No. 18-956

In the Supreme Court of the United States

GOOGLE LLC,
Petitioner,

v.

ORACLE AMERICA, INC.
Respondent.

On Writ of Certiorari to the
United States Court of Appeals for the Federal Circuit

BRIEF OF SAS INSTITUTE INC. AS *AMICUS CURIAE* IN SUPPORT OF RESPONDENT

PETER K. STRIS
ELIZABETH ROGERS BRANNEN
Counsel of Record
DANA BERKOWITZ
DOUGLAS D. GEYSER
JHANIEL JAMES
JOHN STOKES
Stris & Maher LLP
777 S. Figueroa St., Ste. 3850
Los Angeles, CA 90017
(213) 995-6800
elizabeth.brannen@strismaher.com

Counsel for Amicus Curiae

TABLE OF CONTENTS

TABLE OF AUTHORITIES

Page(s)

Cases

Statutes

Other Authorities

INTEREST OF *AMICUS CURIAE*[1]

SAS Institute is one of the world's largest privately held software companies. In 1976, it began with five employees. Today SAS employs nearly 14,000 people and earns over $3 billion in annual revenue.

SAS's flagship product is a suite of business software that facilitates a variety of analyses such as data mining and business intelligence. Its customers include 92 of the top 100 companies on the *2018 Fortune Global 1000*. They use SAS products for important pursuits like fighting cancer.[2] SAS continually improves its products and creates new ones. In 2018, it reinvested 26% of its revenue in research and development.

Unlike Google and many of its amici, which made a business decision to distribute software for free and generate revenue from other sources such as advertising, SAS is a proprietary software company—it licenses its software to commercial users in exchange for payment. But SAS also balances what parts of its software are open, and what parts are closed. It offers various license terms for different kinds of uses. SAS also regularly weighs the business benefits of contributing to industry standardization, such as increased access and influence, against those of keeping its technology strictly proprietary. SAS makes the business decision to participate in industry-standards bodies like the Data

---

[1] No counsel for any party authored this brief in whole or in part, and no person or entity other than amicus or its counsel made a monetary contribution intended to fund the preparation or submission of this brief. Petitioner's consent to the filing of amicus briefs is filed with the Clerk, and respondent has consented to the filing.

[2] *See, e.g.*, Alison Bolen, *Analytics Leads To Cancer Cures*, SAS Institute Inc., http://bit.ly/31NO3mq (last visited Feb. 18, 2020).

Mining Group. When it does so, SAS makes its standards-essential intellectual property available to license under reasonable and nondiscriminatory terms.

SAS can also address first-hand the consequences of curtailing copyright protection for software interfaces. All SAS licenses prohibit reverse engineering and copying without permission. But a British competitor reverse engineered and copied the SAS System to create a drop-in replacement, *i.e.*, a clone of the SAS System. When SAS sued in the United States and Europe, the outcomes were starkly different. Here, the Fourth Circuit affirmed a $79 million award to SAS based on willful breach of license (and other state-law claims). For the same misconduct in the U.K., however, SAS received no redress because Europe has weaker protection for computer programs than the United States. The European courts deemed SAS's software interfaces not copyrightable and its license provisions unenforceable.

SAS Institute's proprietary business model gives SAS a strong interest in robust intellectual-property protection for software, including software interfaces. But as its various license offerings and standards-groups participation demonstrate, SAS also appreciates the desirability of access and balance. Given its first-hand experience, SAS is well-positioned to provide a useful perspective for assessing the questions presented.

SUMMARY OF ARGUMENT

Google copied over 11,000 lines of source code and the accompanying structure, sequence, and organization that Java's creator painstakingly composed. The code fueled Java's success. It took Sun years to create. Google rejected an available open-source license and copied the code into its competing product, Android, virtually

overnight. Oracle Br. 12-15. Innumerable devices with Android operating systems contain, in executable form, the thousands of lines of code that Google copied. C.A. No. 13-1021, J.A. A1092 (citing trial testimony that there are 750,000 daily device activations containing the infringing code). Google made the business decision to proceed without a license because it wanted to attract Java programmers to Android. It seeks a free pass because it copied what it calls "software interfaces," which Google defines as "computer code that allow[s] developers to operate pre-written libraries of code used to perform particular tasks." Google's Opening Brief i ("Br."). The Federal Circuit correctly decided that the Copyright Act prohibits Google's conduct.

I. What Google calls software interfaces are entitled to copyright protection. Google argues that the interfaces are merely a "method of operation" under 17 U.S.C. § 102(b) (codifying the idea/expression dichotomy) and that the merger doctrine applies because (once Google decided to use the declarations it copied) there was only one way to call them. Google's arguments cannot be squared with critical facts: (1) it is widely agreed, including by Google's own "Java guru," that interfaces are the result of an artistic and creative process; and (2) both Sun (the entity that matters for merger) and, in any event, Google, could have written the declaring code in many different ways.

II. Although Google and many of its amici offer "interoperability" as an excuse, Google copied the software interfaces not because it wanted Android applications to interoperate with Java, but so it could attract Java programmers for Android to *replace* Java. "[U]nrebutted evidence" showed "that Google specifically designed Android to be *incompatible* with the Java

platform and not allow for interoperability with Java programs." Pet. App. 46a n.11. No case has found fair use where the defendant copied to produce an *in*compatible product. Regardless, even if Google's resulting product were interoperable with Java, it would not constitute fair use. The notion that software is functional cannot mean it is fair use to copy portions to create a competing alternative. That outcome would eviscerate statutory protection for computer programs and subvert the constitutional goal of incentivizing new works.

