Application Programming Interfaces ("APIs")
A Primer and Discussion of Oracle America v. Google

Editors:
Matthew C. Wagner
Samuel Van Eichner
Yelena Morozova

Contributing Authors:
Matthew C. Wagner
Samuel Van Eichner
Melanie Graham
Shawn Greene
Jordan Joachim
Quinn Stine

Prepared by the Copyright Law Committee of the American Intellectual Property Law Association
Nancy J. Mertzel, Chair
Stefan Mentzer, Vice Chair and Incoming Chair
Matthew C. Wagner, Incoming Vice Chair

August 30, 2013
Table of Contents:

Introduction 1

I. DEFINITION OF AN API 3

II. SUMMARY OF THE DISTRICT COURT DECISION 11

III. ISSUES ON APPEAL 17
   A. Issues on Direct Appeal 18
   B. Issues on Cross-Appeal 24

IV. DISCUSSION OF AMICUS SUBMISSIONS 26
   A. Amici Urging Reversal 26
      1. Amicus Brief of Microsoft Corporation, EMC Corporation, and Netapp, Inc. 26
      2. Amicus Brief of Ralph Oman, former Register of Copyrights of the United States 28
      3. Amicus Brief of Picture Archive Council of America Inc. & Graphic Artists Guild 31
      4. Amicus Brief of BSA | The Software Alliance 33
      5. Amicus Brief of Scott McNealy and Brian Sutphin 35
      6. Amicus Brief of Eugene H. Spafford, Ph.D., Zhi Ding, Ph.D. and Lee A. Hollaar, Ph.D. 36
   B. Amici Urging Affirmance 38
      1. Amicus Brief of the Intellectual Property Law Professors 38
      2. Amicus Brief of Rackspace US, Inc., Application Developers Alliance, TMSOFT, LLC, and Stack Exchange Inc. 39
      3. Amicus Brief of the Software Innovators, Start-ups, and Investors 41
      4. Amicus Brief of the Computer Scientists 43
      5. Amicus Brief of the Computer & Communications Industry Association 45

V. IMPLICATIONS OF A DECISION ON APPEAL 47
   A. Strict Legal Implications 47
   B. Implications for the Software Industry 49
In 2010 Oracle America, Inc., filed suit against Google, Inc., in the Northern District of California (Case No. C 10–03561) (Alsup, J.), and alleged that Google infringed Oracle’s copyright in its Java Application Programming Interface (“API”). The parties and the Court agreed that everyone was and remains free to program in the Java language itself, and that Google was free to use the Java language to write its own API. However, while Google took care to provide fresh line-by-line implementations in 97 percent of the Java API, it generally replicated the overall name organization and functionality of 37 packages in the Java API (three percent).

The main issue to be decided by the District Court was whether this violated the Copyright Act and more fundamentally whether the replicated elements were copyrightable in the first place. Because the vast majority of the code was not copied, Oracle alleged that Google infringed by copying the “structure, sequence and organization” of the Java API. Thus, the District Court was to determine the extent to which, if at all, certain replicated elements of the structure, sequence and organization of the Java API are protected by copyright.

The parties and the District Court agreed that no law was directly on point: no court of appeals has addressed the copyrightability of APIs, much less their structure, sequence and organization. Judge Alsup relied on general principles of copyright law announced by Congress, the Supreme Court and the Ninth Circuit.

On May 31, 2012, after a bench trial, Judge Alsup held that:

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

It is true that the very same functionality could have been offered in Android without duplicating the exact command structure used in Java. This could have been done by re-arranging the various methods under different groupings among the various classes and packages (even if the same names had been used). In this sense, there were many ways to group the methods yet still duplicate the same range of functionality.

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

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

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


On this basis, and after a lengthy discussion, Judge Alsup concluded, “on the specific facts of this case, the particular elements replicated by Google were free for all to use under the Copyright Act.” Judge Alsup therefore dismissed Oracle's copyright infringement claims based on Google's copying of the 37 API packages, including their structure, sequence and organization and granted Google's Rule 50 motions regarding copyrightability. *Id.*, at 1002.

Oracle appealed Judge Alsup’s Order to the Federal Circuit on October 3, 2012, as the original complaint contained both patent and copyright claims (Fed. Cir. Appeal No. 13-1021), and Google filed a cross-appeal on October 19, 2012 (Fed. Cir. Appeal No. 13-1022, consolidated with lead Appeal No. 13-1021). The appeals are now fully briefed by the parties, and a slew of amici have filed briefs for both sides. The consolidated appeals remain pending before the Federal Circuit, and oral argument has not yet been scheduled.
I. DEFINITION OF AN API

An Application Programming Interface ("API") is an abstraction specifying how one software program can request services from another software program. In simpler terms, an API is a way for two pieces of software to communicate.\(^1\) An API is composed of a set of rules a programmer must follow in order to make use of a developer’s program.\(^2\) The purpose of the API is to facilitate interaction between these two programs.

APIs are used by all kinds of software. For example, web services like Twitter use APIs to allow third-party applications to use and interact with their data.\(^3\) A movie studio might use Twitter’s API to create a program that tracks references to their movies in Twitter user statuses. Since the API provides a standard method for searching user statuses for a specified term, a television studio can use essentially the same code to track their new hit TV show. APIs are meant to be adaptable so that they can be used in a variety of ways by programmers. Without an API, Twitter would be effectively closed off from the applications of outside developers.

Programming languages like Java also use APIs. Like APIs for web services, these APIs provide a standard set of rules for interacting with the programming language. The API is the protocol the programmer must follow in order to take advantage of some of the core functionality of the programming language.\(^4\) Java is an object-oriented programming language.

---

\(^1\) Caleb Garling, Dear Oracle: The Java APIs are not a Work of Art, WIR ED (June 4, 2012, 6:30 AM), http://www.wired.com/wiredenterprise/2012/06/google-oracle-api-bookshelf/.

\(^2\) See David Orenstein, QuickStudy: Application Programming Interface (API), COMPUTERWORLD (Jan. 10, 2000), http://www.computerworld.com/s/article/43487/Application_Programming_Interface ("In essence, a program’s API defines the proper way for a developer to request services from that program"). In order to avoid confusion, this paper will refer to the software engineers who write the programs requesting or using the services as "programmers" and the engineers who write the servicing software (e.g. Java) as "developers."

\(^3\) Twitter’s API can be found at REST API v1.1 Resources, https://dev.twitter.com/docs/api/1.1 (last visited August 19, 2013).

\(^4\) See Opening Brief for Defendant at 8, Oracle v. Google, 798 F. Supp. 2d 1111 (N.D. Cal. 2011) (No. 3:10-CV-03561-WHA) (quoting Expert Report of John C. Mitchell at ¶ 60, Oracle v. Google, 798 F. Supp. 2d 1111 (N.D. Cal. 2011) (No. 3:10-CV-03561-WHA)) (arguing that use of the Java APIs are “accepted practice” and “expected” by Java programmers). Some, but not all, of the Java APIs are necessary in order to use the language. See March
Programming languages such as Fortran, C and Java allow programmers to write human-readable code, or source code, that is converted into machine-readable code, or object code. Object-oriented programming languages represent concepts through objects. An “object” is a way of conceptualizing data. Objects are an abstraction made up of fields (or data that describes the object) and methods (functions that the object can perform). In other words, fields are like the properties of objects and methods are their behaviors. For example, Java conceptualizes the word “copyright” as an object. The word “copyright” has fields, for example the number of characters. This field is an integer value assigned to a particular variable, say numChars. “Copyright” also has methods, like a function to output the number of characters. Methods also have names, in this case length(). These methods use the fields to perform a function. To illustrate, the “copyright” length() function outputs the number of characters by returning the field with the number of characters.

Each object belongs to a particular class. The class defines what fields and methods each of its member objects have. The “copyright” object belongs to the string class. As a result, the string class dictates what fields and methods the “copyright” object has. While the number and type of fields is the same for all objects of a class, the value associated with those fields will change. That is, the “copyright” object has different values assigned to its fields than a “patent” object. Both have a field for the number of characters, but the value associated with that field is

---

23, 2012 Brief for Plaintiff at 10, Oracle v. Google, 798 F. Supp. 2d 1111 (N.D. Cal. 2011) (No. 3:10-CV-03561-WHA) (“Only a very small number of elements in the Java APIs are required for the Java programming language.”)
6 See id. at ¶ 39 (describing object-oriented languages).
7 See id. at ¶ 40 (explaining objects).
8 In Java, unlike most object-oriented languages, some classes are called “interfaces”. For the purposes of this paper, interfaces are the same as classes. Moreover, interface in this sense refers to something different from the “interface” in the API acronym. See Oracle America, Inc., v. Google Inc., 872 F.Supp.2d 974, 980 (N.D. Cal. 2012) (Alsup, J.) (noting Java’s use of interfaces) (the “District Court’s Order on Summary Judgment” or “Judge Alsup’s Order”).
different. Similarly, both “copyright” and “patent” have the same `length()` method, but they return different results because of the different values associated with their fields.

In Java and other object-oriented programming languages, the API specifies how objects work. These specifications come in the form of packages that are composed of classes and associated methods.\(^{10}\) The API is the totality of those classes and methods. Each individual API is a unique package containing a number of classes and subpackages.\(^{11}\) Java has 166 individual APIs.\(^{12}\) Each of these packages has a particular name. For example, the string class is located in the “java.lang” package.\(^{13}\) Within each of the classes in a package, the API describes the corresponding methods and fields. For methods, this description includes how to call the method, by specifying the name and required input, and the expected output of that method.\(^{14}\)

Fields are described by name and data type.\(^ {15}\) Moreover, the API describes the relationships between the various packages, classes, methods and fields. For example, the `java.lang.String` API references the `java.lang.Character` API because a string object can be constructed from a sequence of character objects.\(^ {16}\)

The specifications in these packages are very important for programmers because they need objects for almost everything they do in Java. Therefore, in order to create a new object or use a method, programmers often rely on the APIs specifications to tell them how to write functioning code. From a programmer’s standpoint, the API operates as documentation for how

\(^{10}\) The API specifications for the string class can be found at Class String, [http://docs.oracle.com/javase/6/docs/api/java/lang/String.html](http://docs.oracle.com/javase/6/docs/api/java/lang/String.html) (last visited August 19, 2013).


\(^{12}\) Order of Judge Alsup, supra note 8, at 5. Oracle alleges that Google infringed the copyright of 37 of these APIs.

\(^{13}\) Class String, supra note 10.

\(^{14}\) See March 9, 2012 Brief for Plaintiff, supra note 11, at 2 (explaining methods).

\(^{15}\) See id. (explaining fields).

