No. 2013-1021, -1022

IN THE

United States Court of Appeals for the Federal Circuit

ORACLE AMERICA, INC.,

Plaintiff-Appellant,

v.

GOOGLE INC.,

Defendant-Cross Appellant.

ON APPEAL FROM THE UNITED STATES DISTRICT COURT FOR THE NORTHERN DISTRICT OF CALIFORNIA,
IN CASE NO. C 10-10356
JUDGE WILLIAM ALSUP

BRIEF FOR BSA | THE SOFTWARE ALLIANCE
AS *AMICUS CURIAE* IN SUPPORT OF
PLAINTIFF-APPELLEE ORACLE AMERICA, INC.

Paul M. Smith
Matthew S. Hellman
JENNER & BLOCK LLP
1099 New York Avenue, NW
Washington, DC 20001
(202) 639-6000

Counsel for Amicus BSA | The Software Alliance

Date: February 19, 2013

CERTIFICATE OF INTEREST

Pursuant to Federal Circuit Rule 47.4, Matthew S. Hellman, counsel for *Amicus Curiae* BSA | The Software Alliance, certifies the following:

1. The full name of the party represented by me is BSA | The Software Alliance.

2. The name of the real party in interest represented by me is BSA | The Software Alliance.

3. BSA | The Software Alliance is not a subsidiary of any corporation and BSA | The Software Alliance has issued no stock.

4. The names of all law firms and the partners or associates that appeared for the party now represented by me in this proceeding are: Jenner & Block LLP; Paul M. Smith and Matthew S. Hellman.

February 19, 2013

/s/ Matthew. S. Hellman
Matthew S. Hellman
Attorney for *Amicus Curiae*
BSA | The Software Alliance

TABLE OF CONTENTS

TABLE OF AUTHORITIES

OTHER AUTHORITIES

STATEMENT OF INTEREST

BSA | The Software Alliance ("BSA") is an association of the world's leading software and hardware technology companies.[1] On behalf of its members, BSA promotes policies that foster innovation, growth, and a competitive marketplace for commercial software and related technologies. BSA is uniquely situated to aid the Court in this case because its members include the leading developers of software. And because copyright policy is vitally important to promoting the innovation that has kept the United States at the forefront of software development, BSA members have a strong stake in the proper functioning of the U.S. copyright system.

Under the district court's understanding of copyright law, and contrary to the intent of Congress and over 30 years of jurisprudence, copyright protection would be withheld if a computer program contained functional aspects, and fair use defenses would be grafted onto a court's copyrightability determinations. Under such a reading of the Copyright Act, software developers would be subject to substantial

[1]The parties have consented to the filing of this brief pursuant to Fed. R. App. P. 29(a).

uncertainty, creating disincentives to develop new software and hampering innovation.

The members of the BSA include Adobe, Apple, Autodesk, AVEVA, Bentley Systems, CA Technologies, CNC/Mastercam, Dell, Intel, McAfee, Microsoft, Minitab, Oracle, Parametric Technology Corporation, Progress Software, Quest Software, Rosetta Stone, Siemens PLM Software, Symantec, TechSmith, and The MathWorks.

SUMMARY OF ARGUMENT

The Copyright Act seeks to balance "the interests of authors . . . in the control and exploitation of their writings . . . [with] society's competing interests in the free flow of ideas, [and] information." *Sony Corp. of Am. v. Universal City Studios, Inc.*, 464 U.S. 417, 429 (1984). In the more than three decades that have passed since Congress enacted copyright protection for works of software in 1980, the courts have reached a broad consensus that this balance is best achieved by imposing a low threshold for the copyrightability of software. As with other works of authorship like books and music, software is entitled to the protection of copyright so long as it is expressive and original. *E.g., Atari Games Corp. v. Nintendo of Am., Inc.*, 975 F.2d 832, 838 (Fed. Cir.

1992) (explaining that "copyright protection extends to computer programs" just as it does to any other expressive work).

The incredible growth of the software industry since 1980 testifies to the wisdom of this approach. In 2007, the software industry contributed more than $260 billion to the gross domestic product of the United States. *See* http://ww2.bsa.org/GlobalHome.aspx. Software companies and related services employ approximately 2,000,000 U.S. workers, paying salaries that are roughly 195% of the national average. Testimony of Robert W. Holleyman II, BSA President and CEO, Made in America: Increasing Jobs Through Exports and Trade: Hearing before the Subcommittee on Commerce, Manufacturing and Trade of the House Committee on Energy and Commerce, at 2 (Mar. 16, 2011), http://tiny.cc/p3nlow. And software accounts for approximately $36 billion of U.S. exports, with leading software companies deriving as much as 60% of their revenue from overseas sales. *Id.*

