in the  
23 computer software industry” . . . [and] that the pull-down  
24 windows of the two programs look different.

25 *Id.* at 1472–73. Our court of appeals affirmed the district court’s order without elaborating on  
26 the copyrightability rulings quoted above.

27 In *Atari Games Corp. v. Nintendo of America Inc.*, 975 F.2d 832 (Fed. Cir. 1992),  
28 the Federal Circuit had occasion to interpret Ninth Circuit copyright precedent. In *Atari*, the  
claimant Nintendo sued Atari for copying the Nintendo 10NES program, which prevented the  
Nintendo game console from accepting unauthorized game cartridges. Atari deciphered the  
10NES program through reverse engineering and developed its own program to unlock the  
Nintendo game console. Atari’s new program generated signals indistinguishable from 10NES  
but was written in a different programming language. *Id.* at 835–36.

Applying our Ninth Circuit precedents, *Johnson Controls* and *Brown Bag*, the Federal  
Circuit affirmed the district court’s preliminary injunction for copyright infringement.

1 The Federal Circuit held that the 10NES program contained copyrightable expression because  
2 it had organization and sequencing unnecessary to the unlocking function:

3 Nintendo's 10NES program contains more than an idea or  
4 expression necessarily incident to an idea. Nintendo incorporated  
5 within the 10NES program creative organization and sequencing  
6 *unnecessary* to the lock and key function. Nintendo chose  
7 arbitrary programming instructions and arranged them in a unique  
8 sequence to create a purely arbitrary data stream. This data stream  
9 serves as the key to unlock the NES. Nintendo may protect this  
10 creative element of the 10NES under copyright.

11 *Id.* at 840 (emphasis added). The Federal Circuit stated that there were creative elements in the  
12 10NES program

13 beyond the literal expression used to effect the unlocking process.  
14 The district court defined the unprotectable 10NES idea or process  
15 as the generation of a data stream to unlock a console. This court  
16 discerns no clear error in the district court's conclusion.  
17 The unique arrangement of computer program expression which  
18 generates that data stream does not merge with the process so long  
19 as alternate expressions are available. In this case, Nintendo has  
20 produced expert testimony showing a multitude of different ways  
21 to generate a data stream which unlocks the NES console.

22 *Ibid.* (citation omitted). Thus, the Federal Circuit held that the district court did not err in  
23 concluding that the 10NES program contained protectable expression and affirmed the  
24 preliminary injunction.

25 Next came two decisions holding that Section 102(b) bars from copyright software  
26 interfaces necessary for interoperability. The Section 102(b) holdings arose in the context of  
27 larger holdings that it had been fair use to copy software to reverse-engineer it so as to isolate  
28 the unprotectable segments. These two decisions will now be described in detail.

29 In *Sega Enterprises Ltd. v. Accolade, Inc.*, 977 F.2d 1510 (9th Cir. 1992), the accused  
30 infringer had to copy object code in order to understand the interface procedures between the  
31 Sega game console and a game cartridge, that is, how the software in the game console  
32 interacted with the software in the game cartridge to achieve compatibility. *Id.* at 1515–16.  
33 After learning and documenting these interactions (interface procedures), the accused infringer  
34 wrote its own source code to mimic those same interface procedures in its own game cartridges  
35 so that its cartridges could run on the Sega console. Our court of appeals held that the copying  
36 of object code for the purpose of achieving compatibility was fair use. Notably, in its fair-use



1 analysis, our court of appeals *expressly held that the interface procedures for compatibility were*  
2 *functional aspects not copyrightable under Section 102(b)*: “Accolade copied Sega’s software  
3 solely in order to discover the functional requirements for compatibility with the Genesis console  
4 — aspects of Sega’s programs that are not protected by copyright. 17 U.S.C. § 102(b).” *Id.* at  
5 1522. The court used the phrase “interface procedures,” a term describing the interface between  
6 applications, multiple times to describe the functional aspect of the interaction between software  
7 programs and summarized its analysis of copyrightability as follows:

8 In summary, the record clearly establishes that disassembly of the  
9 object code in Sega’s video game cartridges was necessary in  
10 order to understand the functional requirements for Genesis  
11 compatibility. The *interface procedures* for the Genesis console  
12 are distributed for public use only in object code form, and are  
13 not visible to the user during operation of the video game program.  
14 Because object code cannot be read by humans, it must be  
15 disassembled, either by hand or by machine. Disassembly of  
16 object code necessarily entails copying. Those facts dictate our  
17 analysis of the second statutory fair use factor. If disassembly of  
18 copyrighted object code is per se an unfair use, the owner of the  
19 copyright gains a de facto monopoly over *the functional aspects*  
20 *of his work — aspects that were expressly denied copyright*  
21 *protection by Congress.* 17 U.S.C. § 102(b). In order to enjoy a  
22 lawful monopoly over the idea or functional principle underlying a  
23 work, the creator of the work must satisfy the more stringent  
24 standards imposed by the patent laws. *Bonito Boats, Inc. v.*  
25 *Thunder Craft Boats, Inc.*, 489 U.S. 141, 159–64, 109 S.Ct. 971,  
26 982–84, 103 L.Ed.2d 118 (1989). Sega does not hold a patent on  
27 the Genesis console.

18 *Sega*, 977 F.2d at 1526 (emphasis added). In *Sega*, the interface procedure that was required for  
19 compatibility was “20 bytes of initialization code plus the letters S–E–G–A.” *Id.* at 1524 n.7.

20 Our court of appeals found that this interface procedure was functional and therefore not  
21 copyrightable under Section 102(b). The accused infringer Accolade was free to copy this  
22 interface procedure for use in its own games to ensure compatibility with the Sega Genesis game  
23 console. Our court of appeals distinguished the *Atari* decision, where the Federal Circuit had  
24 found that the Nintendo’s 10NES security system was infringed, because there was only one  
25 signal that unlocked the Sega console, unlike the “multitude of different ways to unlock” the  
26 Nintendo console:

27 We therefore reject Sega’s belated suggestion that Accolade’s  
28 incorporation of the code which “unlocks” the Genesis III console  
is not a fair use. Our decision on this point is entirely consistent

1 with *Atari v. Nintendo*, 975 F.2d 832 (Fed. Cir.1992). Although  
2 *Nintendo* extended copyright protection to Nintendo's 10NES  
3 security system, that system consisted of an original program  
4 which generates an arbitrary data stream "key" which unlocks the  
5 NES console. Creativity and originality went into the design of  
6 that program. *See id.* at 840. Moreover, the federal circuit  
7 concluded that there is a "multitude of different ways to generate a  
8 data stream which unlocks the NES console." *Atari*, 975 F.2d at  
9 839. The circumstances are clearly different here. Sega's key  
10 appears to be functional. It consists merely of 20 bytes of  
11 initialization code plus the letters S-E-G-A. There is no showing  
12 that there is a multitude of different ways to unlock the Genesis III  
13 console.

14 *Sega*, 977 F.2d at 1524 n.7.

15 This order reads *Sega* footnote seven (quoted above) as drawing a line between copying  
16 functional aspects necessary for compatibility (not copyrightable) versus copying functional  
17 aspects unnecessary for compatibility (possibly copyrightable). Our court of appeals explained  
18 that in *Atari*, the Nintendo game console's 10NES program had had functionality *unnecessary* to  
19 the lock-and-key function. *See also Atari*, 975 F.2d at 840. Since the accused infringer Atari  
20 had copied the entire 10NES program, it also had copied aspects of the 10NES program  
21 unnecessary for compatibility between the console and game cartridges. This was inapposite to  
22 the facts of *Sega*, where the accused infringer Accolade's final product duplicated *only* the  
23 aspect of Sega's program *necessary* for compatibility between the console and game cartridges.  
24 Thus, the holding of our court of appeals was that the aspect of a program necessary for  
25 compatibility was unprotectable, specifically invoking Section 102(b), but copyrightable  
26 expression could still exist for aspects unnecessary for compatibility.

27 The *Sega* decision and its compatibility reasoning was followed in a subsequent  
28 reverse-engineering decision by our court of appeals, *Sony Computer Entertainment, Inc., v.*  
*Connectix Corporation*, 203 F.3d 596 (9th Cir. 2000). The facts were somewhat different in  
*Sony*. There, the accused infringer Connectix did not create its own games for Sony's  
Playstation game console; instead, the accused infringer created an emulated environment that  
duplicated the interface procedures of Sony's console so that games written for Sony's console  
could be played on a desktop computer running the emulator. In order to do this, the accused  
infringer copied object code for the Sony Playstation's operating software, its BIOS program, in