\(^{16}\) See supra note 10.
to use a major part of the Java language. Using the APIs requires precision, and calling a method in a way not specified either causes an error or executes an entirely different method.\(^\text{17}\)

It is important to distinguish APIs from software libraries. If an API is a set of specifications, its corresponding software library is the actual implementation of those specifications. These libraries are the set of instructions the computer actually follows when an API class or method is called. In other words, they “implement” the abstract functions described in the library. In that sense, it is actually these libraries that do the heavy lifting. The APIs simply provide an abstraction on top of them. The APIs act as the specification hooks that programmers insert into their program to actually access and use these software libraries.\(^\text{18}\) The API in isolation then, is just the list of classes and methods from the library exposed to the programmer. The API simply declares the functions in a library so that they can be controlled and accessed by developers.\(^\text{19}\) For this reason, the API is sometimes referred to as the method declaration.\(^\text{20}\) Note, however, that sometimes the Java libraries are referred to as the API implementations, while the APIs are referred to as the API specifications or declarations.\(^\text{21}\) Under this conception, the library is part of the API. Functionally, this is just another name for the same distinction.

Another example of a Java class is a Queue. A Queue is exactly what it sounds like— a virtualized checkout line. Things are taken off the front of the Queue and things are added to the

\(^{17}\) See supra note 8, at 35 (“To carry out any given function, the method specification as set forth in the declaration must be identical under the Java rules. . . .”).

\(^{18}\) See Oliver Herzfeld, Oracle v. Google: Are APIs Covered by Copyright Law?, FORBES (May 1, 2012, 2:00 PM), http://www.forbes.com/sites/oliverherzfeld/2012/05/01/oracle-v-google-are-apis-covered-by-copyright-law/ (describing APIs as specification “hooks”).

\(^{19}\) See Opening Brief for Defendant, supra note 4, at 7 (arguing that APIs provide “rules” for implementing specifications).


back of the Queue, much like customers at the front of the line are checked out while new customers go to the back of the line. In the Java API, the Queue class has three methods: Queue.add(), Queue.remove() and Queue.element(). Queue.add() adds some data to the back of the queue, Queue.remove() removes the first item from the queue and Queue.element() outputs the first item without removing it. The API specifies each of these methods, exactly how they are called and what their output is. For example, Queue.add() can only be called in that way. Writing code with Queue.push() will return an error. However, the actual implementation of Queue.add() and of the Queue class itself is located in the corresponding software library. In other words, the API is limited to announcing the class names and methods.

For a Queue, the corresponding software library would describe to the computer exactly how to execute Queue.add(), Queue.remove() and Queue.element() as well as create a Queue object in the first place. For Queue.add(), the implementation might instruct the computer to read the element designated by the user, check the Queue object to see where to place the element and then copy the element to that location. The API does not include any of these steps and simply announces the existence of the Queue.add() method and offers a short explanation of how to use it, including what type of data can be added to a Queue.

Metaphors to describe APIs are plentiful, but they all tend to be imperfect. Oracle compares the Java APIs to an outline of a multi-volume history book. Each class and method

---

22 The API specifications for java.util.Queue() can be found at Interface Queue, http://docs.oracle.com/javase/6/docs/api/java/util/Queue.html (last visited August 19, 2013).
23 The function to create an object from a class is commonly called a “constructor” method. See Providing Constructors for Your Classes, JAVA SE TUTORIAL, available at http://docs.oracle.com/javase/tutorial/java/javaOO/constructors.html (defining constructors).
24 This explanation may or may not be an oversimplification of an actual Queue.add() implementation depending on the sophistication of the library and individual developer preferences. The point is that the implementation designates the particular set of steps the computer follows to execute the method.
25 See March 9, 2012 Brief for Plaintiff, supra note 11, at 2 (“[T]he Java API specifications are like the author’s meticulously detailed outline. . . .”). See also Opening Brief for Plaintiff-Appellant at 1, Oracle v. Google, No.
is analogous to the volume, chapter, facts and primary sources of an expansive non-fiction work. Oracle emphasizes the creativity required to create an API, such as that required to author a sprawling volume.\textsuperscript{26} They also emphasize the complexity of APIs.\textsuperscript{27} Google could make a number of arguments to discredit Oracle’s analogy. First, this comparison ignores the main functionality of an API, namely to specify how programmers access the methods implemented in the software libraries, not to outline the contents of these libraries. Secondly, it ignores the purposes of the API. The point of the API is that programmers make use of its functions in their own programs, unlike a history outline, where the point might be to offer a concise summary but not to be used by other authors. On a more fundamental level, a programming language is not like a historical volume. Programming languages are for building things, not just for reading. Unlike a history volume, programming languages do not exist for themselves; they exist to make other programs possible.

Meanwhile, Google likens the APIs to a dictionary, an alphabetical list of classes and methods.\textsuperscript{28} They describe the APIs as purely functional and not creative, much like the alphabetizing of words in a dictionary.\textsuperscript{29} For Google, the creative, expressive content is isolated to the software libraries, where the actual implementations are located. While the APIs are like a list of words in a dictionary, the libraries are like the definitions of those words.\textsuperscript{30} Oracle likewise can expose flaws in Google’s analogy. Like Oracle, Google ignores the functionality and purpose behind APIs. In addition, Google does not give API developers enough credit.

\textsuperscript{26} See March 9, 2012 Brief for Plaintiff, supra note 11, at 9 (describing API specifications as “highly creative”). See also Garling, supra note 1.

\textsuperscript{27} See Opening Brief for Plaintiff-Appellant, supra note 25, at 9 (calling an illustrative package “exceeding complex”).

\textsuperscript{28} See Reply Brief for Defendant, supra note 21, at 1 (analogizing API specifications to a dictionary).

\textsuperscript{29} Id.

\textsuperscript{30} See id. at 2 (“The implementing code, in turn, provides the ‘definitions’ of these words. . . .”)
Unlike dictionaries, the structure of an API is not completely dictated by an unbendable framework. While the amount of creativity involved in writing an API is debatable, conscious development decisions, like what methods to expose and what to call them, go into the creation of the API. However, this metaphor does a capture the idea that APIs are developed to form something new, much like people use words in a dictionary to form sentences and paragraphs.

In his order, Judge Alsup uses an analogy closer to Google’s than Oracle’s. He conceptualizes APIs as a library. Each package is a bookshelf, each class is a book and each method is a chapter in that book. Even ignoring the fact that this metaphor confusingly invokes an API-library distinction, this comparison is perhaps even less accurate than either Oracle’s or Google’s at describing the purpose of an API. However, it is helpful in describing how each of these concepts is organized. Each class is located within an API package and each method is located within a class. Beyond that, however, the metaphor seems to fall apart.

A popular metaphor for describing APIs likens them to the doors of a house. Under this scenario, the libraries are the inside of the house and the programming language itself comprise the bricks, wood and nails that make up the house. Without the doors, the house is closed off, it exists and can work internally, but no one can get in to use it. A programming language without an API is similarly closed off. While the methods and objects exist within the libraries as “private methods”, programmers cannot access them since these methods and constructors are not public. With APIs (as the doors), people can enter the house and use the objects inside. Moreover, the doors describe exactly how to enter the house; it cannot be entered in another way.

31 Compare March 9, 2012 Brief for Plaintiff, supra note 11, at 9 (describing the Java APIs as “highly creative”) with Reply Brief for Defendant, supra note 21, at 2 (describing the Java APIs as purely functional).
32 Order of Judge Alsup, supra note 8, at 5.
33 See, e.g., Orenstein, supra note 2 (quoting Josh Walker, Analyst, Forrester Research Inc.).
34 “Public” methods can be called by outside programs while “private” methods can only be called by other methods inside the same class. See Order of Judge Alsup, supra note 8, at 11.
Similarly, the API specifies how to use the libraries and the objects inside; they cannot be used in another way. Of course, while this metaphor does a better job of describing the role of APIs, it is not very helpful for having a discussion about the copyrightability of an API, which might explain why none of the parties used it in their briefs.

Programming language APIs are important and useful for a number of reasons. The first reason is interoperability and compatibility. In particular, APIs make software applications portable; the same Java code that works on an Android smartphone works on a Window’s desktop or a Linux server because they all have the same APIs. In fact, the original slogan for Java was “write once, run anywhere.”\(^{35}\) The same `Queue.remove()` method can, and in fact has to be called the same way on all platforms using the same APIs. Secondly, APIs allow for flexibility. Specifically, the library or implementation of a method can be changed without breaking any programs using that method. For example, say that Google finds a new, faster way to implement `Queue.remove()`\(^{36}\). If they rewrite the implementation, software using `Queue.remove()` will operate in the same way,\(^ {37}\) even if they were written before the new implementation. Third, a good API promotes stability. As developers become familiar with a particular API, they become reliant on it. Forcing them to learn the proper syntax for a new API with each program would be incredibly onerous.\(^{38}\) As a result, use of an API sometimes

\(^{35}\) Expert Report of John C. Mitchell, supra note 5, at ¶ 42. See also Opening Brief for Plaintiff-Appellant, supra note 25, at 8 (calling “write once, run anywhere”, “Java’s credo”).

\(^{36}\) In fact, Google did rewrite Oracle’s Java implementations, but for very different reasons. While the copyrightability of the APIs is at issue, Google concedes that the implementation of those APIs are copyrightable. Reply Brief for Defendant, supra note 21, at 2. Therefore, Google rewrote the libraries in order to avoid copyright infringement.

\(^{37}\) Technically, the program will not work in exactly the same way because the computer will read its instructions from the new, faster implementation rather than the old, slower one. However, this transition will be seamless, the way the program is called and its output will be unchanged.

\(^{38}\) With “[e]ven the slightest changes to the names or organization of API elements….existing code that used those elements would not run properly, and programmers would have to learn new API element names.” March 9, 2012 Brief for Plaintiff, supra note 11, at 6 (quoting Opening Expert Report of Owen Astrachan at ¶ 131, Oracle v. Google, 798 F. Supp. 2d 1111 (N.D. Cal. 2011) (No. 3:10-CV-03561-WHA)).
becomes an expected part of using the programming language itself.\(^\text{39}\) Lastly, in some cases, APIs can be used to protect a developer’s intellectual property. Other programmers can tap into the software’s functionality without the original programmer having to reveal all of their source code.\(^\text{40}\) Thus, the original software can obtain some of the benefits of open source software without exposing his code and risking copyright or patent infringement.

Defining “API” is difficult for a number of reasons. First, the functionality of APIs vary depending on their context. Twitter’s API is very different from Java’s, even though they share many of the same goals.\(^\text{41}\) Secondly, even within a particular context, the word API can refer to a number of distinct concepts, requiring a further distinction between API specifications and API implementations.\(^\text{42}\) Lastly, even with the right concept in mind, that concept is abstract, and depends upon other abstractions such as objects, classes and methods. Regardless, in the context of this debate, APIs can be best understood as a set of rules for making use of some library of code. They are the specifications for using standard Java libraries for everything from printing words on a screen to creating an encrypted Internet connection. API specifications are the bridge between Java programmers and the thousands of objects and methods at their fingertips.