III. Adopting Google's position would also be bad policy. Proprietary software companies like SAS and Oracle depend on copyright protection to invest the vast sums they do in creating software. That model is thriving in the U.S., where companies can generally count on copyright protection. Permitting competitors to copy software interfaces has undesirable practical consequences, as SAS Institute can aver from first-hand experience. If Google prevails, the incentives for software companies will be exactly backward. Weaker copyright protection will push companies to restrict access to software and invest less in innovation. Affirming meaningful copyright protection, by contrast, will encourage companies like SAS and Oracle to continue investing in creating new works and offering liberal licensing terms. The Court should refrain from revising copyright law to undermine Congress's express protection for computer programs.

ARGUMENT

I. SOFTWARE INTERFACES EMBODY ORIGINAL AND CREATIVE EXPRESSION

The Copyright Act protects software programs as "literary works." 17 U.S.C. § 102(a)(1); *see, e.g.*, Br. 17.

The Act repeatedly recognizes copyright ownership in a "computer program," which is "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. The Act refers to "computer programs" throughout. *See, e.g.*, 17 U.S.C. §§ 109(b)(4), 110(11), 117, 121(b)(2). Google does not dispute that software interfaces, as it defines them, are "literary works" within the meaning of Section 102(a). *See* Br. 17, 19; Pet. App. 141a. As Google frames it, "software interfaces are lines of computer code," and the copied declarations convey instructions to be executed by a computer. Br. i, 1-2, 4.

Google nonetheless insists that software interfaces are a special class of software that does not deserve copyright protection. Google maintains that Oracle's declaring code for Java falls on the wrong side of the "idea/expression dichotomy" of 17 U.S.C. § 102(b). *E.g.*, Br. 17-18. According to Google, the code it copied represents the "one way to perform [the interface's] function." Br. 19. Google thus asks the Court to craft a judicial carve-out from the Copyright Act for software interfaces, which it paints as categorically less expressive and more functional than other computer programs.

Google is wrong. At a basic level, it will often be difficult definitively to distinguish a "software interface" from other software. *Infra* Part I.C. But to the extent Google's definition here essentially equates to declaring code and where interface portions of other code *are* identifiable, Google's argument still fails. That is because such interfaces can, and frequently do, embody creative expression. *See, e.g.*, *Computer Assocs. Int'l, Inc. v. Altai, Inc.*, 982 F.2d 693, 699, 703 (2d Cir. 1992) ("common system interface" was copyrightable subject matter); *Mitel, Inc. v. Iqtel Inc.*, 124 F.3d 1366, 1372 (10th Cir.

1997) (values for setting telecommunications functions could be copyrightable); *Dun & Bradstreet Software Servs., Inc. v. Grace Consulting, Inc.*, 307 F.3d 197, 216 (3d Cir. 2002) ("need to interoperate" did not alter copyrightability of business software); *Sega Enters., Inc. v. Accolade, Inc.*, 977 F.2d 1510, 1520 (9th Cir. 1992) (object code was copyrightable despite implicating "system interface procedures"); *Eng'g Dynamics, Inc. v. Structural Software, Inc.*, 26 F.3d 1335, 1345 (5th Cir. 1994) ("[I]f a best-selling program's interface were not copyrightable, competitors would be free to emulate the popular interface exactly so long as the underlying programs were not substantially similar. This cannot be the law."). There are unlimited ways to write interfaces, and nothing justifies removing them from what the Copyright Act expressly protects. To the contrary, the user-friendly expressive choices Sun made became critical to Java's success.

A. The thousands of lines of Java declaring code and the organization Google copied are intricate, creative expression. They merit "thin" protection only if one misunderstands the nature of the work.

1. The creativity is undeniable. "Google's own 'Java guru' conceded that there can be 'creativity and artistry even in a single method declaration.'" Pet. App. 154a. His concession is well-taken. User-friendly expression is difficult to achieve in any medium. *Cf.* Letter from Henry David Thoreau to Harrison Blake (Nov. 16, 1857), in *The Correspondences of Thoreau* 498 (Walter Harding & Carl Bode eds., 1958) ("[I]t will take a long while to make it short."); William Strunk Jr. & E.B. White, *The Elements of Style* (4th ed. 1999) (advising to, *e.g.*, "[u]se definite, specific, concrete language" and "[o]mit needless words"). Software interfaces are no different. The same Google

Java guru explains that "[c]ode should read like prose." C.A. No. 13-1021, J.A. A3019. The best software interfaces are concise and intuitive. They should be "[e]asy to learn" and "to use" and "[a]ppropriate to audience." *Id.* at A30064, A3019. Designing effective interfaces is therefore "tough," and "[p]erfection is unachievable." *Id.* at A3049. Crafting them takes enormous creative firepower.

Other developers similarly note the challenges of authoring well-designed software interfaces. *See, e.g.*, Keshav Vasudevan, *Best Practices in API Design*, Swagger Blog (Oct. 10, 2016), http://bit.ly/38otnnF (good interfaces "can quickly be memorized" and should be "[h]ard to misuse" and "[c]omplete and concise"); Arnaud Lauret, *The Design of Web APIs* 71 (Oct. 2019) ("Design matters, whatever the type of interface, and APIs are no exception."); Brian Mulloy, *Web API Design: Crafting Interfaces that Developers Love* 4 (2012) (ebook), http://bit.ly/37k9HQG ("You have to get the design [of the interface] right, because design communicates how something will be used. The question becomes—what is the design with optimal benefit for the app developer?").

Poorly designed declaring code can have disastrous real-world consequences. For instance, if programmers misunderstand the declaring code for opening secure internet connections, their apps can suffer security vulnerabilities. *E.g.*, Martin Georgiev et al., *The Most Dangerous Code in the World: Validating SSL Certificates in Non-Browser Software*, *in* CCS '12: Proceedings of the 2012 ACM Conference on Computer and Communications Security (2012), http://bit.ly/2vp3zsY.

SAS Institute, for its part, tasks a committee of its most experienced developers to write and revise its software (including what Google would call the interfaces),

precisely because it is so difficult to craft high-quality expression—*i.e.*, to "communicate[] how something will be used." Mulloy, *supra*. SAS's developers engage in an iterative process in which they exercise considerable judgment in making a wide range of creative choices.

