

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25  
26  
27  
28

IN THE UNITED STATES DISTRICT COURT  
FOR THE NORTHERN DISTRICT OF CALIFORNIA

ORACLE AMERICA, INC.,

No. C 10-03561 WHA

Plaintiff,

v.

**ORDER RE COPYRIGHTABILITY  
OF CERTAIN REPLICATED  
ELEMENTS OF THE  
JAVA APPLICATION  
PROGRAMMING INTERFACE**

GOOGLE INC.,

Defendant.

\_\_\_\_\_ /

**INTRODUCTION**

This action was the first of the so-called “smartphone war” cases tried to a jury. This order includes the findings of fact and conclusions of law on a central question tried simultaneously to the judge, namely the extent to which, if at all, certain replicated elements of the structure, sequence and organization of the Java application programming interface are protected by copyright.

**PROCEDURAL HISTORY**

In 2007, Google Inc., announced its Android software platform for mobile devices. In 2010, Oracle Corporation acquired Sun Microsystems, Inc., and thus acquired Sun’s interest in the popular programming language known as Java, a language used in Android. Sun was renamed Oracle America, Inc. Shortly thereafter, Oracle America (hereinafter simply “Oracle”) sued defendant Google and accused its Android platform as infringing Oracle’s Java-related copyrights and patents.

1 Both Java and Android are complex platforms. Both include “virtual machines,”  
2 development and testing kits, and application programming interfaces, also known as APIs.  
3 Oracle’s copyright claim involves 37 packages in the Java API. Copyrightability of the elements  
4 replicated is the only issue addressed by this order.

5 Due to complexity, the Court decided that the jury (and the judge) would best understand  
6 the issues if the trial was conducted in phases. The first phase covered copyrightability  
7 and copyright infringement as well as equitable defenses. The second phase covered patent  
8 infringement. The third phase would have dealt with damages but was obviated by stipulation  
9 and verdicts.

10 For the first phase, it was agreed that the judge would decide issues of copyrightability  
11 and Google’s equitable defenses and that the jury would decide infringement, fair use, and  
12 whether any copying was de minimis. Significantly, all agreed that Google had not literally  
13 copied the software but had instead come up with its own implementations of the 37 API  
14 packages. Oracle’s central claim, rather, was that Google had replicated the structure, sequence  
15 and organization of the overall code for the 37 API packages.

16 For their task of determining infringement and fair use, the jury was told it should take  
17 for granted that the structure, sequence and organization of the 37 API packages as a whole  
18 *was* copyrightable. This, however, was not a final definitive legal ruling. One reason for this  
19 instruction was so that if the judge ultimately ruled, after hearing the phase one evidence, that  
20 the structure, sequence and organization in question was not protectable but was later reversed  
21 in this regard, the court of appeals might simply reinstate the jury verdict. In this way, the court  
22 of appeals would have a wider range of alternatives without having to worry about an expensive  
23 retrial. Counsel were so informed but not the jury.

24 Each side was given seventeen hours of “air time” for phase one evidence (not counting  
25 openings, closings or motion practice). In phase one, as stated, the parties presented evidence  
26 on copyrightability, infringement, fair use, and the equitable defenses. As to the compilable  
27 code for the 37 Java API packages, the jury found that Google infringed but deadlocked on the  
28 follow-on question of whether the use was protected by fair use. As to the documentation for

1 the 37 Java API packages, the jury found no infringement. As to certain small snippets of code,  
 2 the jury found only one was infringing, namely, the nine lines of code called “rangeCheck.”  
 3 In phase two, the jury found no patent infringement across the board. (Those patents, it should  
 4 be noted, had nothing to do with the subject addressed by this order.) The entire jury portion of  
 5 the trial lasted six weeks.<sup>1</sup>

6 This order addresses and resolves the core premise of the main copyright claims, namely,  
 7 whether the elements replicated by Google from the Java system were protectable by copyright  
 8 in the first place. No law is directly on point. This order relies on general principles of  
 9 copyright law announced by Congress, the Supreme Court and the Ninth Circuit.

10 \* \* \*

11 Counsel on both sides have supplied excellent briefing and the Court wishes to recognize  
 12 their extraordinary effort and to thank counsel, including those behind the scenes burning  
 13 midnight oil in law libraries, for their assistance.

#### 14 **SUMMARY OF RULING**

15 So long as the specific code used to implement a method is different, anyone is free  
 16 under the Copyright Act to write his or her own code to carry out exactly the same function  
 17 or specification of any methods used in the Java API. It does not matter that the declaration or  
 18 method header lines are identical. Under the rules of Java, they *must be identical* to declare a  
 19 method specifying the *same* functionality — even when the implementation is different.  
 20 When there is only one way to express an idea or function, then everyone is free to do so and  
 21 no one can monopolize that expression. And, while the Android method and class names could  
 22 have been different from the names of their counterparts in Java and still have worked, copyright  
 23 protection never extends to names or short phrases as a matter of law.

24 It is true that the very same functionality could have been offered in Android  
 25 without duplicating the exact command structure used in Java. This could have been done

---

26  
 27 <sup>1</sup> After the jury verdict, the Court granted Oracle’s Rule 50 motion for judgment as a matter of law of  
 28 infringement of eight decompiled computer files, which were literally copied. Google admitted to copying eight  
 computer files by decompiling the bytecode from eight Java files into source code and then copying the source  
 code. These files were not proven to have ever been part of Android.

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

4 But the names are more than just names — they are symbols in a command structure  
 5 wherein the commands take the form

6 `java.package.Class.method()`

7 Each command calls into action a pre-assigned function. The overall name tree, of course, has  
 8 creative elements but it is also a precise command structure — a utilitarian and functional set  
 9 of symbols, each to carry out a pre-assigned function. This command structure is a system or  
 10 method of operation under Section 102(b) of the Copyright Act and, therefore, cannot be  
 11 copyrighted. Duplication of the command structure is necessary for interoperability.

## 12 STATEMENT OF FINDINGS

### 13 1. JAVA AND ANDROID.

14 Java was developed by Sun, first released in 1996, and has become one of the world's  
 15 most popular programming languages and platforms.<sup>2</sup> The Java platform, through the use of a  
 16 virtual machine, enables software developers to write programs that are able to run on different  
 17 types of computer hardware without having to rewrite them for each different type. Programs  
 18 that run on the Java platform are written in the Java language. Java was developed to run on  
 19 desktop computers and enterprise servers.<sup>3</sup>

20 The Java language, like C and C++, is a human-readable language. Code written in  
 21 a human-readable language — “source code” — is not readable by computer hardware.

---

22  
 23 <sup>2</sup> For purposes of this order, the term “Java” means the Java platform, sometimes abbreviated to  
 24 “J2SE,” which includes the Java development kit (JDK), javac compiler, tools and utilities, runtime programs,  
 class libraries (API packages), and the Java virtual machine.

25 <sup>3</sup> Rather than merely vet each and every finding and conclusion proposed by the parties, this order has  
 26 navigated its own course through the evidence and arguments, although many of the proposals have found their  
 27 way into this order. Any proposal that has been expressly agreed to by the opposing side, however, shall be  
 28 deemed adopted (to the extent agreed upon) even if not expressly adopted herein. It is unnecessary for this  
 order to cite the record for all of the findings herein. In the findings, the phrase “this order finds . . .” is  
 occasionally used to emphasize a point. The absence of this phrase, however, does not mean (and should not be  
 construed to mean) that a statement is not a finding. All declarative fact statements set forth in the order are  
 factual findings.

1 Only “object code,” which is not human-readable, can be used by computers. Most object code  
2 is in a binary language, meaning it consists entirely of 0s and 1s. Thus, a computer program  
3 has to be converted, that is, compiled, from source code into object code before it can run, or  
4 “execute.” In the Java system, source code is first converted into “bytecode,” an intermediate  
5 form, before it is then converted into binary machine code by the Java virtual machine.

6 The Java language itself is composed of keywords and other symbols and a set of  
7 pre-written programs to carry out various commands, such as printing something on the screen  
8 or retrieving the cosine of an angle. The set of pre-written programs is called the application  
9 programming interface or simply API (also known as class libraries).

10 In 2008, the Java API had 166 “packages,” broken into more than six hundred “classes,”  
11 all broken into over six thousand “methods.” This is very close to saying the Java API had  
12 166 “folders” (packages), all including over six hundred pre-written programs (classes) to carry  
13 out a total of over six thousand subroutines (methods). Google replicated the exact names and  
14 exact functions of virtually all of these 37 packages but, as stated, took care to use different code  
15 to implement the six thousand-plus subroutines (methods) and six-hundred-plus classes.

16 An API is like a library. Each package is like a bookshelf in the library. Each class is  
17 like a book on the shelf. Each method is like a how-to-do-it chapter in a book. Go to the right  
18 shelf, select the right book, and open it to the chapter that covers the work you need. As to the  
19 37 packages, the Java and Android libraries are organized in the same basic way but all of the  
20 chapters in Android have been written with implementations different from Java but solving the  
21 same problems and providing the same functions. Every method and class is specified to carry  
22 out precise desired functions and, thus, the “declaration” (or “header”) line of code stating the  
23 specifications must be identical to carry out the given function.<sup>4</sup>

24 The accused product is Android, a software platform developed by Google for  
25 mobile devices. In August 2005, Google acquired Android, Inc., as part of a plan to develop  
26 a smartphone platform. Google decided to use the Java language for the Android platform.

---

27  
28 <sup>4</sup> The term “declaration” was used throughout trial to describe the headers (non-implementing code)  
for methods and classes. While “header” is the more technically accurate term, this order will remain consistent  
with the trial record and use “declaration” and “header” interchangeably.

1 In late 2005, Google began discussing with Sun the possibility of taking a license to use  
2 and to adapt the entire Java platform for mobile devices. They also discussed a possible  
3 co-development partnership deal with Sun under which Java technology would become  
4 an open-source part of the Android platform, adapted for mobile devices. Google and Sun  
5 negotiated over several months, but they were unable to reach a deal.

6 In light of its inability to reach agreement with Sun, Google decided to use the  
7 Java language to design its own virtual machine via its own software and to write its  
8 own implementations for the functions in the Java API that were key to mobile devices.  
9 Specifically, Google wrote or acquired its own source code to implement virtually all  
10 the functions of the 37 API packages in question. Significantly, all agree that these  
11 implementations — which account for 97 percent of the lines of code in the 37 API packages —  
12 are different from the Java implementations. In its final form, the Android platform also had its  
13 own virtual machine (the so-called Dalvik virtual machine), built with software code different  
14 from the code for the Java virtual machine.

