DRAWING THE LINE BETWEEN IDEA AND EXPRESSION IN *ORACLE V. GOOGLE*: QUESTIONING THE COPYRIGHTABILITY OF JAVA'S APPLICATION PROGRAMMING INTERFACE

by
*Andrew J. Harrington**

The rapid rate of technological innovation in computing has created a challenge in many areas of the law, particularly in the field of copyright. Since technology is now inextricably linked with modern life, the laws governing this field must catch up. Part II of this Article reviews the relevant history of computers and computer programming. Part III of this Article analyzes the current state of copyright law and the copyrightability of computer code, as exemplified by Oracle v. Google. *Part IV of this Article critiques the disposition of that case by the Federal Circuit and clarifies the need for Congress or the Supreme Court to provide clarity in this increasingly convoluted, complex area of the law.*

---

* Intellectual Property Attorney, Tektronix, Inc.; Lewis & Clark Law School, J.D. *magna cum laude* 2016. First and foremost, thank you to my wife, Emily Smith-Harrington, for her constant love, support, and encouragement throughout law school. Second, my thanks and appreciation to Professors Anna B. Laakmann, Douglas Newell, H. Tomás Gómez-Arostegui, and Lydia P. Loren for their outstanding courses in Intellectual Property Law, and to Professors Anne E. Villella and Judith Miller for their instruction in legal writing. Lastly, my sincere gratitude to Tektronix, and especially to Thomas F. Lenihan and Michael A. Nelson, for creating and fostering the in-house Intellectual Property Attorney program.

I. INTRODUCTION

Computing is undoubtedly one of the fastest developing technical fields in history. Since the inception of the modern computer industry in the 1940s, advances in computer technology have unfolded at such a rapid pace that some have identified technological developments in computing as creating an "information revolution" rivaling the agricultural and industrial revolutions of the past.[1] Today, computers are omnipresent. They control critical transportation and utility infrastructure, run banking systems and stock markets, and they are found in our homes, offices, vehicles, appliances, and mobile phones.[2] As the speed and capabilities of computer hardware have steadily advanced, so too has the field of computer software.[3] Originally, programming a computer involved communicating to the machine directly in binary code.[4] This tedious method of programming quickly spurred the development of "high-level programming languages."[5] One such modern high-level programming lan-

---

[1] *See, e.g.*, JOHN L. HENNESSY & DAVID A. PATTERSON, COMPUTER ORGANIZATION AND DESIGN: THE HARDWARE / SOFTWARE INTERFACE 3–4 (2d ed. 1998) (noting, for instance, that had the transportation industry maintained the same pace of progress as the computer industry, people "could travel coast to coast in 5 seconds for 50 cents").

[2] *See id.* at 4.

[3] *See id.* at 4–5.

[4] *Id.* at 6.

[5] *See id.*

guage is Java, currently maintained by Oracle America, Inc. (Oracle).[6] Like many programming languages, Java contains certain pre-written libraries of code that people writing programs in the Java language can freely use to execute certain functions.[7] In Java, these libraries are called the Java Application Programming Interface (API).[8]

Advances in computer technology have strained many areas of the law,[9] but copyright law is one particular area in which the courts and Congress have struggled to adapt to these technological developments, especially changes in the practice of writing computer software.[10] This Article highlights one example of such a struggle. In 2012, Oracle sued Google, Inc. (Google) for copyright infringement, seeking $9 billion in damages.[11] Oracle based its claim on Google's use of certain elements of the Java API in Google's Android operating system for mobile phones and tablets.[12] This Article uses the ongoing copyright dispute[13] between these two giants of Silicon Valley to explore how the courts have wrestled to determine which aspects of computer software are, and are not, copyrightable subject matter. In particular, this Article looks at different approaches that courts have taken to determine the copyrightability of nonliteral elements of a computer program. To set the stage, Part II briefly surveys the history of computers and computer programming. Part III examines the specific facts of the *Oracle v. Google* lawsuit and describes the

---

[6] *See Java Software*, ORACLE, https://www.oracle.com/java/index.html (last visited May 11, 2017).

[7] *See, e.g.*, 1 JAMES GOSLING ET AL., THE JAVA™ APPLICATION PROGRAMMING INTERFACE xvii (1996).

[8] *Id.*

[9] *See, e.g.*, United States v. Jones, 132 S. Ct. 945, 949 (2012) (determining whether attaching a Global-Positioning-System (GPS) unit—a form of computer—to a car was a search under the Fourth Amendment).

[10] *See* ANTHONY LAWRENCE CLAPES, SOFTWARE, COPYRIGHT, AND COMPETITION: THE "LOOK AND FEEL" OF THE LAW 11–18 (1989).

[11] Oracle Am., Inc. v. Google, Inc., 872 F. Supp. 2d 974 (N.D. Cal. 2012), *rev'd and remanded*, Oracle Am., Inc. v. Google, Inc., 750 F.3d 1339 (Fed. Cir. 2014).

[12] *See id.* at 975.

[13] After Google's petition for certiorari was denied in 2015, *see* Google, Inc. v. Oracle Am., Inc., 135 S. Ct. 2887 (2015), the case was remanded back to the Northern District of California for a new trial on the issue of fair use, *see* Oracle Am., Inc. v. Google, Inc., 750 F.3d 1339 (Fed. Cir. 2014). This Article does not address the fair use issues in this case, but instead, focuses solely on the question of whether certain elements of the Java API should be copyrightable at all. Since the original writing of this Article, a jury found in favor of Google on the issue of fair use. Oracle Am., Inc. v. Google Inc., No. 3:10-cv-03561 (N.D. Cal. June 8, 2016) (BL, Court Docket, Final Judgment). After losing its motion for a new trial, Oracle Am., Inc. v. Google Inc., No 3:10-cv-03561 (N.D. Cal. Sept. 27, 2016) (order denying motion for a new trial), Oracle has appealed once again to the Federal Circuit to overturn the jury verdict. Oracle Am., Inc. v. Google Inc., No 3:10-cv-03561 (N.D. Cal. Oct. 26, 2016) (Oracle's notice of appeal to the Court of Appeals for the Federal Circuit). The appeal is currently pending.

precedent and reasoning supporting the opinions of first the district court, and then the appellate court. Part IV offers a critique of the Court of Appeals for the Federal Circuit's opinion and a call for the Supreme Court or Congress to provide more clarity around this increasingly important issue in copyright law.

II. HISTORY OF COMPUTING

A. Computing's roots in mathematics and the physical sciences

Although today's computing devices are often used for personal communication and entertainment, historically, computers have been devices used for performing calculations to solve particular problems.[14] The earliest computers were purely mechanical devices, intimately tied to mathematics.[15] The abacus, for example, is an age-old tool for counting and performing simple calculations, and is still in use today.[16] Other devices modeled laws of physics in an attempt to better understand and describe the natural world.[17] Ancient civilizations built various special-purpose devices for calculating the movement of the sun, moon, planets, and stars, and used these devices for astrology, astronomy, and navigation.[18]

From the Renaissance through the late nineteenth century, discoveries in mathematics spurred the development of machines capable of performing more general-purpose calculations. Galileo's work in the sixteenth and seventeenth centuries to "mathematicize the physical sciences" led to several innovations in mathematics, from the development of algebra to analytical geometry to calculus.[19] All of these innovations were attempts to more accurately describe data observed in the natural world, and to better predict future physical phenomena. Computing machines filled a requirement created by these mathematical developments: the need to efficiently calculate formulas and solve equations.[20] Machines invented by Blaise Pascal and Gottfried Wilhelm Leibniz in the seventeenth and early eighteenth centuries could perform some of the necessary arithmetic operations—addition, subtraction, multiplication, division—and were therefore precursors to modern calculators and com-

---

[14] *See generally* HERMAN H. GOLDSTINE, THE COMPUTER FROM PASCAL TO VON NEUMANN 3–7 (1972) (outlining a brief history of computers).

[15] *Id.* at 39.

[16] *Id.*

[17] *See id.* at 39–40.

[18] *See id.* at 5 (noting the "continuous development of mechanico-graphical scale models" of the movements of the planets, from a Grecian bronze planetarium circa 30 B.C.E. to al-Kāshī's "Plate of Conjunctions" built in the early fifteenth century).

[19] *See id.* at 3–4.

[20] *See id.* at 39–40.

puters.[21] Leibniz viewed his machine as freeing scientists from "los[ing] hours like slaves in the labor of calculation which could safely be relegated to anyone else if machines were used."[22]

B. Babbage's Analytical Engine and Jacquard's Loom

Charles Babbage, born in England in 1791, made great contributions to advance the capabilities of computing machines.[23] Babbage propounded theories about using machines for a task that is very useful for scientists—the calculation of polynomial tables—and he obtained government funding to develop these theories into working devices.[24] Babbage built his first machine, the Difference Engine, specifically to perform polynomial table calculations.[25] His second machine—which Babbage conceived in 1833, but never finished building before his death in 1871—was called the Analytical Engine.[26] The Analytical Engine is a clear ancestor of the modern general-purpose computer.[27] A modern computer is characterized by having the ability to perform the following: some arithmetic operations; the logical operations (e.g., AND, OR, and NOT); load and store operations; testing and branching (e.g., IF-THEN); and communication with input and output.[28] Babbage's design for the Analytical Engine exhibited a majority of these capabilities: It had a memory from which it could load and to which it could store values, it could perform various sequences of arithmetic operations on those values, and it had an input and output.[29]

The Analytical Engine also used a very early form of computer software. The machine was controlled by sequences of punched cards.[30] Babbage's use of cards bearing encoded instructions was definitely inspired by an invention of Joseph Marie Jacquard,[31] made in France a few decades earlier in 1805.[32] Jacquard invented an attachment to the mechani-

---

[21] *See id.* at 7–9.

[22] *Id.* at 8 (citation omitted) (internal quotation marks omitted) (noting that these machines were needed because "at this time the ability to do arithmetic was not generally to be found even among well-educated men").

[23] *See generally id.* at 10–26 (describing Babbage and his work "automating computation").

[24] *See id.* at 15–18 (explaining that since most mathematical and physical functions can be modeled as a polynomial, calculation and recordation of these "polynomial tables" allowed scientists to match an observed set of data to a precomputed table in order to establish a model of a real world phenomenon).

[25] *Id.* at 17–19.

[26] *Id.* at 19.

[27] *Id.*

[28] *See* GENE H. MILLER, MICROCOMPUTER ENGINEERING 38 (2d ed. 1999).

[29] *See* GOLDSTINE, *supra* note 14, at 19–22.

