No. 18-956

In the

Supreme Court of the United States

GOOGLE LLC,

Petitioner,

vs.

ORACLE AMERICA, INC.,

Respondent.

ON A WRIT OF CERTIORARI TO THE UNITED STATES COURT OF APPEALS FOR THE FEDERAL CIRCUIT

BRIEF OF *AMICI CURIAE* INDUSTRY EXECUTIVES JOSEPH M. TUCCI AND PAUL T. DACIER IN SUPPORT OF RESPONDENT

DAVID W. SHAPIRO
Counsel of Record
THE NORTON LAW FIRM P.C.
FRED NORTON
BREE HANN
Attorneys for Amici Curiae
299 Third Street, Suite 106
Oakland, California 94607
(510) 906-4900
dshapiro@nortonlaw.com

Table of Contents

Table of Authorities

I. Interest of Amicus Curiae[1]

Joseph M. Tucci and Paul T. Dacier have between them over eighty years of experience as leading software and technology industry executives.

After starting his career as a programmer, Mr. Tucci served as a senior executive at Unisys and as chairman and CEO at Wang Laboratories, then joined industry giant EMC Corp., where he served as President, CEO, and Chairman. Prior to its acquisition by Dell in 2016, EMC had become the world's largest provider of storage systems, software, and solutions to help customers store, manage, protect, and analyze information, with $25 billion in annual revenues and over 70,000 employees around the globe. Mr. Tucci has also been a member of the President's Council of Advisors on Science and Technology, the Technology CEO Council, and the Business Roundtable, where he has advocated for policies to support and enhance U.S. competitiveness and innovation.

Mr. Dacier entered the computer technology industry in 1984 as an attorney at Apollo Computer Inc.; in 1990, he joined EMC as its sole in-house attorney. Mr. Dacier ultimately rose to General Counsel as EMC grew into one of the software and technology industry's leading companies. While at

[1] Pursuant to Supreme Court Rule 37.6, no counsel for a party authored this brief in whole or in part or made a monetary contribution intended to fund the preparation or submission of the brief, and no person other than amicus curiae or their counsel made such a monetary contribution. All parties have provided general written consent to the filing of this brief pursuant to Supreme Court Rule 37.3(a).

EMC, Mr. Dacier established a reputation as a fierce defender of EMC's intellectual property portfolio against both infringers and patent assertion entities. Mr. Dacier is a past President of the Boston Bar Association and currently serves as Chair of the Judicial Nominating Commission for the Commonwealth of Massachusetts.

In their decades of leadership in the technology industry, Mr. Tucci and Mr. Dacier have played all parts: from tenacious innovators to established standard bearers. *Amici's* experience gives them insight into the real-world effects on the software industry of the copyrightability decision that the Court is asked to make in this case, including the likely consequences on innovation, success, and competition.

II. Summary of Argument

The United States software industry today is enjoying meteoric success and corresponding national significance – employing millions, investing billions, and creating trillions in value. Software creators are able to invest so heavily in people and products because they can rely on the robust system of copyright law to protect their software. With strong copyright protection assured, creators can and do confidently license their software to others – including, frequently, their competitors – and license software from other companies as well. Creators rely on licensing's flexibility, which allows careful, sure, and effective deployment, as well as its strength, which comes from the deterring effect of copyright liability exposure for any entity that goes beyond what the license is designed to permit.

Software companies have long relied on this predictable and sturdy foundation to justify their enormous investments and expenditure of effort in software creation. The Federal Circuit carefully and correctly applied the longstanding principles of copyright law on which *amici* and their companies have long relied. Google's arguments, if accepted by the Court, would immediately, and irrevocably, undermine that foundation, and consequently, the software industry's success.

Google makes two, largely overlapping arguments why the Federal Circuit's copyrightability decision should be reversed: first, that the Java SE declaring code is uncopyrightable because it is a "method of operation" or "system"; and second, that the declaring code is uncopyrightable under the merger doctrine. Neither argument has merit.

