GOOGLE, INC. V. ORACLE AMERICA, INC.: SUPREME COURT DECLINES TO REVIEW REVERSAL OF LANDMARK API COPYRIGHT DECISION

I. INTRODUCTION

The United States Supreme Court delivered several momentous decisions in the 2014–2015 term, but perhaps the most controversial denial of a writ for certiorari came from a Federal Circuit decision with significant potential effects for the future of technology.[1] The legal issue arose from extensive litigation between two technology colossi, Oracle America, Inc. (Oracle) and Google, Inc. (Google). Oracle alleged numerous patent and copyright infringement claims against Google.[2] Judge William Alsup in the Northern District of California heard arguments on the copyrightability of the Application Programming Interface (API) in the popular Java platform.[3] In a well-informed, fact-based decision, Judge Alsup held certain elements of the Java API to be uncopyrightable.[4] The online technology community praised Judge Alsup's opinion as "a textbook for future cases involving intellectual property rights and computer programming languages."[5]

The celebration was premature. A three-judge panel of the Federal Circuit reversed the decision and held the elements copyrightable.[6] The Supreme Court's denial leaves computer programmers in a state of confusion regarding the application of copyright law to software interfaces. More importantly for the field of intellectual property law, the denial relegates Judge Alsup's opinion to the dustbin of history.

This Note pays homage to Judge Alsup for his efforts to master computer programming concepts. Like the law, computer programs are logical sets of rules written in technical language. Like lawyers, computer programmers master the language through years of diligent practice. Lawyers and judges use the legal language to make sense of ambiguous fact patterns in order to render logical decisions. Unlike the human actions and

---

1. Google, Inc. v. Oracle Am., Inc., 135 S. Ct. 2887 (2015) (mem.), *denying cert. to* 750 F.3d 1339 (Fed. Cir. 2014).

2. *See* Oracle Am., Inc. v. Google, Inc., 872 F. Supp. 2d 974, 975–76 (N.D. Cal. 2012), *rev'd*, 750 F.3d 1339 (Fed. Cir. 2014), *cert. denied,* 135 S. Ct. 2887 (2015).

3. *Id.* at 975.

4. *Id.* at 1002.

5. Dan Farber, *Judge William Alsup: Master of the Court and Java*, CNET (May 31, 2012, 6:03 PM), http://www.cnet.com/news/judge-william-alsup-master-of-the-court-and-java/.

6. Oracle Am., Inc. v. Google, Inc., 750 F.3d 1339 (Fed. Cir. 2014), *cert. denied*, 135 S. Ct. 2887 (2015).

circumstances described in the typical fact pattern, however, computer programs are *never ambiguous*. Thus, when elements of a computer program underlie a legal issue, the basis for the legal calculus is semantic. In other words, the judge acts as a translator. Therefore, a judge's mastery of programming languages is critical to proper disposition of a legal issue involving computer program elements.

This Note proves the above thesis with three observations. First, Judge Alsup reached his legal conclusions by thoroughly explaining the programming concepts at issue and stressing the limited precedential value of his opinion.[7] Second, the Federal Circuit reversed all of Judge Alsup's legal conclusions in an opinion heavily based on legal reasoning and lacking in clear understanding of the programming concepts at issue.[8] Third, the Supreme Court failed to recognize the true import of the Federal Circuit reversal. Today's law students cannot solve tomorrow's legal issues without technology law jurisprudence of pedagogical value. The Supreme Court denied today's law students a "textbook for future cases,"[9] while leaving undisturbed an esoteric review of dated copyright jurisprudence.

Part II of this Note introduces the litigants and provides a thorough explanation of the computer programming concepts underlying the Java platform. Part III provides a background to copyright law and its application to computer programs. Part IV outlines the Federal Circuit reversal of the district court. Part V crystallizes the purpose of this Note by comparing the approaches of Judge Alsup and the Federal Circuit panel.

II. FACTS AND HOLDINGS

This Part is divided into three sections. Section A introduces the litigants and the relevant claims. Section B illustrates some basic computer programming concepts. The illustration relies principally on Judge Alsup's thorough explanation in the district

7. Oracle Am., Inc. v. Google, Inc., 872 F. Supp. 2d 974, 977–82 (N.D. Cal. 2012), *rev'd*, 750 F.3d 1339 (Fed. Cir. 2014), *cert. denied*, 135 S. Ct. 2887 (2015); *see id.* at 1002 ("This order does not hold that Java API packages are free for all to use without license. It does not hold that the structure, sequence and organization of all computer programs may be stolen. Rather, it holds on the specific facts of this case, the particular elements replicated by Google were free for all to use under the Copyright Act.")

8. *See generally* Oracle Am., Inc. v. Google, Inc., 750 F.3d 1339, 1354–76 (Fed. Cir. 2014), *cert. denied*, 135 S. Ct. 2887 (2015).

9. Farber, *supra* note 5.

court decision to highlight his mastery of the programming concepts at issue. Section C summarizes the proceedings and holdings in the district court decision, the Federal Circuit reversal, and the Supreme Court denial.

A. THE LITIGANTS AND CLAIMS

Oracle America, Inc. is a subsidiary of Oracle Corporation, a multinational computer technology company offering computer hardware systems, database management, and cloud services.[10] Oracle Corporation is one of the largest software companies in the world, earning $38 billion in revenue in fiscal year 2015.[11] Oracle America is the successor of Sun Microsystems, Inc. (Sun), which was acquired by Oracle Corporation in 2010.[12] Sun built the Java platform at issue in this case.[13]

Google is a multinational technology company whose services include web search, online advertising, cloud services, and software.[14] Google.com is the most visited website in the world.[15] Google's smartphone operating system, Android, is one of the most popular smartphone platforms in the world, used in hundreds of millions of mobile devices.[16] When it built the Android platform, Google incorporated elements of the Java API into the Android API.[17]

---

10. *Oracle Fact Sheet*, ORACLE (Oct. 2015), http://www.oracle.com/us/corporate/oracle-fact-sheet-079219.pdf.

11. *Total Cloud Revenues Up 28% but Would Have Been Up 34% in Constant Currency*, ORACLE (June 17, 2015), http://investor.oracle.com/financial-news/financial-news-details/2015/Total-Cloud-Revenues-Up-28-but-Would-Have-Been-Up-34-in-Constant-Currency/default.aspx.

12. *Company Overview of Oracle America, Inc.*, BLOOMBERG BUS., http://www.bloomberg.com/research/stocks/private/snapshot.asp?privcapId=34903 (last visited Aug. 16, 2015).

13. Jon Swartz & Leslie Cauley, *Oracle to Buy Sun for $7.4B After IBM Drops Bid*, USA TODAY (Apr. 21, 2009, 6:11 PM), http://usatoday30.usatoday.com/tech/techinvestor/corporatenews/2009-04-20-oracle-sun_N.htm. In the article, Oracle Founder and CEO Larry Ellison called Java the "single-most-important software asset we have ever acquired." *Id.*

14. *See Our Products and Services*, GOOGLE, http://www.google.com/about/company/products/ (last visited Aug. 16, 2015).

15. *The Top 500 Sites on the Web*, ALEXA, http://www.alexa.com/topsites (last visited Aug. 16, 2015).

16. *Android, The World's Most Popular Mobile Platform*, ANDROID DEVELOPERS, https://developer.android.com/about/android.html (last visited Aug. 16, 2015).

17. *See* Oracle Am., Inc. v. Google, Inc., 872 F. Supp. 2d 974, 975–76 (N.D. Cal. 2012), *rev'd*, 750 F.3d 1339 (Fed. Cir. 2014), *cert. denied*, 135 S. Ct. 2887 (2015).

Oracle sued Google shortly after the merger with Sun, alleging patent and copyright infringement of its Java platform.[18] The subject matter of Oracle's copyright infringement claims were thirty-seven packages in the Java API,[19] comprised of roughly seven thousand lines of source code.[20] In his disposition of Oracle's claims, Judge Alsup began with a thorough explanation of the thirty-seven packages.[21] His explanation was central to his holdings. Therefore, the next section lays out a firm foundation of computer programming basics to enable a clear understanding of his holdings.

B. TECHNOLOGY BACKGROUND—THE FOUNDATION

For clarity, this section is subdivided into four subsections. Subsection 1 is an explanation of general computer programming concepts. Subsection 2 explains the Java programming language. Subsection 3 explains the Java platform. Subsection 4 continues the discussion of the Java API in the context of this case. These explanations introduce a computer-programming lexicon. Though highly technical, a careful reading of these explanations is essential to understanding the courts' decisions.

1. COMPUTER PROGRAMMING BASICS

A computer is an electronic machine that processes data.[22] A "machine" is a set of parts arranged to transmit energy in a predetermined way.[23] "Electronic" refers to the use of electrical parts.[24] By deduction, "electronic machines" are sets of electrical parts arranged to transmit electricity in a predetermined way. Whereas an electrical machine (a toaster, for example) transforms electric current into another form of energy (heat),

18. Oracle Am., Inc. v. Google, Inc., 872 F. Supp. 2d 974, 975 (N.D. Cal. 2012), *rev'd*, 750 F.3d 1339 (Fed. Cir. 2014), *cert. denied*, 135 S. Ct. 2887 (2015). Judge Alsup describes this case as the "first of the so-called 'smartphone war' cases tried to a jury." *Id.*

19. *Id.* at 975–76.

20. Oracle Am., Inc. v. Google, Inc., 750 F.3d 1339, 1361 (Fed. Cir. 2014), *cert. denied*, 135 S. Ct. 2887 (2015).

21. *Oracle*, at 977–82.

22. *See* D.S. MALIK, C++ PROGRAMMING: FROM ANALYSIS TO PROGRAM DESIGN 3 (4th ed. 2009); *Computer*, MERRIAM-WEBSTER, http://www.merriam-webster.com/dictionary/computer (last visited Mar. 11, 2016).

23. *See Machine*, MERRIAM-WEBSTER, http://www.merriam-webster.com/dictionary/machine (last visited Mar. 11, 2016).

24. *See Electronic*, MERRIAM-WEBSTER, http://www.merriam-webster.com/dictionary/electronic (last visited Mar. 11, 2016).

electronic machines manipulate the electric current to process data.[25] "Data" is simply raw information.[26]

Therefore, a computer is a set of electrical parts arranged to process information.[27] The systematic arrangement in parts is referred to as the computer's "architecture."[28] Computer architectures consist of three main divisions: the central processing unit (CPU), memory, and input/output (I/O) devices.[29] The CPU processes information, the memory stores information, and the I/O devices communicate information.[30]

Computers communicate information in the form of electronic signals.[31] Electronic signals are detectable impulses in electric current.[32] In a computer, the CPU detects electronic signals with switches called "transistors."[33] Transistors are like light switches that turn on and off.[34] The CPU receives a signal from an "input" I/O device such as a keyboard.[35] The CPU then sends a signal to memory to retrieve a set of instructions called a "program."[36] Unlike the parts mentioned heretofore, classified as "hardware," a program is classified as "software."[37] The CPU decodes and executes the instructions and sends a signal to an

---

25. *See* Doug Lowe, *What Is the Difference Between Electronic and Electrical Devices?*, FOR DUMMIES, http://www.dummies.com/how-to/content/what-is-the-difference-between-electronic-and-elec.html (last visited Mar. 11, 2016).

26. *See Data*, MERRIAM-WEBSTER, http://www.merriam-webster.com/dictionary/data (last visited Mar. 11, 2016); *Data vs. Information*, DIFFEN, http://www.diffen.com/difference/Data_vs_Information (last visited Mar. 11, 2016).

27. *See* MALIK, *supra* note 22, at 3.

28. *See Computer Architecture*, PC MAG. ENCYCLOPEDIA, http://www.pcmag.com/encyclopedia/term/40139/computer-architecture (last visited Mar. 11, 2016).

29. *See* MALIK, *supra* note 22, at 4–6; JOHN VON NEUMANN, FIRST DRAFT OF A REPORT ON THE EDVAC 1–3 (Michael D. Godfrey ed., 2011) (1945) https://web.archive.org/web/20130314123032/http://qss.stanford.edu/~godfrey/vonNeumann/vnedvac.pdf (outlining these fundamental elements in a project to build one of the earliest computers).

30. *See* MALIK, *supra* note 22, at 4–6; VON NEUMANN, *supra* note 29, at 1–3.

31. DAVID A. PATTERSON & JOHN L. HENNESSY, COMPUTER ORGANIZATION AND DESIGN: THE HARDWARE/SOFTWARE INTERFACE 11 (4th ed. 2012).

32. *See* ROLAND PRIEMER, INTRODUCTORY SIGNAL PROCESSING 1 (1991).

33. *See* PATTERSON & HENNESSY, *supra* note 31, at 26.

34. *Id.* at 26.

35. *Id.* at 15.

36. ANDREW S. TANENBAUM, STRUCTURED COMPUTER ORGANIZATION 54 (5th ed. 2006); *see also* MALIK, *supra* note 22, at 30 ("A computer program or a program is a sequence of statements whose objective is to accomplish a task.").