[30] *Id.* at 20–22.

[31] *Id.* at 20.

[32] *Id.*

cal loom that automated the process of weaving fabrics.[33] A series of punched cards that represented the specific pattern to be woven controlled the operation of the Jacquard loom.[34] The parallels between the punched cards of the Jacquard loom, the sets of control cards used in the Analytical Engine, and sequences of instructions that today we call a computer "program" are obvious. As described by Lady Ada Lovelace, a student of Babbage, "We may say most aptly that the Analytical Engine weaves algebraical patterns just as the Jacquard-loom weaves flowers and leaves."[35] The idea arising from the Jacquard loom of using punched cards to control machine operation was carried into the late nineteenth century when Herman Hollerith designed a machine that used punched cards to tabulate data for the 1890 census.[36] Hollerith went on to found the Tabulating Machine Company, which would eventually become International Business Machines (IBM), a key player in the early twentieth century computer industry.[37]

C. The dawn of the modern computing era

Developments in computer hardware continued rapidly, and largely in parallel worldwide, into the early twentieth century, generating a slew of new computing machines.[38] Built during World War II, the machine that is generally recognized as the first electronic, general-purpose computer—the ENIAC (Electronic Numerical Integrator and Calculator)—was disclosed to the public in 1946.[39] Other machines quickly followed, having a bewildering array of descriptive acronyms for names.[40] These early machines were largely devoted to research and military applications.[41]

The first commercially successful computer was the UNIVAC I (Universal Automatic Computer), introduced by the Remington-Rand com-

---

[33] *Id.*

[34] *Id.*

[35] *Id.* at 22 (emphasis omitted) (internal quotations omitted).

[36] *See id.* at 65–66.

[37] *Id.* at 66.

[38] *See, e.g.*, *id.* at 111–20 (describing the development of the Mark-I system at Harvard in the 1940s and "concurrent" development of a "partially automatic computer" at Bell Telephone Laboratories); *see also id.* at 349–62 (describing largely contemporaneous worldwide developments "that directly affected progress in the computer field").

[39] HENNESSY & PATTERSON, *supra* note 1, at 32.

[40] *See id.* at 32–35 (chronicling the creation of the EDSAC (Electronic Delay Storage Automatic Calculator) in 1949, the EDVAC (Electronic Discrete Variable Automatic Computer) in 1952, and the Mark-II, -III, and -IV systems during the same period).

[41] *See* GOLDSTINE, *supra* note 14, at 135 (noting the production of ballistics tables was the "*raison d'être* for the first electronic digital computer").

pany in 1951.[42] IBM entered the commercial computer market in 1952, but really established its domination of the market with its introduction of the System/360 family of computers in 1964, greatly expanding the use of computers for business applications.[43] Development of the mini-computer and the personal computer from the mid-1960s through the 1980s exemplified the computer industry's breakneck pace to produce ever smaller, cheaper, yet ever more capable machines.[44] Today's pocket-sized mobile phones are orders of magnitude more powerful than the first computers.[45]

Innovations in the field of computer software generally paralleled the progress made in computer hardware. Originally, computers were programmed by inputting instructions that were specific to the particular model of machine.[46] This tedious method of programming soon gave way to the use of high-level programming languages such as FORTRAN, COBOL, BASIC, C, C++, and others,[47] including Java, the source of the copyright dispute that is the subject of this Article. Part III.B discusses more fully this transformation in the process of computer programing.

III. THE ISSUE OF COPYRIGHTABLE SUBJECT MATTER IN *ORACLE V. GOOGLE*

A. *The parties and their respective computer programs*

The copyright infringement lawsuit brought by Oracle against Google[48] involves the Java programming language, or, more accurately, the "Java platform."[49] In the early 1990s, a team of engineers led by James Gosling at Sun Microsystems (Sun) developed the Java programing language.[50] Although Java was initially targeted for use in a home-entertainment system, Sun's engineers observed the increasing activity and interest around the nascent World Wide Web at the time, and shifted

---

[42] HENNESSY & PATTERSON, *supra* note 1, at 36.

[43] *See id.* at 36–38.

[44] *See id.* at 38–42 (noting that adjusted for inflation, "price/performance has improved by about 240 million in 45 years, or about 54% per year").

[45] *Compare id.* fig.1.30, at 43 (classifying characteristics of key commercial computers from 1951 to 1996), *with iPhone 7 Tech Specs*, APPLE, INC., http://www.apple.com/iphone-7/specs/ (last visited May 11, 2017).

[46] *See* BERNARD A. GALLER, SOFTWARE AND INTELLECTUAL PROPERTY PROTECTION: COPYRIGHT AND PATENT ISSUES FOR COMPUTER AND LEGAL PROFESSIONALS app. A, at 152 (1995).

[47] *See id.* at 154–55.

[48] Oracle Am., Inc. v. Google, Inc., 872 F. Supp. 2d 974, 975 (N.D. Cal. 2012), *rev'd and remanded*, Oracle Am., Inc. v. Google, Inc., 750 F.3d 1339 (Fed. Cir. 2014).

[49] *See, e.g., id.* at 977.

[50] *The History of Java Technology*, ORACLE, http://www.oracle.com/technetwork/java/javase/overview/javahistory-index-198355.html (last visited May 11, 2017).

their attention to Internet-related applications for Java.[51] In 1995, the first publicly released version of Java was distributed over the Internet.[52] Throughout the end of the 1990s and early 2000s, the World Wide Web and the Java programming language both exploded in popularity and, in a sense, came of age together to form part of the foundation of the Internet era.[53] In 2009, Oracle Corporation announced that it was purchasing Sun Microsystems, thereby acquiring all of Sun's intellectual property interests, among other assets.[54] The acquisition was completed in early 2010, giving Oracle "ownership" over the Java programming language, which by that time ran on over a billion devices.[55]

Google is, of course, the company behind the very popular eponymous search engine.[56] Google was incorporated in California in 1998[57] and, like Java, also came of age in the Internet era. Initially strictly a search engine company, Google expanded, sometimes through internal development and sometimes through acquisitions, into many different areas of technology.[58] Most significantly for the purposes of this Article, Google expanded into the area of mobile phone operating systems with its announcement of the Android platform in 2007.[59] According to Google, Android is a "freely-distributed, open-source software stack for mobile devices" developed by the Open Handset Alliance, a group of companies who came together "to accelerate innovation in mobile devices and offer consumers a richer, less expensive, and better mobile experience."[60]

From its beginning, Android was intended to be an important part of the open-source software community,[61] allowing developers free access to

---

[51] *See id.*

[52] *Java Timeline*, ORACLE, http://oracle.com.edgesuite.net/timeline/java/ (last visited May 11, 2017) (scroll to "1995," click on "Java Makes its Debut").

[53] *Id.*

[54] *See* Press Release, Oracle Corp., Oracle Buys Sun (Apr. 20, 2009).

[55] *See Oracle Completes Acquisition of Sun Microsystems*, SAN DIEGO UNION TRIB. (Jan. 27, 1010), http://www.sandiegouniontribune.com/sdut-oracle-completes-acquisition-of-sun-microsystems-2010jan27-story.html.

[56] *See* GOOGLE, http://www.google.com/.

[57] *Our Story: From the Garage to the Googleplex*, GOOGLE, https://www.google.com/about/our-story/ (last visited May 11, 2017).

[58] *See id.; see also Our Products*, GOOGLE, https://www.google.com/about/products/ (last visited May 11, 2017) (listing expansion into products such as the AdWords program, Google Earth, Google Maps, and YouTube, among others).

[59] *See* Miguel Helft & John Markoff, *Google Enters the Wireless World*, N.Y. TIMES (Nov. 5, 2007), http://www.nytimes.com/2007/11/05/technology/05cnd-gphone.html.

[60] Google Inc.'s Answer to Plaintiff's Amended Complaint for Patent and Copyright Infringement and Amended Counterclaims at 17, *Oracle*, 872 F. Supp. 2d 974 (N.D. Cal. 2012) (No. 3:10-cv-03561-WHA) [hereinafter Google Inc.'s Answer].

[61] *See About the Android Open Source Project*, ANDROID, https://source.android.com/ (last visited May 11, 2017); *see generally History of the OSI*, OPEN SOURCE INITIATIVE, https://opensource.org/history (last visited May 11, 2017).

its underlying source code and the ability to freely modify and distribute modified versions of the mobile operating system. In fact, Google believes that the success of the Android platform in the mobile phone market is due to its "open nature."[62] Java also has links into, and has benefitted from, the open-source community. In 2006 and 2007, prior to being acquired by Oracle, Sun released some of the source code for Java to the public subject to the terms of a GNU[63] Public License.[64] Google credits the partial open-source nature of Java with contributing to the platform's "widespread acceptance among software developers."[65]

When Google was deciding whether to develop an open-source operating system for mobile devices, it was initially interested in basing that system on Java, in part to capitalize on its popularity in the software engineering community.[66] In fact, in 2005, Google attempted to negotiate for several months with Sun to modify Java—originally developed for use on desktop and laptop computers—to be used on mobile phones and tablets, but these negotiations were unsuccessful.[67] Around that same time, Sun was facing pressure from other entities in the open-source community, including, coincidentally, from Oracle itself, to "fully open source Java"[68] so that developers could create alternative embodiments of the Java platform. Sun already had an established procedure, called the Java Community Process, for allowing alternative implementations of Java.[69] However, even if an alternative implementation was approved, Sun still imposed field-of-use restrictions on those implementations, one of the most significant being not allowing those alternative implementations to be used on mobile devices.[70] In light of these field-of-use restrictions and the failed negotiations with Sun, Google proceeded to create an alternative implementation of Java that eventually became the Android platform.[71] Despite Oracle's public support for allowing such alternative implementations of Java prior to buying Sun, Oracle's position changed afafter it acquired intellectual property rights in the Java platform.[72] Oracle

---

[62] Google Inc.'s Answer, *supra* note 60, at 20.

[63] GNU, pronounced "g'noo," is an open-source, Unix-like operating system; the name "GNU" is a recursive acronym for "GNU's Not Unix." *What is GNU?*, GNU, https://www.gnu.org/home.en.html (last visited May 11, 2017).

[64] Oracle America, Inc.'s Reply to Defendant Google, Inc.'s Answer to Complaint for Patent and Copyright Infringement and Counterclaims at 2, *Oracle*, 872 F. Supp. 2d 974 (N.D. Cal. 2012) (No. 3:10-cv-03561-WHA).