First, there is no dispute that software code generally is copyrightable under Section 102 of the Copyright Act. Google asserts, however, that the 11,330 lines of Java SE declaring code that it copied are an "entirely functional" "method of operation" and therefore are excluded from copyrightability under 17 U.S.C. § 102(b). But Google makes two mistakes: it misreads Section 102, which does not render an author's *expression* of a method of operation uncopyrightable, and it offers no way for courts or creators to distinguish between "functional" software code that Google concedes is protected by copyright from the "functional" software code that is not protected. Google has offered a rule that cannot be coherently applied – anathema to the software industry, which relies on clear and predictable software protection to justify its investments of time and money.

Second, Google is not truly making a merger doctrine argument. For the actual merger doctrine to bar a work's copyrightability, there must be only a few, very limited ways to express the idea. But here Google concedes that Oracle could have written the declaring code in countless different ways to achieve the same function. Google is instead suggesting something new: that computer programs that were originally protected by copyright lose that protection – not because they embody the sole means of expressing an idea, but because, after their creation, those programs have become the most popular and familiar way to express an idea. This rule, which has no support in the law, would perversely punish software creators whose products succeed the most. Should Google's version of the merger doctrine be adopted, software companies will no longer be incentivized to invest billions of dollars in software research and development, knowing that if they create something great, any competitor will be free to take it for themselves.

The Court should affirm the rulings of the Federal Circuit.

III. Argument

A. The United States Software Industry Is Robust Due To Strong Copyright Protection For Software.

To call the software industry "large" or "important" would be to engage in near-facetious understatement. That industry employs tens of millions of developers across the world; in the United States alone, the federal government estimates 1,365,500 software developer positions exist as of 2018.[2] One technology research group estimates the software industry has directly created 3.1 million total domestic jobs,[3] and it indirectly accounts for nearly *one in ten* United States jobs.[4] Those numbers will continue to swell, with software developer positions alone anticipated to grow 21 percent in the United States from 2018-2028.[5]

The industry does not just employ millions, but also invests billions of dollars in developing new programs and offerings. Software companies invested an estimated $82.7 billion in research and development in 2018 – over 22 percent of all domestic business research and development in the United States that year.[6] These investments, and customers' support

---

[2] *See* U.S. Bureau of Labor Statistics, https://www.bls.gov/ooh/computer-and-information-technology/software-developers.htm. *See also, e.g.*, https://www.idc.com/getdoc.jsp?containerId=US44363318 (International Data Corporation (IDC) estimates approximately 22.3 million software developers worldwide in 2018).

[3] *See* Software.org: BSA Foundation, https://software.org/reports/software-growing-us-jobs-and-the-gdp/?mod=article_inline (3.1 million jobs directly created by the software industry, rising to 14.4 million jobs when "including indirect and induced impacts").

[4] *See id.*

[5] *See* https://www.bls.gov/ooh/computer-and-information-technology/software-developers.htm (U.S. Bureau of Labor Statistics).

[6] See Software.org: BSA Foundation, https://software.org/reports/software-growing-us-jobs-and-the-gdp/?mod=article_inline.

and investment of their own, create proportionately large effects on the nation's economy. For example, one technology research group estimated that, in 2018, customer spending on enterprise software, and "the associated indirect impact of increased productivity and higher-paying jobs," added $1.6 trillion to the United States' gross domestic product.[7]

Those are numbers to make any observer sit up and take immediate notice of what the United States software industry must be doing right to be earning and enjoying such tremendous success. Evidently the software industry, and the rules, policies, and practices that envelop and support it, is working.

That continued success, however, relies on software companies' confidence that their most valuable assets – their software – will continue to be protected by copyright law, as has been true throughout the industry's growth and dominance. Software companies rely on being able to carefully and securely control development of their own copyrighted works, while at the same time using systems, platforms, and infrastructures built by others. They are able to accomplish these two necessary, but occasionally conflicting, goals through licensing. And successful licensing, in turn, is best encouraged by clear,

[7] "Corporate Tech Spending Helps Lift U.S. Economy," *The Wall Street Journal*, September 24, 2019, available at https://www.wsj.com/articles/corporate-tech-spending-helps-lift-u-s-economy-11569367000. *See also* https://software.org/reports/software-growing-us-jobs-and-the-gdp/?mod=article_inline (Software.org: BSA Foundation estimates total value-added GDP in 2018 from software purchases is $1.6 trillion).

consistent, and settled rules for software copyright protection.