37. *See* MALIK, *supra* note 22, at 4, 6.

"output" I/O device such as a computer screen.[38]

As shown, a computer cannot function without the interoperability of its hardware and software components. Components interact at points of contact called "interfaces."[39] To illustrate the ubiquity of interfaces, consider the example of a user who desires to type the letter "A" in a Microsoft Word document. The user initiates the command by pressing the "A" key on the keyboard. The depressed "A" key contacts an electrical component, sending an electronic signal to a piece of hardware called a "keyboard controller."[40] The keyboard controller sends an electronic signal called an "interrupt" to the CPU, requesting software called the "operating system."[41] The operating system is the core program of a computer, whose functions include managing hardware resources and relaying messages between hardware and software.[42] The operating system uses a "device driver" to translate the signals between the technical specifications of a hardware device and the abstract language of application software.[43] In this example, the operating system's keyboard device driver translates the signal that the "A" key was pressed. Microsoft Word provides instructions for how to display the "A" character on the screen. The operating system then uses another device driver to translate the instructions to the screen.

In the above example, an interface exists at each point where a signal is translated. So, the keyboard is an interface between the user and the computer hardware. The keyboard controller is an interface between the keyboard and the CPU. The device drivers allow the operating system to interface between I/O devices and software. The interface between the user and software is called the "graphical user interface" (GUI) that displays on the computer screen.[44] The GUI consists of the

38. *How Computers Work: The CPU and Memory*, U.R.I., http://homepage.cs.uri.edu/faculty/wolfe/book/Readings/Reading04.htm (last visited Mar. 13, 2016).

39. *Interface*, DICTIONARY.COM, http://www.dictionary.com/browse/interface (last visited Mar. 11, 2016).

40. *See Keyboard Controller*, PC MAG. ENCYCLOPEDIA, http://www.pcmag.com/encyclopedia/term/45789/keyboard-controller (last visited Mar. 11, 2016).

41. *See* TANENBAUM, *supra* note 36, at 102–06.

42. PATTERSON & HENNESSY, *supra* note 31, at 10.

43. *See Device Driver*, SEARCHENTERPRISEDESKTOP, http://searchenterprisedesktop.techtarget.com/definition/device-driver (last visited Mar. 11, 2016).

44. *GUI—Graphical User Interface*, WEBOPEDIA, http://www.webopedia.com/TERM/G/Graphical_User_Interface_GUI.html (last visited Mar. 11, 2016).

windows, icons, menus, and mouse.[45] Even the plug that connects the computer to the electrical socket is an interface. Clearly, a user cannot use a computer without interfaces.

To write instructions a CPU can understand, a user must communicate "on" or "off" directly to transistors.[46] To do this, computer programmers use a binary alphabet of "bits."[47] A bit takes a value of 0 or 1 to represent "off" or "on."[48] Early computer programmers wrote instructions in bits to execute functions we would consider very simple today, such as adding two numbers.[49] To execute complex functions, requiring more bits, programmers developed programming languages.[50] A programming language is a set of symbols and syntax rules used to express strings of bits.[51] Programs called "compilers" convert the programming language into an intermediate language called "assembly language," which programs called "assemblers" convert into bits prior to execution.[52] Programmers refer to code written in bits as "machine code."[53] Code written in a programming language is called "source code."[54]

Like users, programmers rely on interfaces to create application software for computers, because interfaces allow for interoperability. A programmer writes an application for a given operating system to allow interoperability between the application and computer hardware.[55] If the programmer tries to run the same application on a different operating system, the application will be incompatible and will not run. The source code must be re-written to be compatible, forcing developers to create as many versions of their applications as there are operating systems.

The Java platform solved this problem.[56] The Java platform

---

45. *GUI—Graphical User Interface*, *supra* note 44.

46. *See* PATTERSON & HENNESSY, *supra* note 31, at 11, 26.

47. *Id.* at 11.

48. *See id.*

49. *See id.*

50. *Id.* at 11–12.

51. *Id.* at 11.

52. PATTERSON & HENNESSY, *supra* note 31, at 11–12.

53. *See* Oracle Am., Inc. v. Google, Inc., 872 F. Supp. 2d 974, 977 (N.D. Cal. 2012), *rev'd*, 750 F.3d 1339 (Fed. Cir. 2014), *cert. denied*, 135 S. Ct. 2887 (2015).

54. *Id.*

55. *See* MALIK, *supra* note 22, at 6; PATTERSON & HENNESSY, *supra* note 31, at 10.

56. *Oracle*, 872 F. Supp. 2d at 977.

allows programmers to write a single version of source code that is compatible across software platforms.[57] Java's slogan sums it up: "Write once, run anywhere."[58] Because the Java platform allows for compatibility across operating systems, Java is one of the most popular programming languages in the world.[59] An explanation of the Java programming language is followed by an explanation of the Java platform.

2. THE JAVA PROGRAMMING LANGUAGE

The Java programming language is a set of symbols and syntax rules. The Java syntax includes "separators" ({ , } , ;), "operators" (+ , - , * , / , < , >), "literal values" (100, 't', "arrival"), "keywords" (if, else, while, return), and "identifiers" (String, java.lang.Object).[60] Programmers use these syntatic elements to write source code.[61]

The Java language is classified as an object-oriented programming language.[62] Object-oriented programming refers to the storing of information in "software objects."[63] Software objects, like any objects, can be described by their state and their behavior.[64] In Java, an object's state is stored in "fields" and its behavior is stored in "methods."[65]

For example, consider a train travelling between stations, loading and unloading at each station. The state of this train at any given time can be described by its speed, number of cars, and weight. Its state changes depending on engine power, the adding or removing of cars, and changes in onboard tonnage. The train can be described in Java as an object with fields "speed," "cars," and "weight." Its methods could be named "enginePower," "carChange," and "tonnageChange." The methods are the

57. Oracle Am., Inc. v. Google, Inc., 872 F. Supp. 2d 974, 977 (N.D. Cal. 2012), *rev'd*, 750 F.3d 1339 (Fed. Cir. 2014), *cert. denied*, 135 S. Ct. 2887 (2015).

58. Oracle Am., Inc. v. Google, Inc., 750 F.3d 1339, 1348 (Fed. Cir. 2014), *cert. denied*, 135 S. Ct. 2887 (2015).

59. *Oracle*, 872 F. Supp. 2d at 977.

60. *See id.* at 979.

61. *See id.* at 977, 979.

62. *See Lesson: Object-Oriented Programming Concepts*, JAVA™ TUTORIALS, https://docs.oracle.com/javase/tutorial/java/concepts/index.html (last visited Mar. 11, 2016).

63. *What Is an Object?*, JAVA™ TUTORIALS, https://docs.oracle.com/javase/tutorialjava/concepts/index.html (last visited Mar. 11, 2016).

64. *Id.*

65. *Id.*

interface between the object and the outside world, translating user inputs into field changes using the method's internal rules.

Note, this object could describe all trains, not just this train. In Java, this train is called an "instance" of a "class" of objects with the common set of characteristics listed above.[66] The class is the basic building block of programs written in Java.[67] A class acts as a blueprint for the creation of objects by implementing the internal rules of methods that translate inputs into field changes.[68] A class can also create instances of another class and invoke its methods to create unique objects from the other class.[69]

An example of an implementation of the Train class illustrates the above concepts. A second example will declare a new class creating instances of the Train class implementation by invoking its methods. Comments appear behind the "//" mark (Java syntax instructing the computer to ignore the comment). In these examples, "int" is an identifier indicating that a value is an integer, "public" is an identifier allowing other classes to "call" the declared method, brackets ({ , }) enclose implementations, semicolons separate implementation statements, and parentheses enclose variables that can take inputs.

66. *See What Is a Class?*, JAVA™ TUTORIALS, https://docs.oracle.com/javase/tutorial/java/concepts/index.html (last visited Mar. 11, 2016).

67. *See id.* Judge Alsup elaborates:

> Classes are a fundamental structural element in the Java language. A Java program is written as one or more classes. More than one method can be in a class. . . . All code in a Java program must be placed in a class. A class declaration (or header) is a line that includes the name of the class and other information that define the class. The body of the class includes fields and methods, and other parameters.

Oracle Am., Inc. v. Google, Inc., 872 F. Supp. 2d 974, 980 (N.D. Cal. 2012), *rev'd*, 750 F.3d 1339 (Fed. Cir. 2014), *cert. denied*, 135 S. Ct. 2887 (2015).

68. *See What Is a Class?*, *supra* note 66; *What is an Object?*, *supra* note 63.

69. *See What Is a Class?*, *supra* note 66.

Program 1

```
class Train {                                          // declares class Train

     int speed = 0;                                    // declares and sets field
     int cars = 5;                                     // declares and sets field
     int weight = 1,000;                               // declares and sets field

     // Next, the methods are declared and implemented.

     public enginePower (int newValue) {               // declares method
                speed = newValue*10;                   // implements method
     }                                                 // closes method

     // The above method declares a public method named "enginePower" that takes
     // integer inputs in a variable named "newValue".  The method
     // implementation changes the value of the "speed" field by multiplying
     // newValue by 10.  For example, if another class calls this method with an
     // input of 3, the method will set the "speed" field to 30.

     // With the above comments as a model, study the next two methods.

     public carChange (int increment) {
          cars = cars + increment;
     }

     // Take note of "increment," which adds an input to the previous value of
     // "cars" to obtain a new value.  For example, an input of 2 would add 2 to
     // the initial value of "cars", 5 (see above), for a new value of 7.

     public tonnageChange (int increment) {
          weight = weight + increment;
     }

     // This method tells the class how to print outputs.
     public printStates( ) {
          system.out.println ( "Speed:" + speed + " Cars:" + cars +
              " Weight:" + weight);
     }
}
```

Next, a new class named "MyJavaTrains" will call the methods in the Train class.

Program 2

```
class MyJavaTrains {
     // First, the main method is declared, telling Java to execute.
     public static void main ( String[ ] args) {

      // two Train instances created, "train1" and "train2".
      Train train1 = new Train( );
      Train train2 = new Train( );

      // Train methods are invoked by both objects, with inputs.
      train1.enginePower(10);
      train1.carChange(1);
      train1.tonnageChange(100);
      train1.printStates( );

      train2.enginePower(15);
      train2.carChange(3);
      train2.tonnageChange(500);
              train2.printStates( );
     }
}

     // Outputs printed according to Train print instructions.
     speed:100  cars:6  weight:1,100
     speed:150  cars:8  weight:1,500.
```

The above examples illustrate how a programmer declares methods that can be invoked elsewhere to do work.[70] First, the programmer declares a class ("Train") to store fields and methods. In the class, the programmer sets fields ("speed," "cars," and "weight") equal to their initial values (10, 5, and 1,000), then declares methods ("enginePower," "carChange," and "tonnageChange"). The methods consist of two parts: a "method header" declaring the method:

```
public enginePower (int newValue)
```

70. *See* Oracle Am., Inc. v. Google, Inc., 872 F. Supp. 2d 974, 979 (N.D. Cal. 2012), *rev'd*, 750 F.3d 1339 (Fed. Cir. 2014), *cert. denied*, 135 S. Ct. 2887 (2015).

and a "method body" implementing the method:

```
{speed = newValue*10;}.[71]
```

The programmer declares a second class ("MyJavaTrains") to create new instances of the Trains class. The MyJavaTrains class has the "main" method, Java's built-in method for program execution. The programmer creates new objects with unique names ("train1" and "train2") and invokes the Train methods with inputs in parentheses. The method implementations in the Train class determine the outputs of the objects in the MyJavaTrains class.

With the above programming language concepts in mind, the next subsection describes the Java platform.

3. THE JAVA PLATFORM

Now for an explanation of the Java platform. The Java platform is comprised of the Java Virtual Machine and the Java API.[72] Java architecture capitalizes on four technologies: (1) the Java programming language, (2) the Java class file format, (3) the Java API, and (4) the Java Virtual Machine.[73]

The Java programming language, detailed in Subsection 2, provides programmers a standard for writing programs compatible across platforms.[74] The Java class file format is another standard that ensures compatibility between programs and the Java API.[75] As defined in Subsection 1, a compiler converts a program's source code into machine code.[76] By contrast, the Java compiler converts a program's source code into class files.[77] Class files are written in a language called "bytecode," the "machine" code of the Java Virtual Machine.[78]

The Java Virtual Machine is an "abstract computer" that

---

71. *See* Oracle Am., Inc. v. Google, Inc., 872 F. Supp. 2d 974, 979–80 (N.D. Cal. 2012), *rev'd*, 750 F.3d 1339 (Fed. Cir. 2014), *cert. denied*, 135 S. Ct. 2887 (2015).

72. Bill Venners, *Inside the Java Virtual Machine: Introduction to Java's Architecture*, ARTIMA DEVELOPER, http://www.artima.com/insidejvm/ed2/introarchP.html (last visited Mar. 14, 2016).