In fact, creative expression matters *more* for code with which users interact than it does for the implementing code Google admits is copyrightable. *Contra, e.g.*, Br. 25 (confining Java's creativity to implementing code). A computer runs implementing code as 0s and 1s; no one but the computer needs to understand it. It "remains a 'black box' to the programmer." Pet. App. 102a. In contrast, Java programmers need to know how to call the declaring code, for that is how they invoke a method. It is therefore critical for interfaces to be clear and easy to use, *i.e.*, to exemplify high-quality expression.

An interface's value thus derives from its quality and user-friendliness, and the declaring code is an important piece of what drove Java's success. Google copied arguably Java's most valuable part. *See* C.A. No. 13-1021, J.A. A3004 ("APIs can be among a company's greatest assets" or its "greatest liabilities."). As one Google subsidiary explains, "APIs are the lynchpin to the success" of several digital businesses, and "it's critical to think about design choices from the app developer's perspective." Apigee, *API Best Practices* 4, 10 (2016) (ebook), http://bit.ly/2HiEbru. "Many app developer[s] prefer [formats that are] more readable, more intuitive, and easier for API developers to implement." *Id.* at 13.

Google's own conduct proves the point. It is no accident that Google copied code that Java programmers liked—Google wanted Java programmers and enticed them by taking code that allowed them to use Android with calls they already knew. *See* Pet. App. 172a ("The compatibility

Google sought to foster was not with Oracle's Java platform or with the JVM central to that platform. Instead, Google wanted to capitalize on the fact that software developers were already trained and experienced in using the Java API packages at issue.").

Google helped itself to a massive benefit. The 37 API packages covered over *six thousand* separate methods, comprising thousands of lines of code and its intricate organizing structure. Pet. App. 129a. Those are 6,000 sets of instructions programmers could use for Android without any further effort—6,000 that Google didn't have to write itself. *Cf.* Oracle Br. 7-11. Google thus had good reason to know it could "accelerat[e] its development process by 'leverag[ing] Java for its existing base of developers.'" Pet. App. 172a (first alteration added); *see* Br. 3.

But Google was no freer to copy that code than a novelist is free to copy prose from another book. Those interfaces are the product of Sun's creative choices—decisions to write the declaring code precisely the way it did, selecting specific words and structure. Like any effective prose, interfaces' value and popularity stem from their concise and intuitive expression. Google's contrary contention that declaring code "is entirely functional," Br. 19, has no grounding in reality.

2. The record provides numerous examples of Java's creativity. For instance, take Oracle's "verify" method (Oracle Br. 5) or Google's simple "max" example (Br. 5-6; Pet. App. 224a-226a). Even with "max" the declaring code notably reflects more creativity than the straightforward implementing code:

(Line 1) **package java.lang**
(Line 2) **public class Math** {
(Line 3) **public static int max (int x, int y)** {

```
(Line 4)                    if (x > y)  return x;
(Line 5)                    else return y;
(Line 6)            }
(Line 7)    }
```

See Pet. App. 224a-225a.[3] To have a computer determine the larger of two integers, Java programmers do not need to know any implementing code, which "remains a 'black box.'" *Id.* at 102a. They only have to know this method's declaring code. *Id.*

The java.text package provides a more complex illustration. Specifically, this package contains a MethodFormat class for constructing messages to display to users. The class includes a "format" method that returns text formatted in a certain way. Its declaring code is:

```
package java.text
Class MessageFormat
public static String format(String pattern,
                            Object... arguments)
```

To avoid an error message, programmers must construct the call for this method using the patterns Sun created. Those are the same patterns Google copied. *Compare MessageFormat*, Java 2 Platform Standard Ed. 5.0 API Specification, http://bit.ly/39y3YIt (last visited Feb. 18, 2020) *with MessageFormat*, Android Platform API Reference, http://bit.ly/2uE8Ixa (last visited Feb. 18, 2020).

---

[3] The first three lines, emphasized in bold, are the declaring code that explains how to invoke the method, while the (simpler) lines, four and five, implement the method.

For instance, to return this sentence—"At 12:30 PM on Jul 3, 2053, there was a disturbance in the Force on planet 7"—a programmer could use this call:

```
int planet = 7;
String event = "a disturbance in the Force";
String result = MessageFormat.format(
    "At {1,time} on {1,date}, there was {2} on planet
    {0,number,integer}.", planet, new Date(), event);
```

MessageFormat, Java 2 Platform Standard Ed. 5.0 API Specification, http://bit.ly/39y3YIt (last visited Feb. 18, 2020). Functionality obviously did not dictate Sun's selection of these patterns.

There are many other possible examples, because Google did not copy a mere handful of methods. The 37 API packages it copied include over *six thousand* methods. Pet. App. 129a. True, programmers must express the calls a certain way to use Sun's declaring code. But that is because Sun made a series of creative choices in composing that code.

In addition to its declaring code, Sun could have expressed the structure of its Java interfaces in any number of ways. Sun elected to express Java's functionality by dividing it into certain packages, classes, and methods. How these names and structures link together represents the result of a distinctly creative process. Sun did not, for example, have to name any of the methods as it did. The Federal Circuit's "Arith.larger" is but one example. Pet. App. 150a. Sun just as easily could have called each one something else. Nor did Sun have to place any given method in a particular class, each of which it also chose to create.

Taken individually, the declaring code for each method is expressive. Taken together, the thousands of lines of code Google copied—because it wanted Android developers to be able to call the exact same methods the exact same way—undeniably constitute valuable expression. As the Federal Circuit explained, "[t]he evidence showed that Oracle had 'unlimited options as to the selection and arrangement of the 7000 lines Google copied.'" Pet. App. 150a.[4] Of those "unlimited options," the "selection and arrangement" it settled on was ultimately enormously creative.