15 As to the 37 packages at issue, Google believed Java application programmers would  
16 want to find the same 37 sets of functionalities in the new Android system callable by the same  
17 names as used in Java. Code already written in the Java language would, to this extent, run on  
18 Android and thus achieve a degree of interoperability.

19 The Android platform was released in 2007. The first Android phones went on sale  
20 the following year. Android-based mobile devices rapidly grew in popularity and now comprise  
21 a large share of the United States market. The Android platform is provided free of charge  
22 to smartphone manufacturers. Google receives revenue through advertisement whenever a  
23 consumer uses particular functions on an Android smartphone. For its part, Sun and Oracle  
24 never successfully developed its own smartphone platform using Java technology.

25 All agree that Google was and remains free to use the Java language itself. All agree  
26 that Google's virtual machine is free of any copyright issues. All agree that the  
27 six-thousand-plus method implementations by Google are free of copyright issues.  
28 The copyright issue, rather, is whether Google was and remains free to replicate the names,

1 organization of those names, and functionality of 37 out of 166 packages in the Java API, which  
2 has sometimes been referred to in this litigation as the “structure, sequence and organization” of  
3 the 37 packages.

4 The Android platform has its own API. It has 168 packages, 37 of which are in  
5 contention. Comparing the 37 Java and Android packages side by side, only three percent  
6 of the lines of code are the same. The identical lines are those lines that specify the names,  
7 parameters and functionality of the methods and classes, lines called “declarations” or “headers.”  
8 In particular, the Android platform replicated the same package, method and class names,  
9 definitions and parameters of the 37 Java API packages from the Java 2SE 5.0 platform.  
10 This three percent is the heart of our main copyright issue.

11 A side-by-side comparison of the 37 packages in the J2SE 5.0 version of Java versus in  
12 the Froyo version of Android shows that the former has a total of 677 classes (plus interfaces)  
13 and 6508 methods wherein the latter has 616 and 6088, respectively. Twenty-one of the  
14 packages have the same number of classes, interfaces and methods, although, as stated, the  
15 method implementations differ.

16 The three percent of source code at issue includes “declarations.” Significantly, the rules  
17 of Java dictate the precise form of certain necessary lines of code called declarations, whose  
18 precise and necessary form explains why Android and Java *must be* identical when it comes to  
19 those particular lines of code. That is, since there is only one way to declare a given method  
20 functionality, everyone using that function must write that specific line of code in the same way.  
21 The same is true for the “calls,” the commands that invoke the methods. To see why this is so,  
22 this order will now review some of the key rules for Java programming. This explanation will  
23 start at the bottom and work its way upward.

## 24 2. THE JAVA LANGUAGE AND ITS API — IMPORTANT DETAILS.

25 Java syntax includes *separators* (e.g., {, }, ;), *operators* (e.g., +, -, \*, /, <, >), *literal*  
26 *values* (e.g., 123, ‘x’, “Foo”), and *keywords* (e.g., if, else, while, return). These elements  
27 carry precise predefined meanings. Java syntax also includes *identifiers* (e.g., String,  
28

1 java.lang.Object), which are used to name specific values, fields, methods, and classes  
2 as described below.

3         These syntax elements are used to form statements, each statement being a single  
4 command executed by the Java compiler to take some action. Statements are run in the sequence  
5 written. Statements are commands that tell the computer to do work.

6         A method is like a subroutine. Once declared, it can be invoked or “called on” elsewhere  
7 in the program. When a method is called on elsewhere in the program or in an application,  
8 “arguments” are usually passed to the method as inputs. The output from the method is known  
9 as the “return.” An example is a method that receives two numbers as inputs and returns the  
10 greater of the two as an output. Another example is a method that receives an angle expressed  
11 in degrees and returns the cosine of that angle. Methods can be much more complicated.

12 A method, for example, could receive the month and day and return the Earth’s declination to  
13 the sun for that month and day.

14         A method consists of the method header and the method body. A method header contains  
15 the name of the method; the number, order, type and name of the parameters used by the method;  
16 the type of value returned by the method; the checked exceptions that the method can throw;  
17 and various method modifiers that provide additional information about the method. At the trial,  
18 witnesses frequently referred to the method header as the “declaration.” This discrepancy has no  
19 impact on the ultimate analysis. The main point is that this header line of code introduces the  
20 method body and specifies very precisely its inputs, name and other functionality. Anyone who  
21 wishes to supply a method with the same functionality must write this line of code in the same  
22 way and must do so no matter how different the implementation may be from someone else’s  
23 implementation.

24         The method body is a block of code that then implements the method. If a method is  
25 declared to have a return type, then the method body must have a statement and the statement  
26 must include the expression to be returned when that line of code is reached. During trial, many  
27 witnesses referred to the method body as the “implementation.” It is the method body that does  
28 the heavy lifting, namely the actual work of taking the inputs, crunching them, and returning an



1 answer. The method body can be short or long. Google came up with its own implementations  
2 for the method bodies and this accounts for 97 percent of the code for the 37 packages.

3       Once the method is written, tested and in place, it can be called on to do its work.  
4 A method call is a line of code *somewhere else*, such as in a different program that calls on  
5 (or invokes) the method and specifies the arguments to be passed to the method for crunching.  
6 The method would be called on using the command format “java.package.Class.method()”  
7 where () indicates the inputs passed to the method. For example,  
8 a = java.package.Class.method() would set the field “a” to equal the return of the method called.  
9 (The words “java.package.Class.method” would in a real program be other names like  
10 “java.lang.Math.max”; “java.package.Class.method” is used here simply to explain the format.)

11       After a method, the next higher level of syntax is the class. A class usually includes  
12 fields that hold values (such as pi = 3.141592) and methods that operate on those values.  
13 Classes are a fundamental structural element in the Java language. A Java program is written as  
14 one or more classes. More than one method can be in a class and more than one class can be in a  
15 package. All code in a Java program must be placed in a class. A class declaration (or header) is  
16 a line that includes the name of the class and other information that define the class. The body of  
17 the class includes fields and methods, and other parameters.

18       Classes can have subclasses that “inherit” the functionality of the class itself. When a  
19 new subclass is defined, the declaration line uses the word “extends” to alert the compiler that  
20 the fields and methods of the parent class are inherited automatically into the new subclass so  
21 that only additional fields or methods for the subclass need to be declared.

22       The Java language does not allow a class to extend (be a subclass of) more than one  
23 parent class. This restrictiveness may be problematic when one class needs to inherit fields  
24 and methods from two different non-related classes. The Java programming language alleviates  
25 this dilemma through the use of “interfaces,” which refers to something different from the word  
26 “interface” in the API acronym. An interface is similar to a class. It can also contain methods.  
27 It is also in its own source code file. It can also be inherited by classes. The distinction is that a  
28

1 class may inherit from more than one interface whereas, as mentioned, a class can only inherit  
2 from one other class.

3 For convenience, classes and interfaces are grouped into “packages” in the same way we  
4 all group files into folders on our computers. There is no inheritance function within packages;  
5 inheritance occurs only at the class and interface level.

6 Here is a simple example of source code that illustrates methods, classes and packages.  
7 The italicized comments on the right are merely explanatory and are not compiled:

```
8  
9 package java.lang; // Declares package java.lang  
10 public class Math { // Declares class Math  
11     public static int max (int x, int y) { // Declares method max  
12         if (x > y) return x ; // Implementation, returns x or  
13         else return y ; // Implementation, returns y  
14     } // Closes method  
15 } // Closes class
```

16  
17 To invoke this method from another program (or class), the following call could be included in  
18 the program:

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

20 Upon reaching this statement, the computer would go and find the max method under the Math  
21 class in the java.lang package, input “2” and “3” as arguments, and then return a “3,” which  
22 would then be set as the value of “a.”

23 The above example illustrates a point critical to our first main copyright issue, namely  
24 that the declaration line beginning “public static” is entirely dictated by the rules of the language.  
25 In order to declare a particular *functionality*, the language *demand*s that the method declaration  
26 take a particular form. There is no choice in how to express it. To be specific, that line reads:

```
27 public static int max (int x, int y) {  
28
```

1 The word “public” means that other programs can call on it. (If this instead says “private,”  
2 then it can only be accessed by other methods inside the same class.) The word “static” means  
3 that the method can be invoked without creating an instance of the class. (If this instead is an  
4 instance method, then it would always be invoked with respect to an object.) The word “int”  
5 means that an integer is returned by the method. (Other alternatives are “boolean,” “char,”  
6 and “String” which respectively mean “true/false,” “single character,” and “character string.”)  
7 Each of these three parameters is drawn from a short menu of possibilities, each possibility  
8 corresponding to a very specific functionality. The word “max” is a name and while any name  
9 (other than a reserved word) could have been used, names themselves cannot be copyrighted, as  
10 will be shown. The phrase “(int x, int y)” identifies the arguments that must be passed into the  
11 method, stating that they will be in integer form. The “x” and the “y” could be “a” and “b” or  
12 “arg1” and “arg2,” so there is a degree of creativity in naming the arguments. Again, names  
13 cannot be copyrighted. (Android did not copy all of the particular argument names used in Java  
14 but did so as to some arguments.) Finally, “{” is the beginning marker that tells the compiler  
15 that the method body is about to follow. The marker is mandatory. The foregoing description  
16 concerns the rules for the language itself. Again, each parameter choice other than the names  
17 has a precise functional choice. If someone wants to implement a particular function, the  
18 declaration specification can only be written in one way.

19 Part of the declaration of a method can list any exceptions. When a program violates  
20 the semantic constraints of the Java language, the Java virtual machine will signal this error to  
21 the program as an exception for special handling. These are specified via “throw” statements  
22 appended at the end of a declaration. Android and Java are not identical in their throw  
23 designations but they are very similar as to the 37 packages at issue.

24 A Java program must have at least one class. A typical program would have more  
25 than one method in a class. Packages are convenient folders to organize the classes.

26 This brings us to the application programming interface. When Java was first introduced  
27 in 1996, the API included eight packages of pre-written programs. At least three of these  
28 packages were “core” packages, according to Sun, fundamental to being able to use the Java

1 language at all. These packages were java.lang, java.io, and java.util. As a practical matter,  
2 anyone free to use the language itself (as Oracle concedes all are), must also use the three core  
3 packages in order to make any worthwhile use of the language. Contrary to Oracle, there is no  
4 bright line between the language and the API.