[65] Google Inc.'s Answer, *supra* note 60, at 15.

[66] *See* Oracle Am., Inc. v. Google, Inc., 872 F. Supp. 2d 974, 978 (N.D. Cal. 2012), *rev'd remanded*, Oracle Am., Inc. v. Google, Inc., 750 F.3d 1339 (Fed. Cir. 2014).

[67] *Id.*

[68] *See* Google Inc.'s Answer, *supra* note 60, at 17.

[69] *Id.* at 16.

[70] *See Id.*

[71] *See Oracle*, 872 F. Supp. 2d at 978.

[72] *See* Google Inc.'s Answer, *supra* note 60, at 15–17.

filed suit against Google in the Northern District of California on August 12, 2010,[73] less than eight months after the acquisition closed.

B. Programming languages and the Java Application Programming Interface (API)

In modern programming practice, a software program is typically embodied in two forms: object code and source code. Object code consists of binary digits—0s and 1s—and is the form of a program that is able to be "read" and "understood" by a computer, but is not generally readily comprehensible to humans.[74] A particular instruction to be executed by a computer processor—for example, an instruction to add two numbers and store the result in a particular memory location—is represented in object code as a particular string of ones and zeros.[75] However, the number of instructions that are recognized as valid by a given processor is finite.[76] The group of valid instructions for a given type of processor is referred to as that processor's "instruction set."[77] To function as intended, the object code to be executed by a particular type of processor must be written in that processor's instruction set.[78] Thus, object code is constrained by and intimately tied to the particular type of processor in a computer; for this reason object code is often said to be written in "machine language."[79]

Source code, on the other hand, is the form of a software program that is human-readable.[80] Source code is written in a programming language, often called a "high-level programming language,"[81] in contrast to a "low-level" machine language. High-level programming languages employ English words and symbols, rather than just binary digits, and therefore allow a programmer to think and write in a more natural manner.[82] In order for a computer to execute a program written in a high-level language, however, the source code must be translated into the machine language that the computer's processor can understand.[83] This translation from high-level language to low-level object code is accomplished by

---

[73] *See* Complaint for Patent and Copyright Infringement, *Oracle*, 872 F. Supp. 2d 974 (N.D. Cal. 2012) (No. 4:10-cv-03561-LB) [hereinafter Complaint].

[74] *See Oracle*, 872 F. Supp. 2d at 977.

[75] *See* GALLER, *supra* note 46, app. A at 151–52.

[76] *See* MILLER, *supra* note 28, at 38.

[77] *Id.*

[78] *See id.*

[79] *See* HENNESSY & PATTERSON, *supra* note 1, at 5.

[80] *See* Oracle Am., Inc. v. Google, Inc., 872 F. Supp. 2d 974, 977 (N.D. Cal. 2012), *rev'd and remanded*, Oracle Am., Inc. v. Google, Inc., 750 F.3d 1339 (Fed. Cir. 2014).

[81] *See* HENNESSY & PATTERSON, *supra* note 1, at 6.

[82] *Id.* at 7.

[83] *See id.* at 6–8.

using a program called a "compiler," which converts human-readable source code into machine-readable object code.[84]

Java is one such high-level programming language used to write human-readable source code.[85] From its beginnings, an overarching goal of Java has been to allow source code written in the Java language to be platform-neutral; that is, able to be executed on any type of computer.[86] Java accomplishes this goal through the use of a type of interpreter program called a "virtual machine."[87] Java source code is compiled by a compiler program, not directly into machine language, but rather into a form called "bytecode."[88] This bytecode is then translated, at run-time, by the virtual machine into the machine language of the particular computer on which it is running.[89]

Of course, Java is merely one of many different programming languages that have waxed and waned in popularity over the last several decades.[90] Java is an object-oriented language, meaning that programs written in the Java language are organized as one or more objects, each called a "class," that each have a particular state and a particular set of behaviors.[91] Information about a class's state is stored in the class's "fields," and the behaviors or functions that a class may perform are called its "methods."[92] These concepts are not unique to Java.[93] The syn-

---

[84] *See id.* Sometimes, a compiler converts code from a high-level language directly into binary machine language. *Id.* at 7. In other cases, the conversion includes an intermediate step wherein the high-level language is first compiled into "assembly language" and is then assembled, by a program called an "assembler," into binary machine language. *Id.* In other cases, rather than being compiled as a whole, prior to program run time, a high-level language may instead be translated line-by-line into one or more binary machine language instructions, during program execution, by a program called an "interpreter." *See* GALLER, *supra* note 46, app. A at 158–59.

[85] *See Oracle*, 872 F. Supp. 2d at 977 (N.D. Cal. 2012).

[86] Sun promoted this feature with its slogan "write once, run anywhere." *See* Jason W. Purdy, *Write Once, Run Anywhere?*, DEVELOPER.COM (Feb. 9, 1998), http://www.developer.com/java/other/article.php/601861/Write-once-run-anywhere.html.

[87] *See Oracle*, 872 F. Supp. 2d at 977.

[88] *See id.*

[89] *See id.*

[90] *See* GALLER, *supra* note 46, app. A at 155 (noting that in 1969 there were at least 200 languages having some use, and listing "FORTRAN, COBOL, PL/1, BASIC, Pascal, C, C++, Ada, LISP, and a few others" as some of the more widely-used languages today).

[91] *See generally What Is an Object?*, ORACLE, https://docs.oracle.com/javase/tutorial/java/concepts/object.html (last visited May 11, 2017). Object-oriented programs typically try to model real-world objects. For example, an object-oriented program might include a class named "LightSwitch." The LightSwitch class would probably have two possible states (off, on) and two behaviors (turn on, turn off). *See id.*

[92] *See id.*

tax of the Java programming language is also not novel—the Java language was heavily influenced by its predecessor programming languages, C and C++.[94] Like C and C++[95]:

> Java syntax includes *separators* (*e.g.*, {,};), *operators* (*e.g.*, +, -, *, /, <, >), *literal values* (*e.g.*, 123, 'x', "Foo"), and *keywords* (*e.g.*, if, else, while, return). These elements carry precise predefined meanings. Java syntax also includes *identifiers* (*e.g.*, String, java.lang.Object), which are used to name specific values, fields, methods, and classes . . . .[96]

A Java programmer uses these elements of syntax to define at least one class and that class will typically have at least one method.[97] As mentioned above, a method performs some particular behavior that is appropriate for the class to which it belongs.[98] In some cases a method will perform its function using data stored in the class's fields; in other cases a method will perform its function using values that are "passed" as arguments when the method is "called on" from somewhere else in the Java program.[99] As a simple example,[100] the following is Java code (with reference line numbers added on the left) that defines a "Math" class, including a "max" method[101] to determine the greater of two numbers (x and y) that are passed in as arguments:

```
1 public class Math {
2 public static int max(int x, int y) {
3 if (x > y) return x;
4 else return y;
5 }
6 }
```

Lines 1 and 6 of this code "declare" the class named "Math;" that is, they give the class a name ("Math") and define what kind of class it is (in this case, "public").[102] Lines 2 and 5 declare the method named "max," a

---

[93] *See Object-Oriented Programming*, TECHTARGET, http://searchmicroservices.techtarget.com/definition/object-oriented-programming-OOP (last visited May 12, 2017).

[94] *See* GOSLING ET AL., *supra* note 7, at xv.

[95] *See, e.g.*, H. M. DEITEL & P. J. DEITEL, C++ HOW TO PROGRAM 27–28, 841 (1998).

[96] Oracle Am., Inc. v. Google, Inc., 872 F. Supp. 2d 974, 979 (N.D. Cal. 2012), *rev'd and remanded*, Oracle Am., Inc. v. Google, Inc., 750 F.3d 1339 (Fed. Cir. 2014).

[97] *See id.* at 980.

[98] *See id.*

[99] *Id.* at 979–81.

[100] The district court used this same example. *See id.* at 980–81.

[101] By convention in Java, class names are capitalized, while method and field names are in lowercase. *See Naming Conventions*, ORACLE, http://www.oracle.com/technetwork/java/codeconventions-135099.html (last visited May 12, 2017).

[102] *See Oracle*, 872 F. Supp. 2d at 981.

method belonging to the Math class.[103] This method "declaration" or "header" gives the method a name ("max"), defines what kind of method it is ("public static"), defines what type of values the method will return as a result ("int," meaning an integer), defines how many and what type of arguments may be passed to the method, and gives these arguments names (two arguments, both integers, "x" and "y").[104] Lines 3 and 4 are the method "body" or "implementation code."[105] In this example, these two lines simply cause the "max" method to compare the values of the two passed arguments "x" and "y." If the value of "x" is greater than the value of "y," then the method will return the value of "x"; otherwise, it will return the value of "y." Every single character of lines 1–6, except for the short names "Math," "max," "x," and "y" are dictated by the syntax and rules of the Java programming language.[106]

The above Math.max "Class.method" example is fairly simple, but Java programs can quickly grow to include multiple classes, each with many associated methods. In order to keep Java source code organized and manageable, the Java language also supports the concept of a "package."[107] Packages are used to group related classes together. Including the line of code "package java.lang;" before line 1 above, would declare a package named "java.lang" and would define the Math class to be a member of that package. The "max" method can be called, or used, from any other Java program that may need to compute the maximum of two numbers.[108] For example, if another program needs to set the value of an integer variable named "a" to the greater of two numbers (2 and 3), the program can simply reuse the already existing "max" method by calling it using its full "package.Class.method" name (java.lang.Math.max) as follows:

```
int a = java.lang.Math.max(2, 3);
```

In this way, the package/class/method structure and hierarchy of Java tends to encourage code re-use, a desirable goal in software development.[109] A programmer can write a method to perform a given function one time and then simply call it whenever needed. This concept is also not unique to Java; many predecessor languages also include "libraries"

---

[103] This method and class is known by virtue of being nested between the opening bracket ("{") of the class declaration on line 1, and the corresponding closing bracket ("}") in line 6. *See id.*

[104] *See id.*

[105] This body is known by virtue of being nested between the opening bracket ("{") of the method declaration on line 2, and the corresponding closing bracket ("}") in line 5. *See id.*

[106] *See id.* at 981 (stating that this fact is "critical to" the copyrightability issue).

[107] *See id.* at 980.

[108] *See id.* at 981.