73. *Id.*

74. *Id.*

75. *See id.*

76. *See supra* text accompanying notes 51–54.

77. Venners, *supra* note 72.

78. *Id.*

simulates computer functions.[79] It consists of two components: the "class loader" and the "execution engine."[80] The class loader loads class files from a compiled program and from the Java API.[81] The execution engine executes the bytecode.[82] By simulating a computer, the Java Virtual Machine provides a standard platform to run programs, no matter where the Java Virtual Machine is implemented.[83]

The Java API makes this possible by providing a "standard way to access the system resources of a host computer."[84] The Java API is a library of pre-written classes of methods organized into "packages."[85] Judge Alsup extended the library analogy:

> An API is like a library. Each package is like a bookshelf in the library. Each class is like a book on the shelf. Each method is like a how-to-do-it chapter in a book. Go to the right shelf, select the right book, and open it to the chapter that covers the work you need.[86]

The Java API contains the methods necessary to make functional use of a host computer's "native" resources.[87] In fact, three of the packages are considered "core" packages whose absence would render the Java language practically useless.[88]

Now that the components of the Java platform are defined, consider how they work together to benefit programmers and users. A programmer who wants to create an application that any user can use writes the application source code in Java. The source code calls methods contained in the Java API to access the host computer's native resources, called "native methods."[89] The Java compiler compiles the source code into class files.[90] Those class files can be run on any computer that implements the Java

---

79. Venners, *supra* note 72.
80. *Id.*
81. *Id.*
82. *Id.*
83. *See id.*
84. *Id.*
85. *See* Oracle Am., Inc. v. Google, Inc., 872 F. Supp. 2d 974, 977, 982 (N.D. Cal. 2012), *rev'd*, 750 F.3d 1339 (Fed. Cir. 2014), *cert. denied*, 135 S. Ct. 2887 (2015).
86. *See id.* at 977.
87. Venners, *supra* note 72.
88. *See Oracle*, 872 F. Supp. 2d at 982.
89. *See* Venners, *supra* note 72.
90. *See id.*

platform.[91]

To run Java applications, a user downloads the Java platform and stores it in memory.[92] When the user opens the programmer's application, the user's operating system launches the Java platform.[93] The Virtual Machine's class loader loads the program's class files and the API class files that were called in the program.[94] In turn, the Java API calls the necessary native methods from the user's operating system.[95] In this way, the Java API interfaces between the application and the operating system. The programmer can specify a particular function in one way, and the Java API acts as translator by calling the native methods unique to the operating system.

The class loader loads the class files from the program and the Java API, and the execution engine executes the bytecode.[96] The application runs smoothly in accord with the user's specific hardware and operating system.

Application developers can market the application to all users who implement the Java platform—increasing revenues—without needing to write new versions of the application—reducing costs. Users can download the application without worrying about compatibility with particular hardware or software. In short, the Java platform benefits developers and users alike by allowing for interoperability.

4. THE SUBSTANCE OF ORACLE'S CLAIM

A further expounding of the Java API is warranted to understand the substance of Oracle's claim. This explanation draws upon Judge Alsup's opinion.

When a programmer calls a method from the Java API, the programmer must type the call in the specific format:

```
java.package.Class.method( ).
```
[97]

---

91. *See* Venners, *supra* note 72.
92. *See id.*
93. *See id.*
94. *See id.*
95. *See id.*
96. *See id.*
97. *See* Oracle Am., Inc. v. Google, Inc., 872 F. Supp. 2d 974, 976 (N.D. Cal. 2012), *rev'd*, 750 F.3d 1339 (Fed. Cir. 2014), *cert. denied*, 135 S. Ct. 2887 (2015).

For example, consider the "max" method. The max method is located in the "Math" class of the "java.lang" package.[98] To call the max method, a programmer must write:

```
java.lang.Math.max (int x, int y).
```
[99]

The format of the call expresses the hierarchical structure of the API.[100] A programmer cannot access the desired method unless the format is replicated exactly.[101] For this reason, Google copied the names and organization of certain Java API packages into its Android API.[102] As a result, Oracle claimed copyright infringement of its package, class, and method names and their organization, as embodied in the declarations and calls associated with the packages.[103] Importantly, Google *did not* copy the implementations of the methods in the thirty-seven packages, just the names and the declarations.[104]

To illustrate, consider this possible implementation of the max method:

```
package java.lang;                              //declares package.
public class Math {                             //declares class.
     public static int max (int x, int y) {  //declares method.
       if (x > y) return x;                     //implementation.
       else return y;                           //implementation.
     }
}
```

Google copied the three declaring lines. Google did not copy the implementations. The difference lies in the fact that the declaring lines must be *identical* to allow programs written in Java to access the same computer functions.[105] The implementation could be written in another way to achieve the same outputs.[106] That is why Google copied the declaring code

98. *See* Oracle Am., Inc. v. Google, Inc., 872 F. Supp. 2d 974, 981 (N.D. Cal. 2012), *rev'd*, 750 F.3d 1339 (Fed. Cir. 2014), *cert. denied*, 135 S. Ct. 2887 (2015).

99. *See id.*

100. *See id.* at 976–77.

101. *See id.* at 978.

102. *See id.* at 977.

103. *See id.* at 978.

104. *See* Oracle Am., Inc. v. Google, Inc., 872 F. Supp. 2d 974, 978 (N.D. Cal. 2012), *rev'd*, 750 F.3d 1339 (Fed. Cir. 2014), *cert. denied*, 135 S. Ct. 2887 (2015).

105. *See id.* at 979.

106. *See id.* at 978.

and not the implementations.[107]

Upon its release in 1996, the Java API contained eight packages.[108] As mentioned, three of the API packages were core packages whose absence would render the language useless.[109] The number of packages proliferated as Sun, collaborating with the Java Community Process, wrote hundreds of programs that execute "nifty functions," which were subsequently organized into the packages composing the Java API today.[110] The API in the Java 2 SE 5.0 platform (the edition at issue in this case) contained 166 packages composed of more than 600 of over 6,000 methods.[111] Thirty-seven of those 166 packages are at issue in this litigation, including the three core packages.[112]

C. JUDGE ALSUP'S REJECTION OF ORACLE'S COPYRIGHT INFRINGEMENT CLAIM

Oracle's copyright infringement claim centered on approximately 7,000 lines of source code in the Android API that Google copied verbatim from the Java API.[113] After the 2005 purchase of Android Inc., Google opted to adapt the Java platform to the Android software, for which it sought a license from Sun.[114] Despite extensive negotiations, the parties failed to agree on licensing terms, so Google decided to build its own platform for Android written in the Java programming language.[115] In the process, Google created its own virtual machine and API class library composed of 168 packages.[116] Google sought to re-create the functions of the thirty-seven Java API packages at issue by

107. *See* Oracle Am., Inc. v. Google, Inc., 872 F. Supp. 2d 974, 997 (N.D. Cal. 2012), *rev'd*, 750 F.3d 1339 (Fed. Cir. 2014), *cert. denied*, 135 S. Ct. 2887 (2015).

108. *Id.* at 982.

109. *See id.* These packages are "java.lang", "java.io", and "java.util". *Id.*

110. *Id.* The Java Community Process is a mechanism that allows interested parties to contribute to the development of standard specifications of the Java API library. *See Welcome to the Java Community Process*, JAVA COMMUNITY PROCESS, https://www.jcp.org/en/home/index (last visited Jan. 28, 2016).

111. *Oracle*, 872 F. Supp. 2d at 977, 979 ("This is very close to saying the Java API had 166 'folders' (packages), all including over six hundred pre-written programs (classes) to carry out a total of over six thousand subroutines (methods).").

112. *Id.* at 978–79.

113. *See* Oracle Am., Inc. v. Google, Inc., 872 F. Supp. 2d 974, 975–76 (N.D. Cal. 2012), *rev'd*, 750 F.3d 1339 (Fed. Cir. 2014), *cert. denied*, 135 S. Ct. 2887 (2015).

114. *Id.* at 978.

115. *Id.* Google, and the public, are free to use the Java programming language to create new software. *See id.*

116. *See id.*

writing its own implementations of the methods in those packages.[117] To allow programmers to adapt software written in Java to the Android API, Google copied the class and method declarations in the thirty-seven packages.[118] Thus, of fifteen million lines of code, Google was accused of copying 7,000 lines[119]—roughly three percent of the source code in the thirty-seven packages.[120]

Oracle accused Google of copyright infringement of the 7,000 lines and the command structure they embody.[121] A brief discussion of Oracle's licensing scheme for the Java platform will elucidate its claim.

Oracle offers three licenses for use of its Java platform: (1) a General Public License, (2) a Specification License, and (3) a Commercial License.[122] The General Public License allows the licensee unrestricted use of the library, free of charge, but the licensee must allow the same terms for its innovations.[123] The Specification License allows the licensee to copy the declaring code and organization of the library, but the licensee must write original implementations.[124] The Commercial License allows unrestricted use of the library for a proprietary purpose in return for payment of royalties.[125]

Because Google and Sun failed to reach an agreement, Oracle and Google never had a licensing agreement for the Java platform.[126] Google's Android platform is open source and free of charge to consumers.[127] As such, it is not clear which of the three

---

117. Oracle Am., Inc. v. Google, Inc., 872 F. Supp. 2d 974, 978 (N.D. Cal. 2012), *rev'd*, 750 F.3d 1339 (Fed. Cir. 2014), *cert. denied*, 135 S. Ct. 2887 (2015).

118. *See id.*

119. Oracle Am., Inc. v. Google, Inc., 750 F.3d 1339, 1353 (Fed. Cir. 2014), *cert. denied*, 135 S. Ct. 2887 (2015).

120. *Oracle*, 872 F. Supp. 2d at 979, 983.

121. *See id.* at 975.

122. *Oracle*, 750 F.3d at 1350.

123. *Id.*

124. *Id.*

125. Oracle Am., Inc. v. Google, Inc., 750 F.3d 1339, 1350 (Fed. Cir. 2014), *cert. denied*, 135 S. Ct. 2887 (2015).

126. *Id.*

127. *See* Charles Arthur & Samuel Gibbs, *The Hidden Costs of Building an Android Device*, GUARDIAN (Jan. 23, 2014, 11:44 AM), http://www.theguardian.com/technology/2014/jan/23/how-google-controls-androids-open-source (noting, however, that there are some hidden costs manufacturers must pay to gain a license to use certain Google services on devices).

licenses is most appropriate for Google. Nevertheless, Oracle's claim parallels the terms set forth in its Specification License.[128] By inference, Oracle's claim for copyright infringement of the declaring code and organization of the library effectively amounts to a request for a judicial declaration that Google should have acquired a Specification License.

Oracle argued the declaring code and the organization of the thirty-seven Java API packages were creative works under the Copyright Act.[129] Judge Alsup rejected this argument, holding the declaring code and organization of the thirty-seven packages to be uncopyrightable.[130]

In his reasons for decision, Judge Alsup held that all are free to write their own implementations of the functions specified by the methods in the Java API.[131] The declarations can be identical because they *have to be* identical to allow a programmer to call a method with the same functionality.[132] Even though Google could have used different names for the packages, classes, and methods in the Android API, copyright never extends to names or short phrases as a matter of law.[133] And even though the overall name tree has creative elements and could have been arranged differently, Judge Alsup held it is also a "utilitarian and functional set of symbols, each to carry out a pre-assigned function. Duplication of the command structure is necessary for interoperability."[134] He held the command structure to be an uncopyrightable system or method of operation under 17 U.S.C. § 102(b).[135]

Judge Alsup pointed to the broad scope of Oracle's claim, noting that 129 of the 166 packages were not violated and, of the remaining thirty-seven, ninety-seven percent of the lines were not

---

128. *See* Oracle Am., Inc. v. Google, Inc., 750 F.3d 1339, 1350 (Fed. Cir. 2014) ("[T]he Specification License . . . provides that the licensee can use the declaring code and organization of Oracle's API packages but must write its own implementing code."), *cert. denied*, 135 S. Ct. 2887 (2015).

129. *See* Opening Brief and Addendum of Plaintiff-Appellant at 34–35, *Oracle*, 750 F.3d 1339 (No. 13-1021), 2013 WL 518611.

130. Oracle Am., Inc. v. Google, Inc., 872 F. Supp. 2d 974, 1001–02 (N.D. Cal. 2012), *rev'd*, 750 F.3d 1339 (Fed. Cir. 2014), *cert. denied*, 135 S. Ct. 2887 (2015).

131. *Id.* at 976.

132. *Id.*

133. *Id.*

134. *Id.* at 976–77.