1 order to discover signals sent between the BIOS and the rest of the game console. *Id.* at 600.  
 2 After uncovering these signals (again, application interfaces), the accused infringer wrote its own  
 3 source code to *duplicate these interfaces* in order to create its emulator for the desktop computer.  
 4 Thus, games written for the Playstation console were playable on Connectix’s emulator for  
 5 the desktop computer. Citing Section 102(b) and *Sega*, our court of appeals stated that the  
 6 Playstation BIOS contained “unprotected functional elements,” and concluded that the accused  
 7 infringer’s intermediate step of copying object code was fair use because it was done for the  
 8 “purpose of gaining access to the unprotected elements of Sony’s software.” *Id.* at 602–03.<sup>6</sup>

9 \* \* \*

10 With apology for its length, the above summary of the development of the law reveals a  
 11 trajectory in which enthusiasm for protection of “structure, sequence and organization” peaked  
 12 in the 1980s, most notably in the Third Circuit’s *Whelan* decision. That phrase has not been  
 13 re-used by the Ninth Circuit since *Johnson Controls* in 1989, a decision affirming preliminary  
 14 injunction. Since then, the trend of the copyright decisions has been more cautious. This trend  
 15 has been driven by fidelity to Section 102(b) and recognition of the danger of conferring a  
 16 monopoly by copyright over what Congress expressly warned should be conferred only by  
 17 patent. This is not to say that infringement of the structure, sequence and organization is a dead  
 18 letter. To the contrary, it is not a dead letter. It is to say that the *Whelan* approach has given way  
 19 to the *Computer Associates* approach, including in our own circuit. *See Sega Enters., Ltd. v.*  
 20 *Accolade, Inc.*, 977 F.2d 1510, 1525 (9th Cir. 1992); *Apple Computer, Inc. v. Microsoft Corp.*,  
 21 35 F.3d 1435, 1445 (9th Cir. 1994).

22 In this connection, since the CONTU report was issued in 1980, the number of software  
 23 patents in force in the United States has dramatically increased from barely a thousand in  
 24 1980 to hundreds of thousands today. *See* Iain Cockburn, *Patents, Tickets and the Financing*

---

25  
 26 <sup>6</sup> *Sega* and *Sony* are not the only Ninth Circuit decisions placing a premium on functionality as  
 27 indicating uncopyrightability. Other such decisions were surveyed in the summary earlier in this order. *See*  
 28 *also Triad Sys. Corp. v. Southeastern Exp. Co.*, 64 F.3d 1330, 1336 (9th Cir. 1995); *Apple Computer, Inc. v.*  
*Microsoft Corp.*, 35 F.3d 1435, 1444 (9th Cir. 1994); *Apple Computer, Inc. v. Formula Intern., Inc.*, 725 F.2d  
 521, 525 (9th Cir. 1984).

1 of *Early-Stage Firms: Evidence from the Software Industry*, 18 JOURNAL OF ECONOMICS &  
 2 MANAGEMENT STRATEGY 729–73 (2009). This has caused at least one noted commentator to  
 3 observe:

4 As software patents gain increasingly broad protection, whatever  
 5 reasons there once were for broad copyright protection of  
 6 computer programs disappear. Much of what has been considered  
 7 the copyrightable “structure, sequence and organization” of a  
 8 computer program will become a mere incident to the patentable  
 9 idea of the program or of one of its potentially patentable  
 10 subroutines.

11 Mark Lemley, *Convergence in the Law of Software Copyright?*, 10 HIGH TECHNOLOGY LAW  
 12 JOURNAL 1, 26–27 (1995). Both Oracle and Sun have applied for and received patents that claim  
 13 aspects of the Java API. *See, e.g.*, U.S. Patents 6,598,093 and 7,006,855. (These were not  
 14 asserted at trial.)<sup>7</sup>

15 \* \* \*

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

- 18 • Under the merger doctrine, when there is only one (or only a few)  
 19 ways to express something, then no one can claim ownership of  
 20 such expression by copyright.
- 21 • Under the names doctrine, names and short phrases are not  
 22 copyrightable.
- 23 • Under Section 102(b), copyright protection never extends to any  
 24 idea, procedure, process, system, method of operation or concept

25 <sup>7</sup> The issue has been debated in the journals. For example, Professor Pamela Samuelson has argued  
 26 that Section 102(b) codified the *Baker* exclusion of procedures, processes, systems, and methods of operation  
 27 for computer programs as well as the pre-*Baker* exclusion of high-level abstractions such as ideas, concepts, and  
 28 principles. Pamela Samuelson, *Why Copyright Law Excludes Systems and Processes from the Scope of  
 Protection*, 85 TEX. L. REV. 1921 (2007). In contrast, Professor David Nimmer (the son of Professor Melville  
 Nimmer) has argued that Section 102(b) should not deny copyright protection to “the expression” of a work  
 even if that work happens to consist of an idea, procedure or process. 1-2 NIMMER ON COPYRIGHT § 2.03[D]  
 (internal citations omitted). Similarly, Professor Jane Ginsburg has argued that the Section 102(b) terms  
 “process,” “system,” and “method of operation” should not be understood literally for computer programs. Jane  
 Ginsburg, *Four Reasons and a Paradox: The Manifest Superiority of Copyright Over Sui Generis Protection of  
 Computer Software*, 94 COLUM. L. REV. 2559, 2569–70 (1994).