[109] *See* HENNESSY & PATTERSON, *supra* note 1, at 8.

of code.[110] The developers of the Java language gathered libraries of code that perform often-used functions and organized them into a package/class/method hierarchy.[111] These libraries of code are referred to as Java's Application Programming Interface.[112]

Just as with other programming languages, Java programmers quickly become accustomed to using the classes and methods included in the Java API. As explained above, in order to use a particular method, a programmer must call it by its exact package.Class.method name, and that name is dictated by the package/class/method organization of the API and the rules of Java syntax.[113] In order to encourage programmers to write programs for its Android platform, Google decided to use some of the exact same package.Class.method names in the Android API.[114] Specifically, Google copied the organizational structure of 37 out of 166 Java API packages.[115] To be clear, for all 37 of these packages, Google wrote its own implementing code;[116] it copied only the lines of declaring code[117] that define the specific package.Class.method names.[118] In total, these copied lines of declaring code comprised three percent of all of the lines of code in the 37 packages at issue.[119]

C. District court proceedings

Oracle's complaint against Google[120] alleged that the Android platform infringed seven patents related to the Java platform[121] that had been assigned by Sun to Oracle, and that Android infringed Oracle's various

---

[110] *See id.*; *see also* DEITEL & DEITEL, *supra* note 95, at 10–11 (documenting the "standard library" of C++ code).

[111] *See Oracle*, 872 F. Supp. 2d at 977 (noting that in 2008, these libraries were organized into "166 'packages,' broken into more than six hundred 'classes,' all broken into over six thousand 'methods'").

[112] *See id.* ("The Java language itself is composed of keywords and other symbols and a set of pre-written programs . . . called the application programming interface or simply API (also known as class libraries)."); *see also* GOSLING ET AL., *supra* note 7, at xvii (describing the Java API as "a standard set of libraries for writing Java programs").

[113] *See Oracle*, 872 F. Supp. 2d at 999–1000.

[114] *Id.* at 978 ("Google believed Java application programmers would want to find the same . . . sets of functionalities in the new Android system callable by the same names as used in Java.Code already written in the Java language would, to this extent, run on Android and thus achieve a degree of interoperability.").

[115] *Id.* at 977.

[116] *Id.* at 978.

[117] *Id.* at 977.

[118] *See id.* at 1000.

[119] *Id.* at 979 ("This three percent is the heart of our main copyright issue.").

[120] Complaint, *supra* note 73.

[121] Specifically, those seven patents are 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. *See id.* at 3–7.

copyright interests in the Java platform.[122] Google answered by denying all claims of patent and copyright infringement, asserting numerous defenses—including, among others, waiver, estoppel, laches, patent invalidity, non-copyrightable subject matter, de minimis copying, implied license, and fair use—and counter-claimed for declaratory judgments of non-infringement and invalidity as to all of the patents, and a declaratory judgment of non-infringement of the asserted copyrights.[123] Both parties demanded a jury trial.[124]

Due to the numerous and complex issues involved, District Court Judge William Alsup split the trial into three phases: The first phase dealt with copyrightable subject matter, copyright infringement, and defenses to infringement; the second phase was for the patent infringement claims; the third phase, which was not reached, would have dealt with damages.[125] After a six-week jury trial, the jury found no infringement as to any of the asserted patents.[126] In order for the jury to determine the issues of copyright infringement and fair use, the jury was instructed to assume that the Java API was copyrightable.[127] On this assumption, the jury found that Google infringed, but deadlocked on the issue of fair use.[128] However, the jury's copyright infringement finding was contingent on the court's ultimate legal determination of whether the "structure, sequence and organization of the 37 API packages as a whole *was* copyrightable."[129] In a detailed and well-reasoned opinion, Judge Alsup held that the structure, sequence and organization—the "overall name tree"—of the Java API was not copyrightable subject matter because it was either a system or a method of operation,[130] both of which are statutorily barred from being copyrighted.[131]

---

[122] Specifically, those copyright interests include the Java 2 Standard Edition versions 1.4 and 5.0. *See id.*, exhibit H, at 8–9.

[123] Google Inc.'s Answer, *supra* note 60, at 8–12, 21–28.

[124] *See id.* at 30; Complaint, *supra* note 73, at 10.

[125] Oracle Am., Inc. v. Google, Inc., 872 F. Supp. 2d 974, 975 (N.D. Cal. 2012), *rev'd and remanded*, Oracle Am., Inc. v. Google, Inc., 750 F.3d 1339 (Fed. Cir. 2014).

[126] *Id.* at 976.

[127] *Id.* at 975.

[128] *Id.* at 976.

[129] *Id.* at 975.

[130] *Id.* at 976–77.

[131] *See* 17 U.S.C. § 102(b) ("In no case does copyright protection for an original work of authorship extend to any idea, procedure, process, *system, method of operation,* concept, principal, or discovery, regardless of the form in which it is described, explained, illustrated, or embodied in such work." (emphasis added)).

D. The district court's copyrightability analysis

1. Originality

In the seminal case *Feist Publications, Inc. v. Rural Telephone Service Co.*, the Supreme Court stated the general requirements for copyrightability of a work.[132] *Feist*, a copyright infringement case involving competing telephone white page directories, dealt primarily with the distinction between uncopyrightable facts and potentially copyrightable compilations of facts.[133] The Court articulated that "[t]he *sine qua non* of copyright is originality,"[134] and explained that this need for originality is a constitutional requirement.[135] For a work to meet the originality requirement, it only needs to satisfy two criteria: First, it must have been independently created; and second, it must "possess[] at least some minimal degree of creativity."[136] The creativity requirement is not stringent—"even a slight amount will suffice."[137] The Court held that in determining whether a compilation of facts possessed the requisite level of creativity, courts should focus on the "selection, coordination, and arrangement" of the compilation.[138]

2. The idea/expression dichotomy

The *Feist* opinion also reiterated a significant limit on the potential scope of copyright in a work: the idea/expression dichotomy.[139] The idea/expression dichotomy reflects the basic principle that copyright law "assures authors the right to their original expression, but encourages others to build freely upon the ideas and information conveyed by a work."[140] In short, copyright protects an author's expression of an idea, but does not allow the author to monopolize the underlying idea itself. Like originality, the idea/expression dichotomy is also constitutionally mandated.[141] The result of this principle is that others are free to use and add to any of the underlying ideas or other unprotected elements of a work. As the Court explained, "This result is neither unfair nor unfortunate. It is the means by which copyright advances the progress of science

---

[132] *See* 499 U.S. 340, 344–45 (1991).

[133] *Id.* at 345.

[134] *Id.*

[135] *Id.* at 346 (explaining that the originality requirement derives from the language of the Intellectual Property Clause of the U.S. Constitution, *see* U.S. CONST. art. I, § 8, cl. 8).

[136] *Id.* at 345.

[137] *Id.*

[138] *Id.* at 358. In *Feist*, the plaintiff's "entirely typical" arrangement of telephone subscriber names alphabetically by last name did not meet the originality standard. *Id.* at 362.

[139] *Id.* at 350.

[140] *Id.* at 349–50 (citing Harper & Row, Publishers, Inc. v. Nation Enters., 471 U.S. 539, 556–57 (1985)).

[141] *Id.* at 349 (citing U.S. CONST. art. I, § 8, cl. 8).

and art."[142] This distinction between unprotectable ideas and protectable expression developed in the common law,[143] but in the Copyright Act of 1976, Congress codified the principle in a new Section 102(b) of Title 17 of the United States Code.[144] The same Act also included granting copyright protection to computer programs as "literary works."[145] Although *Feist* dealt with uncopyrightable facts in a printed phonebook, the idea/expression dichotomy applies to all types of works,[146] including computer programs.

In addition to distinguishing between the copyrightable and non-copyrightable elements of a work, the idea/expression dichotomy also serves to delineate the boundary between copyright law and patent law. As Judge Alsup recognized in *Oracle*, in cases involving questions of software copyrightability, the dividing line between these different forms of protection

> looms large where, as here, the vast majority of the code was *not* copied and the copyright owner must resort to alleging that the accused stole the "structure, sequence and organization" of the work. This phrase—structure, sequence and organization—does not appear in the [Copyright] Act or its legislative history. It is a phrase that crept into use to describe a residual property right where literal copying was absent. A question then arises whether the copyright holder is more appropriately asserting an exclusive right to a functional system, process, or method of operation that belongs in the realm of patents, not copyrights.[147]

The Supreme Court explored this dividing line in *Baker*,[148] a case decided almost a hundred years before Congress granted copyright protection to software. The plaintiff in *Baker* authored a book that instructed the reader on a method of bookkeeping.[149] The book included blank forms to practice the described method with the data fields on the forms named and organized in a particularly useful way.[150] The plaintiff alleged

---

[142] *Id.* at 350; *cf.* U.S. CONST. art. I, § 8, cl. 8 ("The Congress shall have Power . . . To promote the Progress of Science and the useful Arts, by securing for limited Times to Authors and Inventors the exclusive Right to their respective Writings and Discoveries.").

[143] *See, e.g.*, Baker v. Selden, 101 U.S. 99 (1879).

[144] *See* Oracle Am., Inc. v. Google, Inc., 872 F. Supp. 2d 974, 985 (N.D. Cal. 2012) (citing Apple Comput., Inc. v. Microsoft Corp., 35 F.3d 1435, 1443 n.11 (9th Cir. 1994)), *rev'd and remanded*, Oracle Am., Inc. v. Google, Inc., 750 F.3d 1339 (Fed. Cir. 2014).

[145] *Id.* (citing H.R. REP. NO. 94-1476, at 54 (1976)).

[146] *Feist*, 499 U.S. at 350.

[147] *Oracle*, 872 F. Supp. 2d at 984.

[148] *See generally Baker*, 101 U.S. 99.

[149] *Id.* at 99–100.

[150] *Id.* at 100 ("This system . . . by a peculiar arrangement of columns and headings, presents the entire operation, of a day, a week, or a month, on a single page, or on two pages facing each other, in an account-book.").

that the defendant infringed his copyright in the book by printing forms that were similar, but had slightly different heading names and arrangement of the columns.[151] In holding that the plaintiff's blank accounting forms were not copyrightable subject matter,[152] the Court explained that there is a difference between the protectable and unprotectable elements of the plaintiff's book:

> There is no doubt that a work on the subject of book-keeping, though only explanatory of well-known systems, may be the subject of a copyright; but, then, it is claimed only as a book. . . . But there is a clear distinction between the book, as such, and the art which it is intended to illustrate. The mere statement of the proposition is so evident, that it requires hardly any argument to support it.[153]

Recognizing this distinction between an author's expression ("the book, as such") and the underlying idea, system, or method ("the art which it is intended to illustrate")—in other words, recognizing the idea/expression dichotomy—is necessary to maintain the proper, separate roles of copyright law and patent law. Copyright, the Court declared, cannot be used to grant an author a monopoly over a system that she illustrates in words:

> To give to the author of the book an exclusive property in the art described therein, when no examination of its novelty has ever been officially made, would be a surprise and a fraud upon the public. That is the province of letters-patent, not of copyright. The claim to an invention or discovery of an art or manufacture must be subjected to the examination of the Patent Office before an exclusive right therein can be obtained; and it can only be secured by a patent from the government.[154]

Thus, maintaining these separate spheres of patent and copyright protection is necessary to protect the public's right to use knowledge in the public domain. Such knowledge includes any unpatented ideas, systems, or methods[155] that may be embedded in a copyrighted work. Copyright law cannot be used to make an end run around patent law to usurp these unprotectable elements from the public domain.[156]

---

[151] *Id.* at 100–01.