135. *Id.* at 977.

copied.[136] Because the remaining three percent that formed the substance of Oracle's claim fell under exceptions to copyright, Oracle limited its claim to copyright infringement of the command structure.[137] However, this claim amounted to a claim over *all possible implementations* of the Java command structure, though Oracle only copyrighted a single implementation:

> To accept Oracle's claim would be to allow anyone to copyright one version of code to carry out a system of commands and thereby bar all others from writing their own different versions to carry out all or part of the same commands. No holding has ever endorsed such a sweeping proposition.[138]

Judge Alsup stressed the limited scope of the ruling:

> This order does not hold that Java API packages are free for all to use without license. It does not hold that the structure, sequence and organization of all computer programs may be stolen. Rather, it holds on the specific facts of this case, the particular elements replicated by Google were free for all to use under the Copyright Act.[139]

Judge Alsup's bench decision was one half of a bifurcated trial procedure in which a jury was instructed to assume that the API packages were copyrightable.[140] In the jury trial, the jury found in favor of Oracle on the infringement claim, but hung on Google's fair use defense.[141] On appeal of both parties, the Federal Circuit had jurisdiction over the case due to unrelated patent claims.[142] Applying Ninth Circuit law, the Federal Circuit reversed the lower court's copyrightability ruling.[143] Holding Java's unique declaring code and underlying "structure, sequence, and organization" (SSO) copyrightable works of authorship under 17 U.S.C. § 102(a), the Federal Circuit remanded the case to the district court for a retrial of Google's

---

136. Oracle Am., Inc. v. Google, Inc., 872 F. Supp. 2d 974, 1001 (N.D. Cal. 2012), *rev'd*, 750 F.3d 1339 (Fed. Cir. 2014), *cert. denied*, 135 S. Ct. 2887 (2015).

137. *Id.*

138. *Id.* at 1001–02.

139. *Id.* at 1002.

140. *Id.* at 975.

141. *Id.* at 976.

142. Oracle Am., Inc. v. Google, Inc., 750 F.3d 1339, 1353 (Fed. Cir. 2014) (citing 28 U.S.C. § 1295(a)(1)), *cert. denied*, 135 S. Ct. 2887 (2015).

143. *Id.* at 1381 (citing 28 U.S.C. § 1295(a)(1)).

fair use defense.[144] With the Supreme Court denial, the copyrightability of APIs—and software interfaces more generally—remains an unresolved issue.

III. LEGAL BACKGROUND

This part provides the legal background to copyright and computer programs in two sections. Section A covers general copyright law and its adaptation to the rise of computer programs. Section B surveys the different approaches taken by circuit courts to analyze nonliteral elements of expression under copyright law. Ninth Circuit law governed this case, so Section B emphasizes Ninth Circuit jurisprudence.

A. COPYRIGHT LAW AND THE RISE OF COMPUTERS

The United States Constitution expressly delegates to Congress the authority to "promote the Progress of Science and useful Arts, by securing for limited Times to Authors and Inventors the exclusive Right to their respective Writings and Discoveries."[145] In providing for patent and copyright law in a single clause, the Constitution encodes the distinction between patent and copyright law as the difference between the writings of authors and the discoveries of inventors.

A century after the nation's founding, the Supreme Court drew a jurisprudential line between patent and copyright in *Baker v. Selden*.[146] The *Baker* Court analyzed a copyright claim by the creator of a double-entry bookkeeping system.[147] The accused infringer published a unique version of the claimant's bookkeeping system using the same underlying system.[148] In rejecting the claim, the Court held the underlying bookkeeping system to be an uncopyrightable method.[149]

The Copyright Act of 1976 codified the distinction articulated

144. Oracle Am., Inc. v. Google, Inc., 750 F.3d 1339, 1348 (Fed. Cir. 2014) (citing 28 U.S.C. § 1295(a)(1)), *cert. denied*, 135 S. Ct. 2887 (2015). The fair use doctrine—codified in 17 U.S.C. § 107—is an "affirmative defense to copyright infringement" that permits a "'court[] to avoid rigid application of the copyright'" when the result would "'stifle the very creativity'" copyright law seeks to promote. *Id.* at 1372–73 (quoting Campbell v. Acuff-Rose Music, Inc., 510 U.S. 569, 577 (1994)).

145. U.S. CONST. art. I, § 8, cl. 8.

146. Baker v. Selden, 101 U.S. 99 (1879).

147. *Id.* at 100.

148. *Id.*

149. *Id.* at 104–05.

in *Baker*.[150] 17 U.S.C. § 102 provides the general rules for copyright subject matter.[151] Section 102(a) states: "Copyright protection subsists . . . in original works of authorship fixed in any tangible medium of expression . . . ."[152] Original works of authorship include literary works,[153] and the bar for originality is low.[154]

While the general rule of § 102(a) sweeps broadly over original works of authorship, several exceptions to copyright limit its application. Pertinent to this Note are three exceptions to be discussed in turn: the idea-expression dichotomy; the names and short phrases exception; and the merger doctrine.

Section 102(b) prevents copyright claims that would monopolize an idea or method of operation by codifying the idea-expression dichotomy: "In no case does copyright protection for an original work of authorship extend to any idea . . . system, [or] method of operation . . . regardless of the form in which it is described."[155]

The names and short phrases exception is codified in 37 C.F.R. § 202.1(a), which states, "[w]ords and short phrases such as names, titles, and slogans" are not subject to copyright.[156] This prohibition also includes "familiar symbols or designs" and a "mere listing of ingredients or contents."[157] The Ninth Circuit employed the names and short phrases exception in a footnote to

150. Copyright Act of 1976, Pub. L. No. 94-553, § 102(b), 90 Stat. 2541, 2545 (codified at 17 U.S.C. § 102(b) (2012)).

151. 17 U.S.C. § 102 (2012).

152. *Id.* § 102(a).

153. *Id.* § 102(a)(1); *see also* H.R. REP. NO. 94-1476, at 54 (1976) ("The term 'literary works' does not connote any criterion of literary merit or qualitative value: it includes catalogs, directories, and similar factual, reference, or instructional works and compilations of data. It also includes computer data bases, and computer programs to the extent that they incorporate authorship in the programmer's expression of original ideas, as distinguished from the ideas themselves.").

154. Feist Publ'ns, Inc. v. Rural Tel. Serv. Co., 499 U.S. 340, 358 (1991).

155. 17 U.S.C. § 102(b) (2012); *see also* Golan v. Holder, 132 S. Ct. 873, 890 (2012) ("'Due to this [idea/expression] distinction, every idea, theory, and fact in a copyrighted work becomes instantly available for public exploitation at the moment of publication;' the author's expression alone gains copyright expression." (quoting Eldred v. Ashcroft, 537 U.S. 186, 219 (2003))); Mazer v. Stein, 37 U.S. 201, 217 (1954) ("Unlike a patent, a copyright gives no exclusive right to the art disclosed; protection is given only to the expression of the idea—not the idea itself.").

156. 37 C.F.R. § 202.1(a) (2015).

157. *Id.*

Sega Enterprises, Ltd. v. Accolade, Inc.[158] Other circuits have been less willing to apply the doctrine to short phrases.[159]

Unlike the short phrases doctrine, the merger doctrine is entirely a jurisprudential creation.[160] The doctrine originates in the *Baker* decision.[161] Merger's underlying principle is "'when there is essentially one way to express an idea, the idea and its expression are inseparable and copyright is no bar to copying that expression.'"[162] In other words, the idea and expression are *merged* such that protection of the expression would amount to a monopoly over the idea.[163]

Congress passed the Copyright Act of 1976 in reaction to recent scientific and technological developments.[164] However, Congress postponed addressing the copyrightability of computer programs to await further study.[165] Congress formed the Commission on New Technological Uses of Copyrighted Works (CONTU) to recommend changes to copyright law necessitated by nascent technologies.[166] One of its recommendations was the codified definition of "computer program," which Congress added

158. Sega Enters., Ltd. v. Accolade, Inc., 977 F.2d 1510, 1524 n.7 (9th Cir. 1992) (explaining that this doctrine likely excluded from copyright protection an initialization code consisting of twenty bytes and the letters S-E-G-A).

159. *See, e.g.*, Soc'y of Holy Transfiguration Monastery, Inc. v. Gregory, 689 F.3d 29, 52 (1st Cir. 2012) (indicating that short phrases may be copyrightable if sufficiently creative); Softel, Inc. v. Dragon Med. & Sci. Commc'ns, Inc., 118 F.3d 955, 964 (2d Cir. 1997) (adopting cautious approach to copyright claims with uncopyrightable elements at low levels of abstraction, noting that a compilation of uncopyrightable elements may be copyrightable).

160. *See* Comput. Assocs. Int'l, Inc. v. Altai, Inc., 982 F.2d 693, 707–08 (2d Cir. 1992).

161. *See id.* at 707.

162. *Id.* at 707–08 (citing Concrete Mach. Co. v. Classic Lawn Ornaments, Inc., 843 F.2d 600, 606 (1st Cir. 1988)).

163. *See id.* at 708 (citing Herbert Rosenthal Jewelry Corp. v. Kalpakian, 446 F.2d 738, 742 (9th Cir. 1971)).

164. Deborah F. Buckman, Annotation, *Copyright Protection of Computer Programs*, 180 A.L.R. Fed. 1, 17 (2002).

165. H.R. REP. NO. 94-1476, at 57 (1976) ("Some concern has been expressed lest copyright in computer programs should extend protection to the methodology or processes adopted by the programmer, rather than merely to the 'writing' expressing his ideas. Section 102(b) is intended, among other things, to make clear that the expression adopted by the programmer is the copyrightable element in a computer program, and that the actual processes or methods embodied in the program are not within the scope of the copyright law."); Buckman, *supra* note 164, at 17.

166. *See* Comput. Assocs. Int'l, Inc. v. Altai, Inc., 982 F.2d 693, 703–04 (2d Cir. 1992) (citing Act of Dec. 31, 1974, Pub. L. No. 93-573, § 201, 88 Stat. 1873, 1873–74).

by amendment to 17 U.S.C. § 101 in 1980.[167]

CONTU explained that computer programs, when expressed in tangible mediums of expression, are subject to copyright; however, the underlying electromechanical functions are not.[168] The underlying principle of CONTU's recommendations may be expressed as "one is always free to make a machine perform any conceivable process . . . but one is not free to take another's program."[169] CONTU also recognized merger as applicable when specific instructions are the only means of accomplishing a given task.[170]

The development of these exceptions demonstrates the evolution of intellectual property law since the nation's founding as the law has adapted to scientific and economic developments. Today, legal scholars continue to debate the need for a rebalancing of the distinction between patent and copyright in light of changes in the technology industry:

> As software patents gain increasingly broad protection, whatever reasons there once were for broad copyright protection of computer programs disappear. Much of what has been considered the copyrightable "structure, sequence and organization" of a computer program will become a mere incident to the patentable idea of the program or of one of its potentially patentable subroutines.[171]

As the debate continues, the technology industry will continue to seek protection for works under patent or copyright law.

To bring a claim for copyright infringement, the claimant must show: (1) ownership of a valid copyright, and (2) unauthorized copying of the original work by the accused infringer.[172] These two elements are generally referred to as (1) the copyrightability analysis and (2) the infringement

---

167. 17 U.S.C. § 101 (2012) ("A 'computer program' is a set of statements or instructions to be used directly or indirectly in a computer in order to bring about a certain result."); NAT'L COMM'N ON NEW TECH. USES OF COPYRIGHTED WORKS, FINAL REPORT 12 (1978), http://digital-law-online.info/CONTU/PDF/Chapter3.pdf.

168. NAT'L COMM'N ON NEW TECH. USES OF COPYRIGHTED WORKS, *supra* note 167, at 19–20.

169. *Id.* at 20.

170. *Id.*

171. Mark Lemley, *Convergence in the Law of Software Copyright?*, 10 HIGH TECH. L.J. 1, 26–27 (1995).

172. *See, e.g.*, Whelan Assocs., Inc. v. Jaslow Dental Lab., Inc., 797 F. 2d 1222, 1231 (3d. Cir. 1986).

analysis.[173] As discussed above, the district court in this case bifurcated the trial procedure into a bench decision on copyrightability and a jury decision on infringement. The Federal Circuit reversal concerns the district court's copyrightability analysis.

B. COPYRIGHTABILITY OF NONLITERAL COMPUTER PROGRAM ELEMENTS

Throughout the litigation, both courts recognized the subject matter underlying Oracle's second theory of copyright infringement as the "structure, sequence, and organization" (SSO) embodied in the thirty-seven Java API packages.[174] The circuits have approached in various ways the issue of whether the nonliteral components of a computer program, such as SSO, are copyrightable. In attempting to find a jurisprudential basis for SSO copyrightability, both courts surveyed the jurisprudence across the circuits in search of an answer.[175] This survey does the same, with an emphasis on Ninth Circuit jurisprudence.