5 Each package was broken into classes and those in turn broken into methods.  
6 For example, java.lang (a package) included Math (a class) which in turn included max  
7 (a method) to return the greater of two inputs, which was (and remains) callable as  
8 java.lang.Math.max with appropriate arguments (inputs) in the precise form required  
9 (see the example above).

10 After Java's introduction in 1996, Sun and the Java Community Process, a mechanism  
11 for developing a standard specifications for Java classes and methods, wrote hundreds more  
12 programs to carry out various nifty functions and they were organized into coherent packages  
13 by Sun to become the Java application programming interface. In 2008, as stated, the Java API  
14 had grown from the original eight to 166 packages with over six hundred classes with  
15 over six thousand methods. All of it was downloadable from Sun's (now Oracle's) website  
16 and usable by anyone, including Java application developers, upon agreement to certain license  
17 restrictions. Java was particularly useful for writing programs for use via the Internet and  
18 desktop computers.

19 Although the declarations must be the same to achieve the same functionality, the names  
20 of the methods and the way in which the methods are grouped do not have to be the same.  
21 Put differently, many different API organizations could supply the same overall range of  
22 functionality. They would not, however, be interoperable. Specifically, code written for one  
23 API would not run on an API organized differently, for the name structure itself dictates the  
24 precise form of command to call up any given method.

25 To write a fresh program, a programmer names a new class and adds fields and methods.  
26 These methods can call upon the pre-written functions in the API. Instead of re-inventing the  
27 wheels in the API from scratch, programmers can call on the tried-and-true pre-packaged  
28 programs in the API. These are ready-made to perform a vast menu of functions. This is the

1 whole point of the API. For example, a student in high school can write a program that can call  
2 upon `java.lang.Math.max` to return the greater of two numbers, or to find the cosine of an angle,  
3 as one step in a larger homework assignment. Users and developers can supplement the API  
4 with their own specialized methods and classes.

5 The foregoing completes the facts necessary to decide the copyrightability issue but since  
6 Oracle has made much of two small items copied by Google, this order will now make findings  
7 thereon so that there will be proper context for the court of appeals.

### 8 **3. RANGE CHECK AND THE DE-COMPILED TEST FILES.**

9 Oracle has made much of nine lines of code that crept into both Android and Java.  
10 This circumstance is so innocuous and overblown by Oracle that the actual facts, as found  
11 herein by the judge, will be set forth below for the benefit of the court of appeals.

12 Dr. Joshua Bloch worked at Sun from August 1996 through July 2004, eventually  
13 holding the title of distinguished engineer. While working at Sun, Dr. Bloch wrote a nine-line  
14 code for a function called “rangeCheck,” which was put into a larger file, “Arrays.java,” which  
15 was part of the class library for the 37 API packages at issue. The function of rangeCheck was  
16 to check the range of a list of values before sorting the list. This was a very simple function.

17 In 2004, Dr. Bloch left Sun to work at Google, where he came to be the “chief Java  
18 architect” and “Java guru.” Around 2007, Dr. Bloch wrote the files, “Timsort.java” and  
19 “ComparableTimsort,” both of which included the same rangeCheck function he wrote while  
20 at Sun. He wrote the Timsort files in his own spare time and not as part of any Google project.  
21 He planned to contribute Timsort and ComparableTimsort back to the Java community by  
22 submitting his code to an open implementation of the Java platform, OpenJDK, which was  
23 controlled by Sun. Dr. Bloch did, in fact, contribute his Timsort file to OpenJDK and Sun  
24 included Timsort as part of its Java J2SE 5.0 release.

25 In 2009, Dr. Bloch worked on Google’s Android project for approximately one year.  
26 While working on the Android team, Dr. Bloch also contributed Timsort and  
27 ComparableTimsort to the Android platform. Thus, the nine-line rangeCheck function  
28 was copied into Google’s Android. This was how the infringement happened to occur.

1 When discovered, the rangeCheck lines were taken out of the then-current version of Android  
2 over a year ago. The rangeCheck block of code appeared in a class containing 3,179 lines of  
3 code. This was an innocent and inconsequential instance of copying in the context of a massive  
4 number of lines of code.

5 Since the remainder of this order addresses only the issue concerning structure, sequence  
6 and organization, and since rangeCheck has nothing to do with that issue, rangeCheck will not  
7 be mentioned again, but the reader will please remember that it has been readily conceded that  
8 these nine lines of code found their way into an early version of Android.

9 Google also copied eight computer files by decompiling the bytecode from eight Java  
10 files back into source code and then using the source code. These files were merely used as test  
11 files and never found their way into Android or any handset. These eight files have been treated  
12 at trial as a single unit.

13 Line by line, Oracle tested all fifteen million lines of code in Android (and all files used  
14 to test along the way leading up to the final Android) and these minor items were the only items  
15 copied, save and except for the declarations and calls which, as stated, can only be written in one  
16 way to achieve the specified functionality.

## 17 ANALYSIS AND CONCLUSIONS OF LAW

### 18 1. NAMES AND SHORT PHRASES.

19 To start with a clear-cut rule, names, titles and short phrases are not copyrightable,  
20 according to the United States Copyright Office, whose rule thereon states as follows:

21 Copyright law does not protect names, titles, or short phrases or  
22 expressions. Even if a name, title, or short phrase is novel or  
23 distinctive or lends itself to a play on words, it cannot be protected  
by copyright. The Copyright Office cannot register claims to  
exclusive rights in brief combinations of words such as:

- 24 • Names of products or services.
- 25 • Names of business organizations, or groups (including the  
26 names of performing groups).
- 27 • Pseudonyms of individuals (including pen or stage names).
- 28 • Titles of works.

- 1 • Catchwords, catchphrases, mottoes, slogans, or short  
advertising expressions.
- 2
- 3 • Listings of ingredients, as in recipes, labels, or formulas.  
When a recipe or formula is accompanied by an  
4 explanation or directions, the text directions may be  
copyrightable, but the recipe or formula itself remains  
5 uncopyrightable.

6 U.S. Copyright Office, Circular 34; *see* 37 C.F.R. 202.1(a).

7 This rule is followed in the Ninth Circuit. *Sega Enters., Ltd. v. Accolade, Inc.*, 977 F.2d  
8 1510, 1524 n.7 (9th Cir. 1992). This has relevance to Oracle's claim of copyright ownership  
9 over names of methods, classes and packages.

10 **2. THE DEVELOPMENT OF LAW ON THE COPYRIGHTABILITY  
OF COMPUTER PROGRAMS AND THEIR STRUCTURE,  
SEQUENCE AND ORGANIZATION.**

11 Turning now to the more difficult question, this trial showcases a distinction between  
12 copyright protection and patent protection. It is an important distinction, for copyright  
13 exclusivity lasts 95 years whereas patent exclusivity lasts twenty years. And, the Patent and  
14 Trademark Office examines applications for anticipation and obviousness before allowance  
15 whereas the Copyright Office does not. This distinction looms large where, as here, the vast  
16 majority of the code was *not* copied and the copyright owner must resort to alleging that the  
17 accused stole the "structure, sequence and organization" of the work. This phrase — structure,  
18 sequence and organization — does not appear in the Act or its legislative history. It is a phrase  
19 that crept into use to describe a residual property right where literal copying was absent.  
20 A question then arises whether the copyright holder is more appropriately asserting an exclusive  
21 right to a functional system, process, or method of operation that belongs in the realm of patents,  
22 not copyrights.

23 **A. *Baker v. Seldon.***

24 The general question predates computers. In the Supreme Court's decision in *Baker v.*  
25 *Seldon*, 101 U.S. 99 (1879), the work at issue was a book on a new system of double-entry  
26 bookkeeping. It included blank forms, consisting of ruled lines, and headings, illustrating the  
27  
28

1 system. The accused infringer copied the method of bookkeeping but used different forms.

2 The Supreme Court framed the issue as follows:

3 The evidence of the complainant is principally directed to the  
4 object of showing that Baker uses the same system as that which is  
5 explained and illustrated in Selden's books. It becomes important,  
6 therefore, to determine whether, in obtaining the copyright of his  
7 books, he secured the exclusive right to the use of the system or  
8 method of book-keeping which the said books are intended to  
9 illustrate and explain.

7 *Id.* at 101. *Baker* held that using the same accounting system would not constitute copyright  
8 infringement. The Supreme Court explained that only patent law can give an exclusive right to  
9 a method:

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

14 *Id.* at 102. The Supreme Court went on to explain that protecting the method under copyright  
15 law would frustrate the very purpose of publication:

17 The copyright of a work on mathematical science cannot give to  
18 the author an exclusive right to the methods of operation which he  
19 propounds, or to the diagrams which he employs to explain them,  
20 so as to prevent an engineer from using them whenever occasion  
21 requires. The very object of publishing a book on science or the  
22 useful arts is to communicate to the world the useful knowledge  
23 which it contains. But this object would be frustrated if the  
24 knowledge could not be used without incurring the guilt of piracy  
25 of the book.

21 *Id.* at 103. *Baker* also established the "merger" doctrine for systems and methods intermingled  
22 with the texts or diagrams illustrating them:

23 And where the art it teaches cannot be used without employing the  
24 methods and diagrams used to illustrate the book, or such as are  
25 similar to them, such methods and diagrams are to be considered  
26 as necessary incidents to the art, and given therewith to the public;  
27 not given for the purpose of publication in other works explanatory  
28 of the art, but for the purpose of practical application.

27 *Ibid.* It is true that *Baker* is aged but it is not passé. To the contrary, even in our modern era,  
28 *Baker* continues to be followed in the appellate courts, as will be seen below.



1                   **B.       The Computer Age and Section 102(b) of the 1976 Act.**

2                   Almost a century later, Congress revamped the Copyright Act in 1976. By then, software  
3 for computers was just emerging as a copyright issue. Congress decided in the 1976 Act that  
4 computer programs would be copyrightable as “literary works.” *See* H.R. REP. NO. 94-1476,  
5 at 54 (1976). There was, however, no express definition of a computer program until an  
6 amendment in 1980.

7                   The 1976 Act also codified a *Baker*-like limitation on the scope of copyright protection in  
8 Section 102(b). *See Apple Computer, Inc. v. Microsoft Corp.*, 35 F.3d 1435, 1443 n.11 (9th Cir.  
9 1994). Section 102(b) stated (and still states):

10                   In no case does copyright protection for an original work of  
11 authorship extend to any idea, procedure, process, system, method  
12 of operation, concept, principle, or discovery, regardless of the  
13 form in which it is described, explained, illustrated, or embodied in  
14 such work.