As *amici* know well from their own experience, copyright protection for software code establishes the foundation on which the software and technology industry has built extensive and widely beneficial licensing ecosystems, and which are the necessary predicate for investment. During their tenure at EMC, the company spent over $10 billion to acquire over forty software companies, including Data Domain, VMWare, RSA Security, Documentum, and Virtustream. These acquisitions were part of a purposeful strategy to keep pace with the evolving technology industry and transform EMC from a hardware company to a company with market leading capabilities in software, especially storage and big data analytics. *Amici* and EMC were able to identify software innovators and acquire them because *amici* had confidence that existing copyright law would protect the software assets of those multi-billion dollar acquisitions.

Creators can invest the time and considerable resources necessary to develop software, knowing they can recoup that investment by licensing the resulting software to other developers or to end users – and those licenses will have actual consequences for violators. Even more importantly, when software code enjoys broad copyright protection, creators understand they will be able to effectively pursue infringers to enforce their rights. Even Google *amicus* Microsoft admits the proprietary "model still serves an important role in the computer industry." Microsoft Br. at 6. Like Google, and like the technology companies *amici* have led, Microsoft "relies on copyright protection to develop and recover its investment in its products and services." *Id.* at 2.

But software companies understand that they will often be on the other side of the copyright license, as the licensee. Many, and likely most, software customers want each product they purchase to interoperate effectively with other vendors' products. That means software creators must be able to work with other vendors – often, though not always, their competitors – to achieve interoperability between software. In that role, software creators appreciate opportunities for legitimate reverse engineering, competitive analysis, and innovative follow-on development of existing software. Here, too, licensing solves the problem. With the security of copyright protection, creators can carefully and securely control deployment of their own copyrighted works, while at the same time use systems, platforms, infrastructures, and solutions built from connectable offerings provided by multiple vendors.

Licensing achieves both this control and this flexibility through the deterring effect of copyright liability exposure for any entity that goes beyond what the license is designed to permit. In other words, a software creator can be comfortable giving others access to its software – in which the creator has invested countless hours and dollars – through licenses, because it knows it can rely on copyright remedies to make the creator whole should the licensee transgress. Robust copyright protection for software allows its creators the option to license for monetary remuneration, or cross licensing opportunities. Indeed, open source software licenses rely on the idea that what is being licensed is protected by copyright law to ensure that the licensors' interests as authors remain protected.

Without the assurance of copyright protection, in contrast, software companies would be forced to retrench. They would disclose information about their computer programs to users and developers only in more restrictive and costly ways, or in some cases not at all. And of course, without the assurance that investments in software innovation will be protected, some companies will choose not to make those investments in the first place. Diminished confidence in the reliability of copyright protection would strike a devastating blow to innovation in the software industry, cross-industry collaboration, and ultimately to downstream consumers.

As the numbers and ever-increasing success show, the system is *working*. Accepting Google's invitation to upend that system by eliminating copyright protection for creative and original computer software code would not make the system better – it would instead have sweeping and harmful effects throughout the software industry.

B. Google's All-Or-Nothing Approach To Copyrightability And Merger Is Inconsistent With The Copyright Act And Unworkable In Practice.

In seeking reversal of the Federal Circuit's decision regarding copyrightability, Google argues first, that the Java SE declaring code is uncopyrightable because it is a "method of operation" or "system"; and second, that the declaring code is uncopyrightable under the merger doctrine. Google Br. at 19. Neither argument holds up; both would require that this Court revise settled copyright law in ways that would have profound harmful effects on investment and innovation in software.

1. The Java SE Declaring Code Is Copyrightable Even If It Could Also Be Described As "Functional" Or Expressing A "Method of Operation."

Google asserts that the 11,330 lines of Java SE declaring code that it copied are an "entirely functional" "method of operation" under 17 U.S.C. § 102(b). Google Br. at 19. According to Google, the declaring code consequently cannot be copyrighted, no matter how creative or original the form of the expression in that code. This argument fundamentally misapprehends the text of Section 102 in a way that would radically weaken copyright protection for software programs, all of which can be characterized as expressing "methods of operation" for a computer in some sense.