The first test for determining copyrightability of nonliteral computer program elements came from the Third Circuit, in *Whelan Associates, Inc. v. Jaslow Dental Laboratories, Inc.*[176] In *Whelan*, the Third Circuit held the structure of a program copyrightable because there was more than one way to structure the program.[177] In an attempt to draw a bright line between the idea and expression of a program, the court held the purpose or function of the program to be its idea and everything not necessary to that purpose to be expression.[178] The bright-line test enunciated in *Whelan* is sometimes referred to as the "structure, sequence, and organization" test because those words appear for

173. Oracle Am., Inc. v. Google, Inc., 872 F. Supp. 2d 974, 975 (N.D. Cal. 2012), *rev'd*, 750 F.3d 1339 (Fed. Cir. 2014), *cert. denied*, 135 S. Ct. 2887 (2015).

174. *See* Oracle Am., Inc. v. Google, Inc., 750 F.3d 1339, 1348 (Fed. Cir. 2014) (acknowledging lower court's holding that neither of the replicated elements—declaring code nor structure, sequence, and organization—were copyrightable), *cert. denied*, 135 S. Ct. 2887 (2015); *Oracle*, 872 F. Supp. 2d at 978–79 ("The copyright issue, rather, is whether Google was and remains free to replicate the names, organization of those names, and functionality of 37 out of 166 packages in the Java API, which has sometimes been referred to in this litigation as the 'structure, sequence and organization' of the 37 packages.").

175. *See Oracle*, 750 F.3d at 1364–66; *Oracle*, 872 F. Supp. 2d at 984–96.

176. Whelan Assocs, Inc. v. Jaslow Dental Lab., Inc., 797 F.2d 1222 (3d Cir. 1986).

177. *Id.* at 1238–40.

178. *Id.* at 1238.

the first time in *Whelan*.[179]

Three years later, the Ninth Circuit addressed copyrightability of nonliteral software elements in *Johnson Controls, Inc. v. Phoenix Control Systems, Inc.*[180] The *Johnson Controls* court recognized that nonliteral software elements could be copyrightable depending on whether the structure, sequence, and organization of the program is expression.[181] The decision was merely an affirmance of a preliminary injunction under a clear error standard.[182] Thus, the court did not elaborate on how to determine whether SSO was expression.[183] *Johnson Controls* represents the last time the phrase "structure, sequence, and organization" appears in the Ninth Circuit.[184] Nevertheless, Oracle relied on *Johnson Controls* to a degree that Judge Alsup called it "one of Oracle's mainstays herein."[185]

The Second Circuit criticized the Third for taking too narrow a view of the "idea" of a program and instead created a new test, the "abstraction-filtration-comparison" test.[186] To implement the test, a court first dissects the program into its structural parts (the abstraction step).[187] The next step is to filter out the uncopyrightable elements (the filtration step).[188] A court does this by looking for structures dictated by efficiency or by external factors, or structures that are already available in the public domain.[189] Significant to this Note, a court considers merger in the filtration step.[190] Finally, a court then compares the resulting elements with the accused infringer's program (the comparison step).[191]

---

179. *See* Oracle Am., Inc. v. Google, Inc., 872 F. Supp. 2d 974, 988 (N.D. Cal. 2012), *rev'd*, 750 F.3d 1339 (Fed. Cir. 2014), *cert. denied*, 135 S. Ct. 2887 (2015).

180. Johnson Controls, Inc. v. Phoenix Control Sys., Inc., 886 F.2d 1173 (9th Cir. 1989).

181. *See id.* at 1177.

182. *Id.* at 1175.

183. *See Oracle*, 872 F. Supp. 2d at 992.

184. *See id.* at 996.

185. *See* Oracle Am., Inc. v. Google, Inc., 872 F. Supp. 2d 974, 992 (N.D. Cal. 2012), *rev'd*, 750 F.3d 1339 (Fed. Cir. 2014), *cert. denied*, 135 S. Ct. 2887 (2015).

186. Comput. Assocs. Int'l, Inc. v. Altai, Inc., 982 F.2d 693, 706 (2d Cir. 1992).

187. *See Oracle*, 872 F. Supp. 2d at 988 (citing *Altai*, 982 F.2d at 706).

188. *See id.* at 988–89 (citing *Altai*, 982 F.2d at 708–10).

189. *See id.* at 989 (citing *Altai*, 982 F.2d at 708–10).

190. *Altai*, 982 F.2d at 708.

191. *Id.* at 710–11.

Several circuits have applied this test.[192] The Ninth Circuit has criticized *Whelan*[193] and endorsed the abstraction-filtration-comparison test,[194] but has not applied it. In this case, both courts agreed that this test is more consistent with Ninth Circuit jurisprudence than the SSO test from *Whelan*.[195]

Lotus Development Corp. v. Borland International is a significant case because it was the last time the Supreme Court considered the issue of nonliteral software element copyrightability.[196] Unfortunately, the Court established no precedent in an equal four–four affirmance.[197] In *Lotus*, the First Circuit held a menu command hierarchy (with elements such as "copy," "print," and "quit") an uncopyrightable "method of operation" under § 102(b).[198] The court reasoned that, though the claimant made expressive choices in naming and organizing the menu elements, the command hierarchy was necessary to make functional use of program capabilities.[199]

As for the merger doctrine, the circuits are generally split on the issue of whether merger applies to the threshold issue of copyrightability or as an affirmative defense to infringement.[200] Since its holding in *Ets-Hokin v. Skyy Spirits, Inc.*, the Ninth

192. *See* Comput. Mgmt. Assistance Co. v. Robert F. DeCastro, Inc., 220 F.3d 396, 400–01 (5th Cir. 2000); Gates Rubber Co. v. Bando Chem. Indus., Ltd., 9 F.3d 823, 834 (10th Cir. 1993) (adopting the test "[i]n substantial part"); *see also* Bateman v. Mnemonics, Inc., 79 F.3d 1532, 1543–46 (11th Cir. 1996) (concluding that for literal copying either a "filtration step . . . or a separate, yet parallel analysis . . . is necessary").

193. *See* Sega Enters., Ltd. v. Accolade, Inc., 977 F.2d 1510, 1524–25 (9th Cir. 1992).

194. *See* Apple Comput., Inc. v. Microsoft Corp., 35 F.3d 1435, 1445 (9th Cir. 1994) (describing the *Altai* abstraction-filtration-comparison test as same approach taken by Ninth Circuit but articulated differently).

195. *See* Oracle Am., Inc. v. Google, Inc., 750 F.3d 1339, 1357 (Fed. Cir. 2014) (noting the Ninth Circuit's endorsement of the abstraction-filtration-comparison test) *cert. denied*, 135 S. Ct. 2887 (2015); Oracle Am., Inc. v. Google, Inc., 872 F. Supp. 2d 974, 988 (N.D. Cal. 2012) (noting that the Ninth Circuit has adopted a variation of the Second Circuit's approach), *rev'd*, 750 F.3d 1339 (Fed. Cir. 2014), *cert. denied*, 135 S. Ct. 2887 (2015).

196. *See* Lotus Dev. Corp. v. Borland Int'l, 516 U.S. 233 (1996); Lotus Dev. Corp. v. Borland Int'l, 49 F.3d 807 (1st Cir. 1995), *aff'd without opinion by an equally divided court*, 516 U.S. 233 (1996).

197. *Lotus*, 516 U.S. at 234.

198. *Lotus*, 49 F.3d at 815.

199. *Id.* at 816.

200. *See* Ets-Hokin v. Skyy Spirits, Inc., 225 F.3d 1068, 1082 (9th Cir. 2000) (citing cases).

Circuit treats merger as an affirmative defense to infringement.[201] Even before that, merger arguments met with little success in Ninth Circuit jurisprudence.[202]

The foregoing legal background sums up most of the jurisprudence underlying the issues in this litigation. A discussion of the Federal Circuit decision follows.

IV. THE CIRCUIT COURT DECISION

Despite Judge Alsup's restrained, reasoned, and fact-based approach, the Federal Circuit reversed the lower court's rulings on the copyrightability of the declaring code and SSO of the thirty-seven Java API packages at issue.[203]

The Federal Circuit first noted that the district court's decision was subject to de novo review.[204] It then proceeded to analyze Section 102.[205] The court characterized the lower court's reading of Section 102 as a two-part test in which 102(a) extends protection to original works and then 102(b) strips protection from the ideas underlying creative expression.[206] The panel disagreed with this approach, noting 102(b) does not change the scope of 102(a).[207] Instead, 102(a) and 102(b) must be considered collectively, with the court separating protectable expression from unprotectable ideas.[208]

Next, the court discussed the tests for determining copyrightability of nonliteral program elements by analyzing Ninth Circuit jurisprudence. The Ninth Circuit set forth some principles that would guide its application of this copyrightability test.[209] The Federal Circuit in *Oracle* first noted that the abstraction filtration step was clearly part of a copyrightability

201. Ets-Hokin v. Skyy Spirits, Inc., 225 F.3d 1068, 1082 (9th Cir. 2000).

202. *See, e.g.*, CDN Inc. v. Kapes, 197 F.3d 1256, 1261–62 (9th Cir. 1999) (rejecting merger theory applied to price list of projected values); Apple Comput., Inc. v. Formula Int'l, Inc., 725 F.2d 521, 523–24 (9th Cir. 1984) (rejecting merger theory as basis for argument that no computer program is copyrightable except for the expression the computer directly communicates to the user).

203. Oracle Am., Inc. v. Google, Inc., 750 F.3d 1339, 1381 (Fed. Cir. 2014), *cert. denied*, 135 S. Ct. 2887 (2015).

204. *Id.* at 1353 (citing *Ets-Hokin*, 225 F.3d at 1073).

205. *Id.* at 1356–59.

206. *Id.* at 1356.

207. *Id.*

208. *Id.* at 1357.

209. Oracle Am., Inc. v. Google, Inc., 750 F.3d 1339, 1357–59 (Fed. Cir. 2014), *cert. denied*, 135 S. Ct. 2887 (2015).

analysis while the comparison step clearly looked at infringement.[210] However, the court acknowledged a circuit split regarding when to apply the filtration step.[211] The Ninth Circuit has consistently held the merger doctrine—integral to the filtration analysis—to be an affirmative defense to infringement.[212] Thus, only originality may be considered in the copyrightability analysis, while merger is considered in the infringement analysis.[213] The court observed that the district court did not actually apply the test,[214] but kept these principles in mind as it proceeded to review the district court decision.[215]

The court agreed with all of Oracle's assignments of error to the lower court: (1) the conclusion that each line of declaring code was uncopyrightable because the idea and expression had merged; (2) finding the declaring code lines uncopyrightable because they use short phrases; (3) finding the SSO an uncopyrightable "method of operation" under § 102(b); and (4) considering interoperability as part of the copyrightability analysis.[216] This section follows the same order.

A. DECLARING CODE AND THE MERGER DOCTRINE

The court first acknowledged the merger doctrine as "an exception to the idea-expression dichotomy."[217] But the court noted that in the Ninth Circuit, merger is only relevant to the infringement analysis.[218] Thus, the court opined that the district court's merger analysis "appear[ed]" to be irrelevant in deciding whether the thirty-seven packages are copyrightable in the first place.[219] The court then applied Ninth Circuit precedent in holding that merger does not apply to a computer program unless the program can be written in only one way.[220] The court agreed with Oracle's argument that the district court erred in its merger

210. Oracle Am., Inc. v. Google, Inc., 750 F.3d 1339, 1358 (Fed. Cir. 2014), *cert. denied*, 135 S. Ct. 2887 (2015).

211. *Id.*

212. *See id.*

213. *See id.*

214. *Id.*

215. *See id.* at 1358–68.

216. Oracle Am., Inc. v. Google, Inc., 750 F.3d 1339, 1359 (Fed. Cir. 2014), *cert. denied*, 135 S. Ct. 2887 (2015).

217. *Id.*

218. *Id.* at 1359–60.

219. *Id.* at 1360.

220. *Id.* (citing Satava v. Lowry, 323 F.3d 805, 812 n.5 (9th Cir. 2003)).

analysis by "misapply[ing] the merger doctrine and . . . fail[ing] to focus its analysis on the options available to [Sun,] the original author."[221]

The court reasoned that merger does not protect a work from infringement if there are alternative ways to express the underlying idea.[222] Oracle could have chosen any names for the packages, classes, and methods of the Java API.[223] Therefore, merger did not apply to the declaring code in the Java API.[224] Second, merger focuses on the options available to the original author, not those available to the copier.[225] Though Google had to use the same names to allow compatibility between the Android and Java APIs, that consideration is irrelevant in the merger analysis.[226] By focusing on the numerous options available to Sun upon creation of the Java API, the court held that merger did not apply.[227]

B. DECLARING CODE: NAMES AND SHORT PHRASES

The court articulated the general rule that names and short phrases alone are not subject to copyright protection.[228] However, the court noted, "the relevant question for copyrightability purposes is not whether a work uses short phrases . . . but, rather, whether those phrases are original and creative."[229] As an example, the court pointed to the opening line of Charles Dickens's *A Tale of Two Cities*, which, though composed of short phrases, is undisputedly copyrightable.[230] The court reasoned

---

221. Oracle Am., Inc. v. Google, Inc., 750 F.3d 1339, 1361 (Fed. Cir. 2014), *cert. denied*, 135 S. Ct. 2887 (2015).

222. *Id.* (citing Satava v. Lowry, 323 F.3d 805, 812 n.5 (9th Cir. 2003)).