15                   The House Report that accompanied Section 102(b) of the Copyright Act explained:

16                   Copyright does not preclude others from using the ideas or  
17 information revealed by the author’s work. It pertains to the  
18 literary, musical, graphic, or artistic form in which the author  
19 expressed intellectual concepts. Section 102(b) makes clear that  
20 copyright protection does not extend to any idea, procedure,  
21 process, system, method of operation, concept, principle, or  
22 discovery, regardless of the form in which it is described,  
23 explained, illustrated, or embodied in such work.

24                   *Some concern has been expressed lest copyright in computer  
25 programs should extend protection to the methodology or  
26 processes adopted by the programmer, rather than merely to the  
27 ‘writing’ expressing his ideas. Section 102(b) is intended, among  
28 other things, to make clear that the expression adopted by the  
programmer is the copyrightable element in a computer program,  
and that the actual processes or methods embodied in the program  
are not within the scope of the copyright law.*

                  Section 102(b) in no way enlarges or contracts the scope of  
copyright protection under the present law. Its purpose is to  
restate, in the context of the new single Federal system of  
copyright, that the basic dichotomy between expression and idea  
remains unchanged.

1 H.R. REP. NO. 94-1476, at 56–57 (1976) (emphasis added).<sup>5</sup>

2 Recognizing that computer programs posed novel copyright issues, Congress established  
3 the National Commission on New Technological Uses of Copyrighted Works (referred to as  
4 CONTU) to recommend the extent of copyright protection for software. The Commission  
5 consisted of twelve members with Judge Stanley Fuld as chairman and Professor Melville  
6 Nimmer as vice-chairman.

7 The Commission recommended that a definition of “computer program” be added to the  
8 copyright statutes. This definition was adopted in 1980 and remains in the current statute:

9 A “computer program” is a set of statements or instructions to be  
10 used directly or indirectly in a computer in order to bring about a  
certain result.

11 17 U.S.C. 101. Moreover, the CONTU report stated that Section 102(b)’s preclusion of  
12 copyright protection for “procedure, process, system, method of operation” was reconcilable  
13 with the new definition of “computer program.” The Commission explained the dichotomy  
14 between copyrightability and non-copyrightability as follows:

15 Copyright, therefore, protects the program so long as it remains  
16 fixed in a tangible medium of expression but does not protect the  
17 electromechanical functioning of a machine. The way copyright  
18 affects games and game-playing is closely analogous: one may not  
adopt and republish or redistribute copyrighted game rules, but the  
copyright owner has no power to prevent others from playing the  
game.

19 *Thus, one is always free to make a machine perform any*  
20 *conceivable process (in the absence of a patent), but one is not free*  
*to take another’s program.*

21 NAT’L COMM’N ON NEW TECHNOLOGICAL USES OF COPYRIGHTED WORKS, FINAL REPORT 20  
22 (1979) (emphasis added). The Commission also recognized the “merger” doctrine, a rule of  
23 importance a few pages below in this order (emphasis added):

24 The “idea-expression identity” exception provides that copyrighted  
25 language may be copied without infringing when there is but a  
26 limited number of ways to express a given idea. This rule is the  
logical extension of the fundamental principle that copyright  
cannot protect ideas. *In the computer context this means that when*

27  
28 <sup>5</sup> The Court has reviewed the entire legislative history. The quoted material above is the only passage  
of relevance. This order includes a summary of the CONTU report but it came after-the-fact and had little  
impact on the Act other than to include a definition of “computer program.”

1                    *specific instructions, even though previously copyrighted, are the*  
 2                    *only and essential means of accomplishing a given task, their later*  
 3                    *use by another will not amount to an infringement . . . .*

4                    [C]opyright protection for programs does not threaten to block the  
 5                    use of ideas or program language previously developed by others  
 6                    when that use is necessary to achieve a certain result. When other  
 7                    language is available, programmers are free to read copyrighted  
 8                    programs and use the ideas embodied in them in preparing their  
 9                    own works.

10                    *Ibid.* The Commission realized that differentiating between the copyrightable form of a program  
 11                    and the uncopyrightable process was difficult, and expressly decided to leave the line drawing to  
 12                    federal courts:

13                    [T]he many ways in which programs are now used and the new  
 14                    applications which advancing technology will supply may make  
 15                    drawing the line of demarcation more and more difficult.  
 16                    To attempt to establish such a line in this report written in 1978  
 17                    would be futile. . . . Should a line need to be drawn to exclude  
 18                    certain manifestations of programs from copyright, that line should  
 19                    be drawn on a case-by-case basis by the institution designed to  
 20                    make fine distinctions — the federal judiciary.

21                    *Id.* at 22–23.

22                    Congress prepared no legislative reports discussing the CONTU comments regarding  
 23                    Section 102(b). *See* H.R. REP. NO. 96-1307, at 23–24 (1980). Nevertheless, Congress followed  
 24                    CONTU’s recommendations by adding the definition of computer programs to the statute and  
 25                    amending a section of the Act not relevant to this order. *See Apple Computer, Inc. v. Formula*  
 26                    *Intern. Inc.*, 725 F.2d 521, 522–25 (9th Cir. 1984).

27                    Everyone agrees that no one can copy line-for-line someone else’s copyrighted computer  
 28                    program. When the line-by-line listings are different, however, some copyright owners have  
 29                    nonetheless accused others of stealing the “structure, sequence and organization” of the  
 30                    copyrighted work. That is the claim here.

### 31                    **C. Decisions Outside the Ninth Circuit.**

32                    No court of appeals has addressed the copyrightability of APIs, much less their  
 33                    structure, sequence and organization. Nor has any district court. Nevertheless, a review of the  
 34                    case law regarding non-literal copying of software provides guidance. Circuit decisions outside  
 35                    the Ninth Circuit will be considered first.

1           The Third Circuit led off in *Whelan Associates, Inc. v. Jaslow Dental Laboratory, Inc.*,  
2 797 F.2d 1222 (3d Cir. 1986). In that case, the claimant owned a program, Dentalab, that  
3 handled the administrative and bookkeeping tasks of dental prosthetics businesses. The accused  
4 infringer developed another program, Dentcom, using a different programming language.  
5 The Dentcom program handled the same tasks as the Dentalab program and had the following  
6 similarities:

7           The programs were similar in three significant respects . . . most  
8 of the file structures, and the screen outputs, of the programs  
9 were virtually identical . . . five particularly important  
10 “subroutines” within both programs — order entry, invoicing,  
11 accounts receivable, end of day procedure, and end of month  
12 procedure — performed almost identically in both programs.

13 *Id.* at 1228. On these facts, the district court had found, after a bench trial, that the accused  
14 infringer copied the claimant’s software program. *Id.* at 1228–29.

15           On appeal, the accused infringer argued that the structure of the claimant’s program was  
16 not protectable under copyright. In rejecting this argument, the court of appeals created the  
17 following framework to deal with non-literal copying of software:

18           [T]he line between idea and expression may be drawn with  
19 reference to the end sought to be achieved by the work in question.  
20 In other words, *the purpose or function of a utilitarian work would  
21 be the work’s idea, and everything that is not necessary to that  
22 purpose or function would be part of the expression of the idea.*

23 *Id.* at 1236 (emphasis in original). Applying this test, *Whelan* found that the structure of  
24 Dentalab was copyrightable because there were many different ways to structure a program that  
25 managed a dental laboratory:

26           [T]he idea of the Dentalab program was the efficient management  
27 of a dental laboratory (which presumably has significantly  
28 different requirements from those of other businesses). Because  
that idea could be accomplished in a number of different ways with  
a number of different structures, the structure of the Dentalab  
program is part of the program’s expression, not its idea.

*Id.* at 1236 n.28. The phrase “structure, sequence and organization” originated in a passage in  
*Whelan* explaining that the opinion used those words interchangeably and that, although not  
themselves part of the Act, they were intended to capture the thought that “sequence and order  
could be parts of the expression, not the idea, of a work.” *Id.* at 1239, 1248.

1 To summarize, in affirming the district court's final judgment of infringement, *Whelan*  
2 held that the *structure* of the Dentalab program was copyrightable because there were many  
3 other ways to perform the same function of handling the administrative and bookkeeping tasks  
4 of dental prosthetics businesses with different structures and designs. *Id.* at 1238. Others were  
5 free to come up with their own version but could not appropriate the Dentalab structure.  
6 This decision plainly seems to have been the high-water mark of copyright protection for the  
7 structure, sequence and organization of computer programs. It was also the only appellate  
8 decision found by the undersigned judge that affirmed (or directed) a final judgment of  
9 copyrightability on a structure, sequence and organization theory.

10 Perhaps because it was the first appellate decision to wade into this problem, *Whelan*  
11 has since been criticized by subsequent treatises, articles, and courts, including our own court  
12 of appeals. *See Sega Enters., Ltd. v. Accolade, Inc.*, 977 F.2d 1510, 1524–25 (9th Cir. 1992).  
13 Instead, most circuits, including ours, have adopted some variation of an approach taken later  
14 by the Second Circuit. *See Apple Computer, Inc. v. Microsoft Corp.*, 35 F.3d 1435, 1445  
15 (9th Cir. 1994).

16 In *Computer Associates International, Inc. v. Altai*, 982 F.2d 693 (2d Cir. 1992),  
17 the claimant owned a program designed to translate the language of another program into  
18 the particular language that the computer's operating system would be able to understand.  
19 The accused infringer developed its own program with substantially similar structure but  
20 different source code (using the same programming language). The Second Circuit criticized  
21 *Whelan* for taking too narrow a view of the "idea" of a program. The Second Circuit adopted  
22 instead an "abstract-filtration-comparison" test. The test first dissected the copyrighted program  
23 into its structural components:

24 In ascertaining substantial similarity under [the  
25 abstract-filtration-comparison test], a court would first break down  
26 the allegedly infringed program into its constituent structural parts.  
27 Then, by examining each of these parts for such things as  
28 incorporated ideas, expression that is necessarily incidental to  
those ideas, and elements that are taken from the public domain, a  
court would then be able to sift out all non-protectable material.

*Id.* at 706.

1           Then, the test filtered out structures that were not copyrightable. For this filtration step,  
2 the court of appeals relied on the premise that programmers fashioned structures “to maximize  
3 the program’s speed, efficiency, as well as simplicity for user operation, while taking into  
4 consideration certain externalities such as the memory constraints of the computer upon which  
5 the program will be run.” *Id.* at 698. Because these were “practical considerations,” the court  
6 held that structures based on these considerations were not copyrightable expressions.

