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 FOR RESPONDENT

Dorian E. Daley
Deborah K. Miller
Matthew M. Sarboraria
Andrew C. Temkin
ORACLE AMERICA, INC.
500 Oracle Parkway
Redwood Shores, CA 94065

Dale M. Cendali
Joshua L. Simmons
Jordan M. Romanoff
Ari E. Lipsitz
KIRKLAND & ELLIS LLP
601 Lexington Avenue
New York, NY 10022

E. Joshua Rosenkranz
Counsel of Record
Annette L. Hurst
Peter Bicks
Lisa T. Simpson
Andrew D. Silverman
Matthew R. Shahabian
Jeremy Peterman
Hannah Garden-Monheit
Geoffrey Moss
ORRICK, HERRINGTON & SUTCLIFFE LLP
51 West 52nd Street
New York, NY 10019
(212) 506-5000
jrosenkranz@orrick.com

Counsel for Respondent

QUESTIONS PRESENTED

The Copyright Act protects "literary works," 17 U.S.C. §102(a), expansively defined as "works … expressed in words, numbers, or other verbal or numerical symbols or indicia," §101. Computer programs are protected as literary works under the Act. Google copied 11,330 lines of Oracle's original and creative computer code, as well as the intricate organization of its computer program, into a competing software platform, Android. The questions presented are:

1. Under §102(a), computer programs, like all "works of authorship," have "[c]opyright protection," as long as they are "original." The merger doctrine does not make any expression unprotectable except in the rare circumstance where there were very few ways to express the idea. Does the Copyright Act protect the code and organization that Google concedes were original and creative and that Oracle could have written in countless ways to perform the same function?

2. Was the Court of Appeals correct in holding that Google's copying was not fair, where Google conceded it copied for commercial purposes and that the code it copied serves the same purpose and has the same meaning, and Google did not dispute the evidence that Android competes directly with Oracle's work, harming its actual and potential markets?

TABLE OF CONTENTS

Page

TABLE OF AUTHORITIES

Page(s)

Cases

Constitutional Provisions

Statutes & Regulations

Legislative Materials

Other Authorities

INTRODUCTION

Google has a problem. It committed an egregious act of plagiarism and now needs to rewrite copyright law to justify it. It cannot.

Java SE was one of the most creative and intricately designed works of software ever written. Its elegance attracted a wide audience of developers. Manufacturers of all sorts of devices and competing platform makers clamored to license the Java SE platform. Innovation flourished, just as the Framers imagined, and just as the rest of the American software industry thrived under those same constitutional incentives.

Google wanted its own platform. Given its vast resources, it could certainly have written one. But with a looming existential crisis, there was no time to innovate. Google could have taken any of the several Java SE licenses Oracle offered, but Google rejected Oracle's compatibility imperative as inconsistent with its commercial objectives.

So Google opted to plagiarize and take the risk. Google copied 11,330 lines of computer code from Java SE, as well as the intricate organization and relationships among the lines of code. Google put the code in its competing product, Android, and successfully pitched it to Oracle's customers, generating billions of dollars in revenue.

Unauthorized copying into a competing product at this scale is clear-cut copyright infringement. If

Google had taken 11,330 topic sentences from an encyclopedia or the entire structure of a treatise to compete with the original, Google could not credibly argue that what it took was devoid of copyright protection or fair to copy.

Software is no different. Congress chose to treat software the same as any "literary work." All that matters for copyrightability is that the code or structure Google copied was original expression, which Google conceded each was. And all that matters for fair use is that Google used the code for the same purpose in a competing product for commercial advantage, which Google also conceded.

So Google tries to change the law. First, Google carves out from copyright protection a category of computer code that it vaguely calls "interfaces." But the Copyright Act rejects distinctions between kinds of code. Second, Google argues that it "needed" to use Oracle's code to appeal to what was familiar to Oracle's audience of app developers. But Google conceded below that it could have created Android in the Java programming language without copying any of those 11,330 lines. No legal principle justifies copying a work merely because it is popular with an audience that a competitor wants to capture.

Google protests that the decisions below defied "settled expectations" and threaten the software industry. But the U.S. software sector has risen to dominance *because of* copyright protection, not piracy. The real "settled expectation" is the Copyright Act's constitutionally inspired imperative to reward authors'

"individual effort by personal gain." *Eldred v. Ashcroft*, 537 U.S. 186, 212 n.18 (2003) (quoting *Mazer v. Stein*, 347 U.S. 201, 219 (1954)). Accepting Google's invitation to second-guess Congress's judgment is what will upset the status quo and jeopardize innovation.

This Court should affirm.

CONSTITUTIONAL AND STATUTORY PROVISIONS INVOLVED

Relevant constitutional and statutory provisions are reproduced in the appendix to this brief.

STATEMENT OF THE CASE[1]

Sun Develops Java SE To Help Developers Write Their Own Applications

The Java 2 Standard Edition Platform ("Java SE") is one of the most popular and revolutionary works of software ever written. Pet. App. 4a. Its audience is developers who use the platform to help them write programs ("apps"). Its customers are manufacturers that license and install the platform on devices to run those apps. Sun was the original author; Oracle continued the work after acquiring Sun.

---

[1] "GB" is Google's brief. The Joint Appendix and Supplemental Joint Appendix are "JA" and "SJA." The United States' invitation brief is "U.S. Cert. Br." Other amicus briefs are cited as "___ Br." Statutory citations are to Title 17.

Before Java SE, an app typically would not run across myriad devices with different operating systems, like Windows and Mac. App developers had to rewrite their apps for each. Pet. App. 5a. Java SE eliminated that inefficiency by enabling apps written using it to run across operating systems and devices. Hence Sun's credo: "write once, run anywhere." *Id.*

To create Java SE, Sun crafted a collection of ready-to-use programs, essentially modules that developers can incorporate in their own apps. Pet. App. 4a. Each individual program, called a "method," performs a discrete function, like drawing a shape, encrypting text, or solving a type of math problem. *Id.* Sun organized the methods into an intricate collection of "classes" that group related methods and define unique data types on which methods operate, and "packages" that group related classes. *Id.* Sun also created connections called "interfaces" among related methods across packages and classes (not to be confused with what Google calls "interfaces"). Pet App. 224a.

Sun created over 30,000 methods organized in 3000 classes and 166 packages. Pet. App. 5a. These programs save developers time. Pet. App. 4a. But developers don't have to use them; they can "write their own code" in the Java language "to perform those functions." *Id.* (quotation marks omitted).

Google's "Java guru" described writing and organizing the programs as "very much a creative process." JA318-319; Pet. App. 140a-141a, 229a. Like an author crafting a treatise with 30,000 paragraphs, Java SE's authors had "unlimited options" as to what to include and how to describe and organize it all. Pet. App.

150a; JA411-414. No specific approach was required. Pet. App. 165a. The design teaches programmers how to find, use, and remember the programs.

This case is about Google's copying of 11,330 lines of Java SE code and its elaborate organization.

1. The 11,330 lines Google copied are human-readable computer code ("source code"). Those lines would consume roughly 600 Joint Appendix pages. The parties have labeled this code "declaring code" (or "declarations"), as distinguished from "implementing code." The computer processes ("compiles") and reads both declaring and implementing code. Pet. App. 223a-224a. Both are necessary to instruct the computer. *Id.*

There are two differences. First, declaring code is like topic sentences and chapter and section headings, while implementing code serves as the body of paragraphs. Pet. App. 4a-5a. Second, only the declaring code must appeal to a human audience. It is the only code in Java SE that app developers see. Pet. App. 102a. The declarations memorably and vividly explain to app developers what each method and class does, how the computer will use it, and how it relates to other parts of Java SE. JA373-375 (stipulation). For example, here is the declaring code for a method called "verify" that uses a security key to determine whether a signature is valid:

> public boolean verify (PublicKey
> verificationKey, Signature verificationEngine)
> throws InvalidKeyException,
> SignatureException

This code tells the developer not just, "*This is a method called 'verify*,*'*" but also how to use it. In plain English, it says:

> *Give me a security key (which I'll call "verificationKey") that you want me to use to verify a signature that you previously gave me.*
>
> *Also tell me the algorithm I should use to verify the key (I'm calling that "verificationEngine").*
>
> *Caution: You can't just give me any algorithm. The algorithm must meet specified requirements that you can find elsewhere (a class I call "Signature").*
>
> *There are two ways this might not work ("exceptions")—the key might be wrong ("InvalidKeyException") or the algorithm might be wrong ("SignatureException").*
>
> *If so, I'll give you an error message.*
>
> *If the signature is valid, I'll say, "True" (that's what "boolean" means).*

No file label does all of that. JA414-417.