Moreover, this growth has been characterized by constant innovation as the industry produces new and better software for the public each year. In 1980, much software focused on improving business productivity, such as word processors and spreadsheets, and those types of programs have improved immeasurably in the interim. Likewise, there are today many kinds of software that are considered essential – from consumer products such as web browsers, photo editors, and audio players, to commercial products such as e-commerce applications, computer-assisted design tools, and electronic circuit simulators – that did not exist in any form thirty years ago. Software is now indispensable to the operation of financial services, health care, schools at every level from pre-school to post-graduate institutions, as well as for everyday appliances like refrigerators and air-conditioners and automobiles, airplanes and industrial machines. In short, software is today a ubiquitous element of our every-day lives.

But this growth and innovation cannot continue in the absence of clear rules protecting software as copyrighted works. Copyright law must ensure that creativity is rewarded and that intellectual property cannot be misappropriated by those unwilling to pay for the use of another's originality. If developers cannot receive protection for the works they create – or are uncertain about their prospects for protection – then their incentive to create will be removed or reduced, depriving the public and the economy of the benefits of a vibrant software industry.

The decision of the district court in this case threatens to disrupt the well-settled law that has fostered this growth. The particular subject matter here is the copyrightability of Oracle's Java API,[2] but the district court's decision suggests a limited and rigid view of software copyrightability that could have ramifications for all software. The decision thus implicates a major sector of the economy, as well as the benefits we all derive from innovative software at work, at home, and at school.

The threshold for software copyrightability has long been understood to be a low one. *See infra* Part I. Yet the district court's decision misapplied numerous doctrines to create novel obstacles to the copyrightability of software. *See infra* Part II.A. For example, although the district court acknowledged that the API embodied creative

[2]As Oracle explains in its brief, the term API is something of a verbal chameleon, meaning different things at different times. The BSA agrees with Oracle that the subject matter Google copied was the "vast array of Java programs to perform often-needed functions," and the "intricate hierarchy" in which those programs were organized. Oracle Br. 8. For simplicity, this brief will refer to the foregoing as the "Java API" or "API."

expression, it seemed to hold that because the software was also functional in nature, it could not be copyrighted. But of course all software contains functional aspects, and that has never by itself been, nor should it be, a bar to copyrightability. BSA urges this Court to reject the district court's flawed analysis and reaffirm long-settled law holding that software is broadly copyrightable.

The district court also misread the law when it held that Google's asserted goal of interoperability meant that the API was not copyrightable. *See infra* Part II.B. In the first place, this reading of the law and jurisprudence conflates a question of copyrightability with one of infringement. Whether the API is copyrightable depends on whether it contains original expression, not whether there is some justification for allowing another to copy it to further goals such as interoperability.

Nor can Google's actions in this case be justified on the basis of interoperability in any event. *See infra* Part III. BSA's members recognize that interoperability – compatibility between computer product offerings – is an important consideration, but this case does not involve interoperability as that concept is properly understood. The record shows that Google was not trying to make Android interoperate with Java; rather it was directly incorporating copies of the API to make its own competing software more attractive to developers. In short, Android was seeking to replace the Java API programs for mobile devices, not connect to them. That rationale, were it accepted, would severely undermine the protection that copyright affords software. Accordingly, BSA respectfully asks this Court to reverse the decision below.

ARGUMENT

I. Copyright Protection Has Been Broadly Available For Computer Programs For More Than Thirty Years.

It has been well-settled for more than 30 years that there is a relatively low threshold for software copyrightability. That understanding flows from the 1980 amendment to the Copyright Act, in which Congress formally extended copyright protection to software without imposing special conditions on copyrightability. And that broad conception of copyright has been reaffirmed by decades of subsequent case law recognizing that software is copyrightable so long as it contains original expression, subject to generally applicable principles limiting the scope of copyright protection. *See*, *e.g.*, 17 U.S.C. § 102(b) (idea-expression dichotomy); *Soc'y of Holy Transfiguration Monastery,* *Inc. v. Gregory*, 689 F.3d 29, 51-52 (1st Cir. 2012) (words and short phrases are generally excluded from copyright protection), cert. denied (U.S. Nov. Feb. 19, 2013) (No. 12-7513); *Satava v. Lowry*, 323 F.3d 805, 812 & n.5 (9th Cir. 2003) (merger doctrine). *See generally* William F. Patry, *Copyright and Computer Programs: It's All in the Definition*, 14 Cardozo Arts & Ent. L. J. 1, 22-32 (1996).