II. SUMMARY OF THE DISTRICT COURT DECISION

The key aspect of the court’s decision was its holding that portions of the 37 packages of the Java API, which Oracle alleged that Google had copied, as well as the structure, sequence and organization (“SSO”) of the API, were not subject to copyright protection.\(^\text{43}\) Prior to this determination, the District Court chose to divide the case into phases given the complexity of the

\(^{39}\) See supra note 4 and accompanying text.  
\(^{40}\) See Orenstein, supra note 2 (explaining the purpose of APIs).  
\(^{41}\) See supra note 3 and accompanying text (describing Twitter’s API).  
\(^{42}\) See supra note 21 and accompanying text (addressing the specification versus implementation issue). Admittedly, because of this difficulty, an API might aptly be called a “verbal chameleon”. Opening Brief for Plaintiff-Appellant, supra note 25, at 9.  
\(^{43}\) Id. at 5
issues and the case itself. Phase one pertained to the copyright infringement claims, Google’s “fair use” defense, and whether or not copying was de minimis; phase two dealt with the patent claims; phase three would have dealt with damages but was obviated by stipulations and verdicts.\textsuperscript{44} The District Court had instructed the jury to assume that the 37 packages were in fact copyright protected, namely because this would avoid the need for a costly retrial if the District Court were overturned on appeal.\textsuperscript{45} As a result, the jury unanimously found that Google had infringed the SSO of the Java API. Subsequently, Judge Alsup ruled that APIs were non-copyrightable.

Though copyrightability was at the core of the District Court’s order, the discussion of those issues is preceded by a remarkably lengthy explanation of the technical aspects of the Java API, indicating that the District Court went to great lengths to understand the intricacies of the software at issue. Included in this factual synopsis, the District Court covered the history and development of the Java and Android platforms, as well as the more technical aspects of the Java language, and finally the Java API itself, noting that “[c]ontrary to Oracle, there is no bright line between the language and the API.”\textsuperscript{46} The District Court also took great care to discuss factual circumstances that have led to a major point of contention between the parties on appeal, namely, that while the SSO of the Java API does not completely “merge” with use of the Java language itself, the SSO is a prerequisite for interoperability:

Although the declarations must be the same to achieve the same functionality, the names of the methods and the way in which the methods are grouped do not have to be the same. Put differently, many different API organizations could supply the same overall range of functionality. They would not, however, be interoperable.\textsuperscript{47}

\textsuperscript{44} See supra note 8 at 975-76.
\textsuperscript{45} Id.
\textsuperscript{46} Oracle America, Inc., 872 F.Supp.2d at 982.
\textsuperscript{47} Id.
That said, the copyrightability issues addressed by the District Court related not only to the SSO but to the copying of the actual declarations, i.e. the names and headers within the Java API. As to that issue, the District Court found that the declarations within the Java API were not protected because names, titles, or short phrases are not entitled to copyright protection based on a U.S. Copyright Office regulation, which states that “[e]ven if a name, title, or short phrase is novel or distinctive or lends itself to a play on words, it cannot be protected by copyright.” 48 Thus, the District Court dealt with this aspect of the copyrightability question summarily, finding that “[t]his has relevance to Oracle’s claim of copyright ownership over names of methods, classes, and packages.” 49

The District Court acknowledged that the second issue posed “the more difficult question,” namely whether the SSO of the Java API was copyrightable. 50 The District Court approached the question by turning to the merger doctrine as first espoused in Baker v. Seldon, 101 U.S. 99 (1879). In Baker, a copyright owner alleged copyright infringement of a book containing a new system of double-entry bookkeeping consisting of black forms, ruled lines, and headings. The accused infringer copied the bookkeeping method, i.e. the idea behind the system, but used different forms. In Baker, the Supreme Court established the merger doctrine, under which “methods and diagrams are to be considered as necessary incidents to the art, and given therewith to the public” where “the art [the book] teaches cannot be used without employing the methods and diagrams used to illustrate the book.” 51 In other words, to the extent that the methods and diagrams had “merged” with the idea behind the accounting system, those methods

48 Id at 983-84 (citing U.S. Copyright Office, Circular 34; 37 C.F.R. 202.1(a)).
49 Id. at 984.
50 Id.
51 Id at 985 (citing Baker, 101 U.S. at 103).
and diagrams were not protected by copyright because the effect would be to withhold the idea itself from the public.

The District Court noted both the codification of the Baker rule, i.e. the merger doctrine, in section 102(b) of the Copyright Act, and the fact that “[s]ection 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.”\textsuperscript{52} Section 102(b) states that in “no case does copyright protection for an original work of authorship extend to any idea, procedure, process, system, method of operation, concept, principle, or discovery, regardless of the form in which it described, explained, illustrated, or embodied in such work.”\textsuperscript{53} The provision seemingly eviscerates copyright protection for computer programs, and so the District Court mentions the intention behind the codification of the merger doctrine therein, noting the inclusion of “computer program” within the list of items protected by copyright.\textsuperscript{54} The Court also mentions CONTU report’s determination that Section 102(b)’s “preclusion of copyright protection for ‘procedure, process, system, method of operation’ was reconcilable with the new definition of ‘computer program.’”\textsuperscript{55}

The District Court reviewed a number of cases from within the Ninth Circuit and around the country, in order to arrive at its conclusion that the SSO of the Java API was not protected by copyright. In doing so, it noted the high watermark of copyright protection for software SSO in Whelan, from which the SSO concept originates. Whelan found that the SSO of Dentalab—a program that managed dental lab procedures—was protected by copyright because there were

\textsuperscript{53} 17 U.S.C. § 102(b).
\textsuperscript{54} 17 U.S.C. § 101.
\textsuperscript{55} Oracle America, Inc., 872 F.Supp.2d at 986.
many ways to structure such a program.\(^{56}\) The District Court goes on to illustrate a trend away from this approach to software SSO protection, beginning with the Second Circuit’s critique of Whelan in Altai,\(^ {57}\) and including the Sega\(^ {58}\) and Sony\(^ {59}\) decisions in which “intermediate copying” for the purpose of creating something new was deemed non-infringing. The District Court also discussed the Atari\(^ {60}\) decision, where the Federal Circuit applied Ninth Circuit law and found the 10NES program was copyrightable: “Nintendo’s 10NES program contains more than an idea or expression necessarily incident to an idea. Nintendo incorporated within the 10NES program creative organization and sequencing unnecessary to the lock and key function” and created “a purely arbitrary data stream [that] serves as the key to unlock the NES.”\(^ {61}\) The District Court distinguishes the case, citing Sega, where as in Atari the Defendant had copied a security protocol, but had copied only what it needed to achieve interoperability.\(^ {62}\) Consequently, the Ninth Circuit “held that the copying of object code for the purpose of achieving compatibility was fair use.”\(^ {63}\) Thus, the District Court attempts to reveal “a trajectory in which enthusiasm for protection of ‘structure, sequence and organization’ peaked in the 1980s, most notably in the Third Circuit’s Whelan decision,”\(^ {64}\) but which in the Ninth Circuit ultimately gave way to interoperability considerations.

In applying this law to the facts of the case, the District Court returns to its section 102(b) filtration. It notes Oracle’s “best argument” – that the SSO of the Java API is a “taxonomy”

\(^{56}\) Id at 978-88 (citing Whelan Associates, Inc. v. Jaslow Dental Laboratory, Inc., 797 F.2d 1222 (3d Cir.1986)).
\(^{58}\) Sega Enterprises Ltd. v. Accolade, Inc., 977 F.2d 1510 (9th Cir.1992).
\(^{59}\) Sony Computer Entertainment, Inc. v. Connectix Corporation, 203 F.3d 596 (9th Cir.2000).
\(^{60}\) Atari v. Nintendo, 975 F.2d 832 (Fed.Cir.1992).
\(^{61}\) Oracle America, Inc., 872 F.Supp.2d at 993-94 (quoting Atari, 975 F.2d at 840).
\(^{62}\) Id at 995.
\(^{63}\) Id at 994.
\(^{64}\) Oracle America, Inc., 872 F.Supp.2d at 996.
protectable in a manner similar to a phonebook in *Feist*\(^{65}\) or a manual of dental procedures in *American Dental Assoc.*\(^{66}\) But then the Court summarily rejects this conclusion based on section 102(b):

Yes, it is creative. Yes, it is original. Yes, it resembles a taxonomy. But it is nevertheless a command structure, a system or method of operation—a long hierarchy of over six thousand commands to carry out pre-assigned functions. For that reason, it cannot receive copyright protection—patent protection perhaps—but not copyright protection.\(^{67}\)

The Court also falls back on interoperability, noting not only that copying of the SSO of the Java API was necessary to achieve interoperability, but also that “interoperability is at the heart of the [Java API] command structure” and that “fragmentation, imperfect interoperability, and Oracle’s angst over it illustrate the character of the command structure as a functional system or method of operation.”\(^{68}\) The Court again relies on *Sega* and *Sony*, noting that interoperability was imperfect in those cases as well, but that this factual circumstance did not preclude the operation of section 102(b) in either case.\(^{69}\)

In conclusion, the Court notes that it decision does not purport to put an end to the protectability of SSO of computer software as a matter of law. Rather, it notes that “the structure, sequence and organization of a computer program may (or may not) qualify as a protectable element depending on the ‘particular facts of each case’ and always subject to exclusion of unprotectable elements.”\(^{70}\)

---


\(^{66}\) American Dental Association v. Delta Dental Plans Association, 126 F.3d 977 (7th Cir.1997).

\(^{67}\) Oracle America, Inc., 872 F.Supp.2d at 999-1000.

\(^{68}\) Id at 1000.

\(^{69}\) Id at 1000-01.