Every aspect of that instruction was a creative choice that could have been written countless ways. Pet. App. 228a. Sun could have called the method "checkSignature," "analyze," "confirm," "check," or

something more fanciful. Pet. App. 226a. So too for the name of every input and error message—and even whether to include error messages or impose special requirements for the algorithm, what they would be, and where to find them. *Id.*

No wonder Google's "Java guru" described the process of crafting just the declaring code as "an art, not a science," JA517-522, distinguished by "the complexity of figuring out how best to *express* what it is that the programmer wants done," JA380 (emphasis added). He conceded that "there can be 'creativity and artistry even in a single method declaration,'" Pet. App. 154a, just as there can be artistry in a single topic sentence.

Every line reflects multiple choices like these. Java SE's authors struggled for years with those sorts of creative choices for each of its 166 packages, 3000 classes, and 30,000 methods. Pet. App. 5a; JA311-313.

2. Whether in the Java language or the English language, writing good sentences is only part of the creative process. All authors struggle with how to organize and build connections among parts of their written works to make them more appealing to their audience.

Java SE's authors wrestled with the same organizational choices. A computer would run fine if the authors had dumped 30,000 programs in one class. JA417-418; Pet. App. 265a. But that would not appeal to its audience any more than a treatise with 30,000 random and unconnected paragraphs. JA417-419. Java SE's authors had countless creative choices to

make about how to group methods into classes (subclasses, sub-subclasses, etc.) and classes into packages, and what relationships and interdependencies to build between programs. Pet. App. 140a-141a. Those choices reflect the authors' creativity and unique view of what groupings and relationships would be best.

Figure 1 depicts the unique organization the authors chose for one of the packages: "java.security." That package organizes classes and methods that the authors considered security-related. The tiny colored lines on the right represent 362 methods each performing a discrete function. The lengthy declaring code for "verify," for example, is represented by the tiny line with the arrow next to it. The indented names on the left are the classes, subclasses, etc. in which the authors grouped related methods in increasing levels of specificity. Nothing precluded the authors from putting a method into a different class, dividing the methods and classes across multiple packages with more granular security themes, combining other security-themed packages to make a larger package, or having no security theme. Pet. App. 140a-141a. *See also* SJA3-19 (displaying organization, including inter-package relationships for 34 of the packages Google copied).

Figure 1

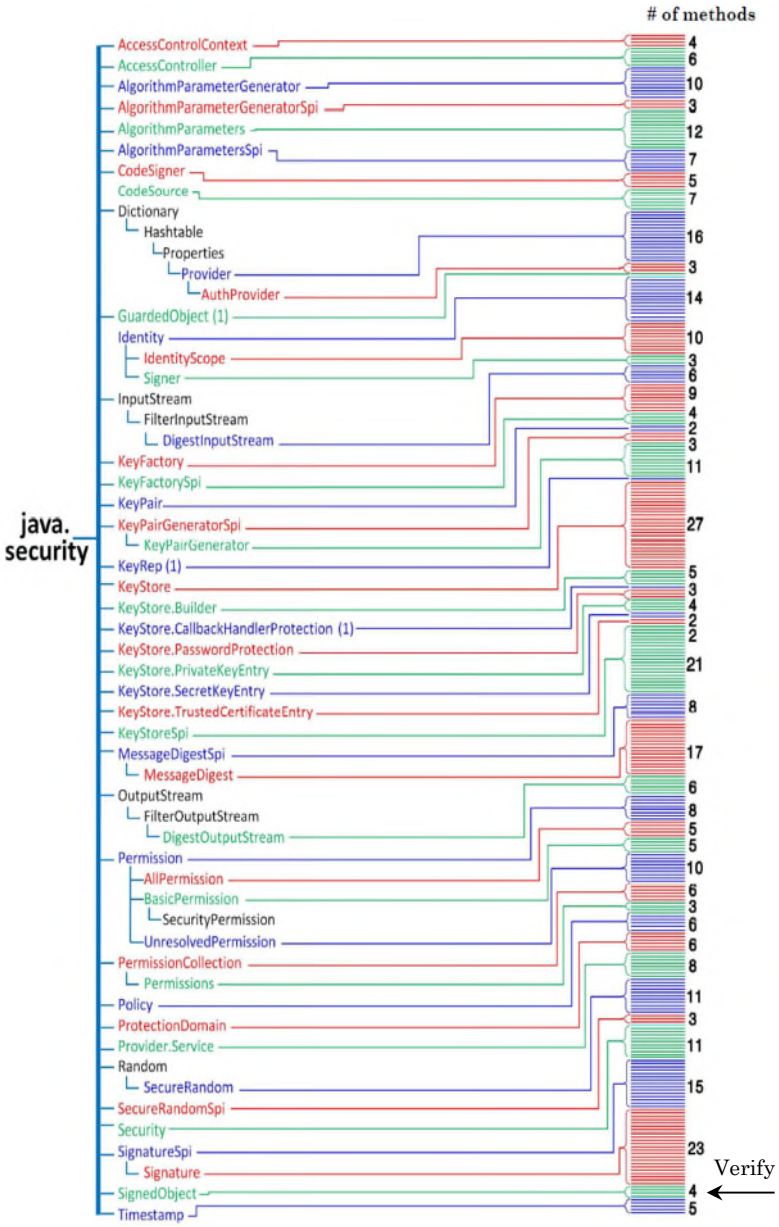
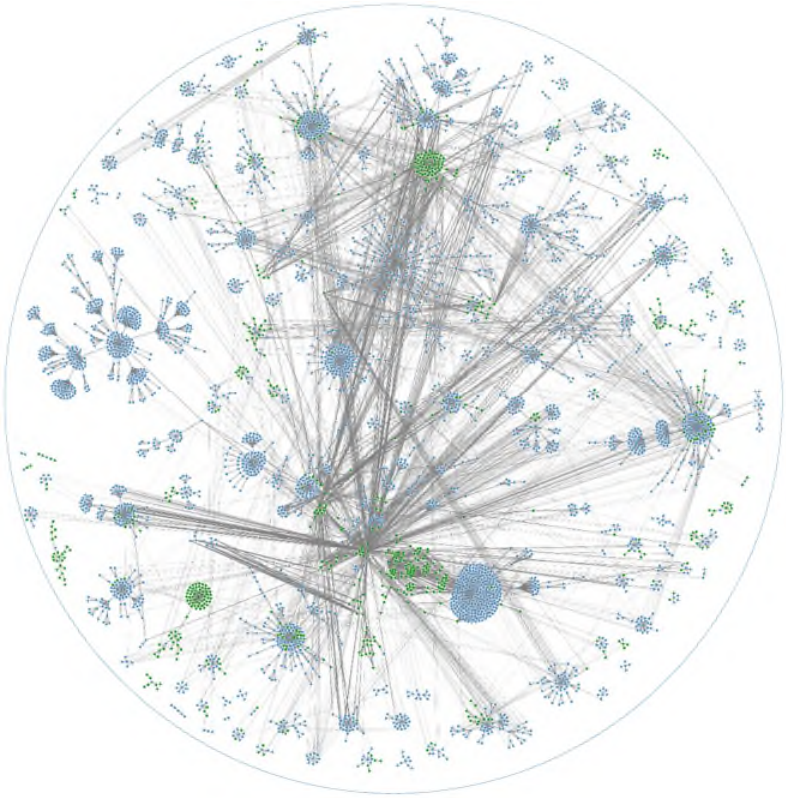

# of methods
java.
security
AccessControlContext 4
AccessController 6
AlgorithmParameterGenerator 10
AlgorithmParameterGeneratorSpi 3
AlgorithmParameters 12
AlgorithmParametersSpi 7
CodeSigner 5
CodeSource 7
Dictionary
Hashtable
Properties
Provider 16
AuthProvider 3
GuardedObject (1) 14
Identity 10
IdentityScope 3
Signer 6
InputStream 9
FilterInputStream 4
DigestInputStream 2
KeyFactory 3
KeyFactorySpi 11
KeyPair
KeyPairGeneratorSpi 27
KeyPairGenerator
KeyRep (1)
KeyStore 5
KeyStore.Builder 3
KeyStore.CallbackHandlerProtection (1) 4
KeyStore.PasswordProtection 2
KeyStore.PrivateKeyEntry 2
KeyStore.SecretKeyEntry 21
KeyStore.TrustedCertificateEntry 8
KeyStoreSpi
MessageDigestSpi 17
MessageDigest
OutputStream 6
FilterOutputStream 8
DigestOutputStream 5
Permission 5
AllPermission 10
BasicPermission 6
SecurityPermission 3
UnresolvedPermission 6
PermissionCollection 6
Permissions 8
Policy 11
ProtectionDomain 3
Provider.Service 11
Random
SecureRandom 15
SecureRandomSpi
Security
SignatureSpi 23
Signature
SignedObject 4
Timestamp 5
Verify

Figure 1 omits an additional layer of complexity: the numerous relationships cutting across packages and classes that combine features in useful ways. Figure 2 shows these relationships as gray lines connecting the clusters of blue classes and green interfaces, further illustrating the authors' numerous expressive choices embodied in the structure of Java SE. JA420-425.