In sum, Sun wrote on a blank slate and had countless ways to compose Java. It made choices about how best to craft and organize the declarations—how to make them clear and concise. Those are classically expressive goals. The resulting work is at the heart of what the Copyright Act protects.

B. Any suggestion that Google had to copy the declaring code and organization to make Java work is incorrect.[5] Just as Google (largely) refrained from copying Oracle's implementing code for each method, Google was free to create new interfaces to call the same functionality performed by those methods using different declarations. Google actually did write its own versions for many Android declarations. Google admittedly could have done the same for the thousands of declarations it copied—just not "without requiring Java developers to learn thousands of new calls." Br. 8. It is simply not the case that Google copied out of technical necessity.

---

[4] After the first trial, the parties stipulated that Google actually copied over 11,000 lines of code. Pet. App. 45a.

[5] Because the parties agreed not to litigate whether the Java language itself merits copyright protection, that question was never at issue at trial or on appeal, and it is therefore not presented here.

Google and its amici nevertheless suggest that it would have been impossible to use Java without the portions Google copied. That is demonstrably false—there are real-world counterexamples. To name a few:

First, Oracle itself wrote new declaring code to perform existing functions. For example, Google copied declaring code for several methods that together allow programmers to open a secure internet connection. Java has included these methods since its original release:

java.net.URL(String spec)
java.net.URL.openConnection()
java.net.HttpURLConnection.getInputStream()

See Pet. App. 126a n.2 (java.net). The corresponding declaring code is reproduced in the Federal Circuit appendix. *See* C.A. No. 13-1021, J.A. A10013-A10028.

With Java 11, Oracle released an alternative—a different way to accomplish this same function, with entirely different declaring code:

java.net.http.HttpClient.newHttpClient()
java.net.http.HttpRequest.newBuilder()
java.net.http.HttpRequest.Builder.uri(URI uri)
java.net.http.HttpRequest.Builder.build()
java.net.http.HttpResponse.body()
java.net.URI.create(String str)
java.net.http.HttpClient.send(HttpRequest request, HttpResponse.BodyHandler<T> handler)

The declaring code is available in the Java Platform Standard Ed. 11 API Specification, http://bit.ly/38oHAAS. Among other advantages, this alternative permits programmers to open a secure connection using a pattern

familiar to them, the "builder pattern." *Introduction to the Java HTTP Client*, http://bit.ly/38vysdQ.

This example refutes any contention that Google could not have written its own interfaces. For Java itself, Sun and Oracle wrote two different interfaces to do the same thing—open a secure connection.

Second, Oracle is not the only one capable of writing its own Java declarations—other entities have done so too. For instance, the java.util package that Google copied offers classes to manipulate "collections," which is Java nomenclature for a group of objects. As an alternative to Java's collections framework, the independent Eclipse Foundation created its own collections framework and designed interfaces it thought preferable to Java's. Eclipse touts its "[r]ich, [c]oncise and [r]eadable APIs." Eclipse Collections, http://bit.ly/38lpfVD (last visited Feb. 18, 2020). Eclipse's collections APIs perform essentially the same functions as Oracle's.

Likewise, Oracle's CEO testified that a company called Spring wrote its own interfaces in Java without copying. As Spring writes, "The Spring team puts a lot of thought and time into making APIs that are intuitive and that hold up across many versions and many years."[6] *Spring Framework Overview*, http://bit.ly/2OLYd1I (last visited Feb. 18, 2020). Spring's solutions are for the Enterprise Edition of Java (not Standard Edition, as here). But they demonstrate that there was no technical constraint preventing Google from creating its own interfaces too.

---

[6] Spring's APIs drew criticism for being "confusing." *See, e.g.*, Zeljka Zorz, *Secure Coding in Java: Bad Online Advice and Confusing APIs*, Help Net Security (Oct. 3, 2017), http://bit.ly/2ONEQoV. This only confirms that not all APIs are created equal.

Google's decision to copy, rather than create, was a *business* decision.

C. In all events, Google's position fails at a more basic level—trying to separate software interfaces from other code is unfounded in law and would often be unmanageable in practice. What counts as a software interface is hard to define consistently (in theory and certainly in practice), and there is no basis to attempt to impose a categorical exclusion; like other creative code, interfaces merit protection.[7]

First, there is no textual basis to distinguish "interfaces" from other computer programs. Code is code. Java methods and declarations both "instruct[]" a "computer" to achieve a "certain result" (17 U.S.C. § 101), and developers can write both in a variety of ways. Declaring code is just as necessary as implementing code to cause a computer to produce an output or bring about a certain result. Declaring code is not a mere "idea" or "method of operation," nor is it the "certain result" of a computer program's operation.[8] It is particular expression

---

[7] Some Google amici acknowledge as much. *E.g.*, Michael Risch Amicus Br. 30 (citing *Bateman v. Mnemonics, Inc.*, 79 F.3d 1532, 1547 (11th Cir. 1996) for "refusing to hold that interface specification is uncopyrightable, but instead applying filtration infringement analysis").

[8] Google's amicus the Electronic Frontier Foundation ("EFF") is therefore incorrect that Java declaring code is "a computer language" rather than code "written in that language" to "achieve[] a 'certain result' upon execution by the computer." EFF Amicus Br. 14; *cf.* Computer Scientists Amicus Br. 16 (arguing that there is no distinction between the Java interfaces and the Java language itself). Moreover, regardless, EFF mistakenly assumes that programming languages themselves are not copyrightable—a question not at issue, which, to SAS's knowledge, no court in this country has decided either

that became exceedingly popular with its intended audience of Java programmers *because of* its expressive qualities.