\(^{70}\) Id at 1001 (quoting Johnson Controls v. Phoenix Control Sys., 886 F.2d 1173, 1175 (9th Cir.1989)).
III. ISSUES ON APPEAL

The main issue on appeal concerns the copyrightability of the declaring source code of thirty seven (37) Java application programming interfaces (“APIs”) or packages, as well as the overall structure, sequence, and organization of those packages.\textsuperscript{71} Based on its interpretation of the merger doctrine as codified in 17 U.S.C § 102(b), the District Court found that Google had not appropriated any protected expression. On appeal, the Federal Circuit will evaluate the line the District Court drew between the expressive and functional aspects of the Java API, and in doing so determine its copyrightability. The difficulty of that task is well illustrated by the immortal words of Judge Learned Hand: “[n]obody has ever been able to fix that boundary and nobody ever can.”\textsuperscript{72} Should Oracle succeed in urging reversal of the district court’s copyrightability determination, the jury’s verdict finding infringement may be reinstated, and the second issue on appeal—whether Google can raise a fair use defense to that infringement—would become directly relevant.\textsuperscript{73} Two final issues are raised in Google’s cross-appeal: whether Google’s copying of rangeCheck was \textit{de minimis}, and whether Oracle’s failure to object to jury instructions precluding consideration of literal infringement of the non-implementing API source code constitutes waiver.\textsuperscript{74}

\textsuperscript{72} Nichols v. Universal Pictures Corporation, 45 F.2d 119, 121 (2d Cir. 1930).
\textsuperscript{73} Oracle America, Inc., 872 F.Supp.2d 974, 976 (N.D. Cal. 2012).
\textsuperscript{74} Brief of Appellee and Cross-Appellant at 74-79. The jury found that Google’s use of the eight (8) decompiled test files was \textit{de minimis}. However, the district court overturned the verdict and granted Oracle’s JMOL on the issue. Id. at 76.
A. Issues on Direct Appeal

As to the copyrightability of the SSO of the Java API, the Federal Circuit will apply the merger doctrine, first espoused in Baker v. Selden,75 and now codified in § 102(b) of the 1976 Copyright Act ("Copyright Act"). It must do so in light of the fact that the Copyright Act protects "works . . . expressed in words, numbers, or other verbal or numerical symbols or indicia" i.e. computer programs.76 "Computer programs" are themselves defined in the Act as "a set of statements or instructions to be used directly or indirectly in a computer in order to bring about a certain result."77 Notably, the Court has expressly recognized that "copyright protection extends to computer programs."78 Nevertheless, the Court must also consider section 102(b) of the Copyright Act, which states that "[i]n no case does copyright protection for an original work of authorship extend to any idea, procedure, process, system, method of operation, concept, principle, or discovery, regardless of the form in which it is described, explained, illustrated, or embodied in such work."79 Finally, it must consider the goal of the merger doctrine and § 102(b), which is to prevent "a monopoly over a given expression [that] would in fact stymie others from expressing the idea embodied therein, such that the idea and its expression 'merge' into one."80

On appeal, Oracle argues that the District Court misapplied § 102(b) by "dissecting [the work] down to the most minute level of abstraction."81 According to Oracle, the result is that "almost nothing [is] copyrightable because original works broken down into their composite parts would usually be little more than basic unprotected elements like letters, colors and

---

75 101 U.S. 99 (1879).
77 Id.
80 Nimmer on Copyright, § 2.18(c)(2).
81 Opening Brief and Addendum of Plaintiff-Appellant at 50.
82 Oracle argues that such a broad application of the merger doctrine and § 102(b) effectively strips computer programs of any and all copyrightability, an effect Congress never could have intended when it included computer programs within copyrightable subject matter.\textsuperscript{83} Oracle further notes that “courts routinely read Baker’s idea/expression principles to find programs protectable simply because their authors seek protection of their own expression—their own way of implementing the function . . . and not the function itself.”\textsuperscript{84} Oracle cites Mitel, Inc. v. Iqtel, Inc., 124 F.3d 1366 (10th Cir. 1997), for the proposition that protection for functional and expressive subject matter are not mutually exclusive: “[s]ection 102(b) does not extinguish the protection accorded in a particular expression of an idea merely because that expression is embodied in a method of operation at a higher level of abstraction.”\textsuperscript{85} In other words, the Mitel court found that “an element . . . may be characterized as a method of operation, . . . [and] nevertheless contain expression that is eligible for copyright protection.”\textsuperscript{86} Oracle also attempts to marginalize the First Circuit’s 1995 decision in Lotus Development Corporation v. Borland International Inc.\textsuperscript{87} (affirmed by an evenly divided Supreme Court), upon which Google relies, noting that “[o]nly one case has ever so much as suggested ‘that expression that is part of a ‘method of operation’ cannot be copyrighted.’”\textsuperscript{88}

Oracle criticizes the District Court’s reliance on the U.S. Copyright Office Regulation concerning “short phrases and names”\textsuperscript{89} in order to find the declarations of the Java API unprotectable, arguing that the Court “over-dissects” the Java API. Specifically, it asserts that the District Court erred when it “dissected the work too minutely—fixating on the individual line of
code rather than the larger arrangement of which it was a part.”90 In urging reversal, Oracle contends that the regulation was meant to “prevent an author from copyrighting a ‘work’ consisting of a naked bit of text,” and that this application “would invalidate practically any computer program” given that “[v]irtually every line of the typical program is a short phrase.”91 Oracle also tackles the District Court’s emphasis on interoperability in its copyrightability holding, arguing that the Court erred in failing to consider copyrightability ex ante, i.e. from the perspective of the original developers of Oracle’s packages, considering what constrained them.”92

Google responds by drawing a distinction between the analyses under section 102(a) and 102(b). Under a section 102(a) analysis, it concedes that “it makes sense to look at the options available to the author when she created the work (ex ante).”93 However, according to Google, the section 102(b) analysis addresses “whether some aspect of the work has become an industry standard or is necessary for compatibility. In that context it only makes sense to look at the options available to the alleged infringer (ex post).”94 Thus, Google defends the District Court decision, asserting that Oracle is not exempt from “normal,” i.e. ex post section 102(b) filtration, including compatibility requirements, design standards, and generally accepted industry practices.95 While acknowledging that “some computer programs ‘may be highly creative and idiosyncratic,’” Google asserts that computer programs are “‘in essence, utilitarian articles—articles that accomplish tasks . . . [and] contain many logical, structural, and visual display

---

90 Opening Brief and Addendum of Plaintiff-Appellant at 55.
91 Id at 55.
92 Id at 34.
93 Brief of Appellee and Cross-Appellant at 48.
94 Id.
95 Brief of Appellee and Cross-Appellant at 41. This aligns with the scholarly understanding of the merger doctrine in the context of infringement. The “merger doctrine should be conceptualized solely as a defense applicable to a given defendant acting in a concrete factual setting.” Nimmer on Copyright, § 2.18(c)(2).
elements that are dictated by the function to be performed.’’ Google thus concludes that the sheer “presence of such unprotected functional elements results in ‘a lower degree of protection’ for computer programs and platforms than for ‘more traditional literary works.’”

**Fair Use**

Section 107 of the Copyright Act sets out a non-exhaustive list of factors to guide a court’s fair use analysis, namely (1) the purpose and character of the use, (2) the nature of the copyrighted work, (3) the amount and substantiality of the portion used in relation to the copyrighted work as a whole, and (4) the effect of the use on the potential market for or value of the copyrighted work. Other factors are likely to be relevant, including (1) whether Google’s use was consistent with generally accepted practices in the computer industry, and (2) whether Google knowingly infringed in bad faith as a result of Google’s interoperability arguments and Oracle’s commercial expediency arguments, respectively.

The first factor – the purpose and character of the use – is generally divided into two questions by courts, namely whether the use is commercial and whether the use is transformative. As Oracle points out, “Google does not dispute that its use is commercial—to the tune of ‘billions of dollars.’” As to whether the use is transformative, Google cites its unique implementations and independently created virtual machine, as well as the fact that Android “accommodates existing works while making new creative works possible,” and

---

96 Id (quoting Sega Enterprises Ltd. v. Accolade, Inc., 977 F.2d 1510, 1524 (9th Cir. 1992)).
97 Id (quoting Sony Computer Entertainment, Inc. v. Connectix Corporation., 203 F.3d 596, 603 (9th Cir. 2000)).
100 Rogers v. Koons, 960 F.2d 301, 309 (2d Cir. 1992).
101 Brief of Appellee and Cross-Appellant at 68-74.
102 Opening Brief and Addendum of Plaintiff-Appellant at 68-77; Response and Reply Brief of Plaintiff-Appellant at 44-59.
103 Patry on Fair Use § 3:1.
104 Response and Reply Brief of Plaintiff-Appellant at 45.
“fosters the ‘dissemination of other creative works’ through interoperability.”\textsuperscript{105} Oracle argues that “[t]o be transformative, the new use must change the very nature of the original, giving it ‘an entirely different function.’”\textsuperscript{106} Oracle adds that “Oracle’s packages allow programmers to use prewritten functions rather than write them from scratch. The same code in Android (e.g., new URL().openConnection()) enables programmers to invoke the same pre-programmed functions in exactly the same way. Use of declaring code and packages in Android does not serve ‘an entirely different function’ from Java.”\textsuperscript{107} Oracle argues that Google used the declaring code “to avoid the drudgery in working up something fresh,” and so its “claim to fairness in borrowing from [Oracle’s] work diminishes accordingly (if it does not vanish).”\textsuperscript{108} Oracle also attacks Google’s interoperability argument by distinguishing Sega Enterprises Ltd. v. Accolade, Inc. and Sony Computer Entertainment, Inc. v. Connectix Corporation as cases of “intermediate copying” for purposes of reverse engineering.\textsuperscript{109} Unlike those cases, Oracle asserts, Google incorporated what it copied from Oracle’s thirty seven (37) packages into its Android platform.

As to the second factor – the nature of the copyrighted work – courts consider whether the copyrighted work is creative or factual.\textsuperscript{110} Google rehashes its argument that “computer programs are ‘essentially utilitarian’ in nature, and, under the Copyright Act, if a work is largely functional, it receives only weak protection.”\textsuperscript{111} Oracle interprets Sega to illustrate “[t]o the extent that there are many possible ways of accomplishing a given task or fulfilling a particular

\begin{footnotesize}
\begin{itemize}
\item[105] Brief of Appellee and Cross-Appellant at 69 (quoting Sega Enterprises Ltd., 977 F.2d at 1523).
\item[106] Response and Reply Brief of Plaintiff-Appellant at 46 (quoting Kelly v. Arriba Soft Corp., 336 F.3d 811, 818 (9th Cir. 2003)).
\item[107] Id at 47 (quoting Kelly, 336 F.3d at 818).
\item[108] Id at 46 (quoting Campbell v. Acuff-Rose Music, Inc., 510 U.S. 569, 580 (1994)).
\item[109] Id at 49.
\item[110] Patry on Fair Use § 4:1.
\item[111] Brief of Appellee and Cross-Appellant at 70 (quoting Sega, 977 F.2d at 1527).
\end{itemize}
\end{footnotesize}
market demand, the [developer’s] choice of program structure and design may be highly creative and idiosyncratic.”\textsuperscript{112}

As to the third factor – the amount and substantiality of the use – courts perform a quantitative and qualitative analysis.\textsuperscript{113} A qualitative analysis will consider whether the protectable elements used constituted the “heart” or most important part of the work.\textsuperscript{114} Google argues that it copied only what it needed to use Java, noting that “[i]f the secondary user only copies as much as is necessary for his or her intended use, then this factor will not weigh against him or her.”\textsuperscript{115} In doing so, Google reiterates the District Court’s finding that “Google replicated what was necessary to achieve a degree of interoperability – but no more, taking care, as said before, to provide its own implementations.”\textsuperscript{116} Oracle counters this, noting that “Google does not dispute that it copied the declaring code of the packages it thought programmers would want in a smartphone platform . . . [n]or does Google dispute that declaring code is all-important, because that is the only code programmers ever see or use.”\textsuperscript{117} In effect, Oracle asserts that Google did not need to copy these aspects order to write programs in Java, and so it in fact replicated more than was necessary.\textsuperscript{118}

As to the fourth factor – the effect of the use on the potential market for, or value of the copyrighted work – courts consider the direct harm from the use at issue and harm that may result from potentially similar future uses.\textsuperscript{119} Google argues that Android does not merely supplant or supersede Java,\textsuperscript{120} and points out that “Oracle never successfully developed its own