Figure 2



SJA1.

That is no filing cabinet. And all these creative choices—both writing the declaring code and organizing the programs—were critical to Java SE's success. "[Q]uality" choices attract app developers. JA388-389 (Google's expert); JA517-521. Sun/Oracle invested hundreds of millions of dollars to deliver that quality. JA297-298, 307-308.

Sun Licenses Java SE, Including Just Declaring Code And Organization

The quality of Sun's code and its broad licensing program spurred boundless innovation. It attracted six million developers and propelled the Java platform to become "the leading platform for developing and running apps on mobile phones," tablets, and personal computers. Pet. App. 6a. By 2005, the Java platform was in over a billion mobile handsets including Samsung, Motorola, and ZTE, JA392, 500, adapting to each new generation of devices. JA501. Danger's T-Mobile Sidekick, a smartphone comparable to the early Android devices, used Java SE. Pet. App. 35a; JA359-360. So did Amazon's Kindle. Pet. App. 50a.

As mobile devices were becoming "as powerful as yesterday's computers," JA498, Sun was exploring a next-generation upgrade based on Java SE. JA227, 535-537. It licensed other companies, like SavaJe, to do the same. Pet. App. 50a-51a; JA429-430.

Sun/Oracle developed a licensing framework to suit all sorts of users and uses. They never charged app developers, who can take a free license to create apps for Java SE. Pet. App. 5a-6a.

To recoup its investment, Sun/Oracle offered a variety of licenses to device manufacturers and competing platform developers (usually large companies) to enable devices to run the apps. These options were all available to Google:

- ***Free license***. This "open-source" license (to a version called "OpenJDK") is *free of charge*, but the licensee and sublicensees must give back their improvements to the public in return. Pet. App. 5a-6a.

- ***Declaring-code license***. This "specification" license for *only* the declaring code and organization was used by big software companies such as IBM, SAP, Red Hat, and Oracle (pre-acquisition) who then "reimplemented" the rest. JA301-302, 304, 402-407. Google ignores this license when it says Sun "never sold or licensed separately" just the declaring code. GB43.

- ***Full-platform license***. Others licensed the entirety of Java SE, including its implementing code, under a "commercial license." Pet. App. 127a.

To retain "write once, run anywhere," the latter two licenses require licensees to prove that their platforms are compatible with Java SE. Pet. App. 127a-128a.

Google Copies 11,330 Lines Of Declaring Code And The Organization Of Java SE

In 2005, Google faced an existential crisis. It was poised to lose a "significant share of an increasingly important portion of the [search] market" if it did not

quickly develop technologies for mobile devices. JA547 (Google 10K). Google's solution was a mobile-device platform called Android. Google knew a successful platform required quality prewritten programs. GB3.

Google tried to develop prewritten programs from scratch, as Apple and Microsoft had. Pet. App. 149a & n.5. But it struggled to write declarations as elegant as Java SE's. After Apple's iPhone launch, Google was "beyond out of time," but its versions were "half-ass at best." JA506, 558 (internal emails).

Google knew it could accelerate Android by copying Java SE. Internal Google documents explain that copying Java SE's declaring code and organization would let Google (1) drastically "reduce[] [Android's] development time," JA480; *see* JA489-490; (2) "leverage" Java SE's "6M[illion] Java developers" to build apps for Android, JA503, Pet. App. 172a; and (3) appeal to device manufacturers and mobile carriers. JA472-474, 482-483.

The problem, as Android's founder advised, was Sun's "APIs are copyrighted." JA492; *see* JA474, 478-479. Google could have taken the open-source license for free. But Google considered the give-back obligation "unacceptable." JA367, 557. Google sought a custom license from Sun, but negotiations cratered when Google insisted on terms that would break "write-once, run anywhere." Google demanded "no limits on modifying the code," which guaranteed that Android would *not* be compatible with Java SE. Pet. App. 6a, 128a. Sun refused. Nevertheless, "Google elected to

'[d]o Java anyway,'" without a license, "making enemies along the way." Pet. App. 6a-7a (quoting Android founder).

"Google copied verbatim the declaring code of the 37 Java API packages—11,500 lines of [Sun's] copyrighted code." Pet. App. 7a.[2] Google also copied the structure and arrangement of those packages. JA372-377 (stipulation); JA95.

The packages Google copied were "central" and "important" to Java SE, JA426; SJA2—the packages "Google believed Java application programmers would want to find" in Android. Pet. App. 219a. Google then "paraphrased the remainder," Pet. App. 140a, partially writing its own implementing code, but largely copying from others, JA361-362.

For all Google's extolling the virtues of interoperability, it bears emphasis: Google admitted that it purposely made Android *in*compatible with Java. Programs written for Android *cannot* run on the Java platform and vice versa. Pet. App. 46a & n.11.

Android Competes Directly With Java SE

Android's founder testified that, overnight, Android became a "competitor" to Java SE, "targeting the same industry with similar products." JA366; *see* Pet. App. 50a-53a. But Google gave Android away for free. GB9. It did not need licensing revenue; Google

[2] The parties stipulated that 170 lines were necessary to use the Java language. Pet. App. 45a; JA386-387 (Google technical expert). Those lines are no longer in the case, leaving 11,330 lines.

makes billions selling advertising based on users' personal information. JA345-346.

Google pitched Oracle's code as a selling point to handset makers and cellular carriers—including Oracle's own customers—touting Android's "Core Java Libraries," "Java API," and "Powerful, Simple Java Application Framework." *E.g.*, JA590-591 (Qualcomm); JA596-597 (LG); JA598, 600 (AT&T). Oracle "customers switched to Android." Pet. App. 7a. For example, Amazon ping-ponged between Java and Android and then leveraged Android to force Oracle to reduce its price by 97.5%. Pet. App. 51a, JA395-397, 438-440. Android competed with Java-SE-powered products, like Danger and SavaJe. Pet. App. 50a; JA584.

Android is now the dominant platform in mobile devices, JA465-466; Pet. App. 7a. As Oracle's CEO vividly put it: It's "very difficult to compete with free, especially since they were using our software." JA397-398.

The Court Of Appeals Finds Google Unfairly Copied Copyrightable Code

Oracle sued Google for copyright infringement. Pet. App. 1a-2a. The jury found Google infringed, but hung on whether Google's use was fair. Pet. App. 130a-131a. After trial, the district court found that the declaring code and organization were both "creative" and "original" but nevertheless held they were not copyrightable. Pet. App. 141a, 165a-166a.

The Court of Appeals unanimously reversed. Pet. App. 123a. The court found it "well established that

copyright protection [for computer programs] can extend to" both their code and their structure and organization. Pet. App. 139a. The court rejected Google's argument that the declaring code's original expression "merge[d]" with unprotectable ideas. Pet. App. 150a-152a. There could be "no merger" because Java SE's authors had "unlimited options" in writing and organizing the declaring code. Pet. App. 150a-151a. The court rejected Google's argument that the organization is an unprotectable "method of operation" under §102(b), finding it contrary to the statutory text and this Court's precedent. Pet. App. 158a-166a. The Court of Appeals remanded on fair use, Pet. App. 184a, and the jury found for Google.

With a full (and different) record on fair use before it, Pet. App. 24a-25a, the Court of Appeals again unanimously reversed, finding no fair use as a matter of law. The court "assume[d] that the jury resolved all factual issues relating to the historical facts in favor of the verdict" and carefully analyzed each of the four fair-use factors in light of those historical facts. Pet. App. 23a. The court concluded that "allowing Google to commercially exploit Oracle's work will not advance the purposes of copyright" because Android is a "superseding use" that "effectively replaced Java SE … and prevented Oracle from participating in developing markets." Pet. App. 53a

SUMMARY OF ARGUMENT

I. Java SE's declaring code and organization are copyrightable.

A. The Copyright Act protects computer programs if they are original. §102(a). Google conceded the 11,330 lines of code and Java SE's organization meet that threshold.

B. Section 102(b) codifies the idea/expression dichotomy. Oracle's copyright protects only its unique expression, not the underlying ideas embodied in Java SE. Anyone can write Java programs that provide app developers the same underlying functionality. They just cannot copy Oracle's exact words and precise organization.

While conceding that §102(b) codifies the idea/expression dichotomy, Google suggests that §102(b) also withdraws protection from declaring code because it is functional. That is wrong because it would negate Congress's decision to protect all computer code, which by statutory definition is always functional.

C. Google invokes merger—a narrow judge-made doctrine that does not apply unless the original author had very few ways to express the idea. It does not apply here because, as Google concedes, Java SE's authors had countless options.

