MORRISON & FOERSTER LLP
MICHAEL A. JACOBS (Bar No. 111664)
mjacobs@mofo.com
KENNETH A. KUWAYTI (Bar No. 145384)
kkuwayti@mofo.com
MARC DAVID PETERS (Bar No. 211725)
mdpeters@mofo.com
DANIEL P. MUINO (Bar No. 209624)
dmuino@mofo.com
755 Page Mill Road, Palo Alto, CA 94304-1018
Telephone: (650) 813-5600 / Facsimile: (650) 494-0792

BOIES, SCHILLER & FLEXNER LLP
DAVID BOIES (Admitted *Pro Hac Vice*)
dboies@bsfllp.com
333 Main Street, Armonk, NY 10504
Telephone: (914) 749-8200 / Facsimile: (914) 749-8300
STEVEN C. HOLTZMAN (Bar No. 144177)
sholtzman@bsfllp.com
1999 Harrison St., Suite 900, Oakland, CA 94612
Telephone: (510) 874-1000 / Facsimile: (510) 874-1460

ORACLE CORPORATION
DORIAN DALEY (Bar No. 129049)
dorian.daley@oracle.com
DEBORAH K. MILLER (Bar No. 95527)
deborah.miller@oracle.com
MATTHEW M. SARBORARIA (Bar No. 211600)
matthew.sarboraria@oracle.com
500 Oracle Parkway, Redwood City, CA 94065
Telephone: (650) 506-5200 / Facsimile: (650) 506-7114

Attorneys for Plaintiff
ORACLE AMERICA, INC.

UNITED STATES DISTRICT COURT

NORTHERN DISTRICT OF CALIFORNIA

SAN FRANCISCO DIVISION

ORACLE AMERICA, INC. Plaintiff, v. GOOGLE INC. Defendant.	Case No. CV 10-03561 WHA **ORACLE'S APRIL 12, 2012 BRIEF REGARDING COPYRIGHT ISSUES** Dept.: Courtroom 8, 19th Floor Judge: Honorable William H. Alsup

I. THE COPYRIGHT ACT PROTECTS A COMPUTER PROGRAMMING LANGUAGE THAT MEETS THE ORIGINALITY REQUIREMENT

The parties agree no case addresses directly whether the Copyright Act protects a computer programming language. However, the most analogous case law, general copyright principles, and the statutory purposes of the Copyright Act support protection. More importantly, none of the arguments advanced for denying protection to a programming language apply to the API specifications and class libraries at issue here.

The cases that have considered works most similar to a computer programming language are those involving codes or compilations of symbols. Courts have upheld copyright protection for codes or compilations of symbols that meet the originality requirement.

In *Reiss v. National Quotation Bureau, Inc.*, 276 F. 717 (S.D.N.Y. 1921), Learned Hand upheld copyright protection for a code book containing coined words that could be given an agreed meaning for the purpose of cable correspondence. Judge Hand compared the code to a language—"a set of words or symbols to form a new abstract speech, with inflections, but as yet with no meaning, a kind of blank Esperanto." 276 F. at 718. He saw "no reason" why the code's set of coined words could not be a "writing" protected by copyright simply "because they communicate nothing," given that "[t]hey may have their uses for all that, aesthetic or practical, and they may be the productions of high ingenuity, or even genius." *Id.* at 719; *see also Lotus Dev. Corp. v. Paperback Software Int.*, 740 F. Supp. 37, 72 (D. Mass. 1990) (rejecting argument "[t]hat not only languages such as English and French but all other languages as well – including Esperanto, and *Reiss*' coined words, *276 F. at 718*, and Pascal – are automatically ineligible for copyright protection").

Similarly, in *Hatfield v. Peterson*, 91 F.2d 998, 1000 (2d Cir. 1937), the Second Circuit upheld copyright protection for a telegraphic code that was a compilation of non-original words and phrases, holding "the copyright is valid because of the originality of the combination."