There is of course no dispute that copyright generally applies to software code, referred to in the Copyright Act as a "computer program," and 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." 17 U.S.C. § 101. Though Google at times suggests that the Java declarations at issue here somehow are ***not*** part of a "computer program," *see* Google Br. at 5, Google itself repeatedly describes those same declarations as "instructions" that are used to invoke or call other computer code to perform a function. *See, e.g., id.* at 19, 20, 25; *see also id.* at 6 & n.4 (example of declaration). That description fits comfortably within the Act's definition of "computer program." Even if Java SE declaring code were not part of a "computer program," however, the declarations would be eligible for copyright protection nonetheless, as they easily meet the broader definition of "literary works" in the Act. *See* 17 U.S.C. § 101 (defining literary works as "works, other than audiovisual works, expressed in words, numbers, or other verbal or numerical symbols or indicia, regardless of the nature of the material objects . . . in which they are embodied."); *see also* 17 U.S.C. § 102(a)(1) (listing "literary works" as the first example of protected works of authorship).[8]

There is also no dispute that the Java SE declaring code satisfies Section 102(a)'s originality requirement for copyrightability. *See* 17 U.S.C. § 102(a) ("Copyright protection subsists . . . in original works of authorship fixed in any tangible medium of expression . . . ."); *see also* Oracle Br. at 7 (quoting Google's "Java guru" as stating that "there can be 'creativity and artistry even in a single method declaration'"); *id.* at 21 (noting Google conceded Section 102(a)'s originality requirement was met).

At this point, Google goes badly astray. Citing Section 102(b), Google argues that the Court can rule that the Java SE declaring code is a "method of operation" or a "system" because "it is entirely

[8] Amici Professors Menell, Nimmer, and Balganesh assert that the "declarations are not code" and that the "Federal Circuit confused the infringement analysis by accepting Oracle's characterization of declarations as 'declaring code.'" Menell, Nimmer, and Balganesh Br. at 35. Declarations are quite obviously code, *see* Oracle Br. at 6, but the answer to the copyrightability question cannot and does not rest on mere labels as copyright protects literary works regardless of whether those works are best characterized as "code" or not.

functional" and thus not copyrightable at all. Google Br. at 19. This argument has two fatal flaws: it misreads Section 102, and it has no coherent limiting principle that would distinguish the "functional" software code that Google concedes is protected by copyright from the "functional" software code that is not protected.

Under Google's view, original, creative code that otherwise meets the prerequisites for copyrightability under Section 102(a) will lose that protection entirely if that same code can also be characterized as a "method of operation" or a "system" under Section 102(b). But Section 102(b) does not disqualify original works of authorship from copyright protection wholesale. As this Court has explained, Section 102(b) codifies the "idea/expression dichotomy" in which "'every idea, theory, and fact in a copyrighted work becomes instantly available for public exploitation at the moment of publication'; the author's expression alone gains copyright protection." *Golan v. Holder*, 565 U.S. 302, 328 (2012) (quoting *Eldred v. Ashcroft*, 537 U.S. 186, 219 (2003)). If the Java SE declaring code expresses a method of operation or a system, copyright protection will not extend to the method or system, but the original expression – the creative way in which the author chose to describe that method or system – is still protected. See *Baker v. Selden*, 101 U.S. 99, 101–02 (1880) (holding that bookkeeping system could not be copyrighted, yet "[t[here is no doubt that a work on the subject of bookkeeping, though only explanatory of well-known systems, may be the subject of copyright"); *id.* at 104 (observing, with respect to Selden, that "no one has a right to print or publish his book, or any material part thereof" even if readers could practice the art his book described).

A brief examination of the text of Section 102(b) makes this clear. It reads, "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 is described, explained, illustrated, or embodied in such work." Google, again, asserts that an "original work of authorship" is not copyrightable at all if it also describes, explains, or embodies a "system" or a "method of operation." By that reasoning, however, any work that describes or embodies an "idea," a "concept," or a "principle" – essentially, anything coherent – would likewise go unprotected. Fortunately, no such absurd result is necessary. Oracle may claim a copyright in its particular expression of the ideas embodied by Java SE. Just as in *Baker v. Selden*, Google can use and apply those ideas so long as it does not copy Oracle's particular expression.