7           Thus, for the filtration step, the court of appeals outlined three types of structures that  
8 should be precluded from copyright protection. *First*, copyright protection did not extend to  
9 structures dictated by efficiency. A court must inquire

10                   whether the use of *this particular set* of modules [is] necessary  
11                   efficiently to implement that part of the program’s process being  
12                   implemented. If the answer is yes, then the expression represented  
13                   by the programmer’s choice of a specific module or group of  
14                   modules has merged with their underlying idea and is unprotected.

15 *Id.* at 708 (emphasis in original). Paradoxically, this meant that non-efficient structures might be  
16 copyrightable while efficient structures may not be. Nevertheless, the Second Circuit explained  
17 its reasoning as follows:

18                   In the context of computer program design, the concept of  
19                   efficiency is akin to deriving the most concise logical proof or  
20                   formulating the most succinct mathematical computation.  
21                   Thus, the more efficient a set of modules are, the more closely  
22                   they approximate the idea or process embodied in that particular  
23                   aspect of the program’s structure.

24                   While, hypothetically, there might be a myriad of ways in  
25                   which a programmer may effectuate certain functions within  
26                   a program — *i.e.*, express the idea embodied in a given  
27                   subroutine — efficiency concerns may so narrow the practical  
28                   range of choice as to make only one or two forms of expression  
workable options.

*Ibid.* Efficiency also encompassed user simplicity and ease of use. *Id.* at 708–09.

*Second*, copyright protection did not extend to structures dictated by external factors.  
The court explained this as follows:

[I]n many instances it is virtually impossible to write a program  
to perform particular functions in a specific computing  
environment without employing standard techniques. This is a  
result of the fact that a programmer’s freedom of design choice  
is often circumscribed by extrinsic considerations such as (1) the  
mechanical specifications of the computer on which a particular

1 program is intended to run; (2) compatibility requirements of  
2 other programs with which a program is designed to operate in  
3 conjunction; (3) computer manufacturers' design standards;  
(4) demands of the industry being serviced; and (5) widely  
accepted programming practices within the computer industry.

4 *Id.* at 709–10.

5 *Third*, copyright protection did not extend to structures already found in the public  
6 domain. The court reasoned that materials in the public domain, such as elements of a computer  
7 program that have been freely accessible, cannot be appropriated. *Ibid.* Ultimately, in the case  
8 before it, the Second Circuit held that after removing unprotectable elements using the criteria  
9 discussed above, only a few lists and macros in accused product were similar to the copied  
10 product, and their impact on the program was not large enough to declare copyright  
11 infringement. *Id.* at 714–15. The copyright claim, in short, failed.

12 The Tenth Circuit elaborated on the abstract-filtration-comparison test in *Gates Rubber*  
13 *Co. v. Bando Chemical Industries, Ltd.*, 9 F.3d 823 (10th Cir. 1993). There, the claimant  
14 developed a computer program that determined the proper rubber belt for a particular machine  
15 by performing complicated calculations involving numerous variables. The program used  
16 published formulas in conjunction with certain mathematical constants developed by the  
17 claimant to determine belt size. The Tenth Circuit offered the following description of a  
18 software program's structure:

19 The program's architecture or structure is a description of how  
20 the program operates in terms of its various functions, which are  
performed by discrete modules, and how each of these modules  
interact with each other.

21 *Id.* at 835. As had the Second Circuit, the Tenth Circuit held that filtration should eliminate the  
22 unprotectable elements of processes, facts, public domain information, merger material, *scenes a*  
23 *faire* material, and other unprotectable elements suggested by the particular facts of the program  
24 under examination. For Section 102(b) processes, the court gave the following description:

25 Returning then to our levels of abstraction framework, we note  
26 that processes can be found at any level, except perhaps the main  
27 purpose level of abstraction. Most commonly, processes will be  
28 found as part of the system architecture, as operations within  
modules, or as algorithms.

1 *Id.* at 837. The court described the *scenes a faire* doctrine for computer programs as follows:

2 The *scenes a faire* doctrine also excludes from protection those  
 3 elements of a program that have been dictated by external factors.  
 4 In the area of computer programs these external factors may  
 5 include: hardware standards and mechanical specifications,  
 6 software standards and compatibility requirements, *Sega*  
*Enterprises Ltd. v. Accolade, Inc.*, 977 F.2d 1510, 1525–27  
 (9th Cir. 1993), computer manufacturer design standards, target  
 industry practices and demands, and computer industry  
 programming practices.

7 \* \* \*

8 We recognize that the *scenes a faire* doctrine may implicate the  
 9 protectability of interfacing and that this topic is very sensitive and  
 10 has the potential to effect [sic] widely the law of computer  
 11 copyright. This appeal does not require us to determine the scope  
 of the *scenes a faire* doctrine as it relates to interfacing and  
 accordingly we refrain from discussing the issue.

12 *Id.* at 838 & n.14 (all citations omitted except *Sega*). Like the Second Circuit, the Tenth Circuit  
 13 also listed many external considerations — such as compatibility, computer industry  
 14 programming practices, and target industry practices and demands — that would exclude  
 15 elements from copyright protection under the *scenes a faire* doctrine. Ultimately, the  
 16 Tenth Circuit remanded because the district court had failed to make specific findings  
 17 that fit this framework.

18 The First Circuit weighed in with its 1995 decision *Lotus Development Corp. v. Borland*  
 19 *International, Inc.*, 49 F.3d 807 (1st Cir. 1995). In *Lotus*, the claimant owned the Lotus 1-2-3  
 20 spreadsheet program that enabled users to perform accounting functions electronically on  
 21 a computer. Users manipulated and controlled the program via a series of menu commands,  
 22 such as “Copy,” “Print,” and “Quit.” In all, Lotus 1-2-3 had 469 commands arranged into more  
 23 than 50 menus and submenus. Lotus 1-2-3 also allowed users to write “macros,” whereby a user  
 24 could designate a series of command choices (sequence of menus and submenus) with a single  
 25 macro keystroke. Then, to execute that series of commands, the user only needed to type the  
 26 single pre-programmed macro keystroke, causing the program to recall and perform the  
 27 designated series of commands automatically. *Id.* at 809–10.

28 The accused infringer Borland developed a competing spreadsheet program.  
 Borland included the Lotus menu command hierarchy in its program to make it compatible



1 with Lotus 1-2-3 so that spreadsheet users who were already familiar with Lotus 1-2-3 would  
2 be able to switch to the Borland program without having to learn new commands or rewrite  
3 their Lotus macros. In so doing, Borland did not copy any of Lotus's underlying source or  
4 object code. (The opinion did not say whether the programs were written in the same language.)

5 The district court had ruled that the Lotus 1-2-3 menu command hierarchy was a  
6 copyrightable expression because there were many ways to construct a spreadsheet menu tree.  
7 Thus, the district court had concluded that the Lotus developers' choice and arrangement of  
8 command terms, reflected in the Lotus menu command hierarchy, constituted copyrightable  
9 expression. *Id.* at 810–11.

10 The First Circuit, however, held that the Lotus menu command hierarchy was not  
11 copyrightable because it was a method of operation under Section 102(b). The court explained:

12 We think that “method of operation,” as that term is used in  
13 § 102(b), refers to the means by which a person operates  
14 something, whether it be a car, a food processor, or a computer.  
15 Thus a text describing how to operate something would not extend  
16 copyright protection to the method of operation itself; other people  
17 would be free to employ that method and to describe it in their  
18 own words. Similarly, if a new method of operation is used rather  
19 than described, other people would still be free to employ or  
20 describe that method.

21 *Id.* at 815.

22 The court reasoned that because the menu command hierarchy was essential to make use  
23 of the program's functional capabilities, it should be properly categorized as a “method of  
24 operation” under Section 102(b). The court explained:

25 The Lotus menu command hierarchy does not merely explain and  
26 present Lotus 1-2-3's functional capabilities to the user; it also  
27 serves as the method by which the program is operated and  
28 controlled . . . . In other words, to offer the same capabilities as  
29 Lotus 1-2-3, Borland did not have to copy Lotus's underlying code  
30 (and indeed it did not); to allow users to operate its programs in  
31 substantially the same way, however, Borland had to copy the  
32 Lotus menu command hierarchy. Thus the Lotus 1-2-3 code is not  
33 a uncopyrightable “method of operation.”

34 *Ibid.* Thus, the court reasoned that although Lotus had made “expressive” choices of what  
35 to name the command terms and how to structure their hierarchy, it was nevertheless an

1 uncopyrightable “method of operation.” The *Lotus* decision was affirmed by an evenly divided  
2 Supreme Court (four to four).

3 The Federal Circuit had the opportunity to apply *Lotus* in an appeal originating from  
4 the District of Massachusetts in *Hutchins v. Zoll Medical Corp.*, 492 F.3d 1377 (Fed. Cir. 2007)  
5 (affirming summary judgment against copyright owner). In *Hutchins*, the claimant owned a  
6 program for performing CPR and argued that his copyright covered the “system of logic  
7 whereby CPR instructions are provided by computerized display, and [] the unique logic  
8 contained in [his] software program.” *Id.* at 1384. The claimant argued that the accused  
9 program was similar because it “perform[ed] the same task in the same way, that is, by  
10 measuring heart activity and signaling the quantity and timing of CPR compressions to be  
11 performed by the rescuer.” *Ibid.* The court of appeals rejected this argument, holding that  
12 copyright did not protect the “technologic method of treating victims by using CPR and  
13 instructing how to use CPR.” *Ibid.* (citing *Lotus*).

#### 14 **D. Decisions in the Supreme Court and in our Circuit.**

15 Our case is governed by the law in the Ninth Circuit and, of course, the Supreme Court.  
16 The Supreme Court missed the opportunity to address these issues in *Lotus* due to the  
17 four-to-four affirmance and has, thus, never reached the general question. Nonetheless, *Baker*,  
18 which is still good law, provides guidance and informs how we should read Section 102(b).