1 regardless of its form. Functional elements essential for  
2 interoperability are not copyrightable.

- 3 • Under *Feist*, we should not yield to the temptation to find  
4 copyrightability merely to reward an investment made in a body of  
5 intellectual property.

### 6 APPLICATION OF CONTROLLING LAW TO CONTROLLING FACTS

7 All agree that everyone was and remains free to program in the Java language itself.  
8 All agree that Google was free to use the Java language to write its own API. While Google  
9 took care to provide fresh line-by-line implementations (the 97 percent), it generally replicated  
10 the overall name organization and functionality of 37 packages in the Java API (the  
11 three percent). The main issue addressed herein is whether this violated the Copyright Act and  
12 more fundamentally whether the replicated elements were copyrightable in the first place.

13 This leads to the first holding central to this order and it concerns the method level.  
14 The reader will remember that a method is like a subroutine and over six thousand are in play  
15 in this proceeding. As long as the specific code written to implement a method is different,  
16 anyone is free under the Copyright Act to write his or her own method to carry out exactly the  
17 same function or specification of any and all methods used in the Java API. Contrary to Oracle,  
18 copyright law does not confer ownership over any and all ways to implement a function or  
19 specification, no matter how creative the copyrighted implementation or specification may be.  
20 The Act confers ownership only over the specific way in which the author wrote out his version.  
21 Others are free to write their own implementation to accomplish the identical function, for,  
22 importantly, ideas, concepts and functions cannot be monopolized by copyright.

23 To return to our example, one method in the Java API carries out the function of  
24 comparing two numbers and returning the greater. Google — and everyone else in the world —  
25 was and remains free to write its own code to carry out the identical function so long as the  
26 implementing code in the method body is different from the copyrighted implementation. This is  
27 a simple example, but even if a method resembles higher mathematics, everyone is still free to  
28 try their hand at writing a different implementation, meaning that they are free to use the same

1 inputs to derive the same outputs (while throwing the same exceptions) so long as the  
2 implementation in between is their own. The House Report, quoted above, stated in 1976 that  
3 “the actual processes or methods embodied in the program are not within the scope of the  
4 copyright law.” H.R. REP. NO. 94-1476, at 57 (1976).

5 Much of Oracle’s evidence at trial went to show that the design of methods in an API  
6 was a creative endeavor. Of course, that is true. Inventing a new method to deliver a new output  
7 can be creative, even inventive, including the choices of inputs needed and outputs returned.  
8 The same is true for classes. But such inventions — at the concept and functionality level —  
9 are protectable only under the Patent Act. The Patent and Trademark Office examines such  
10 inventions for validity and if the patent is allowed, it lasts for twenty years. Based on a single  
11 implementation, Oracle would bypass this entire patent scheme and claim ownership over any  
12 and all ways to carry out methods for 95 years — without any vetting by the Copyright Office  
13 of the type required for patents. This order holds that, under the Copyright Act, no matter  
14 how creative or imaginative a Java method specification may be, the entire world is entitled  
15 to use the same method specification (inputs, outputs, parameters) so long as the line-by-line  
16 implementations are different. To repeat the Second Circuit’s phrasing, “there might be  
17 a myriad of ways in which a programmer may . . . express the idea embodied in a given  
18 subroutine.” *Computer Associates*, 982 F.2d at 708. The method specification is the *idea*.  
19 The method implementation is the *expression*. No one may monopolize the *idea*.<sup>8</sup>

20 To carry out any given function, the method specification as set forth in the declaration  
21 *must be identical* under the Java rules (save only for the choices of argument names). Any other  
22 declaration would carry out some *other* function. The declaration requires precision.  
23 Significantly, when there is only one way to write something, the merger doctrine bars anyone  
24 from claiming exclusive copyright ownership of that expression. Therefore, there can be no  
25 copyright violation in using the identical declarations. Nor can there be any copyright violation

---

27 <sup>8</sup> Each method has a singular purpose or function, and so, the basic function or purpose of a method  
28 will be an unprotectable process. *Gates Rubber Co. v. Bando Chemical Industries, Ltd.*, 9 F.3d 823, 836  
(10th Cir. 1993); see *Apple Computer, Inc. v. Formula Intern. Inc.*, 725 F.2d 521, 525 (9th Cir. 1984) (holding  
that while a particular set of instructions is copyrightable, the underlying computer process is not).