223. *Id.*

224. *Id.*

225. *Id.* (citing Apple Comput., Inc. v. Formula Int'l, Inc., 725 F.2d 521, 524 (9th Cir. 1984)).

226. *Id.*

227. Oracle Am., Inc. v. Google, Inc., 750 F.3d 1339, 1361 (Fed. Cir. 2014), *cert. denied*, 135 S. Ct. 2887 (2015).

228. *Id.* at 1362 (citing 37 C.F.R. § 202.1(a)).

229. *Id.*

230. *Id.* at 1363. The novel begins:

> It was the best of times, it was the worst of times, it was the age of wisdom, it was the age of foolishness, it was the epoch of belief, it was the epoch of incredulity, it was the season of Light, it was the season of Darkness, it was the spring of hope, it was the winter of despair, we had everything before us, we had nothing before us, we were all going direct to Heaven, we were all going direct the other way . . . .

CHARLES DICKENS, A TALE OF TWO CITIES 5 (Michael D. Aeschliman ed., Ignatius

that Google infringed upon 7,000 lines of code, not a single short phrase.[231] Moreover, Google acknowledged that there could be "creativity and artistry even in a single method declaration."[232] Therefore, reasoned the panel, the district court erred in holding the declaring code uncopyrightable under the names and short phrases doctrine.[233]

C. SSO AND METHODS OF OPERATION

The Circuit moved next to the SSO of the thirty-seven Java API packages.[234] First, the panel emphasized the district court's finding that the thirty-seven Java API packages were original and creative.[235] The court then reversed the district court's holding that the SSO was an uncopyrightable method of operation under § 102(b).[236]

First, the court opined, "the district court seems to have relied upon language contained in . . . *Lotus*."[237] The court distinguished this case from *Lotus* in three respects.[238] First, the accused infringer in *Lotus* did not copy any source code, while Google copied 7,000 lines of code "verbatim."[239] Second, the commands at issue in *Lotus* (e.g., "copy" and "print") were not creative, while it is undisputed that the Java API declaring code and SSO are original and creative.[240] Third, the *Lotus* commands were "'essential to operating' the system," while Google chose to copy the declaring code and SSO of the thirty-seven packages—Android would have worked without them.[241] The court also remarked on the inconsistencies between the *Lotus* decision and Ninth Circuit jurisprudence.[242]

Press ed. 2012) (1859).

231. Oracle Am., Inc. v. Google, Inc., 750 F.3d 1339, 1363 (Fed. Cir. 2014), *cert. denied*, 135 S. Ct. 2887 (2015).

232. *Id.* (quoting Google's "Java Guru").

233. *Id.*

234. *Id.* at 1364–68.

235. *Id.* at 1364.

236. *See id.* at 1368.

237. Oracle Am., Inc. v. Google, Inc., 750 F.3d 1339, 1364 (Fed. Cir. 2014) (citing Lotus Dev. Corp. v. Borland Int'l, Inc., 49 F.3d 807 (1st Cir. 1995)), *cert. denied*, 135 S. Ct. 2887 (2015).

238. *See id.* at 1365.

239. *Id.*

240. *Id.*

241. *Id.* (quoting *Lotus*, 49 F.3d at 816).

242. *Id.* at 1365–66. The court distinguished the "method of operation" reasoning in *Lotus* from Ninth Circuit precedent in three ways: (1) Ninth Circuit jurisprudence

After finding no binding precedent within Ninth Circuit jurisprudence, the court held that "a set of commands to instruct a computer to carry out desired operations may contain expression that is eligible for copyright protection."[243] The court repeated the fact that the source code in this case could have been written in any number of ways to perform the same function.[244] Thus, "even if an element directs a computer to perform [an] operation[]," the court must still identify "any separable expression subject to copyright protection."[245] Therefore, though the thirty-seven packages perform functions, the SSO is still subject to copyright protection because: (1) it is original and creative; and (2) the declaring code could have been written and organized in any number of ways to perform the same functions.[246]

D. INTEROPERABILITY IS IRRELEVANT TO COPYRIGHTABILITY

To conclude, the court addressed the district court's finding that duplication of the command structure was necessary for interoperability.[247] The court pointed to the district court's reliance on the Ninth Circuit's decisions in *Sony* and *Sega*.[248] The court argued these cases were inapposite because they both addressed fair use, not copyrightability.[249] Furthermore, Google's reading of these cases as "interoperability exception[s] to copyrightability was inconsistent" with the Ninth Circuit's recognition of the distinction between literal and nonliteral computer program elements and the endorsement of the abstraction-filtration-comparison test in *Sega*.[250]

The court then reiterated an argument against the district

---

holds SSO copyrightable "where it qualifies as expression of an [underlying] idea," (2) Federal Circuit application of Ninth Circuit law extends copyright protection to "'expression of a process or method,'" and (3) the Ninth Circuit has endorsed the abstraction-filtration-comparison test. *Id.* (quoting Atari Games Corp. v. Nintendo of Am., Inc., 975 F.2d 832, 839 (Fed. Cir. 1992)).

243. Oracle Am., Inc. v. Google, Inc., 750 F.3d 1339, 1367 (Fed. Cir. 2014), *cert. denied*, 135 S. Ct. 2887 (2015).

244. *Id.* at 1368.

245. *Id.* at 1367.

246. *Id.* at 1368.

247. *Id.* at 1368–72.

248. *Id.* at 1369 (citing Sega Enters. v. Accolade, Inc., 977 F.2d 1510 (9th Cir. 1992); Sony Comput. Entm't, Inc. v. Connectix, Corp., 203 F.3d 596 (9th Cir. 2000)).

249. Oracle Am., Inc. v. Google, Inc., 750 F.3d 1339, 1369 (Fed. Cir. 2014), *cert. denied*, 135 S. Ct. 2887 (2015).

250. *Id.* at 1370 (quoting Google's brief).

court's merger analysis.[251] Google's requirements for interoperability were irrelevant to the analysis.[252] Rather, only Sun's interoperability concerns were pertinent, and Google presented no evidence in this regard.[253] Furthermore, Google's interoperability argument was belied by record evidence that Android was designed so that it would not be compatible with the Java platform.[254] Finally, the Circuit rejected Google's argument that it was entitled to copy the Java API because Java had become the effective industry standard.[255] The court pointed to a lack of authority suggesting copyrighted works lose protection upon becoming popular.[256]

Having reversed the district court's copyrightability decision, the Federal Circuit proceeded to examine Google's fair use defense to infringement.[257] The court found insufficient factual findings in the record to rule on Google's fair use defense and remanded the case to the district court for a retrial of the issue, consistent with the reversed copyrightability holdings.[258]

Google filed a petition for a writ of certiorari for review of the Federal Circuit's copyrightability holding.[259] Specifically, Google presented the following question: "Whether copyright protection extends to all elements of an original work of computer software, including a system or method of operation that an author could have written in more than one way."[260] Despite a total of eight amici curiae briefs filed (six for petitioner, two for respondent), the Supreme Court denied the petition without opinion.[261] The following analysis incorporates the arguments of amici curiae, who represent many important members of the computer industry and programming community.

---

251. Oracle Am., Inc. v. Google, Inc., 750 F.3d 1339, 1371 (Fed. Cir. 2014), *cert. denied*, 135 S. Ct. 2887 (2015).

252. *Id.*

253. *Id.*

254. *Id.*

255. *Id.* at 1372.

256. *Id.*

257. Oracle Am., Inc. v. Google, Inc., 750 F.3d 1339, 1372–77 (Fed. Cir. 2014), *cert. denied*, 135 S. Ct. 2887 (2015).

258. *Id.* at 1377.

259. *See* Petition for a Writ of Certiorari, Google, Inc. v. Oracle Am., Inc., 135 S. Ct. 2887 (2015) (No. 14-410), 2014 WL 5319724.

260. *Id.* at i.

261. Google, Inc. v. Oracle Am., Inc., 135 S. Ct. 2887 (2015) (mem.), *denying cert. to* 750 F.3d 1339 (Fed. Cir. 2014).

V. ANALYSIS

The rise of computer hardware and software technologies has presented unique challenges for society, including in the practice of law.[262] The CONTU Report recognized that rapid change in technology required flexibility in applying the law.[263] CONTU acknowledged the difficulty inherent in differentiating between the copyrightable form of a program and the uncopyrightable process, and recommended that the demarcation be left to the judge's discretion.[264]

As outlined in the introduction, the application of law to computer programs requires both legal and technical knowledge. Because computer programs are unambiguous, a mapping of law to facts requires the finder of fact to translate technological concepts into legal conclusions. For this reason, deference to the fact-finder is more important in the software context than in other appeals, because the soundness of an appellate court's legal conclusions heavily depends on its grasp of the technological concepts. In this case, depth of understanding of programming concepts explains the conclusions reached by each court.

This part crystallizes the central argument of this Note: rapid change in technology requires that judges have a firm understanding of computer programming concepts. To illustrate this central argument, the first section compares the approaches of the district court and Federal Circuit by applying the abstraction-filtration-comparison test to the thirty-seven Java API packages. The second section discusses the negative effects of a rigid application of copyright law to computer programs.

A. JUDGE ALSUP VS. THE FEDERAL CIRCUIT

In its opinion, the Federal Circuit concluded that the district court did not apply the abstraction-filtration-comparison test, but instead relied on an approach akin to that used by the First

262. *See supra* Section III.

263. NAT'L COMM'N ON NEW TECH. USES OF COPYRIGHTED WORKS, *supra* note 167, at 22–23.

264. *Id.* ("The many . . . new applications which advancing technology will supply may make drawing the line of demarcation more and more difficult. To attempt to establish such a line in this report written in 1978 would be futile. . . . Should a line need to be drawn to exclude certain manifestations of programs from copyright, that line should be drawn on a case-by-case basis by the institution designed to make fine distinctions—the federal judiciary.").

Circuit in *Lotus*.[265] However, close review of the decision reveals that the district court considered the First Circuit decision in conjunction with the approaches of the Second, Third, Ninth, and Tenth Circuits.[266] In fact, the district court based its decision on *Baker*'s interpretation of the jurisprudence later codified in § 102(b).[267] The district court addressed the abstraction-filtration-comparison test[268] and would have reached the same conclusions under an explicit application of the three steps of the test. A hypothetical application of the test follows.

1. ABSTRACTION

In the abstraction step, the court dissects the program into its structural parts.[269] As discussed at length in Part II, the strength of Judge Alsup's opinion was his illustration of the programming elements at issue.[270] The Federal Circuit's understanding of technology did not match that of the lower court.

Despite borrowing heavily from Judge Alsup's factual explanation, the Federal Circuit's description of the technology at issue indicates some conceptual confusion. For instance, the court wrote: "Every package consists of two types of source code—what the parties call (1) declaring code; and (2) implementing code."[271] However packages consist of classes, classes consist of methods, and methods consist of declaring code and implementing code.[272] The court also refers to packages as "application programming interfaces,"[273] though Java has but one "Application Programming Interface"; the API is composed of packages. Then, the court delivered this statement:

---

265. *See supra* text accompanying notes 196–99, 237–42.

266. *See* Oracle Am., Inc. v. Google, Inc., 872 F. Supp. 2d 974, 987–97 (N.D. Cal. 2012) ("Our case is governed by the law of the Ninth Circuit and . . . the Supreme Court. The Supreme Court missed the opportunity to address these issues in *Lotus* due to the four-to-four affirmance and has, thus, never reached the general question"), *rev'd*, 750 F.3d 1339 (Fed. Cir. 2014), *cert. denied*, 135 S. Ct. 2887 (2015).

267. *Id.* at 992.

268. *See* Oracle Am., Inc. v. Google, Inc., 750 F.3d 1339, 1358 (Fed. Cir. 2014) (acknowledging that the district court "mentioned the . . . test"), *cert. denied*, 135 S. Ct. 2887 (2015).

269. *See Oracle*, 872 F. Supp. 2d at 988 (citing Comput. Assocs. Int'l, Inc. v. Altai, Inc., 982 F.2d 693, 706 (2d Cir. 1992)).

270. *See supra* Part II.B.

271. *Oracle*, 750 F.3d at 1349.

272. *See supra* text accompanying notes 62–88.

273. *Oracle*, 750 F.3d at 1347.

> Although it is undisputed that certain Android software contains copies of the 37 API packages declaring code at issue, neither the district court nor the parties specify in which programs those copies appear. Oracle indicated at oral argument, however, that all Android phones contain copies of the accused portions of the Android software.[274]

Of course, the thirty-seven Android API packages contain the copies. All Android phones contain copies of the "accused portions" because all Android phones implement the Android platform, which includes the Android API.