Courts have denied copyright protection to codes or systems of symbols only where they lack originality, not because they are inherently uncopyrightable. In *Toro Company v. R & R Products*, 787 F.2d 1208 (8th Cir. 1986), the Eighth Circuit rejected protection for a machine

parts numbering system because of lack of originality. But far from finding a set of symbols or a language *per se* an uncopyrightable system, the court explained that: "A system that uses symbols in some sort of meaningful pattern, something by which one could distinguish effort or content, would be an original work." 787 F.2d at 1213. Similarly, in *Brief English Systems, Inc. v. Owen*, 48 F.2d 555 (2d Cir. 1931), the court rejected a claim for exclusive right to use a published system of shorthand not because a code or language is inherently not copyrightable but only because "[t]here is no literary merit in a mere system of condensing written words into less than the number of letters usually used to spell them out." 48 F.2d at 558.

Like the code of coined words in *Reiss*, a computer language may qualify for copyright protection if it is sufficiently original. Indeed, to a much greater degree than the *Reiss* coined words, it may and typically does represent very substantial creative work of exactly the kind that copyright is intended to protect and promote, with "enough definite expression so that one may distinguish authorship." *Toro*, 787 F.2d at 1212.

Google argues that a programming language is "an uncopyrightable system or method of operation," "an idea, not expression." Google 4/3 Br. at 14-15. But a work that represents only one of many ways to perform a function is "the expression of a particular idea, not the idea itself." *Toro*, 787 F.2d at 1212; *see Apple Computer, Inc. v. Franklin Computer Corp.*, 714 F.2d 1240, 1253 (3rd Cir. 1983) ("If other programs can be written or created which perform the same function as Apple's operating system program, then that program is an expression of an idea and hence copyrightable.") As the Eighth Circuit explained in *Toro*:

> All that the idea/expression dichotomy embodied in § 102(b) means in the parts numbering context is that appellant could not copyright the idea of using numbers to designate replacement parts. Section 102(b) does not answer the question of whether appellant's particular expression of the idea is copyrightable.

787 F.2d at 1212.

The fundamental "idea" of a computer programming language is to permit the user to create an arrangement of symbolic commands that will direct a computer to perform specified tasks. There may be lower-level ideas that are unprotectable as well, like a programming language directed to a special purpose, or the idea of an object-oriented language. But these ideas

can be expressed in a wide variety of specific forms. While copyrighting a computer language cannot prevent others from designing programming languages that serve the same functions, the detailed vocabulary and written expression of the computer language should be protectable elements if sufficiently original and creative.

Adobe, for example, has asserted copyright protection for its PostScript computer language, and explains:

> The general *idea* of using a page description language is in the public domain. Anyone is free to devise his or her own set of unique commands that constitute a page description language. However, Adobe Systems Incorporated owns the copyright for the list of operators and the written specification for Adobe's Post-Script language. Thus, these elements of the PostScript language may not be copied without Adobe's permission.

Adobe Systems Inc., POSTSCRIPT LANGUAGE REFERENCE xiii and 9 (3d ed. 1999), available at http://www.scribd.com/doc/202357/PostScript-Language-Reference-Third-Edition. In asserting its copyright, Adobe has stated that one of its objectives is to "[m]aintain the integrity of the PostScript language standard." *Id.* at 9.

Google relies primarily on preliminary opinions in a pending English case recently referred to the European Court of Justice ("ECJ"). The ECJ has not yet responded to that referral. *SAS Institute, Inc. v. World Programming Ltd.*, [2010] EWHC (Ch) 1829. The ECJ will not interpret the U.S. Copyright Act; it will decide the case under the extensive relevant provisions of European treaty law and the EEC Software Directive and related case law (*id*. at ¶¶ 149-95), as well as the legislative history of the Directive and its adoption by the European Parliament (*see id*. at ¶¶ 211-227). The English court referred the programming language question to the ECJ for the very reason that its resolution was not "*acte clair*," that is, free from reasonable doubt.

Google relies, in particular, on the opinion to the ECJ by Advocate General Opinion. Opinion of Advocate General Bot, *SAS Institute v. World Programming Ltd.*, Case C-406/10 ("SAS Advocate General Opinion"). Google cites the following passage:

> It seems to me, therefore, that programming language is a functional element which allows instructions to be given to the computer. As we have seen with SAS language, programming language is made up of words and phrases known to everyone and ***lacking originality***. In my opinion, programming language must be regarded as ***comparable to the language by the author of a novel***. It is therefore the means which permits expression to be given, not the expression itself.

Id. at ¶ 71 (emphasis added). This analysis ignores the obvious difference that the language used by a novelist, unlike an original computer language, is not itself the author's work of original creative expression. The novelist does not create the language in which she writes.