\textsuperscript{112} Response and Reply Brief of Plaintiff-Appellant at 53 (quoting Sega, 977 F.2d at 1524).
\textsuperscript{113} Patry on Fair Use § 5:1.
\textsuperscript{115} Brief of Appellee and Cross-Appellant at 71 (quoting Kelly, 336 F.3d at 820-21 (9th Cir. 2003)).
\textsuperscript{116} Id at 72 (quoting Oracle America, Inc., 872 F.Supp.2d at 1000).
\textsuperscript{117} Response and Reply Brief of Plaintiff-Appellant at 54-55.
\textsuperscript{118} Id at 55.
\textsuperscript{119} Patry on Fair Use § 6:1.
\textsuperscript{120} Brief of Appellee and Cross-Appellant at 72.
smartphone platform using the Java technology.”121 Google also argues that its use did not interfere with Oracle’s licensing opportunities,122 analogizing its position to that of Accolade’s in Sega as “a new competitor, the first lawful one that is not a[n] [Oracle] licensee,” and noting the Sega court’s admonition that a decision to the contrary would “mak[e] it impossible for others to compete.”123 Oracle attempts to counter Google’s arguments by illustrating market harm through Amazon’s decision to adopt the free Android platform, even though it had previously licensed Java ME for earlier version of its electronic reader. Oracle further argues that Java at one point dominated the mobile market before smartphones appeared on the scene, and disputes Google’s relatively narrow view of the “market for derivative works,” positing the relevant market as that of “mobile device platforms” rather than “a specific generation of phones.”124 Oracle falls back on the argument that Android at the very least harmed the potential market for a Java smartphone, and distinguishes Sega on the basis that the copying in that case bolstered Sega’s market prospects by contributing to the total number of games available for the console.125

B. Issues on Cross-Appeal

Google’s cross-appeal includes two rulings at the District Court level: a ruling denying Google’s motion to preclude, as a matter of law (JMOL), infringement for copying rangeCheck, and a ruling granting Oracle’s JMOL motion, which overruled a jury verdict that Google did not infringe when it copied eight (8) decompiled files (“security files”).126 The cross-appeal on these motions concerns the correct application of the de minimis doctrine to the aforementioned files and code and the District Court’s “work as a whole” jury instruction.

121 Id (quoting Oracle America, Inc., 872 F.Supp.2d at 978).
122 Id at 73.
123 Brief of Appellee and Cross-Appellant at 72 (quoting Sega Enterprises Ltd., 977 F.2d at 1523-24).
124 Response and Reply Brief of Plaintiff-Appellant at 56-57.
125 Id at 58-59.
126 Brief of Appellee and Cross-Appellant at 74-79.
The District Court instructed the jury that, with respect to the rangeCheck function, the “work as a whole” is the compilable code for the individual rangeCheck file, not the code for the entire API, i.e. the registered work. Google takes issue with this instruction because, viewing the “work as a whole” as only the code contained within the file in which rangeCheck was found, the jury concluded that the copying of rangeCheck was not de minimus. Google focuses upon a complete absence of evidence below as to the file’s ability to “stand alone,” which is the standard for determining the “work as a whole.” Google rejects Oracle’s argument that a file need only be separate in order to stand alone, noting not only an absence of evidence in the record as to the independent storage of the file in question, but reasoning that code can always be placed in separate files, such that viewing the standard this way would “effectively nullify the de minimis doctrine in software-copying cases.” Oracle, alternatively, notes that the files were admitted into evidence separately, each with its own copyright notice and author information, and as such should be treated separately.

Oracle also contends that there is no de minimis defense to copyright infringement in the Ninth Circuit whatsoever, citing Norse v. Henry Holt & Company to show that “the case could not be dismissed on de minimis grounds: ‘The question of whether a copying is substantial enough to be actionable may be best resolved through the fair use doctrine, which permits courts to avoid rigid application of the copyright statute when, on occasion, it would stifle the very creativity which the law is designed to foster.’” Oracle also contends that even if there is a de minimis defense to copyright infringement, Google’s copying was nevertheless qualitatively...

---

127 Reply Brief of Cross-Appellant Google Inc. at 14.
128 Id. at 14-15 (citing Hustler Magazine Inc. v Moral Majority Inc., 796 F.2d 1148, 1154-55 (9th Cir. 1986)).
129 Id. at 16.
130 Response and Reply Brief of Plaintiff-Appellant at 70.
131 991 F.2d 563 (9th Cir. 1993).
132 Response and Reply Brief of Plaintiff-Appellant at 64.
significant, pointing to testimony to the effect that the rangeCheck and security files were important insofar as they were called upon often and governed access “to a network or other resource” in the Java platform.\textsuperscript{133}

Google takes issue with what it terms Oracle’s “no freestanding \textit{de minimis}” theory, arguing primarily that Oracle has waived this argument by failing to raise it at the District Court level.\textsuperscript{134} Google not only relies upon \textit{Newton v. Diamond} to support its position that such a theory does exist, but notes that Oracle in fact relied upon that case for the instructions Oracle proposed on \textit{de minimis} copying, and which in fact “adopted the ‘freestanding \textit{de minimis} doctrine.’”\textsuperscript{135} Thus, Google argues what is effectively a “double waiver,” criticizing Oracle for failing to properly raise the issue.\textsuperscript{136} Google also makes various arguments in favor of the \textit{de minimis} nature of its copying, discrediting Oracle’s arguments as to the number of times rangeCheck is called upon by the Android platform: “[a] typical novel might include the word “the” thousands of times but that does not render the word “the” qualitatively significant to \textit{Moby Dick}.”\textsuperscript{137}

IV. DISCUSSION OF AMICUS SUBMISSIONS\textsuperscript{138}

A. Amici Urging Reversal

1. \textbf{Amicus Brief of Microsoft Corporation, EMC Corporation, and Netapp, Inc.}

Microsoft Corporation, EMC Corporation, and Netapp, Inc. (collectively “Microsoft”) submitted a brief in connection with the appeal and cross-appeal that relates not to the outcome of the District Court decision so much as the analysis that court undertook. Specifically,

\begin{itemize}
  \item \textsuperscript{133} Id. at 67-68.
  \item \textsuperscript{134} Reply Brief of Cross-Appellant Google Inc. at 11.
  \item \textsuperscript{135} Id.
  \item \textsuperscript{136} Id.
  \item \textsuperscript{137} Id at 19.
  \item \textsuperscript{138} For ease of reference, amicus briefs will be cited as “___ Br.” according to name or abbreviation of the lead amicus.
\end{itemize}
Microsoft urges that the Court mistakenly confused the copyrightability analysis with the infringement analysis, and as a result overlooked key aspects of software copyrightability, which could “harm incentives for innovation in the software industry.”¹³⁹

Microsoft urges that the District Court erred in failing to consider all 166 packages within the Java API, rather than only the 37 packages at issue.¹⁴⁰ Microsoft asserts that “[t]his mode of analysis is backwards” and that “by examining only the part of the allegedly infringed work that tracks part of the allegedly infringing work, the district court failed to consider whether and to what extent the allegedly infringed work, in total, merits copyright protection.”¹⁴¹ Microsoft also admonishes that the District Court failed to properly consider whether the selection, coordination and arrangement of the elements within the Java API was copyrightable, and cites Softel’s¹⁴² extension of Feist¹⁴³ to computer programs for that proposition. Microsoft also criticizes the District Court’s “impermissibly stringent originality standard” as illustrated by its consideration of Oracle’s claim that the Java API is a protectable taxonomy analogous to the taxonomy found protectable in American Dental Ass’n.¹⁴⁴ Microsoft reasons that the Java API is entitled to an even greater degree of protection than the taxonomy there, namely because “the Java platform contains a category of creative originality not present in a taxonomy: the choice of what to include in it.”¹⁴⁵ Microsoft also mentions a number of cases that “extensively discuss the copyrightability of critical elements of computer software,” but which the District Court failed to

¹³⁹ Microsoft Br. at 2.
¹⁴⁰ Id at 10.
¹⁴¹ Id.
¹⁴² Softel, Inc. v. Dragon Med. & Scientific Commc’ns, Inc., 118 F.3d 955 (2d Cir. 1997).
¹⁴³ See supra note 65.
¹⁴⁴ See supra note 66.
¹⁴⁵ Microsoft Br. at 13.
cite, while also criticizing the Court for emphasizing cases that relate to infringement and fair use rather than to copyrightability.\footnote{Id at 13-14.}

Microsoft also discusses a number of policy considerations, namely that the Court’s analysis will chill investment in software innovation, namely because developers can no longer expect the structure of their software to be protected.\footnote{Id at 2, 4.} Microsoft further discusses the import of the analysis on the General Public License (GPL), which solves a dilemma between the desire to create and disseminate open source software to the general public and the problem of subsequent proprietizing of that software through copyright law. That GPL, they say, is protected by copyright law, and so to raise the copyrightability bar is to limit the application of copyright law to software licensed under the GPL.\footnote{Id at 16.} Microsoft concludes its submission with a championing of legal certainty in copyright law, noting the absurdity of the parade of horribles cited by the District Court were the Java API to be found copyrightable,\footnote{Id at 20.} and emphasizing the fact that fair use, substantial similarity, and merger are in place to prevent abuses, not the low bar to copyrightability.

2. **Amicus Brief of Ralph Oman, former Register of Copyrights of the United States**

Ralph Oman, former Register of Copyrights of the United States, states at the outset that his brief is principally concerned with “the denial of copyright protection to the creative structure, sequence and organization of Oracle’s software packages.”\footnote{Oman Br. at 2.} Oman also cites concerns similar to Microsoft’s, namely that “[b]y injecting the fair use balance into the threshold determination of copyright protection, the court below erred.”\footnote{Id at 4.} Yet Oman’s primary
concern relates to the protection of the SSO of the Java API, what he terms a “sui generis exception” for software, and comparable to what Oracle terms “software exceptionalism,” on appeal. These are all synonymous for what Oman argues is a lower level of copyright protection for software, and is an incorrect application of copyright law; rather, the appropriate limitations on copyrightability are found in the Copyright Act, including substantial similarity, fair use, merger, and functionality.152 As Oman asserts, “[c]ontrary to the trial court’s conclusion here, nothing in the sophisticated process courts now apply in analyzing copyrightability [e.g. Altair’s abstraction-filtration-comparison test] changes the scope of protection accorded non-literal aspects of software, regardless of whether they are complex menu commands, interfaces, or the structure, sequence or organization of a multitude of aspects of those programs.”153

Having viewed the issue from the 102(a) perspective, Oman next reviews the District Court determination from the 102(b) perspective, noting the error of the Court’s decision to apply 102(b) as broadly as it did. Similar to Oracle’s briefing on appeal, Oman notes that:

“[t]aken to the highest level of abstraction, all computer programming, including source code, can be characterized as a “method of operation” or system of commands, yet no one would argue that none of it is protected by copyright, as that would be flatly contrary to the Act and the intent of Congress.”154

Particularly noteworthy with respect to 102(b) is Oman’s analysis of the District Court’s denial of protection to the SSO of the Java API: “The court below did not hold that the structural components of Oracle’s packages merged with expressive content and were therefore unprotectable. Instead, the court held that that structure was not protectable because, although it “resembles a taxonomy, it is also a command structure.”155 Oman also takes issue with the District Court’s focus on interoperability, and its conclusion that an expression necessary for

152 Id at 5-9.
153 Id at 12-13.
154 Id at 15.
155 Id at 17 (quoting Oracle America, Inc., 872 F.Supp.2d at 999).
interoperability necessarily merges with the idea underlying that expression. Oman interprets the District Court’s conclusion in this regard as a misreading of Altai and its abstraction-filtration-comparison test, specifically of the external factor relating to compatibility which requires “a programmer to writ the program in a specified way, rather than exercising judgment and creativity in choosing from a range of options.” Oman argues that in assessing merger, the compatibility externality as identified in Altai relates to the “extent to which plaintiff-programmer’s choices were strictly dictated by the need to ensure that it could interact with existing third party programs” and the focus was not “on the third-party software or hardware creator’s need to interact with plaintiff’s program.” In other words, Oman argues the focus should be on the interoperability of Oracle’s software, not Google’s software, when assessing the copyrightability of the former.