A. Congress Did Not Impose A High Bar For Software Copyrightability.

In 1974, Congress established the National Commission on New Technological Uses of Copyrighted Works (CONTU) for the purpose of studying and compiling data on, *inter alia*, copyright protection for computer programs. *See* Pub. L. No. 93-573, § 201, 88 Stat. 1873 (1974); Final Report of the National Commission on New Technological Uses of Copyrighted Works 9 (1978) [hereinafter "CONTU Report"]. On July 31, 1978, CONTU issued its final report, and the recommendations contained within it formed much of the basis for the 1980 amendments to the Copyright Act that formally included computer software as copyrightable material.

In its discussion about the copyrightability of computer software, the CONTU Report repeatedly emphasized that software, like any other work of authorship should be protected under the Act if it is original. Its formal recommendation on the subject advocated amending the 1976 Copyright Act "to make it explicit that computer programs, to the extent that they embody an author's original creation, are proper subject matter of copyright." *Id.* at 1. In making this recommendation, CONTU rejected the approach of Commissioner Hersey, who would not have given copyright protection to "a computer program in the form in which it is capable of being used to control computer operations." *Id.* Moreover, the CONTU Report eschewed the approach advocated by Commissioner (and Professor) Nimmer, who believed computer programs could be copyrighted "only when their use leads to copyrighted output." *Id.* at 21.

Rather, CONTU embraced the rule of § 102: a copyrightable work is an "original work[] of authorship fixed in any tangible medium of expression." 17 U.S.C. § 102(a). As such the Commission rejected calls to deny a computer program copyright protection simply because it possessed "utilitarian" aspects or because "the words of a program are used ultimately in the implementation of a process." CONTU Report at 21. Instead, "all that is needed to satisfy both the Constitution and the statute is that the 'author' contributed something more than a 'merely trivial' variation, something recognizably 'his own.'" *Id.* at 25.

The policy choice recommended by CONTU – that the barriers to copyright protection be minimal for computer software – was codified by Congress in the 1980 amendments to the Copyright Act. Congress expressly added computer programs, defined as "a set of statements or instructions to be used directly or indirectly in a computer in order to bring about a certain result," as copyrightable subject matter. *See* 17 U.S.C. §101; Act of Dec. 12, 1980, Pub. L. No. 96-517, § 10, 94 Stat. 3015, 3028. In addition, Congress through section 117, made clear that "it is not an infringement for the owner of a copy of a computer program to make or authorize the making of another copy or adaptation of that computer program," when necessary to the "utilization of the computer program." 17 U.S.C. § 117. Thus, because this specific "utilization" or functional aspect of copying was excepted from the definition of "infringement," more general functional aspects of the program must properly be protectable *and* subject to a finding of infringement.

Otherwise there would be no need for this exception. This is entirely consistent with the definition of a protected computer program as set forth above – to directly or indirectly *produce a result*. *See* Patry *supra* at 36 ("Congress clearly indicated that computer programs are to be protected only if they *do* bring about 'a certain result' . . . ." (emphasis in original)).

In sum, computer programs are like any other copyrightable subject matter. If they are original, *see Feist Publications, Inc. v. Rural Telephone Service Co.*, 499 U.S. 340, 345 (1991) ("The *sine qua non* of copyright is originality."), they are entitled to full copyright protection (subject to generally-applicable principles such as § 102(b)).

B. Courts Have Found Computer Programs to be Broadly Copyrightable.

Following the course charted by Congress, courts have found computer software – including its various specific components – to be broadly copyrightable. A common theme in these cases is their recognition that although software serves a functional purpose, it is nonetheless eligible for protection under the Copyright Act if it is an original work of authorship. As one early court put it in rejecting the argument that operating system instructions were uncopyrightable because they carried out a function: "a 'process' is no more involved because the instructions in an operating system program may be used to activate the operation of the computer than it would be if the instructions were written in ordinary English in a manual which described the necessary steps to activate an intricate complicated machine." *Apple Computer Inc. v. Franklin Computer Corp.*, 714 F.2d 1240, 1251 (3d Cir. 1983), *superseded by statute as stated in Brooktree Corp. v. Advanced Micro Devices*, 977 F.2d 1555 (Fed. Cir. 1992); *see M. Kramer Mfg. Co. v. Andrews*, 783 F.2d 421, 435 (4th Cir. 1986) (same); *see also Williams Elecs. Inc. v. Artic Int'l Inc.*, 685 F.2d 870, 877 (3d Cir. 1982) (rejecting argument that "object code" was uncopyrightable simply because it could be understood only by a computer and explaining that 17 U.S.C. § 101 should not be interpreted "in a manner which would severely limit the copyrightability of computer programs which Congress clearly intended to protect"); *Apple Computer, Inc. v. Formula Int'l Inc.*, 725 F.2d 521, 523 (9th Cir. 1984), *overruled on other grounds by Perfect 10 Inc. v. Google Inc.*, 654 F.3d 976 (9th Cir. 2011) (rejecting a narrow view of copyright protection that would exclude programs "because they control the internal operation of the computer").