In concluding that a computer language is "lacking originality," Advocate General Bot's opinion is also at odds with U.S. case law. Even "[a] factual compilation is eligible for copyright if it features an original selection or arrangement of facts." *Feist Publications, Inc. v. Rural Tel. Serv. Co.*, 499 U.S. 340, 350 (1991). Copyright protection is denied based on lack of originality only to "a narrow category of works in which the creative spark is utterly lacking or so trivial as to be virtually nonexistent." *Id*. at 359. Unlike the white pages at issue in *Feist*, a typical computer programming language represents creative work in its selection and arrangement of symbols that may be sufficient to meet the "originality" requirement.

Moreover, and of more significance to this case, despite opining that a computer programming language may not be copyrightable, Advocate General Bot concluded that an interface may be. He found that the Directive "does not exclude interfaces from copyright protection" and that "if the expression of the interface constitutes a substantial part of the expression of the computer program . . . it is eligible for protection under the Directive." SAS Advocate General Opinion, at ¶¶ 81-82; *see also id.* at ¶ 60 (elements of a program enjoy protection "provided that they contain some of the elements which are the expression of the intellectual creation of the author of the work").

Copyright protection of a computer language is also consistent with the Copyright Act's statutory purpose to "*promote* the creation and publication of free expression" by rewarding authors. *Eldred v. Ashcroft*, 537 U.S. 186, 219 (2003) (emphasis in original). Developing an original, never-before-written language—whether a computer programming language or a newly coined language for a dramatic production such as Na'vi (from the film *Avatar*) and Dothraki (from the HBO series *Game of Thrones*)—may represent years of creative work. Copyright protection rewards and promotes those creative efforts. The estate of J.R.R. Tolkien, for example, has asserted copyright protection for Elvish and other languages used in his works. http://www.theodoramichaels.com/articles/fan-fic.php#Languages

The policy balance to apply to the idea/expression dichotomy was described by the Ninth Circuit in *CDN Inc. v. Kapes*, which affirmed copyrightability of a collectable coin pricing guide:

> As Judge Hand noted, the difference between idea and expression is one of degree. This circuit has held that "[t]he guiding consideration in drawing the line is the preservation of the balance between competition and protection reflected in the patent and copyright laws." *Rosenthal*, 446 F.2d at 742. In this case, the prices fall on the expression side of the line. CDN does not, nor could it, claim protection for its idea of creating a wholesale price guide, but it can use the copyright laws to protect its idea of what those prices are. . . . Drawing this line preserves the balance between competition and protection: it allows CDN's competitors to create their own price guides and thus furthers competition, but protects CDN's creation, thus giving it an incentive to create such a guide.

197 F.3d 1256, 1262 (9th Cir. 1999) (citations omitted). Protecting the expression embodied in a computer language while allowing competitors to create their own languages with parallel purpose and function preserves the balance between competition and protection.

As demonstrated above, the best view of U.S. copyright law is that an original and creative computer language is subject to copyright protection as an "original work[] of authorship." 17 U.S.C. § 102. As far as Oracle has been able to determine, neither Oracle nor Sun has taken a contrary position on the copyrightability of computer languages.

II. HISTORY OF API DEVELOPMENT

Modern APIs owe their origins to the development of modular programming. Subroutines, first invented in 1951, divided programs into units with specific tasks. The first software libraries were collections of common, general-purpose subroutines that could be re-used in different programs. These libraries were not considered part of the language in which they were written. They were sets of reusable program modules, expressed in particular languages.

The use of an API as a specification of how software modules interact arose during the 1970s. One example from that time is *prototypes* written in the C programming language. Prototypes are fragments of code describing the sets of parameters to be passed to different subroutines and the types of their return values. Developers combined these code fragments with English prose specifying the behavior of the subroutines, creating API specifications similar to those written today. Other developers could learn from the API specifications how different libraries worked without having to study their underlying implementations.

The techniques of modular program development are more relevant today than ever. Professor Mitchell will testify that today's software systems are among the most complex products ever created by human beings, and APIs are the core structuring concepts software designers use to manage this complexity. (Mitchell Opp. Rep., ECF No. 397-1 ¶ 18.) Software developers often collaborate on projects from different cities or countries. (*Id.* ¶ 25.) They use APIs to understand the potential dependencies between different sections of code without having to know how the code for an entire project works. A developer in San Francisco, for example, can design an API for a library and then design and implement the library. A colleague in Paris need only consult the API in order to make use of that library; there is no need to know the inner workings of the library.