In closing, Oman notes that the District Court’s decision was based upon two cases, Sega and Sony, and makes much of the fact that, “[w]hile the court acknowledges that these were fair use cases… the court failed to appreciate how this important distinction rendered fits reliance on those cases totally inapposite.” Specifically, in Sega “the only question was whether Accolade’s intermediate copying was a fair use; the Ninth Circuit never reached the issue of whether or not the software code that incorporated the functional aspects that Accolade sought to mimic also contained separable creative expression.” Similarly, in Sony, “the Ninth Circuit only determined, once again in the fair use context, that Sony’s program – like all software –

156 Id at 19.
157 Id at 20.
158 Id at 22.
159 Id at 25.
contained functional aspects; never attempted to identify whether [the program at issue] had expressive elements.”

Oman concludes with an admonishment: that patent and copyright protection are not mutually exclusive. This idea, first posited in Mazer v. Stein and echoed in the CONTU Report, cautions that expressive features of software are not only properly protected under U.S. law, but under GATT/TRIPS and the WIPO Copyright Treaty as well. “It would be ironic if that historic achievement was totally subverted by the faulty reasoning of the court below.”

3. Amicus Brief of Picture Archive Council of America Inc. & Graphic Artists Guild

The Picture Archive Council of America and the Graphic Artists Guild (collectively the “Council”) urges the Federal Circuit, not only to reverse the District Court’s decision on the copyrightability of the Java API, but also to “rule that, as a matter of law, there is no fair use defense to Google’s infringement.” Underlying that request is a concern “about fair use in the context of misappropriation of creative works by a competitor,” and a belief that “there is nothing ‘fair’ about fragmenting Oracle’s markets and undercutting its compelling and recognizable ‘write once, run anywhere’ philosophy.” Clearly, the Council believes that a decision finding Google’s use fair will “affect all creators.”

In showing that the fair use factors disfavor Google, the Council emphasizes the bad faith and commerciality aspects of the first factor, and points to Google’s bad faith in utilizing the Java language without obtaining a license: “Google’s motive – to profit from the unauthorized use of a work that took substantial effort and millions of dollars to create, displacing the original

---

160 Id at 26.
161 Id at 29-30.
162 Id at 30.
163 Council Br. at 6.
164 Id.
165 Id at 7.
work – is the essence of bad faith.”166 The Council proceeds to point out other instances of Google’s conduct that it perceives as instances of utter disregard for copyright law, ranging from the Google books debacle to the Viacom v. YouTube litigation.167 The Council goes on to show that Google’s use is commercial,168 and that there is nothing transformative in Google’s use given that “[Google’s] purpose was exactly the same as Oracle’s – to call upon the Java packages’ library of prewritten source code – and to perform identical functions as in Java, using identical names and structure that were so familiar to Java application programmers.”169

As to the second factor, the Council asserts that the production of the Java API was highly creative, that it is deserving of more than the “thin” protection provided to a telephone book in Feist,170 and that the District Court “confused the copyrightable source code… with the method by which that function is performed.”171 As to the third factor, the Council cites Harper & Row noting that Google appropriated the “heart of what the programming community sought,” and “[h]aving taken the heart, it transplanted the entire circulatory system.”172 As to the fourth factor, the Council accuses Google of stalling Oracle’s derivative smartphone market, and eviscerating its “ability to derive revenue from the smartphone market by licensing a mobile version of Java.”173 The Council also emphasizes the harm done to Oracle’s “write once, run anywhere” principle, which fails to foster interoperability and fragments Oracle’s market.174

---

166 Id at 10.
167 Id at 12-13.
168 Id at 13.
169 Id at 16-17.
170 See supra note 65.
171 Council Br. at 21.
172 Id at 22.
173 Id at 24-25.
174 Id at 27.
4. Amicus Brief of BSA | The Software Alliance

In urging reversal, the BSA | The Software Alliance (hereinafter “BSA”) argues in principal that software deserves copyright protection equivalent to that of other copyrightable works, and that absent this protection, developers will lack the necessary incentives to innovate in the software space. Echoing the concerns of other amici, BSA argues that the District Court “conflates a question of copyrightability with one of infringement.”

BSA also points not only to the fact that interoperability is not a basis for denying copyrightability, but that the situation at bar is not actually one of interoperability given that “Android was seeking to replace the Java API programs for mobile devices, not connect to them.”

BSA begins its argument with a discussion of the CONTU Report and the Congressional decision to accord equivalent copyright protection to computer programs. BSA notes that because 17 U.S.C. § 117 creates an exception from infringement for copying that is “necessary to the ‘utilization of the computer program,’” this must mean that “more general functional aspects of the program must properly be protectable… [o]therwise there would be no need for this exception.” BSA goes on to note Apple Computer Inc. v. Franklin Computer Corp. and the Third Circuit’s analogy between computer software and an instruction manual written in ordinary English on how to activate complex machinery; the Court concluded that “a ‘process’ is no more involved” in one than it is in the other for purposes of section 102(b) filtration. BSA also notes the case law supporting protection of SSO in computer software, namely the Atari and Altai cases.

---

175 BSA Br. at 6.
176 Id. at 7.
177 Id. at 11.
178 714 F.2d 1240 (3d Cir. 1983).
179 BSA Br. at 12 (quoting Apple Computer, Inc. v. Franklin Computer Corp., 714 F.2d 1240, 1251 (3d Cir. 1983)).
180 See supra note 60.
181 See supra note 57.
BSA goes on to point out three errors made by the District Court: “its treatment of functional aspects in software as precluding copyrightability, in its broadening of the § 102(b) method of operation exclusion and the merger doctrine, and in its misapplication of the principle that words and short phrases are not copyrightable.”\textsuperscript{182} As to the first issue, BSA criticizes the District Court’s concomitant acknowledgment of expressive elements in the Java API and subsequent decision to deny those elements protection as part of a command structure; BSA argues this runs counter to \textit{Atâri}.\textsuperscript{183} As to the second issue, BSA asserts that the “creative allocation of thousands of lines of code into a complex structure”\textsuperscript{184} is no method of operation, distinguishing \textit{Lotus}\textsuperscript{185} as dealing with far less creative subject matter, and eschewing the application of the merger doctrine in light of the District Court’s admission that “Google could have structured Android differently.”\textsuperscript{186}

As to the third issue, BSA notes that the Words and Short Phrases Doctrine is little more than recapitulation of the maxim that “\textit{a work must be original}” to obtain copyright protection, arguing that it is “inappropriate for the district court to look only to the length of the smallest components of the API rather than looking to the originality of the API as a whole.”\textsuperscript{187} BSA concludes by discrediting the District court’s reliance on interoperability, distinguishing \textit{Sega}\textsuperscript{188} and \textit{Sony}\textsuperscript{189} as infringement/fair use cases rather than copyrightability cases, and as “intermediate copying” situations rather than an instance of copying to gain a competitive advantage by accommodating developers familiar with Java.\textsuperscript{190} BSA concludes noting that

\textsuperscript{182} BSA Br. at 15.\textsuperscript{183} See supra note 60.\textsuperscript{184} BSA Br. at 21.\textsuperscript{185} See supra note 87.\textsuperscript{186} Id at 23 (citing Oracle America, Inc., 872 F.Supp.2d at 999).\textsuperscript{187} Id at 26.\textsuperscript{188} See supra note 58.\textsuperscript{189} See supra note 59.\textsuperscript{190} BSA Br. at 30.
Google is not eligible for the fair use defense, again because Google did not intend to make Android interoperable with Java, but rather to entice Java developers.\footnote{Id at 33.}

5. \textit{Amicus Brief of Scott McNealy and Brian Sutphin}

In arguing for reversal, Scott McNealy and Brian Sutphin (collectively “McNealy”) first discuss the development of the Java language, and that its ability to facilitate interoperability was the primary reason for its importance in the developer community. McNealy notes that “Java’s success rested in large part upon its elegant and creative set of packages that Sun designed and developed… Merely offering a programming platform in which developers could build cross-system applications would not have been enough to get developers to use it.”\footnote{McNealy Br. at 8-9.} McNealy challenges the notion that software is more functional than literature by nature: “While a novel may have an object of stirring certain thoughts in a reader (a function), writing [sic] a novel or poem is ultimately a highly creative process that requires careful consideration to a variety of literary concerns, including structure, word choice, and pacing.”\footnote{Id at 13.} McNealy points to the creativity inherent in the selecting, naming, and organizing that goes into the Java API packages, and likens the construction of a package to designing a building, which involves many functional components like doors, windows, etc. but is ultimately a creative endeavor.\footnote{Id at 17-18.} To illustrate the point, McNealy discusses setting the time zone in an application using Java, Apple’s iOS, and Microsoft’s Windows Phone, explaining how similar functionality is achieved despite varying creative decisions in the organization naming of various elements within each.\footnote{Id.}

McNealy goes on to discuss Java licensing, including the GPLv2 license that requires developers to give improvements they make back to the developer community, the commercial
and specification licenses that restrict a licensee’s implementation of Java to preserve Java’s “write once, run anywhere” principle. In doing so, McNealy emphasizes that these licenses were available to Google, who declined to take them an instead chose to copy the creative aspects of Java that were familiar to developers. The effect of this was “quick access to Java developers while ensuring that Java cross-platform compatibility was not maintained.” Thus, McNealy points out that, by making the Android platform incompatible with Java, “Google has used the creative aspects of java to undermine its core mission: “Write Once, Run Anywhere.”