Google focuses on the wrong author at the wrong time in arguing that *Google* needed to copy. Regardless, Google did not need to copy. It is undisputed that "nothing prevented Google from writing its own declaring code, along with its own implementing code, to achieve the same result." Pet. App. 151a-152a.

Google's argument is not about necessity at all, but expedience: the desire to save time and make An-

droid familiar to app developers for a commercial advantage. It will always be easier to co-opt someone else's audience than build your own. That does not eliminate protection for the original.

Google reaches this improper result through merger by defining the "idea" of Oracle's work so narrowly as to be literally synonymous with Oracle's expression: the "idea," it says, is to invoke Oracle's precise declaring code. But that's circular: Once Google decided to copy exactly, it *had* to copy exactly. That sort of "necessity" is foreign to copyright law and merger.

Google's interoperability arguments are off-base. Congress declined to exempt copying for interoperability and Google purposely designed Android to be *in*compatible with Java SE.

II. Google's competing commercial use of Oracle's code is the classic superseding use that fair use has always precluded.

A. This Court has already held that the ultimate question of fair use is reviewed *de novo*. The Court of Appeals gave due deference to the jury by assuming it resolved disputed historical facts in favor of the verdict, but Google's copying was unfair as a matter of law.

B. Each fair-use factor confirms Google's superseding use.

1. Copying for a commercial and nontransformative purpose strongly weighs against fair use. Android generated over $42 billion. Google used Oracle's code

for its original purpose without changing its expression, meaning, or message. That makes Google's use nontransformative. Otherwise fair use would swallow the author's exclusive right to create derivative works, which, by definition, add something new.

2. Oracle's code is a work Congress intended copyright to incentivize. It is creative software, which Congress protected as a "literary work," crafted from countless options to appeal to people as well as computers.

3. Copying 11,330 lines of code, and Java SE's intricate organization, is substantial. Given the importance of what Google took, it makes no difference that Google copied only a fraction of a large work.

4. Google's effect on Java SE's market is the most important factor. Android's founder conceded that Android "competed" with Java SE, and the Court of Appeals correctly identified undisputed evidence that Android harmed actual and potential markets for Java SE. If a use like Google's were permissible, Java SE would have no market.

C. Google's contrary policy arguments are legally irrelevant to fair use. First, there is no settled practice of pirating valuable software and incorporating it into competing products. But even if there were, it would not make Google's use fair. Second, Google's argument about "compatibility" and "lock-in" ignores that Android was *in*compatible with Java SE, contradicts Congress's prohibition on superseding use, and fails to prove any real lock-in. Third, Google did not show that its refusal to take a license unleashed innovation, which cannot excuse piracy, anyway, or else it would

be fair to distribute unauthorized copies of popular software, like Adobe Photoshop, to unleash creativity.

III. This Court should reject Google's invitation to rewrite the Copyright Act. In the six years since the Court of Appeals' copyrightability decision, the software industry has continued its meteoric rise. Fears of the industry's demise rest on overreading the Court of Appeals' narrow decisions. Rewriting the Copyright Act to exclude code from protection is what threatens innovation: No company will make the enormous investment required to launch a groundbreaking work like Java SE if this Court declares that a competitor may copy it precisely because it is appealing.

ARGUMENT

I. Java SE's Declaring Code And Organization Are Copyrightable.

A. Java SE's declaring code and organization, which Google conceded are original, are protected under §102(a).

Section 102(a) dictates what works have "copyright protection." Its application here is easy because Google has conceded both the governing law and the dispositive facts.

1. Google admits that the Copyright Act protects computer programs. GB17. Congress achieved that expressly in 1976 by defining "[l]iterary works" expansively as those "expressed in words, numbers, or other verbal or numerical symbols or indicia." §101. In 1980, Congress reaffirmed that protection by defining "computer program," *id.*, and enacting distinct

"[l]imitations" on the scope of computer program copyrights, §117; *see* §§109(b), 506(a).

Google concedes that §102(a)'s text has only one requirement for copyrightability applicable here: "works of authorship" must be "original" to be copyrightable. GB17. This threshold is "minimal"—easily met whenever the work, however "crude [or] humble," reflects some "creative spark." *Feist Publ'ns, Inc. v. Rural Tel. Serv. Co.*, 499 U.S. 340, 345 (1991) (quotation marks omitted).

Like any other literary work, two aspects of a computer program can be sufficiently creative to reflect "original" expression. First, the written code—the "statements or instructions," §101—can be original, much like the prose in a book. 1 Melville B. Nimmer & David Nimmer, *Nimmer on Copyright* §2A.10[B][1] (Nimmer). Second, beyond the lines of code, copyright also covers a program's organization, 2 William F. Patry, *Patry on Copyright* §3:81—just as it covers plot and characters, *see Nichols v. Universal Pictures Corp.*, 45 F.2d 119, 121 (2d Cir. 1930) (L. Hand, J.), and the original arrangement of a compilation of pre-existing material, *Feist*, 499 U.S. at 348.

2. Google resolved the §102(a) inquiry here by "conced[ing] … that [§102(a)'s] originality requirements are met," for both Java SE's declaring code and its organization. Pet. App. 140a-141a; *see* Pet. App. 216a, 267a (district court findings). Google had no choice. All the intricate choices described above (at 4-11) far exceed the minimal "creative spark" required.

Because Google cannot retreat from its concessions, *see Wood v. Milyard*, 566 U.S. 463, 474 (2012), it is unclear what it hopes to achieve with various swipes at the creativity of what it copied. It emphasizes that implementing code is creative, GB25, but that does not prove that Java SE's declaring code and organization are not, Pet. App. 140-141a, 150a-151a, 162a. Google asserts that the declaring code and organization are unprotected because, paradoxically, each "compelled" the other. GB29, 31-32. Having already conceded creativity in both, Google cannot now claim each simultaneously negates the creativity in the other. A plot's creativity does not preclude the eventual dialogue from being creative, and the dialogue does not retroactively strip the plot of creativity.

Google then extends that faulty argument, asserting that once you ignore the elements of the declaring code "compelled" by Java SE's "conceptual" choices, all that remains are unoriginal names. GB29. That is like saying once you choose a plot, the story writes itself. Those "conceptual choices" were themselves expressive; they are the concededly original selection of what programs to write and how to organize and interrelate them. Google cannot just assume away all those expressive choices regarding thousands of declarations and reduce the case down to the expression in one declaration.[3]

---

[3] *Southco, Inc. v. Kanebridge Corp.*, 390 F.3d 276, 279, 282-83 (3d Cir. 2004) (en banc) (Alito, J.), is inapposite because the screw manufacturer's nine-digit part numbers—like "47-10-202-10"—were *un*original—while originality is conceded here. Merger was also not an issue there. *Id.* at 285-86 n.4.

Even looking at one declaration proves Google is wrong. As the "verify" method demonstrates, *supra* 5-7, crafting each declaration involved much more than choosing its name. Google's "names [and] short phrases" argument, which the district court addressed, Pet. App. 215a, is so meritless that Google waived it on appeal. Brief for Appellee and Cross-Appellant Google Inc. at 68, *Oracle Am., Inc. v. Google Inc.*, No. 13-1021 (Fed. Cir. May 23, 2013) (Dkt. No. 100). Regardless, the Court of Appeals correctly explained why it fails on its merits, Pet. App. 153a-155a: The question is whether the *work* is "original," §102(a), not whether each individual phrase standing alone would be copyrightable. U.S. Cert. Br. 15; 37 C.F.R. §202.1(a) (cited at GB29). Otherwise, every literary work could be atomized to unprotectable "short phrases." No poem (of any length) or Mamet play would ever be protected.

B. Section 102(b) codifies the idea/expression dichotomy, and Oracle seeks protection only for its particular expression, not ideas.

Since §102(a) grants "[c]opyright protection" to Oracle's "original" declaring code and organization, the next question is whether §102(b) withdraws that protection. It does not.

1. Section 102(b) "in no way enlarges or contracts the scope of copyright protection." *Feist*, 499 U.S. at 356 (quotation marks omitted). Rather, §102(a) says that an author's expression "gains copyright protection," *Golan v. Holder*, 565 U.S. 302, 328 (2012), while

§102(b) says that protection does not "extend" to prevent others from expressing the author's "idea" in their own words, *see Mazer*, 347 U.S. at 217. Copyright protects the words themselves and their organization—what §102(b) calls "the form in which [the idea] is described, explained, illustrated, or embodied." The author may not "extend" that to claim a monopoly in any "idea" (or "process," "method of operation," or other synonym in §102(b)) described or embodied "in such work."