Moreover, as programs became more complex and courts grew to better understand the various constituent parts of programs, copyright protection for software has been interpreted consistently by courts to apply to both the software itself as well as to structural elements of the software that embodied creative expression. This Court recognized this point applying Ninth Circuit law (also controlling here) in its seminal decision in *Atari Games Corp. v. Nintendo of America*, 975 F.2d 832 (Fed. Cir. 1992). In that case, Nintendo contended that a program created by Atari, which "unlocked" Nintendo's video game machine, infringed Nintendo's copyright. This court observed that "[b]ecause Atari chose a different microprocessor and programming language, the line-by-line instructions of the [Nintendo] and [Atari] programs vary." *Id.* at 836. The court nonetheless concluded that Atari had infringed Nintendo's copyright because Nintendo "exercised creativity in the selection and arrangement of its instruction lines." *Id.* at 840. Another leading decision, *Computer Associates International, Inc. v. Altai, Inc.,* 982 F.2d 693, 706 (2d Cir. 1992), held that the "structural components" at higher "level[s] of abstraction" from the code itself are entitled to copyright protection provided that they reflected expressive elements. *Id.* at 707; *see also Hutchins v. Zoll Med. Corp.*, 492 F.3d 1377, 1383 (Fed. Cir. 2007) (copyright is not necessarily limited to underlying code but can also extend to "design and text, as well as the tangible expressions such as the screen display"); *Gen. Universal Sys., Inc. v. Lee*, 379 F.3d 131, 142 (5th Cir. 2004) (copyright protection "extends not only to the 'literal' elements of computer software – the source code and object code – but also to a program's nonliteral elements, including its structure, sequence, organization, user interface, screen displays, and menu structures" (footnote omitted)).

II. The District Court's Decision Misapplied These Well-Settled Legal Principles.

Against the backdrop of this settled law, the district court in this case charted a very different course. The court's holding that the Java API was not copyrightable rests upon a substantially narrower view of the scope of software copyright, and one that is not reconcilable with the jurisprudence discussed above. The district court's analysis of the law was erroneous in three key ways: in its treatment of functional aspects in software as precluding copyrightabilty, 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.

Moreover, the district court also departed from settled law when it held that considerations of interoperability made the API uncopyrightable. As we explain, interoperability – making certain elements of two programs work together – has arisen in all instances in the case law as a question of *infringement*, not of copyrightability as an initial matter. The district court thus wrongly looked to interoperability as an issue of copyrightability

A. The District Court's Analysis Imposed Too High A Bar For Software Copyrightability.

1. Software Does Not Lose Copyright Protection Simply Because It Has Functional Aspects.

Software by its very nature is functional—that is, all software does something, as the very definition of "computer program" in the Copyright Act makes clear. *See* 17 U.S.C. § 101 (defining a computer program as "statements or instructions to be used . . . *to bring about a certain result*") (emphasis added). But at the same time software can also contain original expressive elements. And as discussed above, so long as the functional aspects of the software are expressed in an original work, they are entitled to copyright protection.

In the decision below, the district court appeared to ignore one half of that equation, suggesting that although the API had expressive elements, it was nevertheless not copyrightable because it also had functional ones. That approach runs counter to the broad swath of authority discussed above, and would defy the text of the Copyright Act by placing a great many – perhaps most – programs outside the scope of copyright protection. It should be rejected by this Court.

The district court did not dispute that the Java API was expressive. To the contrary, the court found that it was "creative," "original," and "resemble[d] a taxonomy," and that "nothing in the rules of the Java language . . . required that Google replicate the same groupings." *Oracle Am., Inc. v. Google Inc.*, 872 F. Supp. 2d 974, 999 (N.D. Cal. 2012). That finding was grounds enough to determine copyrightability (an original work of authorship) and the court should have then moved onto questions of infringement. Yet the court ignored its own conclusions about originality and creativity and nevertheless held that Oracle had created an uncopyrightable "command structure" because it was a "long hierarchy of over six thousand commands to carry out pre-assigned functions." *Id.* at 999-1000. And at another point the court rejected "Oracle's analogy to stealing the plot and character from a movie" on the ground that "movies involve no 'system' or 'method of operation' – scripts are *entirely creative*." *Id.* at 1001 (emphasis added). And the court further seemed to confuse matters by suggesting that because the functional aspects of the software could receive patent protection, the expressive aspects of the software could not also be protected by copyright. *Id.* at 998 (regardless of creativity, "such inventions – at the concept and functionality level – are protectable only under the Patent Act").