6. Amicus Brief of Eugene H. Spafford, Ph.D., Zhi Ding, Ph.D. and Lee A. Hollaar, Ph.D.

In arguing for reversal, Eugene H. Spafford, Ph.D., Zhi Ding, Ph.D. and Lee A. Hollaar, Ph.D. (collectively “Spafford”) asserts that the District Court decision displayed a fundamental misunderstanding of the creativity that went into developing the Java API. Spafford distinguishes between the various items that the term “API” can refer to, including a protocol used to interact with a system, the documentation describing that system’s specification, and a “software API,” which “provides an interface that allows a user, computer, or piece of software to communicate with another piece of software.” Spafford adds to wealth of analogies arguing, similar to the McNeally brief, that “an API is similar to a blueprint… [it] may sp[ecify the building materials, clearances and dimensions, placement of utilities, and methods of entry and egress.” In short, the components are functional, but the choices made about where those functional components fit are creative ones, which Spafford asserts are protectable.

196 Id at 20-23.
197 Id at 25.
198 Id at 25.
199 Spafford Br. at 8.
200 Id at 9.
Spafford further argues that there are many ways to design an API, and uses a software API for drawing simple shapes as an example. Spafford concludes that the result of this “great design flexibility” is that “the expression of an API reflects the author’s imagination and creativity rather than rigid dictates of pre-determined functions.”\(^{201}\) Spafford goes on to cite the many creative choices that the District Court ignored, and which developers of the Java API had to make when developing the Java API, including “the class in which the method is defined; the method’s exposure to other classes; and the containing class’s relationship to other classes, interfaces, and packages.”\(^{202}\) Spafford concludes with a discussion of the fact that Android is not interoperable with Java, and notes that “the court was incorrect, as a technological matter, in its assumption that Google’s copying of the Java API did—or could—allow arbitrary programs written in Java to run on Android.”\(^{203}\) Rather, Spafford cites the fact that Google’s own engineers understood the motivation for using the Java API was “not for interoperability reasons, but merely to make Android more attractive to the pre-existing base of Java developers.”\(^{204}\)

Spafford concludes noting that copyright protection for APIs would create an incentive for innovation, and noting that “unanticipated vulnerabilities and instability” could be caused by modification of the API without any kind oversight by a copyright owner: “[w]e believe that a software API should receive copyright protection to help prevent such failures in information systems.”\(^{205}\)

\(^{201}\) Id at 17.
\(^{202}\) Id at 19.
\(^{203}\) Id at 20.
\(^{204}\) Id at 21.
\(^{205}\) Id at 23.
B. Amici Urging Affirmance

1. Amicus Brief of the Intellectual Property Law Professors

The Intellectual Property Law Professors (collectively “IP Professors”) have argued in their amicus brief that the District Court’s decision should be affirmed on appeal. Unsurprisingly taking a legal approach (as opposed to a policy-oriented approach) to the questions on appeal, the IP Professors argue that “[t]he exclusion of methods and processes from the scope of copyright helps to preserve the distinction between the patent and copyright realms,” citing Taylor Instrument Co. v. Fawley-Brost Co. for that proposition. In that case, the Seventh Circuit found that charts, as components of temperature recording systems, were a “mechanical element of the instrument of which it is an integral part.” Similar to Taylor, the District Court here “noted that ‘[b]oth Oracle and Sun have applied for and received patents that claim aspects of the Java API.’” The IP Professors also cite Sega and Atari as cases that “warn against construing copyrights so broadly that they would grant patent-like protection to subject matters that are ineligible under 17 U.S.C. § 102(b).” Thus, the IP Professors hammer home the District Court’s point that “[f]unctional methods and processes embodied in computer programs may be eligible for patent protection, but not for copyright.”

The IP Professors also echo the District Court’s illustration of a trend away from protection of the SSO of computer software, citing Altai as the high watermark of such SSO protection and noting that “[t]he Ninth Circuit in Sega followed Altai’s lead in holding that interface procedures necessary for achieving interoperability among programs were functional

206 139 F.2d 98 (7th Cir. 1943).
207 IP Professors Br. at 8.
208 Id at 9 (quoting Taylor, 139 F.2d at 100).
209 See supra note 58.
210 See supra note 60.
211 IP Professors Br. at 11.
212 Id at 12.
213 See supra note 57.
elements of programs that copyright did not protect under § 102(b).” 214 Particularly noteworthy is the IP Professors’ mention of Hutchins v. Zoll Medical Corp., 215 where the Court affirmed summary judgment of non-infringement given that copyright protection was “unavailable to the ‘technologic method of treating victims by using CPR and instructing how to use CPR.’” 216 The IP Professors point out that the Federal Circuit endorsed the approach of the First Circuit Court of Appeals in the Lotus case, which is potentially useful to Google on appeal as it provides the Federal Circuit with a relevant precedent more recent than Atairi, 217 and one that can serve as a basis on which to affirm the District Court.

The IP Professors go on to distinguish Atairi 218 by noting that Atari games “copied more than was necessary to achieving compatibility with Nintendo’s programs, which was why the Federal Circuit ruled against its compatibility defense in that case.” 219 In short, the IP Professors seek to further illustrate a trajectory of case law increasingly willing to find that the merger doctrine precludes protection of functional aspects of computer software, beginning with Altai, and culminating with Sega 220 and later Sony. 221


Rackspace, Application Developers Alliance, TMSOFT and StackExchange (collectively “Rackspace”) begin their argument for affirmance of the decision below with an acknowledgment that the industry norm has been that declaring code is functional and non-protectable, and implementing code is expressive and protectable by copyright. To do otherwise,

---

214 IP Professors Br. at 14.
215 492 F.3d 1377 (Fed. Cir. 2007).
216 IP Professors Br. at 15 (quoting Hutchins, 492 F.3d at 1384).
217 See supra note 60.
218 Id.
219 IP Professors Br. at 18.
220 See supra note 58.
221 See supra note 59.

Page 39 of 50
Rackspace says, “would have a devastating impact on the software business as a whole. Thousands of businesses, developers, and even end users would be faced with legal uncertainty if declaring code were afforded copyright protections.”

In addressing the merits of Oracle’s arguments, Rackspace first notes that Oracle has employed a great deal of “discredited ‘sweat of the brow’-type reasoning to argue that it should be entitled to monopolize the functional bits of code necessary for interoperability.” In so noting, Rackspace picks up on the plethora of statements Oracle has made to the effect that it spent years and many millions of dollars developing the Java platform and API, noting their irrelevance to the copyrightability analysis. Also interesting is Rackspace’s decision to include the scenes a faire doctrine as a basis for precluding copyright protection “for the declaratory elements of APIs that are dictated by practical realities like interoperability and file format compatibility.” In fact, the District Court acknowledged scenes a faire as a potentially proper basis for denying Oracle’s SSO claim, but ultimately declined to adopt it as the basis for its decision due to an absence of evidence presented during trial.

Rackspace analogizes the declaring code in APIs to the bumps and holes of Lego bricks, i.e. the attributes of Lego bricks that allow them to be combined in nearly infinite combinations. Meanwhile, the implementing code would be comparable to the color, shape, size and every other aspect of those bricks other than the attributes that allow them to be connected. Rackspace notes the importance of APIs in our daily lives as it relates to everything from speedometer read-outs in a car to cloud computing. Rackspace cites historical

---

222 Rackspace Br. at 4.
223 Id at 5.
224 Id.
225 Oracle America, Inc., 872 F.Supp.2d at 999 n. 9.
226 Rackspace Br. at 7.
227 Id.
228 Id at 8-9.
reliance by those in the software industry over the last forty years that APIs are not copyrightable, proffering the example of Bell Labs’ development of Linux by using the MINIX API.\textsuperscript{229} Rackspace cites Microsoft’s copying of the Corel API in order to allow WordPerfect files to be opened in Microsoft Word.\textsuperscript{230} In short, Rackspace asserts that “[r]e-implementing APIs for the purpose of competing with an existing product is a regular and accepted business practice.”\textsuperscript{231}

In conclusion, Rackspace notes the anticompetitive concerns surrounding attempts to retain APIs as proprietary, and posits that open source software provides enough of an incentive for software innovation to take place.\textsuperscript{232} Rackspace also supports the validity of the District Court’s reasoning, arguing that as a policy matter deciding the issues on copyrightability grounds is preferable to invoking fair use, namely because the latter would allow suits to proceed, and increase transaction costs as well as the inherent uncertainty flowing from the fluidity of the fair use test.\textsuperscript{233}

3. Amicus Brief of the Software Innovators, Start-ups, and Investors

In arguing for affirmance of the District Court's decision, the Software Innovators, Start-ups, and Investors (collectively the "Software Innovators") stress the importance of interoperability, which "supports market entrance by new competitors" and "accelerates the development of new uses, products, and services in a wide variety of fields... fueling competition."\textsuperscript{234} According to the Software Innovators, APIs, which "allow start-ups to create programs that can communicate with and integrate the technology of existing systems," play an

\textsuperscript{229} Id at 10.
\textsuperscript{230} Id at 13.
\textsuperscript{231} Id at 12.
\textsuperscript{232} Id at 16.
\textsuperscript{233} Id at 19.
\textsuperscript{234} Software Innovators Br. at 4.
integral role in facilitating interoperability.\footnote{Id at 3-4.} Specifically, the Software Innovators note that "software programs that implement APIs decrease the amount of time they take to bring products to market by 30%.\footnote{Id at 8.} Were APIs protected by copyright, they argue, "companies that presently provide complementary services by connecting to existing systems would be required instead to undertake a development process fully duplicative of the existing system, limiting offerings to consumers."\footnote{Id at 12.} The Software Innovators further point to the chilling effect that a reversal would have: "investors are much less likely to invest if they fear that their investments will go towards copyright litigation."\footnote{Id at 14.}

In defending the District Court decision, the Software Innovators rely upon the determination that the Java APIs constitute a "method of operation" under section 102(b), arguing that "[m]ethods of operation do not themselves become copyrightable because they relate to some creative expression."\footnote{Id at 28.} They respond to the arguments made by Oracle and those urging reversal, asserting that "available alternative design choices [do not] affect whether a program feature is a method of operation."\footnote{Id.} The Software Innovators also reaffirm the merger and interoperability bases for the decision, but ultimately are more concerned with the policy implications of a reversal on appeal: "It is the changes to well-settled limitations on copyright in computer programs that would disrupt the existing framework and create uncertainty for innovators and their investors alike."\footnote{Id at 31.}
4. Amicus Brief of the Computer Scientists

The Computer Scientists cite concerns similar to those of Software Innovators in arguing for affirmance of the District Court decision on appeal, namely that excluding APIs from copyright protection allows competitors to "challenge established players and advance the state of the art," and as a result "has been essential to the development of modern computers and the Internet." Furthermore, and as with most of the amici supporting affirmance, the Computer Scientists note that "[t]he free and open use of APIs has been routine in the computer industry since its beginning." The Computer Scientists tell the "story of Compaq's and Phoenix's creation of PC clones" of IBM software, where "clean" teams that had never seen the IBM source code were assembled to avoid copyright infringement, but used the BIOS API to create the clones. The Computer Scientists also discuss the successful reimplementations of the UNIX API that resulted in the Linux and OSX operating systems, noting that this free implementation of the UNIX API would not have occurred if copyright protection had extended to it, as well as the reimplementation of the C programming library to create several software platforms. They further note that the functioning of the Internet itself is highly dependent upon reimplementations of various software, including Berkeley Systems Distribution (BSD) sockets, network protocols like HTTP, which are "very similar to APIs," and cloud computing platforms that rely on virtual machines to call functions from APIs.