III. THE JAVA APIS ARE NOT PART OF THE PROGRAMMING LANGUAGE

A. A programming language and an API are distinct things with different purposes

The evidence will show that a programming language and an API are two very different things. A programming language is an artificial language used to create programs that control the behavior of a machine. An API implementation (often referred to as a "class library") is a computer program component that consists of pre-written code. The API specification is the blueprint to the class library. It is a detailed written description of the programs that explains how the programs are structured, how they are to be used, and what they will do. For example, a library implementing a database API will provide database functions, a library implementing a networking API will provide networking functions, a library implementing a mathematics API will provide arithmetic and trigonometric functions, and so forth.

Google was never confused about the distinction between an API and a programming language when this case began or for long afterwards. Google is now straining to change course to take a position it knows is factually incorrect. At the outset, Google acknowledged that the class libraries are distinct from the Java programming language. (*See* ECF No. 51 at 13-14.) Google's expert stated that the Java programming language and the Java APIs are "very different things." (Astrachan Opening Rep., ECF No. 262-1 ¶ 7.) He defined an API as "a particular set of

rules and specifications that software programs can follow to communicate with each other. It serves as an interface between different software programs and facilitates their interaction, similar to the way the user interface facilitates interaction between humans and computers." (*Id.* ¶ 24.)

One area where the parties' experts agree is that there are very few classes within the Java APIs that must be present for the Java language to function. Oracle's expert will testify that "it was not necessary to include any particular class or package (beyond perhaps a very few classes like Object and Class that are tied closely to the Java language) for the Java language to function." (Mitchell Opp. Rep., ECF No. 397-1 ¶ 20.) Google's expert testified in deposition that he agrees:

> Q. Paragraph 20. And Dr. Mitchell says, in around the middle of the paragraph, "It is important to realize that while the robust and elegant API specification and class libraries that implement them have been important to Java's success, it was not necessary to include any particular class or package beyond perhaps a very few classes like object and class that are tied closely to the Java language for the Java language to function." And maybe that's the missing piece here. I mean, do you have an opinion on what classes were required to be included in order for the Java language to function?
>
> A. I think this accurately reflects what the Java language needs to function.

(9/9/11 Astrachan Dep. 230:19-231:8.)

The Java Language Specification defines what is in the Java programming language. That definition includes those few classes, but otherwise does not include these 37 API packages.

B. The Java APIs Are Deliberately Maintained Separately From The Java Programming Language

The Java APIs are deliberately from the Java programming language. This was done very deliberately. They should not be regarded as part of the language.

One example of how the Java APIs are distinct from the Java programming language is that, while the Java programming language has changed very little since it was first released in 1996, the Java APIs have grown explosively. In 1996, there were only 7 API packages. There are now 209 API packages for Java Standard Edition ("SE") alone.

Since developers could program in Java from the time of its first release, it is obvious that these additional 202 APIs are not required to use the programming language. Many of these APIs contain highly specialized functions that would be neither expected nor, in many cases, desirable

in a general purpose programming language. This is true of most of the 37 API packages at issue in the case. Javax.net, for example, concerns a group of methods relating to creating a secure web connection. Javax.crypto, javax.crypto.interfaces, and javax.crypto.spec all relate to methods of encrypting and decrypting data. Java.text is an API package containing methods to help make a computer program usable in multiple natural languages.

The Java APIs are kept separate from the programming language for good reasons. When a programming language is created no one can predict all the ways it will be used. No one can foresee all the APIs that will be needed, and it is a mistake to build too much into the language. For example, there is no reason to build a database API into a general-purpose language like Java. As database technology improves and evolves, developers can create new APIs as needed, but the language should not also have to evolve. The C programming language is one of the most powerful and widely used languages, and it is still recognizable to programmers who used it in the 1970's, even though the uses to which the language is put have changed dramatically.

If a particular API were part of the language, then every change or addition to the API would have a ripple effect through everyone who uses or depends on the language, and would be required to implement the new features. It is for this reason that the Java language has changed only three times since it was first released and the process for changing the language is extremely deliberate and slow. Changes to the language require a super-majority vote of the entire Executive Committee of the Java Community Process and have only occurred after a lengthy public approval process. Changes to API's can be made much more quickly.