1 due to the *name* given to the method (or to the arguments), for under the law, names and short  
2 phrases cannot be copyrighted.

3 In sum, Google and the public were and remain free to write their own implementations  
4 to carry out exactly the same functions of all methods in question, using exactly the same method  
5 specifications and names. Therefore, at the method level — the level where the heavy lifting is  
6 done — Google has violated no copyright, it being undisputed that Google’s implementations  
7 are different.

8 As for classes, the rules of the language likewise insist on giving names to classes and  
9 the rules insist on strict syntax and punctuation in the lines of code that declare a class. As with  
10 methods, for any desired functionality, the declaration line will *always* read the same (otherwise  
11 the functionality would be different) — save only for the name, which cannot be claimed  
12 by copyright. Therefore, under the law, the declaration line cannot be protected by copyright.  
13 This analysis is parallel to the analysis for methods. This now accounts for virtually all of the  
14 three percent of similar code.

15 \* \* \*

16 Even so, the second major copyright question is whether Google was and remains free to  
17 group its methods in the same way as in Java, that is, to organize its Android methods under the  
18 same class and package scheme as in Java. For example, the Math classes in both systems have  
19 a method that returns a cosine, another method that returns the larger of two numbers, and yet  
20 another method that returns logarithmic values, and so on. As Oracle notes, the rules of Java  
21 did not insist that these methods be grouped together in any particular class. Google could have  
22 placed its trigonometric function (or any other function) under a class other than Math class.  
23 Oracle is entirely correct that the rules of the Java language did not require that the same  
24 grouping pattern (or even that they be grouped at all, for each method could have been placed  
25 in a stand-alone class).<sup>9</sup>

---

27 <sup>9</sup> As to the groupings of methods within a class, Google invokes the *scenes a faire* doctrine. That is,  
28 Google contends that the groupings would be so expected and customary as to be permissible under the *scenes a faire* doctrine. For example, the methods included under the Math class are typical of what one would expect to see in a group of math methods. Just as one would expect certain items in the alcove for nuts, bolts and screws

1 Oracle's best argument, therefore, is that while no single name is copyrightable, Java's  
2 overall system of organized names — covering 37 packages, with over six hundred classes, with  
3 over six thousand methods — is a “taxonomy” and, therefore, copyrightable under *American*  
4 *Dental Association v. Delta Dental Plans Association*, 126 F.3d 977 (7th Cir. 1997). There was  
5 nothing in the rules of the Java language that required that Google replicate the same groupings  
6 even if Google was free to replicate the same functionality.<sup>10</sup>

7 The main answer to this argument is that while the overall scheme of file name  
8 organization resembles a taxonomy, it is *also* a command structure for a system or method  
9 of operation of the application programming interface. The commands are (and must be) in  
10 the form

11 `java.package.Class.method()`

12 and each calls into action a pre-assigned function.<sup>11</sup>

13 To repeat, Section 102(b) states that “in no case does copyright protection for an original  
14 work of authorship extend to any idea, procedure, process, system, method of  
15 operation . . . regardless of the form . . . .” That a system or method of operation has thousands  
16 of commands arranged in a creative taxonomy does not change its character as a method of  
17 operation. Yes, it is creative. Yes, it is original. Yes, it resembles a taxonomy. But it is  
18 nevertheless a command structure, a system or method of operation — a long hierarchy of

19 \_\_\_\_\_  
20 in a hardware store, one would expect the methods of the math class to be in, say, a typical math class. At trial,  
21 however, neither side presented evidence from which we can now say that the same is true for all the other  
22 hundreds of classes at issue. Therefore, it is impossible to say on this record that *all* of the classes and their  
23 contents are typical of such classes and, on this record, this order rejects Google's global argument based on  
24 *scenes a faire*.

25 <sup>10</sup> This is a good place to point out that while the groupings appear to be the same, when we drill down  
26 into the detail code listings, we see that the actual sequences of methods in the listings are different. That is, the  
27 sequence of methods in the class Math in Android is different from the sequence in the same class in Java,  
28 although all of the methods in the Java version can be found somewhere in the Android version, at least as  
shown in their respective listings (TX 47.101, TX 623.101). The Court has not compared all six-hundred-plus  
classes. Nor has any witness or counsel so far on the record. Oracle does not, however, contend that the actual  
sequences would track method-for-method and it has not so proven. This detailed observation, however, does  
not change the fact that all of the methods in the Java version can be found somewhere in the Android version,  
classified under the same classes.