Second, there is no reliable way to distinguish "interfaces" from other code. Courts are ill equipped to make the attempt, and Google's artificial definition of "software interfaces" admits no principled reason (or way) to draw the line consistently. Excluding "interfaces" would be an exercise fraught with difficult line-drawing and vulnerable to problematic characterizations by litigants. Rather, Congress and the courts have long followed the better course, namely, recognizing that as with other works, the key to the copyrightability of software is creativity. *See, e.g.*, *Feist Publ'ns, Inc. v. Rural Tel. Serv. Co.*, 499 U.S. 340, 345 (1991) ("minimal degree of creativity" to satisfy Section 102(a)); *Golan v. Holder*, 565 U.S. 302, 328 (2012) (explaining the "idea/expression dichotomy"); Pet. App. 150a-151a (under the "merger doctrine," no copyright protection if there are only a few ways to express the idea at time of original work's creation) (citing *Satava v. Lowry*, 323 F.3d 805, 812 n.5 (9th Cir. 2003) and *Apple Computer, Inc. v. Formula Int'l, Inc.*, 725 F.2d 521, 524 (9th Cir. 1984)). The Federal Circuit correctly applied that precedent.[9]

---

way. *See supra* n.5. For similar reasons, the EFF attacks a strawperson in arguing that "a 'certain result' of a computer program" is generally uncopyrightable. EFF Amicus Br. 19. Declaring code delineates how to achieve the certain result and what form the result may take; it is not the "result" itself.

[9] Some amici criticize the court of appeals for supposedly failing to filter out unprotectable elements before deciding infringement. *See* Michael Risch Amicus Br. 6; Intell. Prop. Scholars Amicus Br. 15-16; Auto Care Ass'n Amicus Br. 13-16. But the court explicitly followed the "'abstraction-filtration-comparison' test formulated by the Second

II. COPYING TO REPLACE A COMPETITOR'S PRODUCT IS NOT "INTEROPERABILITY"

Google and its amici also argue that it is fair use to copy thousands of lines of code verbatim to develop a competing commercial product. *E.g.*, Br. 37-50. One of their primary justifications is "interoperability." *E.g.*, Br. 41.

But Google's notion of interoperability is the opposite of that term's actual meaning. Android is not interoperable with Java at all. As the Federal Circuit observed, there is "unrebutted evidence that Google specifically designed Android to be *incompatible* with the Java platform and not allow for interoperability with Java programs." Pet. App. 46a n.11.

That is, Google copied Oracle's interfaces to make its Android operating system popular enough to *replace* Oracle's products. Oracle Br. 14-15. In the early 2000s, Oracle licensed its own Java-based smartphone operating system that "'quickly became the leading platform for developing and running apps on mobile phones.'" Pet. App. 6a. When Google and Oracle failed to reach licensing terms, Google created its own competing operating system. *Id.* That system was a drop-in replacement for Java SE on mobile devices. Indeed, the evidence shows that as Google's Android phones proliferated, Java SE for mobile died. *Id.* at 6a-8a.

By contrast, *Lexmark Int'l, Inc. v. Static Control Components, Inc.* illustrates what real interoperability looks like and how it relates to fair use. 387 F.3d 522 (6th Cir. 2004). There, plaintiff Lexmark sold toner cartridges "that contained a microchip designed to prevent Lexmark printers from functioning with toner cartridges that

---

Circuit and expressly adopted by several other circuits." Pet. App. 142a.

Lexmark had not refilled." *Id.* at 529. Defendant Static Control Components ("SCC") copied a "Toner Loading Program" from Lexmark's microchip to make its own microchips compatible with Lexmark's printers. *Id.* at 530–531.

The Sixth Circuit held that "pure compatibility requirements justified SCC's copying of the Toner Loading Program" because "if any single byte of the Toner Loading Program is altered, the printer will not function." *Id.* at 542. In the fair-use inquiry, the first factor did not necessarily weigh against SCC because "it is far from clear that SCC copied the Toner Loading Program for its commercial value *as a copyrighted work*," rather than for its value to permit printer functionality. *Id.* at 544; *see also, e.g.*, *Sega*, 977 F.2d at 1526 (explaining that copying was "necessary" to achieve "compatibility"); *Sony Computer Entm't, Inc. v. Connectix Corp.*, 203 F.3d 596, 606-607 (9th Cir. 2000) (explaining that copying "produce[d] a product that would be compatible").[10]

None of the above is true of Oracle's Java interfaces. Google did not copy them to make any hardware or software physically compatible with other hardware or software. There is nothing that Google had to design to a certain specification. Google instead copied the interfaces precisely because of their "value *as a copyrighted work*"—because they are concise and intuitive, and they had

---

[10] *Sega* and *Sony* are also distinguishable because each "final product d[id] not itself contain infringing material" (*Sony*, 203 F.3d at 606 (citing *Sega*, 977 F.2d at 1526-1527); *see also* 203 F.3d at 606 (noting "entirely new object code")), and the copied code was not visible to the user (*Sony*, 203 F.3d at 603; *Sega*, 977 F.2d at 1525-1526). Here, as explained *supra* Part I.A, Android contains infringing material and the copied code is visible to programmers—its visibility provided the whole reason to copy it.

already proved popular among programmers. Google elected to copy not because it had to, but because that was the most direct path to accomplishing its business goal: attracting developers to Android. The Copyright Act does not permit copying popular expression simply to make a different work more appealing.

Contrary to Google's assertion, there is no "well-settled understanding that the functions of earlier computer software may be reimplemented, including by reusing the limited instructions required to replicate the commands known to the earlier product's users." Br. 14. Such an "understanding" would subvert the purposes of the Copyright Act and turn the fair-use inquiry on its head.

Regardless of their preferred business model, commercial competitors cannot properly cut corners by copying to capture the fruits of an original creator's labor—here, the base of six million developers that knew and liked the Java interfaces. The short-cut Google took was so profitable that even if Oracle receives billions in damages in this action, it will *still* have been worth it for Google to appropriate the Java interfaces for Android.[11] That kind of copying is not interoperability, and it is certainly not fair use.[12]

---

[11] Similarly, Arista's CEO, a former Cisco executive, observed that it would have taken "15 years and 15,000 engineers" to have competed "in a conventional way." Adam Lashinsky, *An Ex-Cisco Exec Reflects*, Fortune (Mar. 20, 2014, 2:55 PM), http://bit.ly/2HjCsCe. A jury determined that Arista infringed Cisco's command-line interface copyright but found for Arista based on the scènes à faire doctrine. The case resolved during the appeal.