[152] *Id.* at 107.

[153] *Id.* at 101–02.

[154] *Id.* at 102.

[155] Indeed, some ideas, systems, and methods cannot be the proper subject of patents, either. *See* Alice Corp. Pty. Ltd. v. CLS Bank Int'l, 134 S. Ct. 2347, 2354 (2014) ("Laws of nature, natural phenomena, and abstract ideas are not patentable." (internal quotations omitted) (quoting Ass'n for Molecular Pathology v. Myriad Genetics, Inc., 133 S. Ct. 2107, 2116 (2013))).

[156] For one reason, the Copyright Act grants a much longer duration of monopoly than does Patent Law. *See* Oracle Am., Inc. v. Google, Inc., 872 F. Supp. 2d 974, 998 (N.D. Cal. 2012) ("Based on a single implementation, Oracle would bypass

Closely related to the idea/expression dichotomy is the doctrine of merger, a doctrine developed by the courts to "guard[] the boundaries between ideas and expressions."[157] The merger doctrine is the principle that when there is only one way, or only a limited number of ways, to express an underlying unprotectable idea, the idea and expression "merge," thereby making the expression itself also unprotectable.[158] According to one particular formulation of the doctrine from the Ninth Circuit:

> Where an idea and the expression merge or are inseparable, the expression is not given copyright protection. . . . In addition, where an expression is, as a practical matter, indispensable, or at least standard, in the treatment of a given idea, the expression is protected only against verbatim, or virtually identical copying.[159]

The Copyright Office has incorporated courts' applications of the idea/expression dichotomy and the merger doctrine in its clear-cut administrative rule that excludes words and short phrases, among other things, from copyright protection.[160] As Judge Alsup noted, the Ninth Circuit recognizes the Copyright Office's rule and "[t]his has relevance to Oracle's claim of copyright over names of methods, classes, and packages."[161]

3. Copyrightability of specific lines of declaring code

As discussed above, copyright law has established the following bedrock principles for determining copyrightable subject matter: the idea/expression dichotomy, which is codified in Section 102(b) and reflected in the Copyright Office regulations; the extension of the idea/expression dichotomy through the judicially-developed merger doctrine; and, the fundamental distinction between copyrightable and

---

this entire patent scheme and claim ownership over any and all ways to carry out methods for 95 years—without any vetting by the Copyright Office of the type required for patents."), *rev'd and remanded*, Oracle Am., Inc. v. Google, Inc., 750 F.3d 1339 (Fed. Cir. 2014). *Compare* 17 U.S.C. § 302(c) (2012) (95-year copyright term for a work made for hire), *with* 35 U.S.C. § 154(a)(2) (2012) (20-year patent term).

[157] Matthew J. Faust, Comment, *What Do We Do with a Doctrine Like Merger? A Look at the Imminent Collision of the DMCA and Idea/Expression Dichotomy*, 12 MARQ. INTELL. PROP. L. REV. 131, 141 (2008).

[158] *See* JULIE E. COHEN ET AL., COPYRIGHT IN A GLOBAL INFORMATION ECONOMY 96 (4th ed. 2015).

[159] Johnson Controls, Inc. v. Phx. Control Sys., Inc., 886 F.2d 1173, 1175 (9th Cir. 1989) (internal citations omitted). In addition to merger, this quotation also states the copyright scope-limiting *scenes a faire* doctrine: Characters, settings, plot devices, etc., that are "stock" or expected in a particular genre of work are not given copyright protection. *See Oracle*, 872 F. Supp. 2d at 990.

[160] *See* 37 C.F.R. § 202.1(a) (2015) ("The following are examples of works not subject to copyright . . . : (a) Words and short phrases such as *names*, *titles*, and slogans; familiar symbols or designs . . . ." (emphasis added)); *see also id.* at § 202.1(b) (excluding "[i]deas, plans, methods, systems, or devices").

[161] *Oracle*, 872 F. Supp. 2d at 984.

patentable subject matter as recognized in *Baker*.[162] Based primarily on these fundamental principles, the district court held that the specific lines of declaring code in the disputed 37 packages of the Java API—the lines of code defining the class and method names, inputs, and outputs—were not copyrightable.[163] Although Oracle attempted to prove that the design of the API—including the lines of declaring code—was sufficiently creative to satisfy the constitutional mandate of originality,[164] the court held that, due to the disparate roles of patent and copyright law, the level of creativity involved in writing these lines of code did not matter:

> Inventing a new method to deliver a new output can be creative, even inventive, including the choices of inputs needed and outputs returned. . . . But such inventions—at the concept and functionality level—are protectable only under the Patent Act . . . . [U]nder the Copyright Act, no matter how creative or imaginative a Java method specification may be, the entire world is entitled to use the same method specification (inputs, outputs, parameters) so long as the line-by-line implementations are different.[165]

A straightforward application of the idea/expression dichotomy supports this holding. As the court stated, "The method specification is the *idea*. The method implementation is the *expression*. No one may monopolize the *idea*."[166] Furthermore, in addition to each specification's "concept and functionality" not being copyrightable, the court held that the particular names used in the Java API for each class (e.g., "Math"), method (e.g., "max"), and arguments to methods (e.g., "x" and "y") were also barred from copyrightability.[167]

4. Copyrightability of the API's sequence, structure, and organization

Having disposed of Oracle's infringement claims that had any basis in literal copying, the district court next addressed Oracle's best argu-

---

[162] *Id.* at 997 ("[T]his order concludes that our immediate case is controlled by these principles of copyright law: Under the merger doctrine, when there is only one (or only a few) ways to express something, then no one can claim ownership of such expression by copyright. Under the names doctrine, names and short phrases are not copyrightable. Under Section 102(b), copyright protection never extends to any idea, procedure, process, system, method of operation or concept regardless of its form. Functional elements essential for interoperability are not copyrightable." (bullets omitted)). The court also noted the principle from *Feist* that "we should not yield to the temptation to find copyrightability merely to reward an investment made in a body of intellectual property." *Id.*

[163] *See id.*

[164] *Id.* at 998.

[165] *Id.*

[166] *Id.*

[167] *Id.* ("Nor can there be any copyright violation due to the *name* given to the method (or to the arguments), for under the law, names and short phrases cannot be copyrighted." (emphasis in original)).

ment: "[W]hile no single name is copyrightable, Java's overall system of organized names"[168]—in other words, the sequence, structure and organization of the API—should be found copyrightable expression. The court acknowledged that there were multiple ways that Google could have organized the Android API differently than the Java API.[169] Had Google chosen a different package/class/method organization, then, according to the rules of the Java programming language, users of the Android API would have to call a functionally identical method using a different name than used with the Java API.[170] However, the court held that such a result illustrates why the overall organization (and resulting names) are not copyrightable in the first place; because

> the names are more than just names—they are symbols in a command structure wherein the commands take the form java.package.Class.method()[.] Each command calls into action a pre-assigned function. The overall name tree, of course, has creative elements but it is also a precise command structure—a utilitarian and functional set of symbols, each to carry out a pre-assigned function.[171]

The court held this "command structure" to be a system or method of operation, and therefore explicitly barred from being copyrightable by Section 102(b) of the Copyright Act.[172]

After finding the disputed Java API code not copyrightable as a matter of law, the district court entered final judgment in favor of Google on Oracle's copyright infringement claim.[173] Oracle appealed this copyrightability decision[174] to the Court of Appeals for the Federal Circuit.[175]

E. The appellate court's copyrightability analysis

Applying its interpretation of Ninth Circuit law[176] to conduct a *de novo* review[177] of the district court's decision, the Federal Circuit panel re-

---

[168] *Id.* at 999.

[169] *See id.* at 998–99.

[170] Since a method call name in Java takes the form of "java.package.Class.method()," any change to the package/class/method hierarchy of the API necessarily results in a different method call name. Therefore, the names and the name hierarchy are inextricably linked.

[171] *Oracle*, 872 F. Supp. 2d at 976–77.

[172] *Id.* at 999–1000.

[173] Oracle Am., Inc. v. Google, Inc., 750 F.3d 1339, 1348 (Fed. Cir. 2014).

[174] *Id.*

[175] The Federal Circuit had jurisdiction over this appeal, rather than the Ninth Circuit Court of Appeals, due to the original complaint including patent infringement claims. *See id.* at 1353 (citing 28 U.S.C. § 1295(a) (2012)).

[176] *Id.* ("When the questions on appeal involve law and precedent on subjects not exclusively assigned to the Federal Circuit, [such as copyright,] the court applies the law which would be applied by the regional circuit." (internal quotations omitted)

versed the district court and held that "the declaring code and the structure, sequence, and organization of the 37 Java API packages are entitled to copyright protection."[178] Specifically, the panel found that the district court erred by "fail[ing] to distinguish between the threshold question of what is copyrightable—which presents a low bar—and the scope of conduct that constitutes infringing activity," and by "importing fair use principles, including interoperability concerns, into its copyrightability analysis."[179] The panel divided its opinion into an analysis of, first, the copyrightability of the literal elements of the API packages—the specific lines of declaring source code—and, second, the copyrightability of the non-literal elements—the overall structure, sequence, and organization (SSO) of the Java API packages.[180]

1. Copyrightability of the API's literal elements

As to the literal elements of the API, that is, the class and method declaration lines of code, the panel held that the district court had erred by misapplying the merger and short phrases doctrines to find these lines of code uncopyrightable.[181] The panel opinion took a very constrained view of merger, stating flatly that "the merger doctrine cannot bar copyright protection for any lines of declaring source code unless Sun/Oracle had only one way, or a limited number of ways, to write them."[182] The focus of the merger inquiry, the court said, should be on the options for class and method declarations "that were available to Sun/Oracle *at the time it created* the API packages," not on the options available to Google when it copied these headers into the Android API.[183] Because Oracle was not "selecting among preordained names and phrases to create its packages," and instead had "unlimited options" for the Java API class and method names, the court held that the merger doctrine simply does not apply.[184]

The panel's reasoning for rejecting the district court's application of the short phrases doctrine is thin. Although the court recognized the

---

(quoting Atari Games Corp. v. Nintendo of Am., Inc., 897 F.2d 1572, 1575 (Fed. Cir. 1990))).