The second flaw in Google's "entirely functional" argument is that Google offers no limiting principle of what "functional" means with respect to computer code. As the United States pointed out in its brief opposing certiorari, Google's "effort to distinguish the declaring code at issue here from other copyright-eligible code is unavailing. . . . Both declaring code and implementing code ultimately perform the same practical function: They instruct a computer to work." United States Br. at 12.

This flaw in Google's argument is of especial concern to amici here. ***All*** software code is functional; the very definition of "computer program" in the Copyright Act makes clear that copyright applies to works that are functional: computer programs consist of instructions that "bring about a certain result." 17

U.S.C. § 101. Worse still, Google nowhere offers any coherent definition of what functional means, or any way to distinguish "functional" code from non-functional code, whatever that may be. If Oracle's 11,330 lines of concededly original declaring code are not copyrightable because they are in some sense "functional," or because they consist of numerous discrete "functional" elements, it is entirely uncertain what code is copyrightable. Such uncertainty about what can be protected and what cannot is toxic to investment, innovation, and licensing.

2. Google's Misapplication Of The Merger Doctrine Would Create Uncertainty And Would Undermine Innovation And Investment In Software.

Google further argues that "the declarations can be written only one way to perform their function of responding to the calls already known to Java developers" and consequently the Java SE declaring code is not copyrightable under the merger doctrine. Google Br. at 19. Google suggests that the Court "can decide the case more narrowly by applying the merger doctrine," *id.*, but Google's misapplication of that rule is not narrow at all. Google would fundamentally change the meaning of merger in ways that would upset settled investment expectations in software code and inhibit investment in copyright.

The merger doctrine is an important limitation on copyright that ensures only expression, not ideas, get copyright protection. As Google acknowledges, under the merger doctrine, "copyright protection does not apply when there are only a few ways to express or embody a particular function." *Id.* at 13; *see also*

Oracle Am., Inc. v. Google Inc., 750 F.3d 1339, 1360 (Fed. Cir. 2014) ("Under the merger doctrine, a court will not protect a copyrighted work from infringement if the idea contained therein can be expressed in only one way.").

But the merger doctrine has no application here because Oracle could have written the declaring code in countless different ways to achieve the same function, as Google concedes. *See* Oracle Br. at 7 (citing Pet. App. 5a; JA311-313); Google Br. at 31 (conceding but dismissing "the fact that Sun could have originally chosen different declarations, or the fact that Google later could have created a different organization of methods that responded to different calls.").

In fact, Google is not applying the merger doctrine at all. Instead, Google's "merger doctrine" is a new and different and dangerous thing: computer programs that were originally protected by copyright lose that protection – not because they embody the sole means of expressing an idea, but because at some later date they have become the most popular and familiar way to express an idea.

The odd syntax of Google's argument hints at the illogic of its merger argument. As Google puts it, "the declarations can be written only one way to perform their function of responding to the calls already known to Java developers." Google Br. at 19. Not quite. At the time Oracle wrote the declarations, there were many, many ways to write them. The authors of Java SE were not constrained in any way by the need to "respond to calls already known to Java developers" because at that point, obviously, there were no Java developers. In other words, when Oracle wrote the declarations, what Google now declares was "their function" did not even exist.

Eventually, of course, the Java SE declaring code became very well known to millions of developers and technology companies around the world. This success and familiarity were attributable to the quality of that declaring code, and the copyright-dependent licensing program that allowed Oracle's predecessor Sun to vigorously promote Java while ensuring that reimplementations of Java and Java platforms remained compatible with Java SE.

What Google actually means by its merger argument ("the declarations can be written only one way") is that by the time Google decided to copy Java SE, there was one overwhelmingly popular way to write the declarations – Oracle's way. Thus, Google writes, "the Java language ***would not permit*** Google to write its own declarations for those methods that Android reimplemented ***without requiring Java developers to learn thousands of new calls.***" Google Br. at 8 (first emphasis Google's, second emphasis added). *See also id.* at 10 ("Only the Java SE declarations can create the interface with the calls ***known to Java developers.***" (emphasis added)); *id.* at 13 ("Google reused declarations from the Java SE libraries because – and only because – no other option would recognize ***the calls used by Java developers.***" (emphasis added)); *id.* at 20 ("Only one precisely written set of declarations will perform the function of responding to the corresponding ***calls known to the developer.***" (emphasis added)); *id.* at 31 ("once Sun released Java SE, Google had to use the declarations from the Java SE libraries to respond properly to the existing ***calls that the developers then knew.***" (emphasis added)). When Google describes the Java SE declaring code as "mandatory" or "required," Google Br. at 7, 8, 14, 15, 20, 26, 32, Google really means "convenient."