[12] At trial, Google's fair-use defense centered on the theme that Android was not a drop-in replacement for Oracle's Java platform because "Java SE is on personal computers; Android is on

III. MEANINGFUL COPYRIGHT PROTECTION FOR SOFTWARE INTERFACES IS VITAL TO THE PROGRESS OF SCIENCE

Software interfaces are not only expressive; they are expensive to create. And they require investment throughout the product's life—from the beginning stages where viability is uncertain through maintaining popularity. The revenue SAS Institute earns from licensing its proprietary software is its lifeblood. That revenue supports SAS's substantial reinvestment and innovative software solutions, sustained over 40 years, leading to its recognition as one of the most innovative tech companies in the country. *See* Press Release, *SAS Honored as a Stevie Award Winner in 2019 American Business Awards* (Jun. 26, 2019), http://bit.ly/2HhDHSz.

Companies like SAS Institute invest the resources necessary to conceive and develop software because they can charge customers to use it. A critical part of that exchange of money for goods and services is that proprietary software companies can rely on copyright protection. Unauthorized clones of software interfaces hurt original creators and ruin the market for (and incentive to create) their works. If third parties may simply copy software at will, even "just the interfaces," companies will have less incentive to devote resources to creating software in the first place. Yet at the same time, copiers would reap rewards for the rote act of copying without expending resources—and thereby gain an unfair advantage.

---

smartphones." C.A. No. 17-1118, Oracle Opening Br. 68. But Google knew that it was days away from announcing that "the full functionality of Android would soon be working on desktops and laptops, not just on smartphones and tablets." *Id.* at 67.

Much is at stake. The proprietary software industry is booming in the United States. *See* Greg Ip, *If the Economy Booms, Thank Software,* Wall St. J. (May 29, 2019), https://tinyurl.com/y5ofk6le; BSA Foundation, *The Growing $1 Trillion Economic Impact of Software* (Sept. 2017), https://tinyurl.com/y77xjgke. Over 15,000 American software publishing companies collectively earned nearly $270 billion last year.[13] Dan Cook, *Software Publishing Industry in the US*, IbisWorld, 4 (July 2019), http://bit.ly/2SFKdI9. The industry grew five percent annually between 2014 and 2019, a trend that is expected to continue through 2024. *Id.* at 7, 10. Software publishers now employ more than 660,000 workers. *Id.* at 7. The jobs are high quality. For example, *Fortune Magazine*'s list of best places to work has included SAS for over a decade.

American proprietary software businesses are innovators as well as drivers of economic growth. Three such companies took the top three spots on Forbes's list of the "World's Most Innovative Companies" in 2018. *The World's Most Innovative Companies* (2018), Forbes, http://bit.ly/2SmZmik. American proprietary software companies have also developed some of the most popular computer coding languages. Stephen Cass, *The Top Programming Languages 2019,* IEEE Spectrum (Sept. 6, 2019, 4:30 PM), http://bit.ly/2OP6OAJ (listing, in addition to Java, the languages C and C++ developed by Bell Labs, Swift by Apple, and MATLAB by MathWorks).

In sharp contrast to the American experience, the software industry in Europe lacks the investor confidence and energy found in the U.S. In 2019, European "Software

---

[13] Software publishers are businesses that "disseminate licenses to customers for the right to execute software on their own computers." Cook, *supra*, at 2.

as service" (SaaS) companies received $5 billion in venture investment, compared to $20 billion for U.S. SaaS companies.[14] Philippe Botteri & Maxim Filippov, *2019 Accel Euroscape—The Rise of European SaaS Continues*, http://bit.ly/3bx20JY (last visited Feb. 18, 2020). European software companies also spend comparatively little on R&D. Of the 250 companies that generate nearly two-thirds of global business R&D investment, European software companies represent only about 8 percent of the total spending by software and computer-service firms, compared to *77 percent* by U.S. companies. Jacques Bughin et al., *Innovation in Europe*, McKinsey Global Initiative, 11 (Oct. 2019), https://mck.co/31LWLSi.

That is no accident, as SAS Institute can attest. As recounted below, SAS Institute received contractual protection for its software interfaces in the U.S., whereas a European ruling allowed a U.K. company to clone its software interface in an effort to steal SAS's customers around the world. If Google prevails, SAS and other proprietary software companies will face that prospect in the U.S. too.[15]

A. SAS Institute and others expend tremendous resources to create software. To develop the groundbreaking SAS System, for example, thousands of SAS Institute employees spent many millions of hours

---

[14] "Software as service" is a software licensing and delivery approach where software is licensed to a customer via a subscription. *What is Saas? Software as a Service*, Microsoft, http://bit.ly/3bzEXOE (last visited Feb. 18, 2020).

[15] To be sure, factors other than copyright protection affect the software industry's strength (*e.g.*, tax policy and patent laws), but Google cannot dispute that copyright protection is a critical variable in a software company's success.

over more than four decades. Today, SAS Institute routinely reinvests 23 to 26 percent of its revenue in research and development. And SAS is not alone. Software giants Salesforce and Adobe each spent nearly $2 *billion* on research and development in fiscal year 2019. *See* Press Release, *Adobe Surpasses $11 Billion in Annual Revenue*, 4 (Nov. 29, 2019), https://adobe.ly/37mYXB0; *2019 Annual Report: Celebrating 20 Years of Salesforce*, 39 (2019), http://bit.ly/2OP25yX.