Another reason why the Java APIs are kept separate from the programming language is that Java is used to write programs for devices of very different capabilities, from powerful servers to embedded microcontrollers in single chips. Trying to implement all API elements as part of the Java language would require a heavyweight Java Virtual Machine capable of implementing every part of their functionality regardless of the device. In addition, the same API may provide different functions on different device form factors, and making every API part of the base language would make this impossible. The solution instead is to select, develop, and specify different sets of APIs suitable for different environments. This is exactly what Sun did in

the early days of Java when it divided Java into the language on the one hand and different collections of APIs on the other—Java SE (for desktops and servers), Java EE (for large scale enterprise applications), Java ME (for mobile and embedded devices).

Oracle may properly claim copyright protection over the Java APIs even if the Java programming language is freely available. The evidence at trial will show that copyright notices were prominently and consistently displayed on the API specifications. Hundreds of the world's largest companies pay to license the Java platform. Google was well aware of the requirement to license the specifications. Andy Rubin took a license to the Java specifications when he was at his predecessor company, Danger. Sun expressly rejected the notion that Danger did not require a license because Rubin had created an independent implementation, and therefore Danger paid. Rubin wrote in an internal email at Google and confirmed in his deposition that he was aware that Sun claimed copyright protection for the APIs. (TX0018; 7/27/11 Rubin Dep. 149:18-150:13.)

C. The Existence Of Java APIs And Class Libraries Besides The APIs At Issue Shows That APIs Are Not Inherently Part Of The Language And Why Google's Copyrightability Position Is Incorrect

The evidence will also show that there are many Java-language APIs and class libraries, available from a wide variety of sources, that are also not considered part of the Java programming language.

A good example is Android itself. Android has many APIs besides the 37 packages that give rise to copyright liability in this case. Many of them have the same purpose as some of the Java SE APIs Google did ***not*** copy.[1] Google provides compatibility test suites for its licensees to confirm that they are compatible with its APIs using those test suites. But this does not mean that any given Android API is part of the Java programming language. Oracle's Java APIs should not be analyzed differently.

Beyond Android, the world is full of Java APIs and libraries. None are considered to be part of the Java language either. The website freecode.com lists over 1,100 different Java

[1] This shows the Java APIs are expression, not ideas, since there are corresponding APIs that implement the same idea (*e.g.*, a library to handle screen displays and user interactions) but have different designs.

libraries. *See* http://freecode.com/tags/java-libraries. Many third parties provide class libraries and APIs for specialized Java applications like financial trading. For example, competitors IBM and BEA jointly developed a series of API specifications to create a common data programming environment for their competing server products. And scores of other companies create their own libraries and APIs internally so they can re-use code.

Google's position is that ***none*** of these APIs could ever be copyrighted because "APIs are not copyrightable, regardless of the programming language." (*See* ECF No.778 at 6-7 ("all API specifications, by design, describe precisely the elements of APIs that are needed for compatibility between implementations of the APIs, and with programs that use the APIs").) This does not make sense or comport with the law. The class libraries are copyrightable as a computer program, and their selection, arrangement and structure is copyrightable if sufficiently original and creative. The written description of those class libraries is copyrightable as well.

IV. HOW APIS ARE VIEWED IN OTHER LANGUAGES

The class libraries and APIs in other computer languages are generally not viewed as part of the programming language, even when a core set of APIs is specified in the same document as the programming language specification. The size and richness of these libraries varies widely. For example, C++, an object-oriented programming language similar in some ways to Java, was associated with a much smaller set of libraries. As a result, several different entities have created more comprehensive libraries and APIs, including the Standard Template Library and the Boost C++ Libraries. The Perl programming language also comes with a small set of libraries, called subroutine modules. However, over 100,000 additional modules are available separately in the CPAN Archive. *See* http://www.cpan.org/index.html. The Python software platform took the opposite approach and comes with an extensive set of libraries even larger than Java's, yet the Python documentation is clearly divided into separate descriptions of the language and the libraries. Even with such a large set of built-in libraries, over 20,000 additional packages are available at the Python Package Index. http://pypi.python.org/pypi.

Dated: April 12, 2012

MORRISON & FOERSTER LLP

By: _/s/ Michael A. Jacobs

Attorneys for Plaintiff
ORACLE AMERICA, INC.