These broad statements do not reflect settled law in this area. The mere fact that software (or any other expressive work) can be patented does not preclude it also receiving protection under copyright law. *Mazer v. Stein*, 347 U.S. 201, 217 (1954) ("Neither the Copyright Statute nor any other says that because a thing is patentable it may not be copyrighted."). And while the district court did not actually go so far as to hold that functional or structural aspects render software uncopyrightable, the court's assertion that the Java API, although creative, was not like a creative movie script does not leave much space for software that has utility yet *also* possesses the requisite level of creativity.

In contrast to the analysis of the district court, the proper approach set out by the case law is to recognize the low threshold for copyrightability for computer programs, and *then* engage in a careful and fact-specific analysis in order to set the appropriate metes and bounds of the law's protection for that specific copyrighted program. A good example of the correct approach is this Court's analysis in *Atari*, where the court found Nintendo's 10NES program (which allowed game cartridges to be played on the NES game console) was copyrightable subject matter despite the presence of functional aspects. Conducting a careful analysis, which looked at the extent of creativity in the design of the program and whether external factors or elements from the public domain had played a role in its development, the court concluded, "Nintendo may protect [the] creative element of the 10NES under copyright." *Atari*, 975 F.2d at 840. In stark contrast, the district court here did not appear to apply any recognized test and instead focused on the mere presence of functionality of the API to the exclusion of any other of its characteristics.[3] This is far afield from the way this Court, or any other court, has interpreted the scope of copyright protection in computer software.

2. The District Court Construed The § 102(b) Method Exclusion And Merger Doctrine Overly Broadly.

The district departed from settled law in its suggestion that the Java API was an uncopyrightable "method of operation" under § 102(b), and alternatively was barred by the merger doctrine.

Beginning with the § 102(b) method exclusion, that provision embodies a distinction first discussed in *Baker v. Selden*, 101 U.S. 99, 103 (1879). That case explained that a method of operation – for

[3]Another example of the district court's erroneous approach is the significance it gave to the fact that "the API at issue here [comes] with instructions for use," *Oracle,* 872 F. Supp. 2d at 1001. In making this point, the court cited *American Dental Ass'n v. Delta Dental Plans Ass'n*, 126 F.3d 977 (7th Cir. 1997), a case that had nothing to do with computer software. In that case, the Seventh Circuit found a taxonomy to be copyrightable, and in doing so noted that the taxonomy did not come with instructions. *Id.* at 980-981. Invoking *American Dental*, the court below found the presence of instructions in this case to be strongly suggestive that the API is not copyrightable. *Oracle*, 872 F. Supp. 2d at 1001. But most software comes with instructions of some sort, and it is error to contend that the mere presence of instructions indicates a functionality that weighs against copyrightability.

example a mathematical algorithm – is not copyrightable even if a treatise describing the algorithm would be. Today, the exception is codified at § 102(b) of the Copyright Act, and states "[i]n no case does copyright protection for an original work of authorship extend to any idea, procedure, process, system, [or] method of operation." 17 U.S.C. § 102(b).

In the context of software, which always at some level embodies a method for accomplishing tasks, the method exception has been drawn narrowly and carefully. For example, in *Apple Computer*, the court rejected the defendant's argument that the code constituting an operating system was *per se* uncopyrightable because an operating system is a "method of operation" for a computer. *See Apple Computer, Inc.*, 714 F.2d at 1250. The court distinguished between "the method which instructs the computer to perform its operating functions" and "the instructions themselves" and determined the latter were copyrightable. *See also Mitel, Inc. v. Iqtel, Inc.*, 124 F.3d 1366, 1372 (10th Cir. 1997) ("Section 102(b) does not extinguish the protection accorded a particular expression of an idea merely because that expression is embodied in a method of operation at a higher level of abstraction.”).

Here, the court defined the API as “a long hierarchy of over six thousand commands to carry out pre-assigned functions,” and then determined it was an uncopyrightable method of operation. *Oracle*, 872 F. Supp. 2d at 1000. That analysis expands the method exception beyond recognition. The *concept* of an API, or for that matter the *idea* that a computer program has a structure and internal organization, is not copyrightable and Oracle never argued as much. But the *creative allocation* of thousands of lines of code into a complex structure and the determination of which specific source code terms to use to express the specifics of that structure is not a method of operating the API, even if the API as a whole accomplishes functions.