Although creating original software (including the interfaces) is resource-intensive for everyone, companies recoup their investments in different ways. Some companies, like SAS Institute, monetize their software directly by offering commercial licenses in exchange for payment. Even such traditional software businesses, however, frequently offer free or low-cost limited licenses for educational use. Oracle and SAS mix-and-match models, both commercially licensing software and, depending on the product and circumstances, simultaneously making certain software available on open-source or less expensive terms for particular uses.

Others give their software away or make it widely available on open-source terms to attract users, then make money from their customer base. For example, Google famously offers its software and other content for free, then reaps billions in advertising revenue. Even Google's model, however, circumscribes customers' uses via licenses. *See, e.g.*, Pet. App. 7a; *Licenses*, Android Open Source Project, http://bit.ly/2SkymQy (last visited Feb. 18, 2020).

The Google model has gained adherents in recent years, including among Google's amici, but different companies will have different business reasons to prefer

one model or another.[16] Like Google and many of its amici, SAS Institute must decide which of its software products should be offered open source, which should be closed source but intentionally interoperable, and which should be closed and not made public.

Adopting Google's position would constrain that choice. But companies should remain free and encouraged to pursue proprietary software models. For SAS Institute and others, the extraordinary investment required to create cutting-edge software is worth it because copyright protection allows them to get paid for the works they create.

B. That ability to recoup the fruits of their engineering labor is critical to software companies' capacity to innovate. And robust copyright protection is essential to that process—that is "[t]he economic philosophy behind the" Copyright Clause, namely, "the conviction that encouragement of individual effort by personal gain is the best way to advance public welfare." *Mazer v. Stein*, 347 U.S. 201, 219 (1954).

Google's amici disagree. They assert that for software interfaces, the premise of the Copyright Clause and the Copyright Act is wrong: copyright protection in fact stifles

---

[16] Whether public policy should favor either model is a question for Congress. Google's business model has led to some controversy. Providing software for "free" to consumers and then monetizing their data can have grave privacy implications. *See* Rani Molla, *Why Your Free Software is Never Free,* Vox (Jan. 29, 2020), http://bit.ly/2SDWCMv; *see also, e.g.*, Charlie Warzel & Ash Ngu, *Google's 4,000-Word Privacy Policy Is a Secret History of the Internet*, N.Y. Times (July 10, 2019), https://nyti.ms/2UKTTne. Recently, Facebook's CEO faced a hostile Congressional hearing on the company's practice of selling users' data to third parties. *Mark Zuckerberg Testimony: Senators Question Facebook's Commitment to Privacy*, N.Y. Times (Apr. 10, 2018), https://nyti.ms/2OOQmjW.

innovation. One amicus speculates, for instance, that copyright protection for software interfaces "discourages innovation and inhibits competition in the technology industries." CCIA Amicus Br. 1-2; *see also* AAI Amicus Br. 9; Auto Care Ass'n Amicus Br. 5; Computer Scientists Amicus Br. 17; Lunney Amicus Br. 5.

1. SAS Institute's real-world experience shows otherwise. In dueling litigation in the United States and Europe, SAS recently felt first-hand the effects of lesser protections for software interfaces.

SAS distributed a limited version of its flagship SAS System product under a "Learning Edition" license, which included "a prohibition on 'reverse engineering,' as well as a restriction requiring use only for 'non-production purposes.'" *SAS Institute, Inc. v. World Programming Ltd.*, 874 F.3d 370, 376 (4th Cir. 2017), *cert. denied*, 139 S. Ct. 67 (2018). A U.K. company, World Programming Limited ("WPL") acquired that limited license, then violated its terms by cloning the SAS software to create a competing product. *Id.* at 382-383.

SAS Institute sued WPL in the U.K. and in the Eastern District of North Carolina. *Id.* at 376. The U.K. High Court referred several questions about the legal protections for computer programs under European law to the Court of Justice of the European Union ("CJEU"). *Id.* The CJEU ruled that the software interfaces of the SAS System were not copyrightable and that licensees are categorically entitled to reverse engineer computer programs. *Id.* (discussing Case C-406/10, *SAS Institute Inc. v. World Programming Ltd.* (May 2, 2012), http://bit.ly/2tWiRFa). SAS Institute thus took nothing in the U.K. action. *Id.* at 377.

American courts saw the matter differently. The district court granted summary judgment to SAS

Institute on liability for breach of the license agreement but granted summary judgment to WPL on the copyright-infringement claim. *Id.* After a jury trial and trebling of damages for state-law violations, SAS secured an award of over $79 million. *Id.* On appeal, the Fourth Circuit affirmed the damages award and vacated as moot the district court's copyright ruling. *Id.* at 378-379 (calling the copyright question "close," noting the direct conflict between North Carolina and E.U. public policy, and observing that "the United States has taken an approach that is more protective of intellectual property, and North Carolina courts have taken an approach that is more protective of the sanctity of contract").

As detailed above, SAS's experience illustrates that copyright protection *is* critical to incentivize software companies like SAS and Oracle to invest in creating and improving their products. The reason proprietary software companies can invest so much capital in research and development is because third parties cannot freely copy their work. As Oracle's CEO explained at trial: "If people could copy our software, in other words create cheap knock offs of our products, we wouldn't get paid for our engineering and we wouldn't be able to continue to invest at the rate we invest." C.A. No. 13-1021, J.A. 20454-20455. The same goes for SAS Institute and other traditional software companies that form a cornerstone of the software economy.

While some companies are willing to recoup their investments by *indirectly* monetizing software, there is no reason to force *every* company to adopt that business model. Nor is there any reason to believe such a model would adequately foster development of new and improved software. After all, the model is relatively new. The modern proprietary software business in this country

has been around twice as long. And that traditional model is what has produced most, if not all, of the software that others are clamoring to "reimplement" without permission.

2. Following the European example would invite additional negative repercussions. The likely response to weaker copyright protection is more restrictive license terms, or secrecy—outcomes that hurt consumers.