Given the foregoing, it is doubtful the court would be able to properly abstract the elements to filter. Improper abstraction leads to ambiguity in the filtration analysis. Ambiguity raises the likelihood of filtering copyrightable, or comparing uncopyrightable, expression. Though the abstraction-filtration-comparison test was not applied in full, a misunderstanding of API basics may have played a role in the Federal Circuit reversal.

2. FILTRATION

Assuming a proper abstraction of structural elements of the API packages, a court proceeds to the filtration step.[275] To filter out the uncopyrightable elements, the court looks for structures dictated by efficiency, by external factors, or that are already available in the public domain.[276]

Judge Alsup recognized the copyrightability of the method implementations and recognized creativity in the names and their organization.[277] Nevertheless, he held that the names were not subject to copyright[278] and he would have filtered the names out before the comparison. In fact, he would have filtered out all the declaring code because the rules of the Java language dictate its structure.[279] Because duplication of the declaring code is necessary for interoperability between the Android API (written

---

274. Oracle Am., Inc. v. Google, Inc., 750 F.3d 1339, 1351 (Fed. Cir. 2014), *cert. denied*, 135 S. Ct. 2887 (2015).

275. *See* Oracle Am., Inc. v. Google, Inc., 872 F. Supp. 2d 974, 988 (N.D. Cal. 2012) (citing Comput. Assocs. Int'l, Inc. v. Altai, Inc., 982 F.2d 693, 706 (2d Cir. 1992)), *rev'd*, 750 F.3d 1339 (Fed. Cir. 2014), *cert. denied*, 135 S. Ct. 2887 (2015).

276. *See supra* text accompanying note 188–89.

277. *See Oracle*, 872 F. Supp. 2d at 976–77.

278. *See id.* at 983–84.

279. *See id.* at 976.

in Java) and applications written in Java,[280] the declaring code merges with the underlying functions. By filtering the declaring code, the SSO it embodies would also be filtered. The remainder would be the method implementations.

The Federal Circuit would disagree, pointing to Ninth Circuit jurisprudence treating merger as an affirmative defense to infringement.[281] Instead, the Federal Circuit would ignore merger in the filtration analysis. The Federal Circuit would also hold the names to be sufficiently creative to warrant copyright protection. It is not clear that the Federal Circuit would filter *any* of the API elements as uncopyrightable, given that the names and short phrases exception does not apply and merger is an affirmative defense to infringement.

3. COMPARISON

Hence, the Federal Circuit would compare all of the code in the thirty-seven Android packages with the code in the thirty-seven Java packages (assuming it had identified the relevant Android code for comparison). The declaring code would be identical, so the Federal Circuit would find infringement, subject to any affirmative defenses. The district court, having filtered the declaring code, would compare the method implementations with the declaring code and implementations in the Android API. Because the implementations in the Android API were original, the district court would find no infringement.

Thus, even if the courts had fully applied the abstraction-filtration-comparison test to resolve Oracle's nonliteral software element copyright claim, they would likely arrive at opposing conclusions. The confusion tends to support granting the writ for certiorari; nevertheless, the Supreme Court denied Google's writ. The effects of denial are discussed in the next section.

B. EFFECTS OF DENYING CERTIORARI: LEGAL CONFUSION AND ECONOMIC UNCERTAINTY

By denying certiorari, the Supreme Court left several important questions of copyright law as applied to software unresolved. The overarching issue of the interface of patent and copyright law is closely related to that of the idea-expression

280. *See* Oracle Am., Inc. v. Google, Inc., 872 F. Supp. 2d 974, 976 (N.D. Cal. 2012), *rev'd*, 750 F.3d 1339 (Fed. Cir. 2014), *cert. denied*, 135 S. Ct. 2887 (2015).

281. For discussion of the Federal Circuit decision, see *supra* Part IV.

dichotomy codified in § 102. CONTU was well aware of the difficulty in applying a rigid distinction between copyrightable programs and uncopyrightable ideas.[282] CONTU recommended giving judges wide discretion to apply § 102 to the particular facts of each case.[283] Despite Judge Alsup's mastery of the programming concepts in the record, the Federal Circuit reversed his narrowly fact-bound findings.[284] The reversal creates legal confusion in applying copyright to computer programs and economic uncertainty for programmers worldwide. The legal and economic implications are discussed below.

1. LEGAL CONFUSION

The Federal Circuit held both the declaring code and the command structure of thirty-seven packages in the Java API to be copyrightable works of authorship under § 102(a).[285] However, from the point of view of amici curiae, interfaces are uncopyrightable methods of operation under § 102(b), and their patentability is questionable.[286] Software interfaces allow communication between programs and computers.[287] They allow for software compatibility and interoperability and are thus necessary to make functional use of program capabilities.[288] "Potentially patentable methods of operation are not themselves copyrightable expression."[289] Allowing copyright to extend to potentially patentable methods of operation would grant the right holder a ninety-five-year monopoly over an idea without subjection to the more rigorous standards for patent protection.[290] For these reasons, the Federal Circuit's decision may have

282. *See* NAT'L COMM'N ON NEW TECH. USES OF COPYRIGHTED WORKS, *supra* note 167, at 22–23.

283. *Id.* at 23.

284. *See* Oracle Am., Inc. v. Google, Inc., 750 F.3d 1339, 1381 (Fed. Cir. 2014), *cert. denied*, 135 S. Ct. 2887 (2015).

285. *See id.*; *see supra* Part IV.B–D

286. *See* Brief of Amicus Curiae Public Knowledge in Support of the Petition at 18–19, Google, Inc. v. Oracle Am., Inc., 135 S. Ct. 2887 (2015) (No. 14-410), 2014 WL 5868949 (arguing that APIs are concepts that are neither copyrightable nor patentable).

287. *Id.* at 5.

288. Brief of Amici Curiae Hewlett-Packard Co. et al. in Support of Petitioner at 3, 15, *Google*, 135 S. Ct. 2887 (No. 14-410), 2014 WL 5868947.

289. Brief of Amicus Curiae Public Knowledge in Support of the Petition, *supra* note 286, at 19.

290. Brief of Amici Curiae Hewlett-Packard Co. et al. in Support of Petitioner, *supra* note 288, at 17; *see also* 17 U.S.C. § 302(c) (2012) (establishing a ninety-five-year term for the copyright of a "work made for hire").

implications for the scope of patent and copyright protection of computer software that give members of the technological community reason for concern.

The Federal Circuit erred in its merger and short phrases analyses by focusing on the *ex ante* alternatives available to the creator of the API and not on the lack of alternatives available to users of the API. To summarize, Google incorporated declaring code contained in thirty-seven Java API packages to allow for interoperability with pre-existing applications written in Java and to allow future application developers to write their programs in Java.[291] The declaring code had to be copied verbatim to allow for compatibility with programs and applications written in the Java language.[292] Because the declaring code had to be copied verbatim, subsequent programmers were constrained in their choices.[293]

This constraint invokes the merger doctrine.[294] Contrary to the Circuit's repeated argument that the scope of merger should be limited to the *ex ante* alternatives available to the interface creator, restraints on subsequent programmers should be the focus of a merger analysis applied to computer programs.[295] Admittedly, Sun had multiple alternatives for naming its packages, classes, and methods in its Java API library. But once Sun had named the package, class, or method, the name merged with the underlying function. Subsequent programmers using the Java language who wished to execute the underlying function had to use the name and command structure selected by the programmers at Sun to invoke that underlying function.

Furthermore, the names of the packages, classes, and methods in the Java API library fit squarely under the names

---

291. *See supra* text accompanying notes 101–12.

292. *See supra* text accompanying notes 134–35.

293. *See* Brief of Intellectual Prop. Professors in Support of Grant of Petition at 18, 24, Google, Inc. v. Oracle Am., Inc., 135 S. Ct. 2887 (2015) (No. 14-410), 2014 WL 5868948.

294. *See id.* at 18–24.

295. *See* Brief of Amicus Curiae Public Knowledge in Support of the Petition, *supra* note 286, at 14–17; *see also* NAT'L COMM'N ON NEW TECH. USES OF COPYRIGHTED WORKS, *supra* note 167, at 20 ("The 'idea-expression identity' exception provides that copyrighted language may be copied without infringing when there is but a limited number of ways to express a given idea. . . . In the computer context this means that when specific instructions, even though previously copyrighted, are the only and essential means of accomplishing a given task, their later use by another will not amount to an infringement.").

and short phrases doctrine. By referencing the opening lines of Charles Dickens's *A Tale of Two Cities*,[296] the Federal Circuit indicated further misunderstanding of API basics. *A Tale of Two Cities* is a novel with no functional aspects whatsoever. The Java API is not a novel; it is a command hierarchy. The Java API was designed to enable use of the Java programming language, and the Java programming language is free for all to use.[297] Copyright of the Java API, especially of the three core packages, amounts to a copyright of the language itself.

Programmers use the Java language to create programs, just as Charles Dickens used the English language to create *A Tale of Two Cities*. Copyright protection extends to both programs and novels.[298] Copyright protection does not extend to the English language, its grammar, or the method for its expression on paper.[299] Likewise, copyright does not extend to the Java programming language, its syntax, or to the command hierarchy that enables its communication with hardware.[300] Even if Sun exhibited creativity in organizing and naming the elements of the command hierarchy, copyright protection does not extend over them.

As noted throughout, application of the abstraction-filtration-comparison test in a proceeding bifurcated into copyrightability and infringement trials can produce inconsistent results. In this case, the district court held the copyrightability and infringement proceedings concurrently, instructing the jury to assume copyrightability.[301] Under Ninth Circuit precedent, the exceptions to copyrightability, such as merger, are considered affirmative defenses to infringement.[302]

In order for the district court to apply this test in the bifurcated trial, the judge would clearly have responsibility for the abstraction step and the jury would have responsibility for

296. *See* Oracle Am., Inc. v. Google, Inc., 750 F.3d 1339, 1363 (Fed. Cir. 2014), *cert. denied*, 135 S. Ct. 2887 (2015); *see supra* note 230.

297. *See supra* text accompanying notes 126–27.

298. *See* Comput. Assocs. Int'l, Inc. v. Altai, Inc., 982 F.2d 693, 702 (2d Cir. 1992).

299. *See* 17 U.S.C. § 102(b) (2012).

300. *See* Oracle Am., Inc. v. Google, Inc., 872 F. Supp. 2d 974, 978 (N.D. Cal. 2012) ("All agree that Google was and remains free to use the Java language itself."), *rev'd*, 750 F.3d 1339 (Fed. Cir. 2014), *cert. denied*, 135 S. Ct. 2887 (2015).

301. *See id.* at 975–76.

302. *See supra* text accompanying notes 200–02.

the comparison step.[303] But the filtration step would have to be shared: the judge would have to filter out the uncopyrightable material to rule on copyrightability, while the jury would have to consider the copyrightability exceptions—including merger—before proceeding to the comparison step.[304] The judge would consider factors for determining whether an element of a computer program was a method of operation, including whether the idea and its expression have merged.[305] Thus, the judge and jury could arrive at inconsistent conclusions. Such a result renders application of the abstraction-filtration-comparison test ineffective.

The confusion inherent in applying copyright jurisprudence to software arises from judicial and legislative attempts to mold dated copyright law to evolving technologies. Concepts such as merger, interoperability, and industry practices are specialized legal attempts to draw lines between expression and function. In this case, both courts cite jurisprudence applying the filtration step of the abstraction-filtration-comparison test.[306] Every court considers merger, interoperability, and industry practices in identifying elements to filter. By inference, these concepts are variations of a situation in which expression and function are identical.

The district court approached this issue by applying relevant law to clearly understood facts. Judge Alsup properly deconstructed the API into its component parts and then considered whether each part represented programming artistry or an underlying electromechanical function. Judge Alsup was better suited than the Federal Circuit to perform this type of analysis because he had a firm grasp of computer programming concepts. He identified the declaring code as representative of

303. *See* Oracle Am., Inc. v. Google, Inc., 750 F.3d 1339, 1358 (Fed. Cir. 2014) ("In all circuits, it is clear that the first step is part of the copyrightability analysis and that the third is an infringement question. It is at the second step of this analysis where the circuits are in less accord."), *cert. denied*, 135 S. Ct. 2887 (2015).

304. *See supra* text accompanying note 281.

305. *Cf.* Oracle Am., Inc. v. Google, Inc., 872 F. Supp. 2d 974, 976–77 (N.D. Cal. 2012) ("When there is only one way to express an idea or function, then everyone is free to do so and no one can monopolize that expression."), *rev'd*, 750 F.3d 1339 (Fed. Cir. 2014), *cert. denied*, 135 S. Ct. 2887 (2015).

306. *See Oracle*, 750 F.3d at 1358 (citing Atari Games Corp. v. Ninento of Am. Inc., 975 F.2d 832, 839 (Fed. Cir. 1992)); *Oracle*, 872 F. Supp. 2d at 989–90 (citing Comput. Assocs. Int'l, Inc. v. Altai, 982 F.2d 693, 708–10 (2d Cir. 1992)).

underlying electromechanical function in the Java language.[307] He recognized creativity in the names and their hierarchy, but held the names uncopyrightable as a matter of law.[308] By deduction, the hierarchy of names is uncopyrightable because its exact replication was required to access the underlying electromechanical function in the Java language.