Baker v. Selden, 101 U.S. 99 (1880), is the source of this "idea/expression" dichotomy and much of the language in §102(b). It holds that although copyright protected how Selden described his accounting method in a book (the "expression"), it did not protect the system itself (the "idea" or "method of operation"). Accordingly, Selden's copyright did not give him a right to preclude others from achieving "similar … [accounting] results." *Id.* at 100-01.

Oracle does not seek to protect the ideas embodied in Java SE—or, in *Baker*'s words, to preclude anyone from achieving "similar … results." Oracle claims rights only in its particular expression of those ideas.

Consider the methods, on which Google myopically focuses one declaration at a time. The idea of a method is the functionality that prewritten program performs. The idea of a method's declaration is to establish and describe what the method does, how to use it, and how it relates to the rest of the platform's organization. For example, the idea embodied in Java SE's "verify" method is a program developers can invoke to confirm a valid signature. *Supra* 5-6.

In contrast, the method's precise declaring and implementing code are Oracle's particular expression of the idea. Anyone is free to write another program to confirm a valid signature and to describe it with declaring code—in §101's parlance, to "bring about a certain result" in a computer. There is no dispute that Oracle, Google, or anyone else could write different code in Java—including different declaring code—to perform that exact same function. Pet. App. 150a-151a & n.6. What they cannot do is use the identical "statements or instructions" contained in Oracle's "verify" program's declaring code, along with thousands of other Oracle declarations. *See* H.R. Rep. No. 94-1476, at 57 (1976) ("Section 102(b) … make[s] clear that the expression adopted by the programmer is the copyrightable element in a computer program" while "the actual processes or methods embodied in the program are not.").

Now look beyond the methods to Java SE's creative organization, including the 37 packages and nearly 600 classes Google copied (which Google ignores). The idea of Java SE is to provide a collection of modular programs that are helpfully organized and described to enable developers to use them in writing their apps. Anyone is free to create and organize their own platform that appeals to developers—including one that provides exactly the same functions. Anyone can create a package of programs organized around security functions, or a class of related programs for authenticating data. They simply cannot duplicate Oracle's organization. Pet. App. 164a-165a. The code and organization Google copied are protected because they are expression, not ideas.

Similarly, protecting Oracle's code and organization does not undermine "limits on software patents." GB26. Oracle is not claiming protection in an "abstract" idea practiced through "generic computer implementation," like the idea of using declaring code to create and organize programs. *Alice Corp. Pty. Ltd. v. CLS Bank Int'l*, 573 U.S. 208, 223, 225 (2014); *see also Mazer*, 347 U.S. at 217 (no "[s]tatute" precludes a thing from receiving patent and copyright protection).

2. Google concedes that §102(b) "codifies the 'idea/expression' dichotomy." GB17. Yet it contradicts that concession, §102(b)'s plain language, and this Court's holdings by simultaneously suggesting that the provision does far more. Google implies that "original" expression protected under §102(a) loses protection whenever it can *also* be described as a "functional" "method of operation" or "system." GB19.

Impossible. *All* "computer programs are by definition functional," Pet. App. 162a-163a, because they "bring about a certain result," §101. "[S]ince we know that Congress did determine in 1980 to protect computer programs, the terms 'process,' 'system,' or 'method of operation' [in §102(b)] must not be understood" to withdraw copyright protection from computer programs. Jane C. Ginsburg, *Four Reasons and a Paradox: The Manifest Superiority of Copyright Over Sui Generis Protection of Computer Software*, 94 Colum. L. Rev. 2559, 2569-70 (1994).

Google insists that *some* code (implementing code) is protectable, just not *this* code (declaring code). GB25. Its rationale is that because developers use a

portion of the declaring code (which Google labels "interfaces") to invoke Oracle's prewritten programs, only *that* code is the "method of operation." GB19. But Google cannot draw a line between "interfaces" and any other code. Declaring and implementing code both operate the computer, which is why both, like "*all* computer code[,] could be described as a method of operating a computer." U.S. Cert. Br. 12 (emphasis added). Moreover, Congress explicitly rejected Google's purported distinction. It defined "computer program" to cover code whether "used *directly or indirectly*" to operate a computer. §101 (emphasis added). That definition prohibits Google's carveout for code that "interact[s] with, or operate[s]" other code. GB5 n.2.

Congress made a considered legislative judgment not to put courts in the unsuitable position of differentiating between different types of code. As CONTU member Arthur Miller explained in his influential article, CONTU and Congress "reject[ed]" "exceptions to copyright law for … the *copying of program interfaces*." Arthur R. Miller, *Copyright Protection for Computer Programs, Databases, and Computer-Generated Works: Is Anything New Since CONTU?*, 106 Harv. L. Rev. 977, 1013 (1993) (emphasis added); *see* Nat'l Comm'n on New Tech. Uses of Copyrighted Works, Final Report 27 (1979) (CONTU Report) (dissent of Commissioner Hersey) (discussing rejection of proposal to deny protection to interfaces, described there as "machine-control element[s]"). "No matter how artfully phrased," an exception for "interfaces" "would create an unbounded opportunity to appropriate programs and to foment litigation." Miller, *supra*, 1034.

Google also flouts the Act in arguing Java SE's organization is an unprotectable "system" comparable to a "filing cabinet[]." GB19. Congress protected an original "arrange[ment]" of "collected data"—including the utilitarian organization of facts "so that they may be used effectively." *Feist*, 499 U.S. at 348. Google's argument that Oracle's organization is an unprotectable "system" would vitiate that decision. Regardless, Oracle's work is no filing cabinet, nor is it a collection of facts. It is an elegant arrangement of computer code that is itself original expression, as Google conceded. *Supra* 21.

C. Google's merger argument is meritless.

Google's main argument now is merger—an argument Google did not even present in its first cert. petition. This is another improper effort to achieve an atextual carveout for "interfaces" from the protection §102(a) grants. The merger doctrine applies only in the narrow situation where there are very few ways to express an idea. That is not this case. Google's argument stretches the conceded facts and the merger doctrine beyond recognition, in ways that would jeopardize protection for all sorts of works.

1. Merger is inapplicable because Java SE's authors had countless ways to express the ideas embodied in the platform.

The merger doctrine is a judge-made corollary to the idea/expression dichotomy, found nowhere in the Copyright Act. It applies in the "rare instance" where the author has very few ways to express the idea of

the work, 2 Patry on Copyright §4:47, such that expression and idea are "indistinguishable" and merge. *Sid & Marty Krofft Television Prods., Inc. v. McDonald's Corp.*, 562 F.2d 1157, 1168 (9th Cir. 1977). When there are a "variety of ways to perform the same function" or to describe the same idea, merger is inapplicable. *Atari Games Corp. v. Oman*, 888 F.2d 878, 885 (D.C. Cir. 1989) (Ginsburg, J.).

Under "the Copyright Act's basic design," merger focuses exclusively on "the choices available to [Sun] ex ante when it created Java." U.S. Cert. Br. 14; *accord* Oman Br. §II.A. The Act says "[c]opyright in a work … *subsists* from its creation and[] … *endures* for [the copyright] term." §302(a) (emphasis added); *accord* §102(a) ("protection subsists")."An author gains 'exclusive rights' in her work immediately upon the work's creation," *Fourth Estate Public Benefit Corp. v. Wall-Street.com, LLC*, 139 S. Ct. 881, 887 (2019), and copyright protection cannot "retroactively divest" based on later events, U.S. Cert. Br. 14. Likewise, §410(a) requires the Copyright Office to assess a work's copyrightability—and merger—at the time of registration. *See Atari*, 888 F.2d at 884-85. There is no mechanism for deregistering a work for merger later.

That is why every circuit to consider the issue has concluded that what matters are the options available to the author creating the original work—not the copyist. ACUF Br. §III.A.2 (collecting cases). The best Google can muster to argue that merger considers the copyist's options *ex post* is an out-of-context quote from CONTU. GB30. But CONTU agrees that protection is assessed at inception, and in any event, cannot

negate three separate provisions of the Copyright Act.[4]

Properly considering the options available to Java SE's authors, merger "is not implicated." U.S. Cert. Br. 13. As both courts below concluded, those authors had "unlimited options" in writing the declaring code and organizing the packages to achieve their various functions. Pet. App. 150a, 215a, 227a-228a; *see* McNealy Br. §II. Of the creative choices described above (at 4-11), no particular expression was *necessary*. That is dispositive.

2. Copying Java SE's exact words and organization was not necessary for Google to express the ideas.

Google does not argue that this is the "rare instance" in which Java SE's authors had very limited ways to write a platform that achieves their result. Instead, it looks to the wrong author at the wrong time, arguing that "*Google's* engineering team … had no other choice." GB21 (emphasis added).