<sup>11</sup> The parentheses indicate that inputs/arguments may be included in the command.



1 over six thousand commands to carry out pre-assigned functions. For that reason, it cannot  
2 receive copyright protection — patent protection perhaps — but not copyright protection.

3 \* \* \*

4 Interoperability sheds further light on the character of the command structure as a system  
5 or method of operation. Surely, millions of lines of code had been written in Java before  
6 Android arrived. These programs necessarily used the `java.package.Class.method()` command  
7 format. These programs called on all or some of the specific 37 packages at issue and  
8 necessarily used the command structure of names at issue. Such code was owned by  
9 the developers themselves, not by Oracle. *In order for at least some of this code to run on*  
10 *Android, Google was required to provide the same `java.package.Class.method()` command*  
11 *system using the same names with the same “taxonomy” and with the same functional*  
12 *specifications.* Google replicated what was necessary to achieve a degree of interoperability —  
13 but no more, taking care, as said before, to provide its own implementations.

14 That interoperability is at the heart of the command structure is illustrated by Oracle’s  
15 preoccupation with what it calls “fragmentation,” meaning the problem of having imperfect  
16 interoperability among platforms. When this occurs, Java-based applications may not run  
17 on the incompatible platforms. For example, Java-based code using the replicated parts of the  
18 37 API packages will run on Android but will not if a 38th package is needed. Such imperfect  
19 interoperability leads to a “fragmentation” — a Balkanization — of platforms, a circumstance  
20 which Sun and Oracle have tried to curb via their licensing programs. In this litigation, Oracle  
21 has made much of this problem, at times almost leaving the impression that if only Google had  
22 replicated *all* 166 Java API packages, Oracle would not have sued. While fragmentation is a  
23 legitimate business consideration, it begs the question whether or not a license was required in  
24 the first place to replicate some or all of the command structure. (This is especially so inasmuch  
25 as Android has not carried the Java trademark, and Google has not held out Android as fully  
26 compatible.) The immediate point is this: fragmentation, imperfect interoperability, and  
27 Oracle’s angst over it illustrate the character of the command structure as a functional system or  
28 method of operation.

1 In this regard, the Ninth Circuit decisions in *Sega* and *Sony*, although not on all fours, are  
 2 close analogies. Under these two decisions, interface procedures required for interoperability  
 3 were deemed “functional requirements for compatibility” and were not copyrightable under  
 4 Section 102(b). Both decisions held that interface procedures that were necessary to duplicate in  
 5 order to achieve interoperability were functional aspects not copyrightable under Section 102(b).  
 6 Here, the command structure for the 37 packages (including inheritances and exception throws),  
 7 when replicated, at least allows interoperability of code using the replicated commands. To the  
 8 extent of the 37 packages — which, after all, is the extent of Oracle’s copyright claim — *Sega*  
 9 and *Sony* are analogous. Put differently, if someone could duplicate the interfaces of the Sony  
 10 BIOS in order to run the Playstation games on desktops (taking care to write its own  
 11 implementations), then Google was free to duplicate the command structure for the 37 packages  
 12 in Android in order to accommodate third-party source code relying on the 37 packages (taking  
 13 care to write its own implementations). Contrary to Oracle, “full compatibility” is not relevant  
 14 to the Section 102(b) analysis. In *Sony*, the accused product implemented only 137 of the  
 15 Playstation BIOS’s 242 functions because those were the only functions invoked by the games  
 16 tested. Connectix’s Opening Appellate Brief at 18, available at 1999 WL 33623860, (9th Cir.  
 17 May 27, 1999). Our court of appeals held that the accused product “itself infringe[d] no  
 18 copyright.” *Sony*, 203 F.3d at 608 n.11. This parallels Google’s decision to implement some  
 19 but not all of the Java API packages in Android.