In summary, instead of focusing on merger, interoperability, and industry practices as separate considerations, Judge Alsup focused on whether a line of code could replicate the same electromechanical function if written in a different way. This approach is consistent with the one suggested by CONTU.[309] Under this test, Judge Alsup's holding is clear: "The method specification is the *idea*. The method implementation is the *expression*. No one may monopolize the *idea*."[310]

The Federal Circuit, on the other hand, used purely legal reasoning while misunderstanding the subject matter. In so doing, the panel mischaracterized both the facts and Judge Alsup's application of the law. Unlike Judge Alsup, the Federal Circuit addressed merger, names, interoperability, and industry practices as separate considerations.[311] This treatment allowed the court to summarily dismiss each of the considerations in turn without having to understand the whole picture of the technology at issue.

The court dismissed merger and interoperability with one argument based on focusing the analysis on the software creator.[312] This distinction makes little sense in the API context, because an API is software written for software writers. When software writers write software in a free language, they are subject to constraints imposed by the language. A developer of an alternative API for a free language must consider the limitations imposed by the language as well. Because of the nature of an API, a copyright over the code that limits the use of the API amounts to a copyright of the language itself. Languages are not

307. Oracle Am., Inc. v. Google, Inc., 872 F. Supp. 2d 974, 976–77 (N.D. Cal. 2012), *rev'd*, 750 F.3d 1339 (Fed. Cir. 2014), *cert. denied*, 135 S. Ct. 2887 (2015).

308. *Id.*

309. *See supra* text accompanying notes 168–70.

310. *Oracle*, 872 F. Supp. 2d at 998.

311. *See* Oracle Am., Inc. v. Google, Inc., 750 F.3d 1339, 1359–72 (Fed. Cir. 2014), *cert. denied*, 135 S. Ct. 2887 (2015).

312. *See id.*

copyrightable.[313] The panel failed to recognize this legal consequence because it did not understand the API.

The court also confused the Java *platform* and the Java *language*.[314] In dismissing the lower court's interoperability argument, the panel pointed to record evidence that Google did not intend to make programs written for the Android platform compatible with the Java platform.[315] The panel missed the issue—Google sought compatibility with the *language*, not the *platform*. The court similarly revealed its confusion by dismissing Google's argument that the Java language was the effective industry standard.[316] Google was justified in catering to software developers who wrote in a language considered the effective industry standard. Oracle was not justified in claiming a copyright over functional elements to prevent other platforms from using the effective industry standard. Needless to say, Oracle's actions will likely drive users away from its *proprietary* Java platform towards open source platforms written in the *free* Java language.

The Federal Circuit also mischaracterized Judge Alsup's reasoning by attributing his opinions to cases he discussed and then attacking those cases. For instance, Judge Alsup held the SSO embodied by the declaring code to be an uncopyrightable method of operation under Section 102(b).[317] The panel opined that this holding "seems to have relied" on *Lotus*, then differentiated *Lotus* from the facts of this case.[318] The panel interpreted *Lotus* to stand for the proposition that nonliteral elements of a program are uncopyrightable under Section 102(b), and attributed this reasoning to Judge Alsup.[319] The panel neglected to mention Judge Alsup's conclusion in which he expressly denied such a broad holding, but limited his holding to the facts of this case.[320] The panel used the same "straw-man" tactic in rejecting his interoperability argument by calling it an

313. *See* 17 U.S.C. § 102(b) (2012).

314. *See* Oracle Am., Inc. v. Google, Inc., 750 F.3d 1339, 1371 (Fed. Cir. 2014), *cert. denied*, 135 S. Ct. 2887 (2015).

315. *Id.*

316. *Id.* at 1372 (describing the API as the effective industry standard though Google's argument points to the Java language the industry standard).

317. Oracle Am., Inc. v. Google, Inc., 872 F. Supp. 2d 974, 977 (N.D. Cal. 2012), *rev'd*, 750 F.3d 1339 (Fed. Cir. 2014), *cert. denied*, 135 S. Ct. 2887 (2015)

318. *Oracle*, 750 F.3d at 1364–66.

319. *Id.* at 1365–66.

320. *Oracle*, 872 F. Supp. 2d at 1002.

"interoperability exception," attributing the argument to two Ninth Circuit fair use cases, and dismissing the fair use cases as inapposite in a copyrightability ruling.[321]

Judge Alsup did survey the jurisprudence on nonliteral element copyrightability, and he did draw legal principles from his survey.[322] Nevertheless, his decision was based on whether a line of code could perform the same electromechanical function if written in a different way. The panel never recognized this distinction. Instead, the panel broadly held that nonliteral elements of computer programs were copyrightable to the extent they represented expression,[323] and then decided the declaring code and SSO constituted expression as a matter of law.[324] Such purely legal reasoning runs directly counter to the CONTU recommendations,[325] especially in the context of a complete reversal of a judge well-versed in computer programming. For this reason, the Federal Circuit reversal is particularly unsettling.

Fortunately for software developers, the Federal Circuit precedent is restricted to Federal Circuit applications of Ninth Circuit law. This may be one reason why the Supreme Court denied Google's writ. However, the Federal Circuit's application of copyright to API declaring code may have an effect disproportionate to its precedential value, as discussed in the next subsection.

2. ECONOMIC UNCERTAINTY

The Federal Circuit's decision will have negative effects on the worldwide technological community. A review of the history of computer technology reveals the impossibility of foreseeing the future of technological change. Nonetheless, an analysis of the history of computer technology and international consensus on technological standards reveals the importance of uncopyrightable interfaces in the rise of computer technology.

"Uncopyrightable interfaces were essential to the

321. Oracle Am., Inc. v. Google, Inc., 750 F.3d 1339, 1368–71 (Fed. Cir. 2014), *cert. denied*, 135 S. Ct. 2887 (2015).

322. Oracle Am., Inc. v. Google, Inc., 872 F. Supp. 2d 974, 983–97 (N.D. Cal. 2012), *rev'd*, 750 F.3d 1339 (Fed. Cir. 2014), *cert. denied*, 135 S. Ct. 2887 (2015).

323. *Oracle*, 750 F.3d at 1355–56.

324. *Id.* at 1359–68.

325. *See supra* text accompanying notes 168–70, 263–64.

development of modern computers and the Internet."[326] The first IBM-compatible personal computer used an interface called BIOS, an early operating system, to allow for user communication.[327] Today, modern personal computers have operating systems that reimplement the open source UNIX API to allow for user communication.[328] The ubiquitous "C" programming language became universal because it is an uncopyrightable interface.[329] In addition, open interface standards allow computers to communicate over the Internet.[330] Most recently developed, cloud computing also relies on APIs.[331]

Unlicensed use of interfaces is "ubiquitous and essential" to compatibility.[332] The open source use of APIs lowers development costs, increases competition, and expands consumer choice.[333] Uncopyrightable interfaces spur software innovation through the development of new software allowing for program compatibility and new capabilities.[334]

An international consensus has developed over twenty-five years that copyright should not interfere with compatibility.[335] Pro-compatibility advocacy started in the United States in reaction to court decisions extending copyright to functional software elements.[336] Consensus spread through the branches of United States government.[337] Today, U.S. bilateral free trade agreements with many nations mandate compatibility protection.[338] The compatibility law of the European Union and

326. Brief of Amici Curiae Comput. Scientists in Support of Petitioner at 6, Google, Inc. v. Oracle Am., Inc., 135 S. Ct. 2887 (2015) (No. 14-410), 2014 WL 5868950.

327. *Id.* at 7.

328. *Id.* at 9–10.

329. *See id.* at 11.

330. *Id.* at 13.

331. *See id.* at 16.

332. Brief of Amici Curiae Hewlett-Packard Co. et al. in Support of Petitioner, *supra* note 288, at 8.

333. Brief of Amici Curiae Open Source Initiative et al. in Support of Petitioner at 12, Google, Inc. v. Oracle Am., Inc., 135 S. Ct. 2887 (2015) (No. 14-410), 2014 WL 5868951.

334. *See* Brief of Amici Curiae Comput. Scientists in Support of Petitioner, *supra* note 326, at 21.

335. Brief Amicus Curiae of the Comput. & Comm'ns Indus. Ass'n in Support of Petitioner at 13, *Google*, 135 S. Ct. 2887 (No. 14-410), 2014 WL 5868946.

336. *Id.* at 13–14.

337. *Id.* at 14–16.

338. *Id.* at 16–17.

several other countries mirrors U.S. law.[339] Copyright law worldwide encourages compatibility.[340]

The disruption of settled expectations about the scope of copyright and the freedom to build interoperable systems is causing "great uncertainty."[341] Amici curiae representing many of the major actors in the technology community filed briefs with the Court, indicating the level of concern over the issue. The briefs highlight the negative effects on software innovation,[342] the threat of undermining international consensus,[343] and the "orphan software" problem.[344]

Amici curiae Software Freedom Law Center and Free Software Foundation supported denial of Google's writ not because amici agreed with the result below, but because its precedential weight is "essentially nil."[345] Amici reasoned that Google could have freely used the Java API under the General Public License, rendering the dispute moot.[346] Further, amici reasoned, the district court's decision was fact-specific, narrowing the scope of its reversal.[347] Nonetheless, amici criticized the Federal Circuit's broad holding that ignored the district court's factual finding that the Java API declaring code was a method of operation.[348] Amici denounced the broad scope of the Federal Circuit's holding for allowing copyrightability if the creators "could have used different names for basic functions, and organized its grammar differently."[349]

---

339. Brief Amicus Curiae of the Comput. & Comm'ns Indus. Ass'n in Support of Petitioner, *supra* note 336, at 18.

340. *Id.* at 17.

341. Brief of Amici Curiae Hewlett-Packard Co. et al. in Support of Petitioner, *supra* note 288, at 14.

342. *Id.*

343. *Id.*

344. Brief of Amici Curiae Comput. Scientists in Support of Petitioner, *supra* note 326, at 23–26. Orphan software results when software is no longer supported by new technologies—preventing access to the software on new technologies—and the copyright holder has disappeared such that the software cannot be licensed for reimplementation for use on new technology. *Id.* at 23. The orphan software problem has a disproportionate effect on the public sector. *Id.* at 26.

345. Brief of Software Freedom Law Ctr. & Free Software Found., Amici Curiae in Support of Respondent 3, Google, Inc. v. Oracle Am., Inc., 135 S. Ct. 2887 (2015) (No. 14-410), 2014 WL 6967821.

346. *Id.* at 12–14.

347. *Id.* at 11–12.

348. *Id.*

349. *Id.* at 11 (citing Oracle Am., Inc. v. Google, Inc., 750 F.3d 1339, 1363 (Fed. Cir. 2014)).

The future effects of this decision are difficult to predict given rapid change in technology. If this decision had been adopted shortly after the *Lotus* decision in 1995, millions of products would have been affected.[350] Furthermore, legal uncertainty has dramatic effects for investment in new technologies.[351] Despite these potentially deleterious effects, the Supreme Court must have given greater weight to judicial considerations of mootness and narrow scope in denying Google's writ. Under the circumstances—a federal law claim, a complete reversal of a fact-based decision, and a host of potential economic effects—the Supreme Court's silence is deafening.

VI. CONCLUSION

Following the denial for its writ of certiorari, Google announced removal of the thirty-seven packages from the next version of the Android platform, opting instead for an open source implementation of the Java platform called OpenJDK.[352] As of this writing, the API copyright litigation still pends before Judge Alsup in the Northern District of California. Oracle is seeking $1 billion in damages from Google.

By denying Google's petition, the Supreme Court missed its chance to correct a misapplication of copyright law to APIs that has upset decades of international consensus. Judge Alsup's opinion exemplifies the proper application of law to facts in the software context. Its reversal shows the importance of technological understanding in software cases. The Supreme Court's denial relegates Judge Alsup's decision to the dustbin of jurisprudential history. His opinion could have served as an educational tool for future law students and practitioners alike. Its reversal leaves the world uncertain.

Nicholas A. Holton

350. Brief of Amici Curiae Open Source Initiative et al. in Support of Petitioner, *supra* note 333, at 23.

351. *See generally* SaveTheInternet.com, *Lawrence Lessig: "Neutral Networks Work"*, YOUTUBE (Apr. 18, 2008), https://www.youtube.com/watch?v=_mYbYG-nXVA (video of comments made by Lessig at an FCC hearing in Stanford, California on April 17, 2008).

352. Ryan Whitwam, *Google Plans to Remove Oracle's Java APIs from Android N*, EXTREMETECH (Dec. 30, 2015, 1:29 PM), http://www.extremetech.com/mobile/220136-google-plans-to-remove-oracles-java-apis-from-android-n.