The Java API thus is far more expressive than say, the “computer menu command hierarchy” that the First Circuit found to be an unprotected method in *Lotus Development Corp. v. Borland International, Inc.*, 49 F.3d 807, 815 (1st Cir. 1995), *aff'd* 516 U.S. 233 (1996). In that case, the court addressed whether the series of menus and subsidiary commands in Lotus 1-2-3 was copyrightable. The court found the menu command hierarchy fell within the § 102(b) exception because it "provide[d] the means by which users control and operate Lotus 1-2-3." *Id.* As the court described it, a method of operation "refers to the means by which a person operates something, whether it be a car, a food processor, or a computer." *Id.* Thus, because without the command terms "users would *not be able* to access and control … Lotus 1-2-3's functional capabilities," the court found they constituted a method of operation. *Id.* (emphasis added).

Key to this definition is that the component of the program be the only means through which the program can be operated. Conversely, the district court acknowledged that Google could have structured Android differently and chosen different ways to express the functionality and that the portions of the Java API copied by Google reflected a set of fundamentally creative choices, hardly the predetermined and only way to express the ideas in the program. This is the essence of what makes a work original and that should qualify for copyright protection. The district court's belief that it was a mere method should be rejected.[4]

These same considerations reveal the district court's error in its application of the merger doctrine. *See Oracle*, 872 F. Supp. 2d at 1001. As the district court correctly explained, "when there is only one way to write something, the merger doctrine bars anyone from claiming exclusive copyright ownership of that expression." *Id* at 998. But here, the district conceded that the Java API's packages *could* have been structured differently. *See id.* at 999 (acknowledging there was "nothing in the rules of the Java language that required that Google replicate the same groupings"). There were myriad ways in which the API's packages could have been organized, and it was error to hold that the merger doctrine deprived copyright protection to Oracle's particular choice of organization.

[4]While Google copied the 37 Java packages at issue in this case, Google showed that it was possible to make its own, original and creative expression in other areas of Android. There were numerous other areas where Google could have copied Java but choose not to, most notably the 131 packages it did not copy. Google could have done its own original and creative expression in the area of the 37 packages as well. That there was such choice and flexibility supports a finding that the subject Java API is original and copyrightable.

3. The District Court Misapplied The Words and Short Phrases Doctrine

The district court's discussion of the principle that words and short phrases are not copyrightable provides yet another example of how it construed too narrowly the scope of copyright protection for software.

Computer programs, like other literary works, consist of words and phrases. That does not make them uncopyrightable. The question is whether the words and phrases were assembled into an original work of authorship. When the relevant Copyright Office regulations (37 C.F.R. § 202.1) provide "[w]ords and short phrases" as an example of "works not subject to copyright," they do no more than recapitulate the basic principle that a *work* must be original to be protected by copyright. *See* 1 Melville B. Nimmer & David Nimmer, Nimmer on Copyright, § 2.01[B]. And as a result, the question for copyrightability is not whether a work uses short phrases, but whether those phrases exhibit creativity. *See Soc'y of Holy Transfiguration Monastery, Inc.*, 689 F.3d at 52 (the exception "very much turns on the specific short phrases at issue, as not all short phrases will automatically be deemed uncopyrightable"); *CMM Cable Rep, Inc. v. Ocean Coast Properties, Inc.*, 97 F.3d 1504, 1520 n.20 (1st Cir. 1996) ("Of course, we recognize that not all short, simple, declarative sentences fall within the meaning of [37 C.F.R. §202.1(a)]."); 1 Nimmer on Copyright §2.01[B], at 2-17 ("[A] short phrase may command copyright protection if it exhibits sufficient creativity.").

Here, however, the court deemed the length of the declarations copied by Google to be determinative of the copyrightability of the API, rather than look at the totality of Oracle's work. That approach is not faithful to the principles described above. The content of a declaration, including the name of the method and the method's location in the API, is not some pre-determined label, but is rather the result of the creative structuring of the API. And the API itself is not merely a collection of disconnected words but rather a complex and creative platform created by Sun's developers out of infinite possibilities. Oracle is not trying to protect a specific short phrase or word, but rather the larger collection that makes up the various packages and thousands of line of source code in the API that was copied. As in *Atari*, Oracle "exercised creativity in the selection and arrangement" of method names and packages when it created its API. 975 F.2d at 840; *see also* 17 U.S.C. §§ 101, 103 (recognizing copyright protection for compilations "that are selected, coordinated, or arranged in such a way that the resulting work as a whole constitutes an original work of authorship").

It was thus 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 and the expressive role the method names served within the API.