20 \* \* \*

21 This explains why *American Dental Association v. Delta Dental Plans Association*,  
 22 126 F.3d 977 (7th Cir. 1997), is not controlling. Assuming arguendo that a taxonomy is  
 23 protectable by copyright in our circuit, see *Practice Mgmt. Info. Corp. v. Am. Med. Ass’n*,  
 24 121 F.3d 516 (9th Cir. 1997), the taxonomy in *ADA* had nothing to do with computer programs.  
 25 It was not a system of commands, much less a system of commands for a computer language.  
 26 The taxonomy there subdivided the universe of all dental procedures into an outline of numbered  
 27 categories with English-language descriptions created by the ADA. This was then to be used  
 28 by insurance companies and dentists to facilitate billings. By contrast, here the taxonomy is

1 composed entirely of a system of commands to carry out specified computer functions. For a  
2 similar reason, Oracle’s analogy to stealing the plot and character from a movie is inapt, for  
3 movies involve no “system” or “method of operation” — scripts are entirely creative.

4 In *ADA*, Judge Frank Easterbrook (writing for the panel) suggested that a “system” under  
5 Section 102(b) had to come with “instructions for use.” 126 F.3d at 980. Because the taxonomy  
6 there at issue had no instructions for use, among other reasons, it was held not to be a system.  
7 By contrast, the API at issue here does come with instructions for use, namely, the  
8 documentation and embedded comments that were much litigated at trial. They describe every  
9 package, class and method, what inputs they need, and what outputs they return — the classic  
10 form of instructions for use.

11 In our circuit, the structure, sequence and organization of a computer program may (or  
12 may not) qualify as a protectable element depending on the “particular facts of each case” and  
13 always subject to exclusion of unprotectable elements. *Johnson Controls v. Phoenix Control*  
14 *Sys.*, 886 F.2d 1173, 1175 (9th Cir. 1989). Contrary to Oracle, *Johnson Controls* did not hold  
15 that all structure, sequence and organization in all computer programs are within the protection  
16 of a copyright. On a motion for preliminary injunction, the district court found that the structure,  
17 sequence and organization of the copyrighted program, on the facts there found, deserved  
18 copyright protection. (The structure, sequence and organization features found protectable were  
19 not described in the appellate decision.) On an appeal from the preliminary injunction, our court  
20 of appeals merely said no clear error had occurred. Again, the appellate opinion stated that the  
21 extent to which the structure, sequence and organization was protectable depended on the facts  
22 and circumstances of each case. The circumstances there are not the circumstances here.

23 In closing, it is important to step back and take in the breadth of Oracle’s claim. Of the  
24 166 Java packages, 129 were not violated in any way. Of the 37 accused, 97 percent of the  
25 Android lines were new from Google and the remaining three percent were freely replicable  
26 under the merger and names doctrines. Oracle must resort, therefore, to claiming that it owns,  
27 by copyright, the exclusive right to any and all possible implementations of the taxonomy-like  
28 command structure for the 166 packages and/or any subpart thereof — even though it

1 copyrighted only one implementation. To accept Oracle's claim would be to allow anyone  
2 to copyright one version of code to carry out a system of commands and thereby bar all others  
3 from writing their own different versions to carry out all or part of the same commands.  
4 No holding has ever endorsed such a sweeping proposition.

5 **CONCLUSION**

6 This order does not hold that Java API packages are free for all to use without license.  
7 It does not hold that the structure, sequence and organization of all computer programs may be  
8 stolen. Rather, it holds on the specific facts of this case, the particular elements replicated by  
9 Google were free for all to use under the Copyright Act. Therefore, Oracle's claim based on  
10 Google's copying of the 37 API packages, including their structure, sequence and organization  
11 is **DISMISSED**. To the extent stated herein, Google's Rule 50 motions regarding copyrightability  
12 are **GRANTED** (Dkt. Nos. 984, 1007). Google's motion for a new trial on copyright infringement  
13 is **DENIED AS MOOT** (Dkt. No. 1105).

14  
15 **IT IS SO ORDERED.**

16  
17 Dated: May 31, 2012.

18   
\_\_\_\_\_  
19 WILLIAM ALSUP  
20 UNITED STATES DISTRICT JUDGE  
21  
22  
23  
24  
25  
26  
27  
28

**FINAL JUDGMENT**

**DATED JUNE 20, 2012**

United States District Court  
For the Northern District of California

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25  
26  
27  
28

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

ORACLE AMERICA, INC.,

Plaintiff,

No. C 10-03561 WHA

v.

GOOGLE INC.,

Defendant.