---

[4] The passage Google quotes merely uses "previously" and "later" to describe the inevitable temporal relationship *between author and copyist*. CONTU Report 20. CONTU then acknowledged that the "design of the Act of 1976 … was clearly to protect all works of authorship from the moment of their fixation." *Id.* at 21. Google omits that CONTU's discussion of merger reiterated that "one is not free to take another author's program" and that merger does not apply "[w]hen other language *is* available" to "achieve a certain result." *Id.* at 20.

No choice? Google had a choice to take one of the readily available licenses, as other platform developers did. *Supra* 12. It certainly had the resources to innovate and write its own code. A company facing an existential crisis for failure to innovate quickly enough may feel like it has no choice, but that is not the sort of need merger excuses.

As the Court of Appeals correctly explained, even looking to Google's choices there is no merger: It was not necessary for Google to copy Oracle's exact words and organization. Pet. App. 148a. Both courts below found as undisputed fact that "nothing prevented Google from writing its own declaring code, along with its own implementing code, to achieve the same result." Pet. App. 151a-152a; Pet. App. 215a (Google could have "offered in Android" "the very same functionality" embodied in Java SE "without duplicating"); *see also* Spafford Br. §III.E.

Google gives three shifting explanations of why it had no choice—why copying was "necessary."

a. First, without even acknowledging the contrary factual findings below, Google asserts that each declaration "can be written only one way," GB23, and that the declarations it copied "were the *only* instructions that could perform their functions." GB2; *see* GB14-15. That is false. Just look at what Apple and Microsoft did: They each wrote their own platforms, which provide app developers prewritten programs with much the same functionality as Java SE. Pet. App. 149a-150a n.5; *e.g.*, Pet. App. 165a n.14 (three different expressions for same idea of setting time zones).

b. Next, Google claims that copying was "required by the Java language" and that it copied "no more than what the Java language absolutely required." GB20; *see* GB3-4, 8, 47. Again, false. Google agreed that only 170 lines of code were necessary to write in the Java language, and those lines of code are no longer at issue in this case. *Supra* 14 n.2. And just look at what others did: The Spring platform, for example, uses the Java language with Spring's "own set of prewritten programs" and declaring code. JA299-300. Log4J is a competing package that uses the Java language to "solve exactly the same kinds of problems" as Java SE programs, but with "different class names, different method names, different interfaces, and different relationships." JA316, 412-413.

c. Google ultimately lands on another theory for why it was "absolutely required" to copy: to make Android more accessible to Oracle's loyal audience. GB20. By copying, Google could more readily capture the app developers who had come to know and love Java SE by letting them put their Java SE knowledge to use in Android. But that's just expedience, not the sort of "need" merger recognizes.

It is anathema to copyright law, and foreign to merger, to "treat the current popularity of [Oracle's] work among developers as retroactively divesting the work of copyright protection." U.S. Cert. Br. 14. No work (or part of a work) loses its protection just because the audience has expended effort to learn it. A director could not set his musical to leitmotifs from the iconic *Star Wars* score because the pit orchestra already knows them. This "popularity" argument for

software "would not be taken seriously if the copyrighted work[] were Steinbeck's *Grapes of Wrath*." Miller, *supra*, 1020. Equally misplaced are Google's arguments about denying protection to preserve app developers' "efforts … in learning how to use" Java SE. GB28. What an audience has learned, or grown to love, has no bearing on whether the work is protected or on merger.

Google reaches this improper result only by defining the idea of the work so narrowly as to be literally synonymous with Oracle's expression. Google says: "[o]nly one precisely written set of declarations will perform the function of responding to the corresponding calls *known to the developer*." GB20 (emphasis added). So, it argues, the "idea" it copied is "the function of responding properly to the developers' [Java SE] calls." GB21.

But, as Google acknowledges, the "calls" are lines of code that directly reference *Oracle's particular declarations*. GB5. *After* Java SE's authors chose to write and organize Java SE in their particular way and specified in the declaring code how to invoke each program, an app developer who wants to invoke a particular Java SE program will type those particular Java SE calls. Pet. App. 151a. But Google was free to write its own declarations and match them to its own calls to describe and invoke the same functionality without copying Java SE's declarations.

Google's view of "necessity" is hopelessly circular: *Once Google decided* to copy parts of Java SE that developers already knew, Google "had no other choice" but to copy Oracle's declaring code. GB21. Google says

there is only one set of "Java instructions that would do the same thing," GB21, but only if the "thing" is granularly defined as using Oracle's precise words and identical organization in exactly the same way. Google's "necessity" is born entirely of Google's own choice to plagiarize. When the ideas embodied in each method, class, package, and the whole platform are properly framed around what functionality each provides or purpose each achieves, *supra* 24-25, Google undisputedly did not need to copy.

By Google's logic, a plagiarist could define J.K. Rowling's idea as "a story about Harry Potter, Ron Weasley, and Hermione Granger who attend Hogwarts" and steal the characters and their back stories. Or she could market detailed knock-offs of bestsellers by declaring that she "had no other choice" but to reproduce verbatim the 11,300 most memorable sentences or scenes because they were "necessary" to allow fans to use their existing knowledge. *Contra Castle Rock Entm't v. Carol Publ'g Grp., Inc.*, 955 F. Supp. 260, 265-66 (S.D.N.Y. 1997) (Sotomayor, J.) (holding trivia book that copied events depicted in *Seinfeld* episodes violated copyright protection), *aff'd*, 150 F.3d 132 (2d Cir. 1998). The idea in "any work" can be described at different degrees of "generality." *Nichols*, 45 F.2d at 121. If the idea embodied in a work is circularly defined as its particular expression, then merger would *always* apply.

Contrary to Google's assertion, GB30-31, *Baker* does not condone that sort of circularity. *Baker* merely "held that a copyrighted book on a peculiar system of bookkeeping was not infringed by" forms that "made

a different arrangement of the columns and used different headings." *Mazer*, 347 U.S. at 217. There, no one could ever have achieved results like Selden's accounting method without forms having analogous arrangements of "columns and headings." *Baker*, 101 U.S. at 100. *Baker* would be like this case if Oracle claimed the right to prohibit platform developers from using packages, classes, and interfaces to organize methods. But Oracle is not doing that. Rather, Oracle created its own specific design and then filled in the blanks—30,000 times over—and seeks to protect only *that* fully realized expression.

3. Google's proposed interoperability exception is misplaced and inconsistent with the Act.

Much of Google's merger argument revolves around the policy assertion that copying is "critical" to achieve compatibility "between and among programs, platforms, and … devices." GB28. But it is undisputed that Google designed Android to *defeat* compatibility: Apps written for Java SE cannot run on Android, and vice versa. Pet. App. 46a n.11, 172a. What Google really wanted was to give Android a boost in the market by offering a "Java-like" platform without accepting any of the license conditions that ensure *actual* compatibility ("write once, run anywhere") or code sharing (the open-source license's give-back requirement). Google's self-interested choice cannot justify denying copyright protection to the original.

Indeed, the Act's text defeats Google's proposal. Congress wrestled with software "interoperability"

and decided not to create a carveout on that basis. *See* §117 ("limitations" on computer program copyrights). Separately, §1201(f) creates a special interoperability defense to a charge of software hacking, but only if the circumvention does "not constitute infringement." In other words, Congress *presumed* copyrightability and crafted an interoperability safe harbor in circumstances even Google does not contend are applicable here. Google cannot undo Congress's choices by rewriting the judge-made merger doctrine to suit its business interests.

II. Google's Superseding Use Of Oracle's Copyrighted Work Was Not Fair Use.

No court has found fair use where, as here, someone copied so much valuable expression into a competing product to serve the same purpose as the original in the marketplace. That is because the "doctrine has always precluded a use that 'supersede[s] the use of the original.'" *Harper & Row Publ'rs, Inc. v. Nation Enters.*, 471 U.S. 539, 550 (1985) (quoting *Folsom v. Marsh*, 9 F. Cas. 342, 344-345 (C.C. Mass. 1841) (Story, J.)).

Congress enacted §107 to codify this judicial doctrine. *Campbell v. Acuff-Rose Music, Inc.*, 510 U.S. 569, 577 (1994). It directed courts to determine fair use by comparing the challenged use to classic fair uses and assessing four historically significant factors listed in §107. Google does not even suggest its use resembles any of the statutory examples—"criticism, comment, news reporting, teaching ..., scholarship [and] research." This alone militates against fair use.

Moreover, undisputed facts establish that the statutory factors favor Oracle and that Google committed an unfair superseding use.

A. The Court of Appeals applied the correct standard of review.

This Court has established that review is *de novo*. *Harper & Row* held that "[f]air use is a mixed question of law and fact." 471 U.S. at 560. Where, as here, the record contains "facts sufficient to evaluate each of the statutory factors, an appellate court … may conclude as a matter of law that the challenged use does not qualify as a fair use." *Id.* (quotation marks and brackets omitted). *Harper & Row* could not have so held unless the ultimate question of fair use were a legal question a court resolves without deference.