19 Another Supreme Court decision, *Feist Publications, Inc. v. Rural Telephone Services*  
20 *Co., Inc.*, 499 U.S. 340 (1991), which dealt primarily with the copyrightability of purely factual  
21 compilations, provided some general principles. In *Feist*, the Supreme Court considered the  
22 copyrightability of a telephone directory comprised of names, addresses, and phone numbers  
23 organized in alphabetical order. The Supreme Court rejected the notion that copyright law was  
24 meant to reward authors for the “sweat of the brow.” This meant that we should not yield to the  
25 temptation to award copyright protection merely because a lot of sweat went into the work.  
26 The Supreme Court concluded that protection only extended to the original components of an  
27 author’s work. *Id.* at 353. The Supreme Court concluded:

28 This inevitably means that the copyright in a factual compilation  
is thin. Notwithstanding a valid copyright, a subsequent compiler

1           remains free to use the facts contained in another's publication to  
2           aid in preparing a competing work, so long as the competing work  
          does not feature the same selection and arrangement.

3       *Id.* at 349.

4           Turning to our own Ninth Circuit, our court of appeals has recognized that non-literal  
5       components of a program, including the structure, sequence and organization and user interface,  
6       can be protectable under copyright depending on whether the structure, sequence and  
7       organization in question qualifies as an expression of an idea rather than an idea itself.  
8       *Johnson Controls, Inc. v. Phoenix Control Sys., Inc.*, 886 F.2d 1173, 1175 (9th Cir. 1989).  
9       This decision arrived between the Third Circuit's *Whelan* decision and the Second Circuit's  
10      *Computer Associates* decision. *Johnson Controls* is one of Oracle's mainstays herein.

11           In *Johnson Controls*, the claimant developed a system of computer programs to  
12      control wastewater treatment plants. The district court found that the structure, sequence and  
13      organization of the program was expression and granted a preliminary injunction even though  
14      the accused product did not have similar source or object code. *Id.* at 1174. Therefore, the  
15      standard of review on appeal was limited to abuse of discretion and clear error. Our court  
16      of appeals affirmed the preliminary injunction, stating that the claimant's program was very  
17      sophisticated and each individual application was customized to the needs of the purchaser,  
18      indicating there may have been room for individualized expression in the accomplishment  
19      of common functions. Since there was some discretion and opportunity for creativity in the  
20      structure, the structure of the program was expression rather than an idea. *Id.* at 1175.  
21      *Johnson Controls*, however, did not elaborate on which particular structures deserved copyright  
22      protection.

23           In *Brown Bag Software v. Symantec Corp.*, 960 F.2d 1465 (9th Cir. 1992), our court  
24      of appeals outlined a two-part test for determining similarity between computer programs:  
25      the extrinsic and intrinsic tests. This pertained to infringement, not copyrightability.  
26      The claimant, who owned a computer program for outlining, alleged that an accused infringer  
27      copied his program's non-literal features. *Id.* at 1472. The claimant alleged that seventeen  
28

1 specific features in the programs were similar. On summary judgment, the district court had  
2 found that each feature was either not protectable or not similar as a matter of law:

3 The district court ruled that one group of features represented a  
4 claim of copyright in “concepts . . . fundamental to a host of  
5 computer programs” such as “the need to access existing files,  
6 edit the work, and print the work.” As such, these features, which  
7 took the form of four options in the programs’ opening menus,  
8 were held to be unprotectable under copyright.

9 A second group of features involved “nine functions listed in  
10 the menu bar” and the fact that “virtually all of the functions of  
11 the PC-Outline program [ ] can be performed by Grandview.”  
12 The district court declared that “these functions constitute the idea  
13 of the outlining program” and, furthermore, “[t]he expression of  
14 the ideas inherent in the features are . . . distinct.” The court also  
15 held that “the similarity of using the main editing screen to enter  
16 and edit data . . . is essential to the very idea of a computer  
17 outlining program.”

18 The third group of features common to PC-Outline and Grandview  
19 concerned “the use of pull-down windows.” Regarding these  
20 features, the district court made three separate rulings. The court  
21 first found that “[p]laintiffs may not claim copyright protection of  
22 an . . . expression that is, if not standard, then commonplace in the  
23 computer software industry” . . . [and] that the pull-down  
24 windows of the two programs look different.

25 *Id.* at 1472–73. Our court of appeals affirmed the district court’s order without elaborating on  
26 the copyrightability rulings quoted above.

27 In *Atari Games Corp. v. Nintendo of America Inc.*, 975 F.2d 832 (Fed. Cir. 1992),  
28 the Federal Circuit had occasion to interpret Ninth Circuit copyright precedent. In *Atari*, the  
claimant Nintendo sued Atari for copying the Nintendo 10NES program, which prevented the  
Nintendo game console from accepting unauthorized game cartridges. Atari deciphered the  
10NES program through reverse engineering and developed its own program to unlock the  
Nintendo game console. Atari’s new program generated signals indistinguishable from 10NES  
but was written in a different programming language. *Id.* at 835–36.

Applying our Ninth Circuit precedents, *Johnson Controls* and *Brown Bag*, the Federal  
Circuit affirmed the district court’s preliminary injunction for copyright infringement.

1 The Federal Circuit held that the 10NES program contained copyrightable expression because  
2 it had organization and sequencing unnecessary to the unlocking function:

3 Nintendo's 10NES program contains more than an idea or  
4 expression necessarily incident to an idea. Nintendo incorporated  
5 within the 10NES program creative organization and sequencing  
6 *unnecessary* to the lock and key function. Nintendo chose  
7 arbitrary programming instructions and arranged them in a unique  
8 sequence to create a purely arbitrary data stream. This data stream  
9 serves as the key to unlock the NES. Nintendo may protect this  
10 creative element of the 10NES under copyright.

11 *Id.* at 840 (emphasis added). The Federal Circuit stated that there were creative elements in the  
12 10NES program

13 beyond the literal expression used to effect the unlocking process.  
14 The district court defined the unprotectable 10NES idea or process  
15 as the generation of a data stream to unlock a console. This court  
16 discerns no clear error in the district court's conclusion.  
17 The unique arrangement of computer program expression which  
18 generates that data stream does not merge with the process so long  
19 as alternate expressions are available. In this case, Nintendo has  
20 produced expert testimony showing a multitude of different ways  
21 to generate a data stream which unlocks the NES console.

22 *Ibid.* (citation omitted). Thus, the Federal Circuit held that the district court did not err in  
23 concluding that the 10NES program contained protectable expression and affirmed the  
24 preliminary injunction.

25 Next came two decisions holding that Section 102(b) bars from copyright software  
26 interfaces necessary for interoperability. The Section 102(b) holdings arose in the context of  
27 larger holdings that it had been fair use to copy software to reverse-engineer it so as to isolate  
28 the unprotectable segments. These two decisions will now be described in detail.

29 In *Sega Enterprises Ltd. v. Accolade, Inc.*, 977 F.2d 1510 (9th Cir. 1992), the accused  
30 infringer had to copy object code in order to understand the interface procedures between the  
31 Sega game console and a game cartridge, that is, how the software in the game console  
32 interacted with the software in the game cartridge to achieve compatibility. *Id.* at 1515–16.  
33 After learning and documenting these interactions (interface procedures), the accused infringer  
34 wrote its own source code to mimic those same interface procedures in its own game cartridges  
35 so that its cartridges could run on the Sega console. Our court of appeals held that the copying  
36 of object code for the purpose of achieving compatibility was fair use. Notably, in its fair-use

1 analysis, our court of appeals *expressly held that the interface procedures for compatibility were*  
 2 *functional aspects not copyrightable under Section 102(b)*: “Accolade copied Sega’s software  
 3 solely in order to discover the functional requirements for compatibility with the Genesis console  
 4 — aspects of Sega’s programs that are not protected by copyright. 17 U.S.C. § 102(b).” *Id.* at  
 5 1522. The court used the phrase “interface procedures,” a term describing the interface between  
 6 applications, multiple times to describe the functional aspect of the interaction between software  
 7 programs and summarized its analysis of copyrightability as follows:

8           In summary, the record clearly establishes that disassembly of the  
 9 object code in Sega’s video game cartridges was necessary in  
 10 order to understand the functional requirements for Genesis  
 11 compatibility. The *interface procedures* for the Genesis console  
 12 are distributed for public use only in object code form, and are  
 13 not visible to the user during operation of the video game program.  
 14 Because object code cannot be read by humans, it must be  
 15 disassembled, either by hand or by machine. Disassembly of  
 16 object code necessarily entails copying. Those facts dictate our  
 17 analysis of the second statutory fair use factor. If disassembly of  
 18 copyrighted object code is per se an unfair use, the owner of the  
 19 copyright gains a de facto monopoly over *the functional aspects*  
 20 *of his work — aspects that were expressly denied copyright*  
 21 *protection by Congress.* 17 U.S.C. § 102(b). In order to enjoy a  
 22 lawful monopoly over the idea or functional principle underlying a  
 23 work, the creator of the work must satisfy the more stringent  
 24 standards imposed by the patent laws. *Bonito Boats, Inc. v.*  
 25 *Thunder Craft Boats, Inc.*, 489 U.S. 141, 159–64, 109 S.Ct. 971,  
 26 982–84, 103 L.Ed.2d 118 (1989). Sega does not hold a patent on  
 27 the Genesis console.

18 *Sega*, 977 F.2d at 1526 (emphasis added). In *Sega*, the interface procedure that was required for  
 19 compatibility was “20 bytes of initialization code plus the letters S–E–G–A.” *Id.* at 1524 n.7.

20 Our court of appeals found that this interface procedure was functional and therefore not  
 21 copyrightable under Section 102(b). The accused infringer Accolade was free to copy this  
 22 interface procedure for use in its own games to ensure compatibility with the Sega Genesis game  
 23 console. Our court of appeals distinguished the *Atari* decision, where the Federal Circuit had  
 24 found that the Nintendo’s 10NES security system was infringed, because there was only one  
 25 signal that unlocked the Sega console, unlike the “multitude of different ways to unlock” the  
 26 Nintendo console:

27           We therefore reject Sega’s belated suggestion that Accolade’s  
 28 incorporation of the code which “unlocks” the Genesis III console  
 is not a fair use. Our decision on this point is entirely consistent

1 with *Atari v. Nintendo*, 975 F.2d 832 (Fed. Cir.1992). Although  
2 *Nintendo* extended copyright protection to Nintendo's 10NES  
3 security system, that system consisted of an original program  
4 which generates an arbitrary data stream "key" which unlocks the  
5 NES console. Creativity and originality went into the design of  
6 that program. *See id.* at 840. Moreover, the federal circuit  
7 concluded that there is a "multitude of different ways to generate a  
8 data stream which unlocks the NES console." *Atari*, 975 F.2d at  
9 839. The circumstances are clearly different here. Sega's key  
10 appears to be functional. It consists merely of 20 bytes of  
11 initialization code plus the letters S-E-G-A. There is no showing  
12 that there is a multitude of different ways to unlock the Genesis III  
13 console.