B. The Scope of Copyrightability Is Not Limited By A Desire For Interoperability.

The district court also misread the law when it invoked the concept of interoperability as a ground for denying copyrightability. According to the district court, "millions of lines of code had been written in Java before Android arrived," and "[i]n order for at least some of this code to run on Android, Google was required to provide the same java.package.Class.method() command system using the same names with the same 'taxonomy' and with the same functional specifications." *Oracle*, 872 F. Supp. 2d at 1000.

As an initial matter, and as discussed in more detail in Part III, the district court's conclusion that Google was "required" to copy and use the Java code is not supported by the evidence. It was certainly easier for Google to simply copy and use the code, but copyright law does not excuse such an act of expedience. But even if interoperability were at issue here, the district court erred in characterizing interoperability as an issue of *copyrightability*, rather than an issue of *infringement*. The district court should have assessed whether the Java API was copyrightable, and *then* assessed whether Google's purported desire for interoperability could be excused as a fair use. The reason for this is simple: the question the copyrightabilty inquiry asks is whether the program at issue is an original work of authorship or whether instead it was "dictated by considerations of efficiency, so as to be necessarily incidental to that idea; required by factors external to the program itself." *Altai*, 982 F.2d at 707, 721.

That question can and must be answered with respect to the work in question standing by itself, at the time it was created. Conversely, the idea of "interoperability" presupposes that there are two works – an original work, and a new work that must also meet the law's originality requirement on its own merits and that can interoperate with the first work. The question courts have had to decide in these circumstances is whether the second program contains infringing elements and whether the inclusion of such elements is excused by the fair use defense. Courts have also had to consider whether in the course of developing a second interoperable program, its developers engaged in acts, such as unauthorized reproduction of the original program, that constitute infringement, and whether those acts are excused.

The district court appeared to hold that because it was theoretically *possible* for some piece of a user-written Android code to be interoperable with the Java platform, the Java API was uncopyrightable, regardless of whether Android actually was interoperable with the Java platform. *See Oracle*, 872 F. Supp. 2d at 1000 (noting that Oracle's concern over "imperfect interoperability" served to "illustrate" that that the Java API was a "method of operation"). But this analysis is mistaken. The copyrightability of a computer program should not depend on whether some other program *might* interoperate with it, regardless of the particular actions of the infringer. Even if there were *actual* interoperability, not just *hypothetical* interoperability, interoperability is not a relevant basis to determine copyrightability.

The district court cited no case law in support of its proposition that purely theoretical interoperability renders a piece of software uncopyrightable. The district court characterized *Sega Enterprises Ltd. v. Accolade, Inc.*, 977 F.2d 1510 (9th Cir. 1992), and *Sony Computer Entertainment, Inc. v. Connectix Corp.*, 203 F.3d 596 (9th Cir. 2000), as "close analogies," *see Oracle*, 872 F. Supp. 2d at 1000, but they are not. Both were cases where infringement had been found and the issue before the court was whether the fair use defense should be available to the infringer. In fact, in *Sega* copyrightability was not even litigated, and in *Sony*, the Ninth Circuit held that the original software was copyrightable, but that the copying was excused under the fair use defense.

The district court focused on the *Sega* court's statement, in the context of its fair use analysis, that "functional requirements for compatibility with the Genesis console" were "aspects of Sega's programs that are not protected by copyright." *Oracle*, 872 F. Supp. 2d at 994 (quoting *Sega*, 977 F.2d at 1522). Furthermore, Accolade's sole contention on appeal was that it disassembled object code "in order to gain an understanding of the ideas and functional concepts embodied in the code." 977 F.2d at 1517-18. Here, Google did not merely "gain an understanding of the ideas and functional concepts embodied" in the Java API. It copied the entire underlying structure of 37 separate Java packages, which required the replication of thousands of lines of source code, solely for the purpose of making it easier for developers to write Android programs, not to make Android interoperate with Java. The Ninth Circuit has made clear that *Sega* does not extend to such copying. *Practice Mgmt. Info. Corp. v. AMA*, 121 F.3d 516, 520 n.8 (9th Cir. 1997) (holding *Sega* prohibits "the owners of copyrights in expressions mandated by industry standards to use their copyrights to stifle independent creative expression in the industry," but does not authorize "wholesale copying of an existing system"). *See also Sony*, 203 F.3d at 609-10.[5]

[5]Also in the *Sega* and *Sony* cases there was real interoperability, where the second work – the competing video game – would actually interoperate with the first company's game console. There is not such interoperability here and Google said as much in its public statements. See A2205, A21172, and A21179.

Finally, the district court's view that interoperability is a copyrightability issue would inject considerable uncertainty into copyright law. Under the district court's holding, as long as a sufficiently creative expert witness could state a *hypothetically* interoperable need, an original computer program could be stripped of copyright protection. Software developers would have no way of predicting in advance of litigation whether this would occur. The resultant uncertainty would create a disincentive to create new software and would be a drag on innovation.