[177] *Id.* at 1353 ("[I]n the Ninth Circuit, whether particular expression is protected by copyright law is 'subject to de novo review.'" (quoting Ets-Hokin v. Skyy Spirits, Inc., 225 F.3d 1068, 1073 (9th Cir. 2000))).

[178] *Id.* at 1354.

[179] *Id.*

[180] *See id.* at 1356.

[181] *See id.* at 1359–63.

[182] *Id.* at 1361 (citing Satava v. Lowry, 323 F.3d 805, 812 n.5 (9th Cir. 2003) ("Under the merger doctrine, courts will not protect a copyrighted work from infringement if the idea underlying the copyrighted work can be expressed in only one way, lest there be a monopoly on the underlying idea.")).

[183] *Id.* (emphasis added).

[184] *Id.*

Copyright Office's regulation regarding short phrases,[185] the court stated that "the relevant question . . . is not whether the work at issue contains short phrases—as literary works often do—but, rather, whether those phrases are creative."[186] Furthermore, the court pointed out that *Feist* made clear that compilations of unprotectable elements can be copyrightable, provided that the compilation exhibits the requisite level of originality.[187] The court offered the analogy of a passage from a novel:

> [T]he opening of Charles Dickens' A Tale of Two Cities is nothing but a string of short phrases. Yet no one would contend that this portion of Dickens' work is unworthy of copyright protection because it can be broken down into these shorter constituent components. The question is not whether a short phrase or series of short phrases can be extracted from the work, but whether the manner in which they are used or strung together exhibits creativity.[188]

Choosing to view the "7,000 lines" of class and method declarations as a compilation, rather than individually, the court held as follows: "Because Oracle 'exercised creativity in the selection and arrangement' of the method declarations when it created the API packages and wrote the relevant declaring code, they contain protectable expression that is entitled to copyright protection."[189]

2. Copyrightability of the API's non-literal elements

As to the non-literal sequence, structure, and organization of the Java API packages, the Federal Circuit panel held that the district court erred in finding the SSO to be an uncopyrightable "system" or "method of operation" under Section 102(b) of the Copyright Act.[190] According to the panel, the district court mistakenly appeared to have "relied on language" in an opinion from the First Circuit Court of Appeals, *Lotus Development Corp. v. Borland International, Inc.*[191] The district court's opinion did indeed discuss *Lotus* in its survey of cases decided outside the Ninth Circuit related to copyrightability of non-literal elements, such as the

---

[185] *See id.* at 1362 (citing 37 C.F.R. § 202.1(a)).

[186] *Id.* (citing Soc'y of Holy Transfiguration Monastery, Inc. v. Gregory, 689 F.3d 29, 52 (1st Cir. 2012); 1 MELVILLE B. NIMMER & DAVID NIMMER, NIMMER ON COPYRIGHT § 2.01[B] (2016)).

[187] *See id.* (citing Softel, Inc. v. Dragon Med. & Sci. Commc'ns, Inc., 118 F.3d 955, 964 (2d Cir. 1997)).

[188] *Id.* at 1363.

[189] *Id.* (quoting Atari Games Corp. v. Nintendo of Am. Inc., 975 F.2d 832, 840 (Fed. Cir. 1992)).

[190] *See id.* at 1368.

[191] *See id.* at 1364 (citing Lotus Dev. Corp. v. Borland Int'l, Inc., 49 F.3d 807 (1st Cir. 1995)).

SSO, of a computer program.[192] The district court did not actually cite *Lotus* when reaching the conclusion that the Java API SSO was not copyrightable.[193] However, it may as well have, because the copyright dispute in *Oracle* is highly analogous to the underlying issue in *Lotus*.[194]

a. The "method of operation" analysis in Lotus

The *Lotus* case involved two competing spreadsheet programs: the plaintiff's "Lotus 1-2-3," which at the time had been the market-leader, and the defendant's "Quattro" program.[195] Users of Lotus 1-2-3 controlled the program using a set of menu commands.[196] Commands were executed either by highlighting them on screen, or by typing the first letter of the command.[197] To save time when needing to repeatedly perform the same sequence of commands, users could also write and save "macros"[198]—essentially small computer programs "written in" the Lotus 1-2-3 command "language." When developing the Quattro programs, Borland purposefully copied "the words and structure of Lotus's menu command hierarchy."[199] Like Google, it did so for compatibility reasons, "so that spreadsheet users who were already familiar with Lotus 1-2-3 would be able to switch to the Borland programs without having to learn new commands or rewrite their Lotus macros."[200] The district court in *Lotus* had held that the command structure was copyrightable expression because "[a] very satisfactory menu tree can be constructed using different commands and a different command structure from those of Lotus 1-2-3."[201] In other words, because the authors of Lotus 1-2-3, at the time of its creation, had "literally millions" of possible menu trees to choose from, both the particular command names that they selected, and the particular way in which those commands were arranged, were protectable expression.[202]

On appeal, the First Circuit panel eschewed an analysis of the expressive choices made by Lotus at the time it designed the command tree

---

[192] *See* Oracle Am., Inc. v. Google, Inc., 872 F. Supp. 2d 974, 990–91 (N.D. Cal. 2012), *rev'd and remanded*, Oracle Am., Inc. v. Google, Inc., 750 F.3d 1339 (Fed. Cir. 2014).

[193] *See id.* at 999–1000 (basing holding on direct application of the text of 17 U.S.C. § 102(b)).

[194] *See Lotus*, 49 F.3d at 809 (stating the issue as "whether a computer menu command hierarchy is copyrightable subject matter").

[195] *Id.*

[196] *Id.*

[197] *Id.*

[198] *Id.* at 809–10.

[199] *Id.* at 810.

[200] *Id.*

[201] *Id.* (quoting Lotus Dev. Corp. v. Borland Int'l, Inc., 799 F. Supp. 203, 217 (D. Mass. 1992)).

[202] *See id.* at 810–11 (internal quotations omitted) (quoting *Lotus*, 799 F. Supp. at 217).

in favor of focusing on whether the command tree, as a whole, was statutorily barred from copyright protection.[203] The court held that any expression embodied in "choosing and arranging the Lotus command terms" was not copyrightable because it is part of Lotus 1-2-3's "method of operation,"[204] and is therefore, according to Section 102(b), not eligible for copyright protection.

In the *Oracle* appellate opinion, the Federal Circuit found the district court's "reliance on *Lotus*" to be misplaced because, according to the panel, "*Lotus* is inconsistent with Ninth Circuit case law recognizing that the structure, sequence, and organization of a computer program is eligible for copyright protection where it qualifies as an expression of an idea, rather than the idea itself."[205] Specifically, the court characterized *Lotus* as having set down a "hard and fast rule" that "elements which perform a function can never be copyrightable," and it rejected this rule and the *Lotus* court's Section 102(b) "method of operation" analysis as being "at odds with the Ninth Circuit's endorsement of the abstraction-filtration-comparison analysis" for copying of non-literal elements, such as the SSO, of a computer program.[206]

b. Altai *and the abstraction-filtration-comparison Analysis*

The abstraction-filtration-comparison analysis process comes from a copyright infringement case in the Second Circuit, *Computer Associates International, Inc. v. Altai, Inc.*,[207] which involved "the challenging question of whether and to what extent the 'non-literal' aspects of a computer program, that is, those aspects that are not reduced to written code, are protected by copyright."[208] In *Altai*, the plaintiff created a scheduling

---

[203] *See id.* at 816 ("*The fact that Lotus developers could have designed the Lotus menu command hierarchy differently is immaterial* to the question of whether it is a 'method of operation.' In other words, our initial inquiry is not whether the Lotus menu command hierarchy incorporates any expression. Rather, our initial inquiry is whether the Lotus menu command hierarchy is a 'method of operation.'" (emphasis added)).

[204] *Id.*

[205] Oracle Am., Inc. v. Google, Inc., 750 F.3d 1339, 1365–66 (Fed. Cir. 2014) (citing Johnson Controls, Inc. v. Phx. Control Sys., Inc., 886 F.2d 1173, 1175–76 (9th Cir. 1989)). The panel also found *Lotus* to be distinguishable on its facts for three reasons. *Id.* at 1365 ("First, while the defendant in *Lotus* did not copy any of the underlying code, Google concedes that it copied portions of Oracle's declaring source code verbatim. Second, the *Lotus* court found that the commands at issue there . . . were not creative, but it is undisputed here that the declaring code and the structure and organization of the API packages are both creative and original. Finally, while the court in *Lotus* found the commands at issue were 'essential to operating' the system, it is undisputed that . . . Google did not need to copy [the SSO] of the Java API packages to write programs in the Java language.").

[206] *Id.* at 1366.

[207] Computer Assocs. Int'l, Inc. v. Altai, Inc., 982 F.2d 693 (2d Cir. 1992).

[208] *Id.* at 696.

program that included a sub-program named "ADAPTER."[209] The ADAPTER program functioned as a "translator" between the scheduling program and the operating system of the computer on which it was running.[210] The defendant owned a competing scheduling program and eventually adopted a program structure similar to the plaintiff's, including employing a similar translator program that it called "OSCAR."[211] Originally, OSCAR was written by a former employee of the plaintiff who had literally copied significant portions of the ADAPTER source code into the OSCAR source code.[212] However, after this fact came to the defendant's attention, the defendant engaged a team to completely rewrite the OSCAR source code without having access to the ADAPTER source code.[213] The trial court found that the rewritten version of OSCAR was non-infringing.[214] Nevertheless, the plaintiff appealed, asserting that "the district court failed to account sufficiently for a computer program's non-literal elements."[215]

The Second Circuit panel recognized that copyright protection for a computer program, as for all literary works, extends beyond the literal program code to its non-literal aspects.[216] Finding the analytical approaches taken by other courts to have addressed non-literal copying to be "less than satisfactory,"[217] the Second Circuit announced a systematic three-step process for assessing substantial similarity of non-literal elements:

> [A] court would first break down the allegedly infringed program into its constituent structural parts. Then, by examining each of these parts for such things as incorporated ideas, expression that is necessarily incidental to those ideas, and elements that are taken from the public domain, a court would then be able to sift out all non-protectable material. Left with a kernel, or possible kernels, of creative expression after following this process of elimination, the court's last step would be to compare this material with the structure of an allegedly infringing program. The result of this comparison will determine whether the

---

[209] *Id.* at 698.