Interoperability and compatibility are among the Computer Scientists' chief concerns, given that reimplementations of APIs facilitate innovation and growth in the software ecosystem.

242 Computer Scientists Br. at 2.  
243 Id at 4.  
244 Id at 6-8.  
245 Id at 9-10.  
246 Id at 12-13.  
247 Id at 15.  
248 Id at 18.
They cite the Wine Project as an example of useful software the creation of which would be deterred by onerous licensing demands, or worse, by copyright litigation involving statutory damages.\textsuperscript{249} They also discuss supercomputers and the manner in which developers must create programs that work on both supercomputers and normal computers, which in turn necessitates a shared API. After coding for a particular proprietary API and making a substantial investment in doing so, developers would "would find themselves 'locked in' to the vendor they worked with at the start of their project."\textsuperscript{250}

Finally, the Computer Scientists discuss what they refer to as the "orphan software" problem: "Software creators go bankrupt or stop supporting their creations, and the intellectual property in software is often bought and sold when startups are acquired or divisions of companies spin off or shut down."\textsuperscript{251} The result is orphan software, for which the only remedy may be reimplemention of the associated API in order to salvage the time and money invested in the orphaned software. Yet according to the Computer scientists Section 102(b) "solves this problem… [b]y keeping interface specifications free of copyright" and allowing "developers to build compatible systems."\textsuperscript{252} The Computer Scientists cite the collapse of the bookmarking site Delicious and Pinboard's reimplemention of the Delicious API to allow former Delicious users to retain their Delicious-based applications,\textsuperscript{253} as well as the impending shutdown of Google Reader and Feedly's use of the associated API to attract users.\textsuperscript{254} The Computer Scientists conclude noting that the orphan problem disproportionately affects the public sector and not-for-

\textsuperscript{249} Id at 23.
\textsuperscript{250} Id at 23-24.
\textsuperscript{251} Id at 27.
\textsuperscript{252} Id at 28.
\textsuperscript{253} Id at 29.
\textsuperscript{254} Id at 30-31.
profits, who are "especially susceptible to the orphan programs problem... since their tight
budgets often force them to use outdated technology."255

5. Amicus Brief of the Computer & Communications Industry Association

The Computer & Communications Industry Association (hereinafter the "Association")
begin their brief in support of affirmance by noting the fundamental differences between a novel
and a computer program: "A novel stands by itself. A computer application, however, can
function only in conjunction with hardware and other software."256 The Association stresses that
copyright protection for APIs will stifle innovation, allow companies to leverage their market
position to keep competitors out, and restrict consumer choices by locking them into a particular
operating system or software environment.257 The Association echoes the reasoning of the
District Court as well as the many of the arguments made by Google on appeal, namely that
copyright protection for software is "thin," and that the logic of Whelan258 and Franklin259 has
been discredited and rejected by subsequent courts.260 Yet it ultimately seeks to provide "an
overview of how U.S. courts, Congress, and jurisdictions around the world have, or the past 25
years, arrived at a consensus interpretation of the copyright question of interoperability."261

In presenting this overview, the Association begins by noting that "decision makers
around the world" have agreed that "interface specifications" are not protected by copyright, nor
are copies made pursuant to reverse engineering.262 The Association cites five judicial decisions
where reverse engineering activities were protected by fair use, and a great deal more cases
dealing with interface specifications, concluding that "U.S. copyright law ultimately settled on a

255 Id at 31.
256 Association Br. at 3.
257 Id at 5.
258 See supra note 56.
259 See supra note 179.
260 Association Br. at 5-6.
261 Id at 8.
262 Id at 9-10.
rule now internationally recognized: that copyright protection does not extend to interface specifications necessary for interoperability.”\textsuperscript{263} Notably, the Association discusses section 1201(f) of the Digital Millennium Copyright Act (DMCA), which effectively creates an exception to the DMCA’s prohibition on circumvention of technological security measures for reverse-engineering and interoperability purposes,\textsuperscript{264} the intent being to allow developers to exercise "fair use privileges recognized in Sègà and its progeny.”\textsuperscript{265} The Association also cites the various U.S. Free Trade Agreements that mirror the DMCA’s reverse-engineering exceptions,\textsuperscript{266} as well as the European Union Software Directive and its exceptions from liability in Article 5(3) for "reverse black box engineering," and in Article 6 for decompilation or disassembly, i.e. translation of object code into source code.\textsuperscript{267} The Association also discusses the reverse-engineering and/or interoperability exceptions as they exist under the laws of Australia, Hong Kong, Singapore, Philippines, Taiwan, New Zealand, Malaysia, India, Kenya, Israel, and Canada.\textsuperscript{268}

The Association concludes by noting that amici in support of Oracle "largely acknowledge as a matter of law and policy what jurisdictions around the world have concluded: copyright does not and should not apply to program elements necessary to achieve interoperability.”\textsuperscript{269} Specifically, the Association notes that BSA and Spafford acknowledge these interoperability exceptions under the present law as well as the reach of fair use following

\textsuperscript{263} Id at 14.
\textsuperscript{264} Id at 16-17. Section 1201(f)(4) defines "interoperability" as "the ability of computer programs to exchange information, and of such programs mutually to use the information which has been exchanged." 17 U.S.C. § 1201(f)(4).
\textsuperscript{265} Association Br. at 17.
\textsuperscript{266} Id at 19-20.
\textsuperscript{267} Id at 21. The Association concedes that the Software Directive does not expressly mention the protectability of interface specifications. Id. Yet it notes that "Commentators interpreted this to mean that interface information necessary to achieve interoperability must fall on the idea side of the idea/expression dichotomy." Id at 22.
\textsuperscript{268} Association Br. at 23-29.
\textsuperscript{269} Id at 29.
Sega, and improperly take issue with the District Court's reasoning on the copyrightability of the Java APIs, as well as with determinations of fact relating to Google's motivation for implementing the Java APIs.\textsuperscript{270}

V. IMPLICATIONS OF A DECISION ON APPEAL

A. Strict Legal Implications

The main legal question to be answered on appeal is whether the District Court erred in finding that the declarations and SSO of the Java APIs are not copyrightable. Thus, perhaps the most significant legal implications of a decision on appeal would flow from a determination on the copyrightability of the Java API. If the Federal Circuit affirms, then protection for software SSOs will become more narrow, as will protection for literal declarations in source code. Put another way, the case might stand as precedent for the proposition that copyright in software is "thin" and extends only to a precise iteration of the source code at issue, namely because the interoperability rationale that served as the basis for the decision here could be invoked in almost any instance involving software. One would be hard pressed to identify software for which a case for interoperability could not be made, particularly given that all software is necessarily interoperable and does not "stand by itself."\textsuperscript{271}

Even without the interoperability rationale, the District Court's section 102(b) rationale, if allowed to stand, would seem to do something similar, given that all software is ultimately "functional." Thus, if the Federal Circuit affirms on either rationale, Oracle may have a colorable appeal to the Supreme Court, requiring it to opine on the nature of protection for "computer programs." In taking the appeal, the Supreme Court might rule on the level of abstraction courts should use when applying the merger doctrine in software cases, particularly given the degree of

\textsuperscript{270} Id at 30-31.
\textsuperscript{271} See supra note 256.
confusion the doctrine has caused and continues to cause amongst the various courts,\footnote{See supra note 72.} and on the meaning of section 102(b).

Alternatively, if the Federal Circuit reverses the copyrightability determination and remands to the District Court for consideration of Google’s fair use defense, the implications of such a decision would be less clear. Although the jury hung on that issue, nine jurors allegedly found that Google had proved its fair use defense.\footnote{See A131, 24622-24.; U.S.C.A. for the Federal Circuit, Oracle America, Inc. v. Google, Inc. No. 10-CV-3561. (May 23, 2013)} Thus, while the Federal Circuit may rule, as Oracle urges it to do, that Google's fair use defense is precluded as a matter of law, a newly formed jury could find that Google’s use was fair if the Court reverses on copyrightability and remands on fair use. On the other hand, the jury could also find that the use was not fair, or hang once again.

A finding that Google's use of the Java APIs was fair would likely be confined to the facts of the case, would have less precedential value, and could go either way given the fluidity and unpredictability of the fair use analysis. On the other hand, a finding that Google’s fair use defense is precluded as a matter of law could have major legal implications. The Federal Circuit will be applying Ninth Circuit law in deciding the issues on appeal, and a finding that the fair use defense is precluded on the facts of this case would cut strongly against the Ninth Circuit's holdings in Sega\footnote{See supra note 58.} and Sony.\footnote{See supra note 59.} The result would arguably be a circuit split warranting Supreme Court review, and a high court pronouncement on fair use in the software space.

A fair use appeal to the Supreme Court would also be possible in the event that the Federal Circuit holds as a matter of law that Google's fair use defense is precluded. The Supreme Court has not centrally addressed the fair use doctrine since its decision in Campbell v. Acuff-
Rose Music, Inc., and has never before found the defense to apply in the context of computer software. In doing so, the Court would primarily grapple with whether the creation of new software that copies APIs for purposes of interoperability constitutes a transformative use, as well as whether a pre-existing licensing market for the API favors the copyright holder notwithstanding that party's failure to enter the specific mobile marketplace at issue (i.e. smartphones). The Court would then have to consider the scope of the relevant marketplace for purposes of the fourth fair use factor, another point of contention among the parties.277

B. Implications for the Software Industry

Many have argued that the outcome of this case will have a profound impact on the software industry in terms of interoperability and innovation. Computer scientists, tech industry leaders, the Electronic Frontier Foundation (“EFF”) and others have filed amicus briefs that focus upon stimulating innovation, the economy, and interoperability in the software industry. Oracle and its amici argue that a lack of copyrightability for the Java APIs will translate into a lack of incentive to create, invest, and innovate in software, whereas Google and its amici argue that copyrightability for the Java APIs will stifle competition by deterring new market entrants that rely upon use of pre-existing software technologies.

These opposing views present a window into the balance that copyright law seeks to strike — one where enough protection is granted to incentivize innovation, but without granting too much to stifle it. This balance is largely a policy decision to be struck by the legislature, not

276 See supra note 108.
277 Response and Reply Brief of Plaintiff-Appellant at 57 ("To differentiate some smartphones from others (as Google does) slices the 'market' too thin.")
by the courts, and the more persuasive legal arguments will likely relate to Congressional intent.\textsuperscript{278}

\textsuperscript{278} For example, the CONTU Report is cited \textit{ad infinitum} by the parties and \textit{amici}, as its findings are an indication of Congressional intent in deciding to include “computer programs” within the protectable subject matter under the Copyright Act. \textit{See e.g. supra} note 59, 161, 177.