14 *Sega*, 977 F.2d at 1524 n.7.

15 This order reads *Sega* footnote seven (quoted above) as drawing a line between copying  
16 functional aspects necessary for compatibility (not copyrightable) versus copying functional  
17 aspects unnecessary for compatibility (possibly copyrightable). Our court of appeals explained  
18 that in *Atari*, the Nintendo game console's 10NES program had had functionality *unnecessary* to  
19 the lock-and-key function. *See also Atari*, 975 F.2d at 840. Since the accused infringer Atari  
20 had copied the entire 10NES program, it also had copied aspects of the 10NES program  
21 unnecessary for compatibility between the console and game cartridges. This was inapposite to  
22 the facts of *Sega*, where the accused infringer Accolade's final product duplicated *only* the  
23 aspect of Sega's program *necessary* for compatibility between the console and game cartridges.  
24 Thus, the holding of our court of appeals was that the aspect of a program necessary for  
25 compatibility was unprotectable, specifically invoking Section 102(b), but copyrightable  
26 expression could still exist for aspects unnecessary for compatibility.

27 The *Sega* decision and its compatibility reasoning was followed in a subsequent  
28 reverse-engineering decision by our court of appeals, *Sony Computer Entertainment, Inc., v.*  
*Connectix Corporation*, 203 F.3d 596 (9th Cir. 2000). The facts were somewhat different in  
*Sony*. There, the accused infringer Connectix did not create its own games for Sony's  
Playstation game console; instead, the accused infringer created an emulated environment that  
duplicated the interface procedures of Sony's console so that games written for Sony's console  
could be played on a desktop computer running the emulator. In order to do this, the accused  
infringer copied object code for the Sony Playstation's operating software, its BIOS program, in

1 order to discover signals sent between the BIOS and the rest of the game console. *Id.* at 600.  
 2 After uncovering these signals (again, application interfaces), the accused infringer wrote its own  
 3 source code to *duplicate these interfaces* in order to create its emulator for the desktop computer.  
 4 Thus, games written for the Playstation console were playable on Connectix’s emulator for  
 5 the desktop computer. Citing Section 102(b) and *Sega*, our court of appeals stated that the  
 6 Playstation BIOS contained “unprotected functional elements,” and concluded that the accused  
 7 infringer’s intermediate step of copying object code was fair use because it was done for the  
 8 “purpose of gaining access to the unprotected elements of Sony’s software.” *Id.* at 602–03.<sup>6</sup>

9 \* \* \*

10 With apology for its length, the above summary of the development of the law reveals a  
 11 trajectory in which enthusiasm for protection of “structure, sequence and organization” peaked  
 12 in the 1980s, most notably in the Third Circuit’s *Whelan* decision. That phrase has not been  
 13 re-used by the Ninth Circuit since *Johnson Controls* in 1989, a decision affirming preliminary  
 14 injunction. Since then, the trend of the copyright decisions has been more cautious. This trend  
 15 has been driven by fidelity to Section 102(b) and recognition of the danger of conferring a  
 16 monopoly by copyright over what Congress expressly warned should be conferred only by  
 17 patent. This is not to say that infringement of the structure, sequence and organization is a dead  
 18 letter. To the contrary, it is not a dead letter. It is to say that the *Whelan* approach has given way  
 19 to the *Computer Associates* approach, including in our own circuit. *See Sega Enters., Ltd. v.*  
 20 *Accolade, Inc.*, 977 F.2d 1510, 1525 (9th Cir. 1992); *Apple Computer, Inc. v. Microsoft Corp.*,  
 21 35 F.3d 1435, 1445 (9th Cir. 1994).

22 In this connection, since the CONTU report was issued in 1980, the number of software  
 23 patents in force in the United States has dramatically increased from barely a thousand in  
 24 1980 to hundreds of thousands today. *See* Iain Cockburn, *Patents, Tickets and the Financing*

---

25  
 26 <sup>6</sup> *Sega* and *Sony* are not the only Ninth Circuit decisions placing a premium on functionality as  
 27 indicating uncopyrightability. Other such decisions were surveyed in the summary earlier in this order. *See*  
 28 *also Triad Sys. Corp. v. Southeastern Exp. Co.*, 64 F.3d 1330, 1336 (9th Cir. 1995); *Apple Computer, Inc. v.*  
*Microsoft Corp.*, 35 F.3d 1435, 1444 (9th Cir. 1994); *Apple Computer, Inc. v. Formula Intern., Inc.*, 725 F.2d  
 521, 525 (9th Cir. 1984).



1 of *Early-Stage Firms: Evidence from the Software Industry*, 18 JOURNAL OF ECONOMICS &  
 2 MANAGEMENT STRATEGY 729–73 (2009). This has caused at least one noted commentator to  
 3 observe:

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

8 Mark Lemley, *Convergence in the Law of Software Copyright?*, 10 HIGH TECHNOLOGY LAW  
 9 JOURNAL 1, 26–27 (1995). Both Oracle and Sun have applied for and received patents that claim  
 10 aspects of the Java API. *See, e.g.*, U.S. Patents 6,598,093 and 7,006,855. (These were not  
 11 asserted at trial.)<sup>7</sup>

12 \* \* \*

13 In view of the foregoing, this order concludes that our immediate case is controlled by  
 14 these principles of copyright law:

- 15 • Under the merger doctrine, when there is only one (or only a few)  
 16 ways to express something, then no one can claim ownership of  
 17 such expression by copyright.
- 18 • Under the names doctrine, names and short phrases are not  
 19 copyrightable.
- 20 • Under Section 102(b), copyright protection never extends to any  
 21 idea, procedure, process, system, method of operation or concept

22

---

23 <sup>7</sup> The issue has been debated in the journals. For example, Professor Pamela Samuelson has argued  
 24 that Section 102(b) codified the *Baker* exclusion of procedures, processes, systems, and methods of operation  
 25 for computer programs as well as the pre-*Baker* exclusion of high-level abstractions such as ideas, concepts, and  
 26 principles. Pamela Samuelson, *Why Copyright Law Excludes Systems and Processes from the Scope of*  
 27 *Protection*, 85 TEX. L. REV. 1921 (2007). In contrast, Professor David Nimmer (the son of Professor Melville  
 28 Nimmer) has argued that Section 102(b) should not deny copyright protection to “the expression” of a work  
 even if that work happens to consist of an idea, procedure or process. 1-2 NIMMER ON COPYRIGHT § 2.03[D]  
 (internal citations omitted). Similarly, Professor Jane Ginsburg has argued that the Section 102(b) terms  
 “process,” “system,” and “method of operation” should not be understood literally for computer programs. Jane  
 Ginsburg, *Four Reasons and a Paradox: The Manifest Superiority of Copyright Over Sui Generis Protection of*  
*Computer Software*, 94 COLUM. L. REV. 2559, 2569–70 (1994).

1           regardless of its form. Functional elements essential for  
2           interoperability are not copyrightable.

- 3           • Under *Feist*, we should not yield to the temptation to find  
4           copyrightability merely to reward an investment made in a body of  
5           intellectual property.

#### 6           **APPLICATION OF CONTROLLING LAW TO CONTROLLING FACTS**

7           All agree that everyone was and remains free to program in the Java language itself.

8           All agree that Google was free to use the Java language to write its own API. While Google  
9           took care to provide fresh line-by-line implementations (the 97 percent), it generally replicated  
10          the overall name organization and functionality of 37 packages in the Java API (the  
11          three percent). The main issue addressed herein is whether this violated the Copyright Act and  
12          more fundamentally whether the replicated elements were copyrightable in the first place.

13          This leads to the first holding central to this order and it concerns the method level.

14          The reader will remember that a method is like a subroutine and over six thousand are in play  
15          in this proceeding. As long as the specific code written to implement a method is different,  
16          anyone is free under the Copyright Act to write his or her own method to carry out exactly the  
17          same function or specification of any and all methods used in the Java API. Contrary to Oracle,  
18          copyright law does not confer ownership over any and all ways to implement a function or  
19          specification, no matter how creative the copyrighted implementation or specification may be.  
20          The Act confers ownership only over the specific way in which the author wrote out his version.  
21          Others are free to write their own implementation to accomplish the identical function, for,  
22          importantly, ideas, concepts and functions cannot be monopolized by copyright.

23          To return to our example, one method in the Java API carries out the function of  
24          comparing two numbers and returning the greater. Google — and everyone else in the world —  
25          was and remains free to write its own code to carry out the identical function so long as the  
26          implementing code in the method body is different from the copyrighted implementation. This is  
27          a simple example, but even if a method resembles higher mathematics, everyone is still free to  
28          try their hand at writing a different implementation, meaning that they are free to use the same

1 inputs to derive the same outputs (while throwing the same exceptions) so long as the  
2 implementation in between is their own. The House Report, quoted above, stated in 1976 that  
3 “the actual processes or methods embodied in the program are not within the scope of the  
4 copyright law.” H.R. REP. NO. 94-1476, at 57 (1976).

5 Much of Oracle’s evidence at trial went to show that the design of methods in an API  
6 was a creative endeavor. Of course, that is true. Inventing a new method to deliver a new output  
7 can be creative, even inventive, including the choices of inputs needed and outputs returned.  
8 The same is true for classes. But such inventions — at the concept and functionality level —  
9 are protectable only under the Patent Act. The Patent and Trademark Office examines such  
10 inventions for validity and if the patent is allowed, it lasts for twenty years. Based on a single  
11 implementation, Oracle would bypass this entire patent scheme and claim ownership over any  
12 and all ways to carry out methods for 95 years — without any vetting by the Copyright Office  
13 of the type required for patents. This order holds that, under the Copyright Act, no matter  
14 how creative or imaginative a Java method specification may be, the entire world is entitled  
15 to use the same method specification (inputs, outputs, parameters) so long as the line-by-line  
16 implementations are different. To repeat the Second Circuit’s phrasing, “there might be  
17 a myriad of ways in which a programmer may . . . express the idea embodied in a given  
18 subroutine.” *Computer Associates*, 982 F.2d at 708. The method specification is the *idea*.  
19 The method implementation is the *expression*. No one may monopolize the *idea*.<sup>8</sup>

20 To carry out any given function, the method specification as set forth in the declaration  
21 *must be identical* under the Java rules (save only for the choices of argument names). Any other  
22 declaration would carry out some *other* function. The declaration requires precision.  
23 Significantly, when there is only one way to write something, the merger doctrine bars anyone  
24 from claiming exclusive copyright ownership of that expression. Therefore, there can be no  
25 copyright violation in using the identical declarations. Nor can there be any copyright violation

---

27 <sup>8</sup> Each method has a singular purpose or function, and so, the basic function or purpose of a method  
28 will be an unprotectable process. *Gates Rubber Co. v. Bando Chemical Industries, Ltd.*, 9 F.3d 823, 836  
(10th Cir. 1993); see *Apple Computer, Inc. v. Formula Intern. Inc.*, 725 F.2d 521, 525 (9th Cir. 1984) (holding  
that while a particular set of instructions is copyrightable, the underlying computer process is not).