[210] *See id.* at 699 ("ADAPTER's function is to translate the language of a given program [e.g., CA-SCHEDULER] into the particular language that the computer's own operating system can understand.").

[211] *See id.* at 699–700.

[212] *See id.*

[213] *See id.* at 700.

[214] *Id.* at 701.

[215] *Id.*

[216] *Id.* (citing Nichols v. Universal Pictures Corp., 45 F.2d 119, 121 (2d Cir. 1930)).

[217] *Id.* at 696.

> protectable elements of the programs at issue are substantially similar so as to warrant a finding of infringement.[218]

The first step in this process, the "abstraction" step, draws from the abstractions "test"[219] described by Judge Learned Hand in *Nichols*.[220] In this step, "a court should dissect the allegedly copied program's structure and isolate each level of abstraction contained within it."[221] In the second step, "filtration," a court is to examine each identified level of abstraction and filter out all elements that are "nonprotectable expression."[222] Ideas are nonprotectable, as are elements that are "dictated by considerations of efficiency so as to be necessarily incidental to [those] idea[s]; required by factors external to the program itself; or taken from the public domain."[223] Finally, step three requires a court to compare the defendant's program to the plaintiff's "core of protectable expression"—the work's "golden nugget"—that may remain after sifting out all of the nonprotectable elements identified in step two.[224]

The *Altai* court characterized this abstraction-filtration-comparison analysis as "break[ing] no new ground; rather, it draws on such familiar copyright doctrines as merger, *scenes a faire*, and public domain."[225] The court also noted that the filtration step functions to "defin[e] the scope of [the] plaintiff's copyright."[226] The court specifically rejected an alternate approach to filtration previously used by the Third Circuit[227] in *Whelan Associates, Inc. v. Jaslow Dental Laboratory, Inc.*[228] The *Whelan* court, which had also dealt with allegations of copyright infringement of the

---

[218] *Id.* at 706.

[219] Some commentators have expressed the view that Judge Hand never intended his observations in *Nichols* to be a rigid "test." *See* Jon O. Newman, *New Lyrics for an Old Melody: The Idea/Expression Dichotomy in the Computer Age*, 17 CARDOZO ARTS & ENT. L.J. 691, 694 (1999). Judge Newman urges courts to consider using words other than "idea" and "expression" for, respectively, the unprotectable and protectable elements of a computer program, since these terms "all too readily enlist[] the doctrines and case law developed in cases involving written texts. . . ." *Id.* at 700. Others agree that different vocabulary may be useful to avoid the temptation for parties to try to get expert witnesses to label a program element an "idea." *See, e.g.*, GALLER, *supra* note 46, at 21.

[220] *Comput. Assoc.*, 982 F.2d at 706 (citing *Nichols*, 45 F.2d at 121).

[221] *Id.* at 707.

[222] *Id.*

[223] *Id.*

[224] *Id.* at 710.

[225] *Id.* at 706.

[226] *Id.* at 707 (internal quotations omitted) (quoting Brown Bag Software v. Symantec Corp., 960 F.2d 1465, 1475 (9th Cir. 1992)).

[227] *See id.* at 705–06 (discussing *Whelan*'s "mixed review" in the courts and criticism from the academic community).

[228] 797 F.2d 1222 (3d Cir. 1986).

overall structure of a computer program,[229] had expressed the test for separating unprotectable ideas from protectable expression as "the purpose or function of a utilitarian work would be the work's idea, and everything that is not necessary to that purpose or function would be part of the expression of the idea."[230] The *Altai* court rejected this test as "descriptively inadequate," primarily because it simplistically assumes that there may be only one singular idea in a program.[231] The Second Circuit's more nuanced abstraction-filtration-comparison approach for separating idea from expression, the court thought, "not only comports with, but advances the constitutional policies underlying the Copyright Act."[232]

In the *Oracle* appeal, the Federal Circuit panel noted that the Ninth Circuit has endorsed the abstraction-filtration-comparison approach,[233] and faulted the district court for not following this approach when determining whether the Java API SSO contained any protectable expression.[234] According to the panel, the district court's conclusion that the SSO was a "system or method of operation," and therefore an unprotectable, functional element of the Java API, is contrary to Ninth Circuit law that "an original work—even one that serves a function—is entitled to copyright protection as long as the author had multiple ways to express the underlying idea."[235] Based on the district court's findings that "the SSO is original and creative, and that the declaring code could have been written and organized in any number of ways and still achieved the same functions," the Federal Circuit reversed the district court and held that, under Ninth Circuit law, the SSO of the Java API was copyrightable.[236]

---

[229] *See id.* at 1224 (introducing the phrase "structure (or sequence and organization)," with respect to computer programs). The *Whelan* court used the terms "structure," "sequence," and "organization" (i.e., SSO) interchangeably. *Id.* at 1224 n.1.

[230] *Id.* at 1236 (emphasis omitted). The computer program at issue in *Whelan* was for running a dental laboratory. *See id.* at 1224. The court identified the program's purpose and, thus, its "idea," as "the efficient managing of a dental laboratory." *Id.* at 1236 n.28. According to the court, everything in the program structure not necessary to achieve this purpose was protectable expression. *Id.*

[231] *See Comput. Assocs.*, 982 F.2d at 705–06 (2d Cir. 1992).

[232] *Id.* at 711 ("The interest of the copyright law is not in simply conferring a monopoly on industrious persons, but in advancing the public welfare through rewarding artistic creativity, in a manner that permits the free use and development of non-protectable ideas and processes.").

[233] *See* Oracle Am., Inc. v. Google, Inc., 750 F.3d 1339, 1357 (Fed. Cir. 2014) (citing Sega Enters. Ltd. v. Accolade, Inc., 977 F.2d 1510, 1525 (9th Cir. 1992)).

[234] *Id.* at 1358.

[235] *Id.* at 1366–67.

[236] *Id.* at 1368.

IV. THE FEDERAL CIRCUIT'S OPINION MISCONSTRUES NINTH CIRCUIT LAW

The fundamental disagreement between the district court and the Federal Circuit panel in *Oracle* relates to how a court should draw the perpetually elusive boundary between unprotectable ideas and protectable expression.[237] The Federal Circuit characterized the district court's approach as a "two-step" analysis "wherein Section 102(a) grants copyright protection to original works, while Section 102(b) takes it away if the work has a functional component."[238] But, the proper approach, according to the Federal Circuit panel, is for Sections 102(a) and 102(b) "to be considered collectively so that certain expressions are subject to greater scrutiny."[239] Under the Federal Circuit's approach, a district court should bias its analysis in favor of protectability by initially "ferret[ing] out apparent expressive aspects of a work," then separating this protectable expression from unprotectable aspects.[240] This seems to be a fairly unimportant distinction and, if properly applied, both modes of analysis should lead to the same result.

A. Section 102(b) clearly excludes systems and methods of operation from copyright protection

Copyright law is statutory;[241] therefore, courts must apply the words of the statute. Regardless of whether Section 102(a) is considered before, or "collectively with," Section 102(b), and regardless of when in the analysis the unprotected elements of a work are "filtered" out, the plain language of Section 102(b) is clear: "*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."[242] Thus, on its face, it is apparent that Section 102(b) limits *the extent*, or *the scope*, of any copyright protection granted to a work under Section 102(a). Contrary to the Federal Circuit, this does not mean that Section 102(a) grants protection and that Section 102(b) operates to "take it away." Rather, 102(b) clarifies that copyright protection never extends to unprotectable elements of a work *in the first place*.

This copyright scope-limiting function of Section 102(b) is confirmed by its legislative history and by courts' construction of the statute.

---

[237] *See* Nichols v. Universal Pictures Corp., 45 F.2d 119, 121 (2d Cir. 1930) ("Nobody has ever been able to fix that boundary, and nobody ever can.").

[238] *Oracle*, 750 F.3d at 1356.

[239] *Id.* at 1357.

[240] *Id.*

[241] *See* 17 U.S.C. § 101 (2012).

[242] *Id.* (emphasis added).

As the district court in *Oracle* explained, at the same time the 1976 Copyright Act made explicit that computer programs are eligible for copyright protection as "literary works," the Act "also codified a *Baker*-like *limitation on the scope* of copyright protection in Section 102(b)."[243] That is, Section 102(b) codified *Baker*'s fundamental principle that any underlying idea (or procedure, process, system, method of operation, concept, principle, or discovery) is not protectable under copyright law.[244] Likewise, the Supreme Court has recognized that the exclusive rights granted to copyright holders by Section 106 are shaped and limited by the First Amendment considerations embodied in the idea/expression dichotomy and the concept of fair use.[245] The Ninth Circuit has also recognized that Section 102(b) limits the scope of copyright protection in a work.[246]

Thus, the district court's approach in *Oracle* of finding the sequence, structure, and organization of the Java API uncopyrightable as a matter of law under Section 102(b)[247] was entirely appropriate according to both Ninth Circuit and Supreme Court precedent. Contrary to the Federal Circuit panel, the district court did not find the SSO of the Java API packages uncopyrightable "just because they also perform functions."[248] Rather, the district court specifically identified the SSO as a system or method of operation, irrespective of the large number of class and method names included:

> That a system or method of operation has thousands of commands arranged in a creative taxonomy does not change its character as a method of operation. Yes, it is creative. Yes, it is original. Yes, it resembles a taxonomy. But, it is nevertheless a command structure, a system or method of operation—a long hierarchy of over six thousand commands to carry out preassigned functions. For that reason, it cannot receive copyright

---

[243] Oracle Am., Inc. v. Google, Inc., 872 F. Supp. 2d 974, 985 (N.D. Cal. 2012) (emphasis added) (citing Apple Comput., Inc. v. Microsoft Corp., 35 F.3d 1435, 1443 n.11 (9th Cir. 1994)), *rev'd and remanded*, Oracle Am., Inc. v. Google, Inc., 750 F.3d 1339 (Fed. Cir. 2014).

[244] *See id.* at 985–86 ("'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.'" (quoting H.R. REP. NO. 94-1476, at 56–57)).

[245] *See* Golan v. Holder, 132 S. Ct. 873, 890 (2012) (describing the idea/expression dichotomy and fair use as shaping the "traditional contours" of copyright protection).