For instance, Google's amici IBM and Red Hat suggest that, as an alternative to copyright protection, companies keep their interfaces as trade secrets. IBM Amicus Br. 1 n.2; *id.* at 6-7. At the same time, they assert that software interfaces are essential to interoperability (and ergo innovation). *E.g.*, *id.* at 6. IBM and Red Hat thus urge that copyright protection is unnecessary because trade-secret protection would suffice.

Those arguments are incompatible. If software interfaces are critical to interoperability and thus innovation, then the law should encourage broad *disclosure*, not secrecy. That is precisely what a robust copyright regime achieves. Combined with the enforcement of licensing restrictions against reverse engineering, copyright protection allows creators to disseminate their works liberally without forfeiting all their value. Google's position, however, will drive precisely the opposite outcome.[17]

---

[17] Nor does patent protection alone suffice. Patents may protect functional and innovative aspects of a "procedure, process, system, [or] method of operation" that Section 102(b) clarifies do not receive copyright protection. But Congress wisely determined that software authors are entitled to copyright protection for the expression in their works without undertaking the expense and delay of seeking patent protection, and regardless of whether their expression discloses a patentable innovation.

That conclusion holds true even for open-source software. Although many of Google's amici contend that software interfaces are routinely available for free use, like Android, even those uses are subject to licenses. *See, e.g.*, *The Red Hat Enterprise Agreement*, https://red.ht/2SIMz8Y (last visited Feb. 18, 2020); *Mozilla Public License*, https://mzl.la/2ONbC9Z (last visited Feb. 18, 2020); *About The Licenses,* Creative Commons, http://bit.ly/39vF6RC (last visited Feb. 18, 2020); *Introducing the Permissive License Stack,* Protocol Labs (Feb. 14, 2019), http://bit.ly/2w7yw5e.

Without copyright protection, one cannot generally hope to enforce license terms—certainly not license terms that make source code openly available. Any requirement of contractual privity would be easy to avoid. Without copyright protection, one party could acquire a license and breach terms against reverse engineering or unauthorized copying, then others could use that code with impunity.

Software creators require rights against the world, not just rights against contracting parties. Google's position would eviscerate important protection for software interfaces, whereas the challenged decision wisely preserves it. A sea change would be ill-advised and should be imposed, if at all, only by Congress.

C. Finally, the preceding discussion shows that Google's amici are wrong to assert that the Federal Circuit's decision will upset the status quo. *See, e.g.*, Small, Medium, and Open Source Tech. Orgs. Amicus Br. 12-13, 18-21; Software Innovators Amicus Br. 11-22; Microsoft Amicus Br. 21-22; Gov't Engineers Amicus Br. 16-19; Software Freedom Law Center Amicus Br. 16-18; IBM Amicus Br. 13-18. Those amici do not cite a single case that allowed copying creative software code where the copier was trying to create a product that was *in*compatible with

the original; and none of their cases stands for the proposition that copyright protection does not extend to genuinely creative source code.[18] Instead, they cite cases involving true interoperability, like *Lexmark*. Or they incorporate the misguided argument that software interfaces embody an idea, not expression, or at best can be expressed only one way. But for the reasons discussed *supra* Part I and as the Federal Circuit recognized, Oracle faced "unlimited options" in determining how to write the declaring code. Pet. App. 150a. The Federal Circuit properly understood the cases amici cite. *See, e.g.*, Pet. App. 32a, 53a-54a (discussing *Sony* and *Sega*); *id.* at 142a-143a (discussing *Altai*).[19]

In actuality, it is the position of Google and its amici that would upend copyright law. Congress extended copyright protection to computer code, and courts have long determined copyrightability by asking whether the

---

[18] *Cf., e.g.*, *Lotus Dev. Corp. v. Borland Int'l, Inc.*, 49 F.3d 807, 810 (1st Cir. 1995) ("Borland did not copy any of Lotus's underlying computer code"); Pet. App. 159a-160a (explaining that in *Lotus* the commands were not at all creative and, unlike here, "were 'essential to operating' the system"); *Sega*, 977 F.2d at 1526; *Sony*, 203 F.3d at 606-607; *Altai*, 982 F.2d at 696, 702 (no literal copying).

[19] Moreover, the district court improperly excluded evidence that disproves Google's claim that everyone in the industry believed it was legal to copy Java APIs because "Sun/Oracle made the Java API declarations free and open." C.A. No. 17-1118, Oracle Opening Br. 74-78. For example, the court *sua sponte* redacted a sentence from an industry actor's email stating that copying the Java declaring code "makes us [Apache] *already* doing illegal things (in fact, Android using Harmony code is illegal as well)." *Id.* at 75. The court also improperly excluded as hearsay Sun's statement to the European Union that "Sun believes that . . . the Android runtime environment[] [is] an unauthorized derivative work of Java SE." *Id.* at 76. Compounding these errors, the court allowed Google to tell the jury at closing that these documents did not even exist. *Id.* at 78.

code constitutes creative expression that could have been written differently. To be sure, courts need guidance about how to distinguish unprotectable ideas, systems, or methods from protectable expression in the software context. *See, e.g.*, *SAS Institute*, 874 F.3d at 388 ("The area of software copyrights is a murky one, and federal courts have struggled with it for decades."). But the Federal Circuit assessed "the line between functionality and creativity," *id.*, in an exemplary manner.

At bottom, Google asks the Court to craft an atextual statutory exception for "software interfaces" or to extend the fair use doctrine to excuse blatantly unfair commercial copying. The court of appeals correctly rejected Google's ill-advised arguments.

CONCLUSION

This Court should affirm.

Respectfully submitted,

PETER K. STRIS
ELIZABETH ROGERS BRANNEN
Counsel of Record
DANA BERKOWITZ
DOUGLAS D. GEYSER
JHANIEL JAMES
JOHN STOKES
Stris & Maher LLP
777 S. Figueroa St., Ste. 3850
Los Angeles, CA 90017
(213) 995-6800
elizabeth.brannen@strismaher.com

Counsel for Amicus Curiae

February 19, 2020