**FINAL JUDGMENT**

The pleadings in this action asserted the following: Oracle asserted infringement of seven patents, U.S. Patent Nos. 6,125,447; 6,192,476; 5,966,702; 7,426,720; RE38,104; 6,910,205; and 6,061,520. Oracle further asserted infringement of its copyrights in the code, documentation, specifications, libraries, and other materials that comprise the Java platform. Oracle alleged that the infringed elements included Java method and class names, definitions, organization, and parameters; the structure, organization and content of Java class libraries; and the content and organization of Java’s documentation. In turn, Google asserted declaratory judgments of non-infringement and invalidity, and equitable defenses. Before trial, Oracle dismissed with prejudice all claims for relief based on the ’447, ’476, ’702, ’720, and ’205 patents. During trial, Google abandoned claims for relief for invalidity declarations as to the ’104 and ’520 patents.

United States District Court  
For the Northern District of California

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25  
26  
27  
28

Based upon the verdicts by the jury and orders entered by the Court, it is now  
**ORDERED, ADJUDGED, AND DECREED** that:


With respect to Oracle’s claim for relief and Google’s counterclaim for declaratory judgment of non-infringement for the ’520 and ’104 patents, judgment is entered for Google and against Oracle. With respect to Google’s counterclaims for declaratory judgment of invalidity for the ’520 and ’104 patents, judgment is entered for Oracle and against Google, such counterclaims having been abandoned during trial. With respect to the five remaining patents, claims for relief by Oracle were completely dismissed with prejudice by Oracle (and may not be resurrected except as indicated in the orders of May 3, 2011, and March 2, 2012, with respect to new products). In this regard, it is the intent of this judgment and order that general principles of merger of claims into the judgment and res judicata shall be applicable.

With respect to Oracle’s claim for relief for copyright infringement, judgment is entered in favor of Google and against Oracle except as follows: the rangeCheck code in TimSort.java and ComparableTimSort.java, and the eight decompiled files (seven “Impl.java” files and one “ACL” file), as to which judgment for Oracle and against Google is entered in the amount of zero dollars (as per the parties’ stipulation).

With respect to Google’s equitable defenses, judgment is entered for Oracle and against Google as to waiver and implied license. As to equitable estoppel and laches, no ruling need be made due to mootness.

**IT IS SO ORDERED.**

Dated: June 20, 2012.

  
\_\_\_\_\_  
WILLIAM ALSUP  
UNITED STATES DISTRICT JUDGE

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

**DATED JULY 13, 2012**



1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25  
26  
27  
28

United States District Court  
For the Northern District of California

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

ORACLE AMERICA, INC.,

No. C 10-03561 WHA

Plaintiff,

v.

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

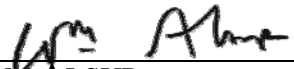
GOOGLE INC.,

Defendant.

Plaintiff Oracle America, Inc. moves for judgment as a matter of law under Rule 50(b), or in the alternative, for a new trial under Rule 59, on issues of patent and copyright infringement. Oracle's arguments are repetitive of its Rule 50(a) motions and rely on the same evidence. For reasons stated in prior orders (Dkt. Nos. 1119, 1165, 1201, 1202, 1203, 1211), Oracle's motion is **DENIED**. The hearing scheduled for July 26 is **VACATED**.

**IT IS SO ORDERED.**

Dated: July 13, 2012.

  
WILLIAM ALSUP  
UNITED STATES DISTRICT JUDGE

## CERTIFICATE OF SERVICE

I hereby certify that on February 11, 2013, I caused the foregoing Opening Brief And Addendum of Plaintiff-Appellant Oracle America, Inc., to be electronically filed with the Clerk of the Court using CM/ECF, which will automatically send email notification of such filing to the following counsel of record:

Christa M. Anderson  
Robert A. Van Nest  
Steven Hirsch  
Michael Kwun  
Dan Jackson  
Keker & Van Nest, LLP  
633 Battery Street  
San Francisco, CA 94111

By: /s/ E. Joshua Rosenkranz  
*Attorney for Plaintiff-Appellant*

**CERTIFICATE OF COMPLIANCE  
UNDER FEDERAL RULES OF APPELLATE PROCEDURE  
32(a)(7) AND FEDERAL CIRCUIT RULE 32**

Counsel for Plaintiff-Appellant certifies that the brief contained herein has a proportionally spaced 14-point typeface, and contains 13,998 words, based on the “Word Count” feature of Word 2007, including footnotes and endnotes. Pursuant to Federal Rule of Appellate Procedure 32(a)(7)(B)(iii) and Federal Circuit Rule 32(b), this word count does not include the words contained in the Certificate of Interest, Table of Contents, Table of Authorities, Abbreviations, and Statement of Related Cases.

Dated: February, 11, 2013

Respectfully submitted,

By: /s/ E. Joshua Rosenkranz  
*Attorney for Plaintiff-Appellant*