[246] *See* Brown Bag Software v. Symantec Corp., 960 F.2d 1465, 1476 (9th Cir. 1992) ("To the extent a plaintiff's work is unprotected or unprotectable under copyright, the scope of the copyright must be limited.").

[247] *See Oracle*, 872 F. Supp. 2d at 1000.

[248] Oracle Am., Inc. v. Google, Inc., 750 F.3d 1339, 1368 (Fed. Cir. 2014).

protection—patent protection perhaps—but not copyright protection.[249]

B. The Federal Circuit panel's reliance on taxonomy cases is misplaced

The Federal Circuit seized on the district court's mention of a "taxonomy" to refer to a group of so-called taxonomy cases as support for its assertion that "classifying a work as a 'system' does not preclude copyright for the particular expression of that system."[250] However, the cases cited by the Federal Circuit are all distinguishable on their facts: They all had to do with numbering "systems" for parts catalogs or medical procedures,[251] not hierarchies of functional command names in a computer program. The distinction between the "systems" of names in these taxonomy cases, and a functional command name hierarchy is captured by the district court's succinct observation in *Oracle* that, in the Java API, "the names are more than just names—they are symbols in a command structure."[252]

Furthermore, despite the panel's assertion that "taxonomies, in varying forms, have generally been deemed copyrightable,"[253] several cases actually hold these types of taxonomies to be uncopyrightable subject matter.[254] Moreover, in the cases that do find a copyrightable taxonomy, the reasoning is distinguishable. For example, in *American Dental*, the Seventh Circuit held that a taxonomy of codes for dental procedures was not an uncopyrightable "system" in part because it "does not come with instructions for use."[255] In contrast, the Java API package.Class.method names *do* come with instructions for use; in fact, the names themselves *are* the instructions. A declaring line of code inherently instructs the user as to the name of the command (e.g., java.lang.Math.max), the arguments upon which this command will operate (e.g., int x, int y), and the expected form of the result or return of the command (e.g., an integer).[256]

---

[249] *Oracle*, 872 F. Supp. 2d at 999–1000.

[250] *Oracle*, 750 F.3d at 1366 (citing Toro Co. v. R & R Prods. Co., 787 F.2d 1208, 1212 (8th Cir. 1986)).

[251] *See* Practice Mgmt. Info. Corp. v. Am. Med. Ass'n, 121 F.3d 516, 517 (9th Cir. 1997) (AMA's codes "to identify particular medical procedures with precision"); Am. Dental Ass'n v. Delta Dental Plans Ass'n, 126 F.3d 977, 977 (7th Cir. 1997) (codes for dental procedures); *Toro*, 787 F.2d at 1210 (part numbers for lawn care equipment).

[252] *Oracle*, 872 F. Supp. 2d at 976.

[253] *Oracle*, 750 F.3d at 1367 n.12.

[254] *See Toro*, 787 F.2d at 1213 (holding part numbering taxonomy is not copyrightable due to lack of originality in using "arbitrary and random" numbers); *see also* ATC Distribution Grp., Inc. v. Whatever It Takes Transmissions & Parts, Inc., 402 F.3d 700, 705–06 (6th Cir. 2005) (holding part numbering system not copyrightable as to both the individual part numbers and the catalog as a whole).

[255] *Am. Dental*, 126 F.3d at 980.

[256] *Oracle*, 872 F. Supp. 2d at 986. *See supra* note 90 and accompanying text. Interestingly, when the declaring lines of code are viewed as a hierarchy of names,

Thus, at least under the logic of *American Dental*, a case cited by the Federal Circuit panel, the Java API command name hierarchy should be found to be an uncopyrightable "system."

C. *The* Lotus *approach is compatible with Ninth Circuit law*

The Federal Circuit panel's opinion in *Oracle* stated, "Notably, no other circuit has adopted the First Circuit's 'method of operation' analysis."[257] While it may be true that no circuit has *expressly* adopted the First Circuit's definition of a "method of operation" given in *Lotus*,[258] that may be simply because no other circuit has had the opportunity to do so. Other circuits, including the Ninth Circuit, have heard cases involving copyrightability of "interfaces," but these have generally been user interfaces,[259] or hardware compatibility interfaces.[260] No other circuit, except the First Circuit in *Lotus*, has had to squarely determine the copyrightability of a *command* interface, such as the menu name hierarchy in Lotus 1-2-3 and the package.Class.method name hierarchy of the Java API. And, no other circuit has expressly *rejected* the *Lotus* method of operation analysis.

Lotus is factually the closest case to *Oracle*, and the Federal Circuit's rejection of *Lotus*'s method of operation analysis as being contrary to Ninth Circuit law is misguided. The Ninth Circuit has *implicitly* accepted that at least some menu commands are not protectable at all, regardless of whether they embody some quantum of expression.[261] Furthermore, in the recent case *Bikram's Yoga College of India, L.P. v. Evolation Yoga, LLC*, decided after the Federal Circuit's *Oracle* opinion, a panel of the Ninth Circuit directly applied Section 102(b) to find a sequence of yoga poses

---

they may not even qualify as a "computer program" under the Copyright Act. Section 101 defines a "computer program" as "a set of statements or instructions to be used directly or indirectly in order to bring about a certain result." *See* 17 U.S.C. § 101. As a mere hierarchy of names, without any implementing code, the SSO itself is just a skeleton, incapable of bringing about *any* result.

[257] *Oracle*, 750 F.3d at 1366.

[258] Lotus Dev. Corp. v. Borland Int'l, Inc., 49 F.3d 807, 816 (1st Cir. 1995).

[259] These user interfaces typically include audiovisual elements. *See, e.g.*, Apple Comput., Inc. v. Microsoft Corp., 35 F.3d 1435 (9th Cir. 1994) (graphical user interface); Data E. USA, Inc. v. Epyx, Inc., 862 F.2d 204, 204–05 (9th Cir. 1988) (graphical elements of "Karate Champ" video game).

[260] *See, e.g.*, Sony Comput. Entm't, Inc. v. Connectix Corp., 203 F.3d 596, 598 (9th Cir. 2000) (basic input-output system (BIOS) for Sony Playstation); Sega Enters. Ltd. v. Accolade, Inc., 977 F.2d 1510, 1514 (9th Cir. 1992) (compatibility key for Sega Genesis); Atari Games Corp. v. Nintendo of Am. Inc., 975 F.2d 832, 835 (Fed. Cir. 1992) (protection scheme for Nintendo Entertainment System).

[261] *See* Brown Bag Software v. Symantec Corp., 960 F.2d 1465, 1472 (9th Cir. 1992) (affirming the district court's finding that a group of features that "took the form of four options in the programs' opening menus" were unprotected).

uncopyrightable subject matter.[262] Admittedly, *Bikram's Yoga* involved subject matter very different from computer software, but its copyrightability reasoning is still appropriate. Not only did the *Bikram's Yoga* court find the sequence of poses to be per se statutorily uncopyrightable as a "process," the court also held the sequence not copyrightable as a compilation.[263] The court specifically found that "[i]t makes no difference that similar results could be achieved through a different organization" of the sequence, "the [s]equence is nevertheless a process and is therefore ineligible for copyright protection."[264]

When this reasoning is applied to the Java API SSO at issue in *Oracle*, it directly contradicts the Federal Circuit's holding that because "the declaring code could have been written and organized in any number of ways and still achieved the same functions . . . Section 102(b) does not bar the packages from copyright protection."[265] Thus, it appears that the Federal Circuit misconstrued the law when it stated the rule that "under Ninth Circuit law, an original work—even one that serves a function—is entitled to copyright protection as long as the author had multiple ways to express the underlying idea."[266] On the contrary, *Bikram's Yoga* makes clear that if the subject matter is an uncopyrightable idea, process, system, or method of operation under Section 102(b), then it does not matter how many hypothetical ways there might have been for an author to organize the work; the subject matter is still uncopyrightable.

D. Courts and producers of software need improved criteria for what constitutes an uncopyrightable system or method of operation under Section 102(b)

Nevertheless, what is still needed, either from Congress or the Supreme Court, is criteria for, or a clear statement of, what constitutes an idea, process, system, or method of operation in the context of computer programs. As Judge Alsup noted, when the Supreme Court affirmed *Lotus* without opinion, it "missed the opportunity" to clarify for the courts, and for the public, when a computer program's command structure should be considered an uncopyrightable "method of operation."[267] Likewise, by denying Google's petition for certiorari in *Oracle*, the Supreme Court has missed another golden opportunity to define when a

---

[262] *See* Bikram's Yoga Coll. of India, L.P. v. Evolation Yoga, LLC, 803 F.3d 1032, 1041 (9th Cir. 2015) ("[T]he Sequence [of yoga poses] is an idea, process, or system; therefore, it is not eligible for copyright protection. That the Sequence may possess many constituent parts does not transform it into a proper subject of copyright protection.").

[263] *See id.*

[264] *Id.* at 1042.

[265] Oracle Am., Inc. v. Google, Inc., 750 F.3d 1339, 1368 (Fed. Cir. 2014).

[266] *Id.* at 1367.

[267] Oracle Am., Inc. v. Google, Inc., 872 F. Supp. 2d. 974, 992 (N.D. Cal. 2012), *rev'd and remanded*, Oracle Am., Inc. v. Google, Inc., 750 F.3d 1339 (Fed. Cir. 2014).

computer programming language's application programming interface, or other elements of a computer program, are properly considered a "system" or "method of operation" under Section 102(b).

Until the Supreme Court or Congress provides guidance on this critical application of the idea/expression dichotomy, courts are left struggling to apply old precedents to new technological circumstances. Long ago in *Baker*, the Court cautioned that it is antithetical to use copyright law to prevent people from employing the "useful knowledge" contained in a work.[268] In the dispute between Oracle and Google, Google used the "knowledge" embedded in the command name hierarchy of the Java API.[269] This allowed third parties to write programs to run on Android, an operating system for mobile phones that has been widely adopted. The Federal Circuit's decision finding the command name hierarchy copyrightable ignores *Baker*, has the potential of chilling technological progress of the kind made by Google, and runs counter to copyright law's constitutionally-directed purpose of advancing Science and the useful Arts.

[268] *See* Baker v. Selden, 101 U.S. 99, 103 (1879) ("The very object of publishing a book on science or the useful arts is to communicate to the world the useful knowledge which it contains. But this object would be frustrated if the knowledge could not be used without incurring the guilt of piracy of the book.").

[269] *See Oracle*, 750 F.3d at 1350.