De novo review also follows from *U.S. Bank National Association v. Village at Lakeridge, LLC*, 138 S. Ct. 960, 967-68 (2018). *Accord* Pet. App. 16a-19a. Evaluating fair use "entails primarily legal work," *U.S. Bank*, 138 S. Ct. at 967, as Congress enacted §107 to preserve the "judicial doctrine" where "courts … ruled upon the fair use doctrine over and over again," H.R. Rep. No. 94-1476, at 65-66 (1976). Those courts routinely determined "*in point of law*" whether the use "constitute[s] a piracy." *Folsom*, 9 F. Cas. at 348 (emphasis added). Similarly, fair use entails "developing auxiliary legal principles," which is why this Court has issued four opinions—*Harper & Row*, *Campbell*, *Stewart v. Abend*, 495 U.S. 207 (1990), and *Sony Corp. of America v. Universal City Studios, Inc.*, 464 U.S. 417 (1984)—"elaborat[ing]" on fair use. *U.S. Bank*, 138 S. Ct. at 967-68. Courts "most frequently"

resolve fair use as a matter of law "at summary judgment" (like *Stewart*). *TCA Television Corp. v. McCollum*, 839 F.3d 168, 178 (2d Cir. 2016). Even Google relied on generalized legal issues in its petition and presses them here, such as the standard for transformative use, whether a copyist's good faith supports fair use, and the standard for harm to potential markets, GB38-39, 47, 49-50; Pet. 24-29.

The Court of Appeals correctly identified the effect of a jury verdict. Pet. App. 22a-25a & n.4. First, the court defers to the factfinder's role by accepting any genuinely disputed historical fact that favors the verdict. Second, it considers those presumed facts—along with any "uncontradicted and unimpeached" evidence, *Reeves v. Sanderson Plumbing Products, Inc.*, 530 U.S. 133, 151 (2000)—to determine whether a use "further[s] [the Copyright Act's] essential purpose," GB37. *See Harper & Row*, 471 U.S. at 569 (addressing fair use *de novo* using district court's factual findings).

Google cannot overcome this body of precedent by declaring that the opinion below is "unprecedented," just because there was a jury trial. GB35. Fair use jury trials are almost unheard of, precisely because fair use so rarely revolves around disputed facts. And a trial does not turn a legal question into a factual one. If Google wanted this Court to overrule cases establishing the standard of review, it needed to do more than baldly assert a different standard. GB37, 42.

The Court of Appeals correctly found no fair use as a matter of law because undisputed facts foreclosed Google's defense.

B. Google's copying is an unfair superseding use.

1. Factor one: Google's use was commercial and for the same purpose as Oracle's.

The first factor—the "purpose and character of" Google's copying—weighs against fair use, based entirely on undisputed facts.

A commercial use weighs against fair use when the copyist "stands to profit from exploitation of the copyrighted material." *Harper & Row*, 471 U.S. at 562. Far from "educational" or "nonprofit," §107(1), Google's purpose in copying was to rush a product to market to protect its competitive position. It has collected $42 billion (and counting). JA463-464. It does not get more "commercial" than that. That is why Google admitted it copied Oracle's original expression for "purely commercial purposes." Pet. App. 182a. Nothing Google now says—about Google's additional purposes or how much Oracle might (or might not) have profited but for the infringement, GB43-44—changes the fact that Google profits enormously from the use.

Factor one, therefore, favors Oracle, unless Google meaningfully transformed Oracle's expression. *See Campbell*, 510 U.S. at 579. It did not.

a. A use is not transformative unless it "adds something new, with a further purpose or different character, altering the first with new expression, meaning, or message." *Id*. Google could not transform what it copied just by "adding something new." Google fell far short of transforming Oracle's code, because, it concedes, every line of code it copied has the same meaning, and serves "the same purpose," in Android as in Java SE. GB45; Pet. App. 31a-33a; JA627-630. The whole point was to appeal to Java SE's audience with material that was familiar.

Courts must enforce the rule that transformation requires a change in expression or meaning, because crucial rights depend on it. Congress explicitly granted authors, including software authors, the exclusive right "to prepare derivative works." §106(2). A "'derivative work' is a work based upon … [a] preexisting work[]" in "any … form in which a work may be recast, *transformed*, or adapted." §101 (emphasis added). A classic derivative work is a sequel or a movie adapted from a book. For software, the quintessential derivative is a "version" adapted for the next generation of devices or different operating systems. *See, e.g.*, U.S. Copyright Office, Compendium of U.S. Copyright Office Practices §721.8 (3d ed. 2017), https://tinyurl.com/y742m3zn. The kinds of changes that qualify as transformative for fair use must be different in kind from the transformation inherent in adapting a work to create a derivative, or else fair use would swallow the derivative-work right.

This Court has accordingly drawn a consistent line between transformative works and derivative

works. On the one hand, it is not fair to create a derivative work, such as adapting a short story into a movie (*Stewart*), serializing a memoir into excerpts (*Harper & Row*), or recasting a pop song into a rap cover (*Campbell*). On the other hand, it may be fair to create a parody or a criticism. The difference is "there is no protectible derivative market" for such works as authors rarely pillory their own works. *Campbell*, 510 U.S. at 592.

Google copied Oracle's code to create a nontransformative derivative: a sequel that adapted Oracle's software for an improved generation of devices. Congress granted Oracle alone the right to create or license such a sequel. Just because Google had the resources to crank out the sequel more quickly does not make it fair.

b. Given Google's concession that it did not change any meaning or purpose, it tries to change settled law instead. Google argues Android is transformative because "the new work as a whole … added something new to the computing world." GB45. What Google means is that it moved the code from the context of computers (desktops and laptops) to smaller computers (tablets and smartphones). GB43. The Court of Appeals correctly rejected that argument for two independent reasons.

The first is a dispositive point that Google fails to address: The court held that Google's premise—that Java SE was not in those smaller computers—is false. The Court of Appeals pointed to undisputed facts establishing that "Java SE was already being used in

smartphones." Pet. App. 35a. *Google* witnesses testified that the Danger smartphone (T-Mobile Sidekick) was equivalent to early Android smartphones and used Java SE. JA359-360, 369-370; *see* JA430-432. SavaJe (the smartphone platform) and the Amazon Kindle (a tablet) both used Java SE too. Pet. App. 50a. That means Google did not put Android into a new context.

Second, Google's legal conclusion is wrong. To start, Congress directed courts to consider just "the [infringing] *use*" for factor one while another factor (three) considers "the copyrighted worked *as a whole*." §107 (emphasis added); *see* Pierre N. Leval, *Toward a Fair Use Standard*, 103 Harv. L. Rev. 1105, 1112 (1990) ("Courts must consider the question of fair use for each challenged passage and not merely for the secondary work overall."). Thus, the Court of Appeals correctly "focused on whether the *reused material* itself was transformed rather than on whether the *new work as a whole*" was different. GB45.

Moreover, as explained (at 40-41), neither "adding something new" nor putting the code in a new context is transformative, unless the code's meaning or purpose changes. *See TCA Television*, 839 F.3d at 182. Otherwise, transformative use would swallow the derivative-work right because every derivative work "adds something new." *See Kienitz v. Sconnie Nation LLC*, 766 F.3d 756, 758 (7th Cir. 2014). Movies, for example, convert books' descriptions and prose to images and dialogue, just as Google updated the implementing code for resource-constrained devices. GB43. Movies commonly take snippets of the original and

add new material. And movies require significant innovation. *E.g.*, Nick Clark, *How did they bring the 'unfilmable' Life of Pi to our screens?*, Independent, Dec. 8, 2012, https://tinyurl.com/tr8y562. But producers cannot assert that their added innovation insulates them from infringement.

Ultimately, Google's argument devolves into yet another atextual assertion that Oracle's code must be treated differently because it is "impossible to reuse declarations in a software environment for a different function." GB45. Google does not explain why that concern justifies special treatment only for declarations, as all code is functional. In any event, Congress decided that §107's fair-use inquiry is the same for all works. Besides, there are plenty of transformative uses for Oracle's declaring code that align with the classic uses listed in §107, such as copying to teach, analyze, or critique code, develop a tool for detecting code plagiarism, or research how to make an interoperable program that does not itself infringe. *Sony Comput. Entm't, Inc. v. Connectix Corp.*, 203 F.3d 596, 606-07 (9th Cir. 2000).

Here, however, Google took popular, recognizable expression and used it "to get attention" and "avoid the drudgery in working up something fresh." *Campbell*, 510 U.S. at 580. That is not transformative.

2. Factor two: Google copied creative and expressive portions of Oracle's work.

Factor two considers whether the nature of the copyrighted work is one that Congress intended to incentivize. *Campbell*, 510 U.S. at 586. When Congress defined software as a "literary work," it left no doubt that it intended copyright to incentivize software creation. *Supra* 20-21; *see Miller*, *supra*, 983.