Google's argument has nothing at all to do with merger. Google is not arguing that there is only one way to write declaring code that will carry out the functions of Java SE. *Zalewski v. Cicero Builder Dev., Inc.*, 754 F.3d 95, 103 (2d Cir. 2014) (merger applies when an idea can be expressed in such a limited number of ways that idea and expression "merge"). In fact, it appears that there are and were alternative ways to write such declaring code.[9] Google's version of merger is novel and pernicious: As Google would have it, if a creative, original copyrighted computer program becomes sufficiently well-known and popular, the author is not rewarded for its ingenuity or creativity. To the contrary, the work will lose the copyright protection it previously enjoyed. New entrants will be relieved of the need to innovate or compete, but instead may copy the successful work of others with impunity. The effects on innovation and investment in software would be obvious and perverse. Companies will not invest over $80 billion per year in software R&D, nor collaborate on licensing and compatibility, if they are faced with the prospect that at some point in the future, their most successful and popular copyrighted works could suddenly be transferred to the public domain, for reasons wholly beyond their control.

[9] In its brief, Oracle points out that Apple and Microsoft were able to achieve the same functionality on their own platforms without using Java or the Java declaring code, and that others like Spring and Log4J wrote their own declaring code for Java. Oracle Br. at 31–32.

As the Federal Circuit correctly observed, "Google cites no authority for its suggestion that copyrighted works lose protection when they become popular, and we have found none." 750 F.3d at 1372. To the contrary, courts have held that external constraints on the plagiarist's choices at the time of infringement are irrelevant to the question of whether the work was copyrightable in the first place. *See, e.g., Dun & Bradstreet Software Servs., Inc. v. Grace Consulting, Inc.*, 307 F.3d 197, 215 (3d Cir. 2002); *Mitel, Inc. v. Iqtel, Inc.*, 124 F.3d 1366, 1375 (10th Cir. 1997). To advance this new and expanded theory of merger, Google argues in this Court that "at least in the context of computer software, merger is evaluated at the time material is reused."[10] Google Br. at 30 (citing CONTU report at 20). There is no support for Google's argument that copyrightability in general, or software copyrightability in particular, flips on and off depending on subsequent events. Congress did create special rules for computer software in certain contexts, *see* 17 U.S.C. §§ 109, 117, but not with respect to copyrightability generally or with respect to the doctrine of merger embodied in Section 102(b). Furthermore, if a work is entitled to copyright protection at the time of creation, it retains that protection for the term of the copyright. *See* 17 U.S.C. § 302.

Google's argument would radically change the doctrine of merger to add special rules, apparently applicable only to software, that do not appear anywhere in the text of the Copyright Act. Those

---

[10] Throughout its brief, Google uses the word "reuse" rather than "copy" to describe what it did with the 11,330 lines of Java SE declaring code.

special rules would undermine successful authors by allowing competitors to copy the authors' original work, not because that work is the only way to express an idea, but because it is the most popular. The purpose of the Copyright Act is to "promote the Progress of Science and the useful Arts," U.S. Const. Art. I, § 8, cl. 8, not to punish those who innovate, create, and invest. This Court should reject Google's merger argument.