1 due to the *name* given to the method (or to the arguments), for under the law, names and short  
2 phrases cannot be copyrighted.

3 In sum, Google and the public were and remain free to write their own implementations  
4 to carry out exactly the same functions of all methods in question, using exactly the same method  
5 specifications and names. Therefore, at the method level — the level where the heavy lifting is  
6 done — Google has violated no copyright, it being undisputed that Google’s implementations  
7 are different.

8 As for classes, the rules of the language likewise insist on giving names to classes and  
9 the rules insist on strict syntax and punctuation in the lines of code that declare a class. As with  
10 methods, for any desired functionality, the declaration line will *always* read the same (otherwise  
11 the functionality would be different) — save only for the name, which cannot be claimed  
12 by copyright. Therefore, under the law, the declaration line cannot be protected by copyright.  
13 This analysis is parallel to the analysis for methods. This now accounts for virtually all of the  
14 three percent of similar code.

15 \* \* \*

16 Even so, the second major copyright question is whether Google was and remains free to  
17 group its methods in the same way as in Java, that is, to organize its Android methods under the  
18 same class and package scheme as in Java. For example, the Math classes in both systems have  
19 a method that returns a cosine, another method that returns the larger of two numbers, and yet  
20 another method that returns logarithmic values, and so on. As Oracle notes, the rules of Java  
21 did not insist that these methods be grouped together in any particular class. Google could have  
22 placed its trigonometric function (or any other function) under a class other than Math class.  
23 Oracle is entirely correct that the rules of the Java language did not require that the same  
24 grouping pattern (or even that they be grouped at all, for each method could have been placed  
25 in a stand-alone class).<sup>9</sup>

---

26  
27 <sup>9</sup> As to the groupings of methods within a class, Google invokes the *scenes a faire* doctrine. That is,  
28 Google contends that the groupings would be so expected and customary as to be permissible under the *scenes a faire* doctrine. For example, the methods included under the Math class are typical of what one would expect to see in a group of math methods. Just as one would expect certain items in the alcove for nuts, bolts and screws

1 Oracle's best argument, therefore, is that while no single name is copyrightable, Java's  
 2 overall system of organized names — covering 37 packages, with over six hundred classes, with  
 3 over six thousand methods — is a “taxonomy” and, therefore, copyrightable under *American*  
 4 *Dental Association v. Delta Dental Plans Association*, 126 F.3d 977 (7th Cir. 1997). There was  
 5 nothing in the rules of the Java language that required that Google replicate the same groupings  
 6 even if Google was free to replicate the same functionality.<sup>10</sup>

7 The main answer to this argument is that while the overall scheme of file name  
 8 organization resembles a taxonomy, it is *also* a command structure for a system or method  
 9 of operation of the application programming interface. The commands are (and must be) in  
 10 the form

11 `java.package.Class.method()`

12 and each calls into action a pre-assigned function.<sup>11</sup>

13 To repeat, Section 102(b) states that “in no case does copyright protection for an original  
 14 work of authorship extend to any idea, procedure, process, system, method of  
 15 operation . . . regardless of the form . . . .” That a system or method of operation has thousands  
 16 of commands arranged in a creative taxonomy does not change its character as a method of  
 17 operation. Yes, it is creative. Yes, it is original. Yes, it resembles a taxonomy. But it is  
 18 nevertheless a command structure, a system or method of operation — a long hierarchy of

19 \_\_\_\_\_  
 20 in a hardware store, one would expect the methods of the math class to be in, say, a typical math class. At trial,  
 21 however, neither side presented evidence from which we can now say that the same is true for all the other  
 22 hundreds of classes at issue. Therefore, it is impossible to say on this record that *all* of the classes and their  
 contents are typical of such classes and, on this record, this order rejects Google's global argument based on  
*scenes a faire*.

23 <sup>10</sup> This is a good place to point out that while the groupings appear to be the same, when we drill down  
 24 into the detail code listings, we see that the actual sequences of methods in the listings are different. That is, the  
 25 sequence of methods in the class Math in Android is different from the sequence in the same class in Java,  
 26 although all of the methods in the Java version can be found somewhere in the Android version, at least as  
 27 shown in their respective listings (TX 47.101, TX 623.101). The Court has not compared all six-hundred-plus  
 classes. Nor has any witness or counsel so far on the record. Oracle does not, however, contend that the actual  
 sequences would track method-for-method and it has not so proven. This detailed observation, however, does  
 not change the fact that all of the methods in the Java version can be found somewhere in the Android version,  
 classified under the same classes.

28 <sup>11</sup> The parentheses indicate that inputs/arguments may be included in the command.

1 over six thousand commands to carry out pre-assigned functions. For that reason, it cannot  
2 receive copyright protection — patent protection perhaps — but not copyright protection.

3 \* \* \*

4 Interoperability sheds further light on the character of the command structure as a system  
5 or method of operation. Surely, millions of lines of code had been written in Java before  
6 Android arrived. These programs necessarily used the `java.package.Class.method()` command  
7 format. These programs called on all or some of the specific 37 packages at issue and  
8 necessarily used the command structure of names at issue. Such code was owned by  
9 the developers themselves, not by Oracle. *In order for at least some of this code to run on*  
10 *Android, Google was required to provide the same `java.package.Class.method()` command*  
11 *system using the same names with the same “taxonomy” and with the same functional*  
12 *specifications.* Google replicated what was necessary to achieve a degree of interoperability —  
13 but no more, taking care, as said before, to provide its own implementations.

14 That interoperability is at the heart of the command structure is illustrated by Oracle’s  
15 preoccupation with what it calls “fragmentation,” meaning the problem of having imperfect  
16 interoperability among platforms. When this occurs, Java-based applications may not run  
17 on the incompatible platforms. For example, Java-based code using the replicated parts of the  
18 37 API packages will run on Android but will not if a 38th package is needed. Such imperfect  
19 interoperability leads to a “fragmentation” — a Balkanization — of platforms, a circumstance  
20 which Sun and Oracle have tried to curb via their licensing programs. In this litigation, Oracle  
21 has made much of this problem, at times almost leaving the impression that if only Google had  
22 replicated *all* 166 Java API packages, Oracle would not have sued. While fragmentation is a  
23 legitimate business consideration, it begs the question whether or not a license was required in  
24 the first place to replicate some or all of the command structure. (This is especially so inasmuch  
25 as Android has not carried the Java trademark, and Google has not held out Android as fully  
26 compatible.) The immediate point is this: fragmentation, imperfect interoperability, and  
27 Oracle’s angst over it illustrate the character of the command structure as a functional system or  
28 method of operation.



1 composed entirely of a system of commands to carry out specified computer functions. For a  
2 similar reason, Oracle’s analogy to stealing the plot and character from a movie is inapt, for  
3 movies involve no “system” or “method of operation” — scripts are entirely creative.

4 In *ADA*, Judge Frank Easterbrook (writing for the panel) suggested that a “system” under  
5 Section 102(b) had to come with “instructions for use.” 126 F.3d at 980. Because the taxonomy  
6 there at issue had no instructions for use, among other reasons, it was held not to be a system.  
7 By contrast, the API at issue here does come with instructions for use, namely, the  
8 documentation and embedded comments that were much litigated at trial. They describe every  
9 package, class and method, what inputs they need, and what outputs they return — the classic  
10 form of instructions for use.

11 In our circuit, the structure, sequence and organization of a computer program may (or  
12 may not) qualify as a protectable element depending on the “particular facts of each case” and  
13 always subject to exclusion of unprotectable elements. *Johnson Controls v. Phoenix Control*  
14 *Sys.*, 886 F.2d 1173, 1175 (9th Cir. 1989). Contrary to Oracle, *Johnson Controls* did not hold  
15 that all structure, sequence and organization in all computer programs are within the protection  
16 of a copyright. On a motion for preliminary injunction, the district court found that the structure,  
17 sequence and organization of the copyrighted program, on the facts there found, deserved  
18 copyright protection. (The structure, sequence and organization features found protectable were  
19 not described in the appellate decision.) On an appeal from the preliminary injunction, our court  
20 of appeals merely said no clear error had occurred. Again, the appellate opinion stated that the  
21 extent to which the structure, sequence and organization was protectable depended on the facts  
22 and circumstances of each case. The circumstances there are not the circumstances here.

23 In closing, it is important to step back and take in the breadth of Oracle’s claim. Of the  
24 166 Java packages, 129 were not violated in any way. Of the 37 accused, 97 percent of the  
25 Android lines were new from Google and the remaining three percent were freely replicable  
26 under the merger and names doctrines. Oracle must resort, therefore, to claiming that it owns,  
27 by copyright, the exclusive right to any and all possible implementations of the taxonomy-like  
28 command structure for the 166 packages and/or any subpart thereof — even though it



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

5 **CONCLUSION**

6 This order does not hold that Java API packages are free for all to use without license.  
7 It does not hold that the structure, sequence and organization of all computer programs may be  
8 stolen. Rather, it holds on the specific facts of this case, the particular elements replicated by  
9 Google were free for all to use under the Copyright Act. Therefore, Oracle's claim based on  
10 Google's copying of the 37 API packages, including their structure, sequence and organization  
11 is **DISMISSED**. To the extent stated herein, Google's Rule 50 motions regarding copyrightability  
12 are **GRANTED** (Dkt. Nos. 984, 1007). Google's motion for a new trial on copyright infringement  
13 is **DENIED AS MOOT** (Dkt. No. 1105).

14  
15 **IT IS SO ORDERED.**

16  
17 Dated: May 31, 2012.

18   
\_\_\_\_\_  
19 WILLIAM ALSUP  
20 UNITED STATES DISTRICT JUDGE  
21  
22  
23  
24  
25  
26  
27  
28