***

For all the reasons stated above, BSA urges this Court to reaffirm the settled principles set forth in other cases governing the copyrightability of software. Today's software programs are extraordinarily complex and sophisticated, and the large role software companies have in the economy make it crucial that there is clarity as to their ability to protect computer programs that are original and creative works. Software companies have come to rely on case law that affirms a broad understanding of the copyrightability of software, both at the source code level and at higher levels of abstraction. Without a stable foundation of copyright protection for those programs, software developers will not expend the significant time and money necessary to create them.

III. Google's Use of Java's Code Did Not Have A Purpose of Achieving Interoperability And Thus Is Not Eligible for the Fair Use Defense.

Google has argued in the alternative that even if the API were copyrightable, its copying should be permitted under the fair use doctrine. BSA urges this Court to reject this argument because the settled case law on software copyright fair use required the defendant to be creating interoperability between computer programs, something Google was *not* trying to achieve.

BSA recognizes that interoperability between computer programs is in many instances desirable both from the perspective of developers and their customers. Operating systems work harmoniously with microprocessors; applications work harmoniously with operating systems; and different types of computers work harmoniously when interacting over the Internet.

To those ends *Sega* recognizes that fair use may excuse limited copying of computer programs to make new programs interoperable with existing software or hardware. But under that case law, the *sine qua non* is that the copying is both *intended to*, and *strictly necessary for*, achieving physical interoperability. *See Sega,* 977 F.2d at 1527 (fair use permits copying for interoperability when it is "the *only* way to gain access to the ideas and functional elements embodied in a copyrighted computer program"); *see also Atari*, 975 F.2d at 843 ("holding that [a]ny reproduction of protectable expression must be strictly necessary to ascertain the bounds of protected information within the work"). Neither *Sega* nor any subsequent court has suggested that mere convenience to a programmer familiar with a particular software program was a sufficient justification for fair use.

Yet that is what Google sought to do here. Google was not trying to make Android interoperable with Java (or vice versa), it was instead trying to make it easier and more convenient for developers skilled in Java to write Android programs. Google made no showing that copying 37 out of 166 packages was narrowly tailored to any purpose of ensuring interoperability. Rather, such copying and use "as part of the ordinary operation of" its software would not qualify as "fair use" under the Sega analysis. *See DSC Commc'ns Corp. v. Pulse Commc'ns, Inc.*, 170 F.3d 1354, 1363 (Fed. Cir. 1999). Authorizing fair use of Google's direct copies rather than only copies made as an intermediate step to ascertain the bounds of protected information within the work would destabilize settled practices in the software industry.

CONCLUSION

For the foregoing reasons, BSA respectfully urges this Court to reverse the decision of the district court. The copyright regime envisioned by the district court would undermine the protections for innovation – and the corresponding incentives to innovate – that have served the public, the economy, and the software industry so well over the past 30 years. This Court should reaffirm that the bar to software copyrightability is a low one, and that interoperability cannot justify infringement for the sake of convenience. The judgment of the district court should be reversed.

Respectfully submitted,

/s/ Matthew S. Hellman
Paul M. Smith
Matthew S. Hellman
JENNER & BLOCK LLP
1099 New York Avenue, NW
Washington, DC 20001
(202) 639-6000

Date: February 19, 2013

CERTIFICATE OF SERVICE

I hereby certify that on February 19, 2013, I caused the foregoing Brief for BSA | The Software Alliance as *amicus curiae* in Support of Plaintiff-Appellee Oracle America, Inc., to be electronically filed with the Clerk of the Court using CM/ECF, which will send notification to all registered users.

Dated: February 19, 2013

Respectfully submitted,

/s/ Matthew S. Hellman
Attorney for *Amicus Curiae*
BSA | The Software Alliance

CERTIFICATE OF COMPLIANCE UNDER FEDERAL RULES OF APPELLATE PROCEDURE 32(a)(7) AND FEDERAL CIRCUIT RULE 32

Counsel for *amicus curiae* BSA | The Software Alliance certifies that the brief contained herein has a proportionally spaced 14-point typeface, and contains 6,622 words, based on the "Word Count" feature of Word 2007, including footnotes. Pursuant to Federal Rule of Appellate Procedure 32(a)(7)(B)(iii) and Federal Circuit Rule 32(b), this word count does not include the words contained in the Certificate of Interest, Table of Contents, and Table of Authorities.

Dated: February 19, 2013

Respectfully submitted,

/s/ Matthew S. Hellman
Attorney for *Amicus Curiae*
BSA | The Software Alliance