3. Google's Misapplication Of The Merger Doctrine Is Not Saved By Its Policy Arguments For "Reimplementation."

In the middle of its merger argument, Google makes an appeal to purported industry practices and policy. According to Google, its wholesale copying of the 37 packages of the Java SE declaring code was a straightforward example of an accepted industry practice of "reimplementation," whereby new entrants in a software market "write the extensive code that performs the relevant *functions* from the legacy product" but "reuse the more limited code that is required – because it cannot be written any other way – to allow users to use commands they already know from the legacy product." Google Br. at 26. Google appears to argue that the merger doctrine should be expanded to endorse and sanction this practice even when it involves copying, rather than "reimplementing," other code. *Id.* Otherwise, Google argues, "Oracle would require Java developers to learn thousands of new calls to replace those they already know, with no benefit to anyone." *Id.* at 27. Google complains that this is equivalent to "requiring the developers to learn an entirely new programming language, simply to invoke the same functions using different labels." *Id.*

First, this argument actually has nothing to do with the merger doctrine. Recognizing copyright for Java SE declarations does not allow Oracle to monopolize any *function*, it merely requires those who refuse to license Java SE to find their own way to express those functions (what Google disparagingly calls "using different labels"). To the extent Google wants a rule that would allow it to write its own code to implement the functions that Java SE performs, Google does not need the merger doctrine. It just needs to write its own declaring code.

Second, despite Google's argument that "reimplementation" of another party's copyrighted code without authorization is commonplace and accepted in the software industry, and dictated by the merger doctrine, the concept is starkly missing from copyright decisions. Simply put, there is not a single reported decision from any court, at any level, discussing unauthorized "reimplementation" of software code as an appropriate practice, an example of merger, or as a copyright defense, except for Google's arguments in this case. On the rare occasion the term does appear, it is an example of infringement, not a defense to it. *See XimpleWare Corp. v. Versata Software, Inc.*, No. C 13-05160 SI, 2014 WL 490940, at *2 (N.D. Cal. Feb. 4, 2014) (holding that plaintiff adequately alleged copyright infringement where it alleged that defendant "without authorization, copied, reproduced, distributed, and re-implemented" plaintiff's source code as part of another computer program). This is likely because *actual* reimplementation – writing new and original code to perform the same functions as legacy code, often using a "clean room" – typically will not raise copyright concerns. What predictably does raise copyright concerns is when a company does not "reimplement" code to replicate functions but instead *copies* it without permission – what Google did here.

Third, Google complains that making "reimplementation" a version of merger is necessary to prevent Oracle from creating "serious obstacles" to the creation of "an innovative platform like Android and the creation of applications for it." Google Br. at 27. This is a scare tactic.[11] Software developers have an interest in seeing their products distributed as widely as possible, and often work to ensure compatibility with other companies' software and hardware for that very reason. Like many industry participants, Oracle's predecessor in Java, Sun Microsystems, accomplished this through licensing. Oracle Br. 11–12. The only apparent obstacle that Oracle presented to the "innovative" Android platform was Oracle's insistence that Google license Java SE on terms that would ensure the compatibility of Android with Java SE. *Id.* at 13. Google refused, and there is no dispute that its "reimplementation" is incompatible with Java SE.

Once again, Google's approach to copyright threatens innovation and investment. If new entrants and competitors can simply "reimplement" or "reuse" (i.e., copy) an existing firm's software code on the theory that doing so takes less effort than writing their own, it is difficult to imagine that firms will undertake the investment and dedication that made a software platform like Java – or many other successful software platforms – possible. The same is

[11] To the extent there are genuine concerns about a company using its control over expressive language to hold up innovation, new entrants have other remedies, such as competition law, that are more appropriate and sensible than advocating a radical change to the scope of the copyright laws for software.

true if those new entrants and competitors can spurn the copyright holder's license terms because they perceive those terms as an "obstacle." Certainly *amici* could not and would not have been willing to invest billions of dollars in software acquisitions to keep EMC at the forefront of technology industry trends, had they suspected such loopholes in the copyright protection for those software assets.

IV. Conclusion

Clear, settled, consistent rules for copyright protection are essential to foster investment and innovation, particularly with respect to software. The principles of copyrightability that Google advances in this case would, if adopted, unnecessarily create uncertainty and doubt among software innovators and their licensees. The Court should not accept Google's invitation, and instead should affirm the sound decision of the Federal Circuit below.

Respectfully submitted,

David W. Shapiro
Counsel of Record
Fred Norton
Bree Hann
The Norton Law Firm P.C.
299 Third Street, Suite 106
Oakland, CA 94607

February 19, 2020

Counsel for Amici Joseph M. Tucci and Paul T. Dacier