If ever there were software that deserves robust protection, it is the code Google copied. All the creative choices documented above (at 4-11) were directed at appealing to an audience and making the code memorable. That was why Google's Java guru described crafting the code as "an art, not a science." Pet. App. 41a; JA521. Congress could not have intended for code shaped by expressive considerations to receive less protection than implementing code developers never see. If anything, code specifically designed to communicate to people deserves *greater* protection. SAS Inst. Br. §I.

Google ignores all this when it asserts that "the declarations were functional, not creative." GB46. It is not either/or. Functionality alone cannot be dispositive because all software is functional. §101. Besides, functional works can be highly creative. *Harper & Row*, 471 U.S. at 563 (biography). And even minimally creative functional works receive protection against copying their original elements. *Feist*, 499 U.S. at 348.

3. Factor three: Google's copying was substantial.

Factor three favors Oracle because the "quality and … quantity" of material Google copied was significant. *Campbell*, 510 U.S. at 586-87; Pet. App. 45a-47a. Google admits it copied the packages most valuable to create a derivative version of Java SE for mobile devices, GB7—"central" and "important" Java packages, *supra* 14. It did not copy some trivial line of code with no expressive value. *Cf. Lexmark Int'l, Inc. v. Static Control Components, Inc.*, 387 F.3d 522, 544 (6th Cir. 2004).

It makes no difference that Google copied a fraction of the code in a very large work or that the code was "scattered within the copyrighted work." GB47. Copying 11,330 lines of code is a lot regardless of the overall size of the work. Moreover, statistics cannot trump quality: *Harper & Row* found infringement where the copyist copied scattered passages amounting to merely 0.15% of the original. 471 U.S. at 569; *id.* at 598 (Brennan, J., dissenting).

Google cannot disprove the value of what it copied by arguing that the implementing code is *also* important or that it alone "carr[ies] out the functionality." GB47. Both are essential for the computer to achieve a result. *Supra* 5. Moreover, that major companies license the declaring code without the implementing code, *supra* 12, disproves Google's assertion that "declarations … have no value independent of the implementing code," GB47.

4. Factor four: Google's concededly "competing" product harmed Java SE in actual and potential markets.

The "effect of the use upon the potential market for or value of the copyrighted work" "is undoubtedly the single most important" factor. *Harper & Row*, 471 U.S. at 566 (quotation marks omitted). It considers any effects on the market for the original work and derivative works, as well as whether unrestricted and widespread conduct like the defendant's would cause harm. *Id.* at 568-69; *Campbell*, 510 U.S. at 590. Relying only on facts Google did not dispute—and still refuses to address—the Court of Appeals properly held factor four decisively favors Oracle. Pet. App. 47a-53a.

Current markets. Google packaged Oracle's code into its own product and then appealed to Oracle's customers touting those "Core Java Libraries" as a selling point. JA590-591. The court found *undisputed* the following harm:

- Android's chief admitted Android and Java SE are "competitor[s]," "targeting the same industry with similar products." JA365-366.

- Amazon switched from the Java platform to Android, then leveraged its ability to use Android for free to secure a 97.5% price concession from Oracle. *Supra* 15.

- Smartphone platform SavaJe licensed Java SE, and Google admitted Android harmed it. *Supra* 11, 15; JA584.

Although Google points to *other* supposedly "disputed factual questions," it does not contest these *undisputed* facts, GB48-49—any one of which is dispositive. *See Harper & Row*, 471 U.S. at 567 (single lost license).

Instead, Google makes *legal* arguments that are irrelevant and unaccompanied by authority. Factor four does not weigh the illusory "respects in which Android *benefitted* Oracle." GB49. A filmmaker cannot excuse ripping off a novel just because the film increased book sales. "If the defendant's work adversely affects the value of any of the rights in the copyrighted work … the use is not fair." *Harper & Row*, 471 U.S. at 568 (quotation marks omitted).

Nor could a jury refuse to find market harm because Oracle licensed a free open-source version of Java SE, OpenJDK. GB49. It is undisputed that *Android*—not OpenJDK—caused each of the *commercial* harms described above. Pet. App. 51a n.13; JA407-409, 447-448. Oracle's commercial customers switched to Android, not OpenJDK, and leveraged Android, not OpenJDK, for 97.5% discounts. Companies (like Google and others) considered OpenJDK's give-back terms "unacceptable." *Supra* 12-13. Moreover, Google's argument that offering an open-source option makes *any* copying fair would effectively make such licensing terms unenforceable.

Potential markets. The Court of Appeals also cited only *undisputed* evidence in finding that Oracle suffered harm to potential markets, Pet. App. 51a-52a, which alone suffices, *see Stewart*, 495 U.S. at 238. Since smartphones were existing markets, not just

"potential" ones, for Oracle, the question is whether the *next* generation of smartphone platforms—for even "smarter" smartphones—was a potential market for Oracle. Was it a market that Oracle would "in general develop or license others to develop"? *Campbell*, 510 U.S. at 592.

It was, for the undisputed reasons the court gave: (1) Oracle licensed Java SE to SavaJe for that exact market and (2) the "lengthy licensing negotiations" with Google "demonstrate[d] that Oracle was attempting to license its work for" next generation "smartphones." Pet. App. 51a-52a & n.14. These were concrete steps to enter the emerging market—not some "mere *wish*." GB49; *see* 4 Nimmer §13.05 ("use is not fair, even if plaintiff has not yet exercised that right"); *Campbell*, 510 U.S. at 592 (original author has the exclusive right to "license others to develop" derivative works); Copyright Alliance Br. §IV (discussing "potential market").

Widespread use. Finally, "to negate fair use one need only show that if the challenged use should become widespread, it would adversely affect the potential market for" Java SE. *Harper & Row*, 471 U.S. at 568 (quotation marks and emphasis omitted). If what Google did was permissible, IBM, Danger, and others would not have licensed Sun's declaring code or complied with "write once, run anywhere." If everyone could copy the declaring code without a license, Java SE would lose value, as anyone could "reimplement" a knock-off. JA399. This undisputed evidence negates Google's defense as a matter of law.

***

All the factors lead to the same conclusion: Google's copying is a classic, unfair superseding use. And that conclusion would not change even if some factors favor Google. *See* Pet. App. 53a-54a; *Castle Rock*, 955 F. Supp. at 272.

C. Google's additional considerations cannot establish fair use.

Tellingly, Google leads fair use with arguments divorced from §107 and copyright's purpose of ensuring that copyright owners receive "a fair return for their labors." *Harper & Row*, 471 U.S. at 546. It ignores this Court's admonition against giving "insufficient deference to the scheme established by the Copyright Act for fostering … original works." *Id.* at 545-46, 560. Google's arguments are legally irrelevant to fair use and are addressed by other doctrines.

Industry "practice." Google starts—and infuses its brief—with an assertion that is irrelevant and wrong: that there is a settled "practice of software companies and developers … reimplementing declarations" *without permission*. GB37-38; *see* GB2, 26-27. Google tried to prove such a practice with expert testimony. But the district court excluded Google's expert for trying to prove an industry practice of *unlicensed* copying with evidence of *licensed* copying. JA470.

The record evidence proves Google wrong. Sun/Oracle insisted on licensing the declaring code. JA511-516. Major companies licensed just the declaring code and the organization. *Supra* 12. Sun policed unlicensed uses. JA611-612, 448-450. Before joining

Google, Android's creator licensed the declaring code for Danger's "Java … SE" "implementation." JA370-371, 436-437. Even Google claims copyright protection in its own search and advertising declarations, JA561-583, and prohibits copying that "compete[s] with [its] products or services," JA561.

Stepping away from the record (as Google and its amici do), no one in the industry could have thought it "settled" that unlicensed reimplementations were lawful. Numerous cases found infringement for copying computer code and the organization of computer programs. Oman Br. §II. In a high-profile case, Microsoft paid for a *license* to reimplement Java SE, but then violated the compatibility requirements in "a concerted effort" to undermine "write once, run anywhere." *Sun Microsystems, Inc. v. Microsoft Corp.*, 87 F. Supp. 2d 992, 1000 (N.D. Cal. 2000). Sun won an injunction. *Id.* at 1006-07; *see also* McNealy Br. §III (discussing industry practice).

In any event, "everybody does it" is not a viable defense. Contrary to Google's assertion (GB37-38), the common law did not support that view. At common law, an "author's reasonable consent" was not some separate factor, but a shorthand to describe uses, unlike Google's, that do not "supplant the market for or value of the original." William F. Patry, *The Fair Use Privilege in Copyright Law* 17 (1985). This theory is also no basis to salvage the verdict because the district court did not instruct the jury on it. *E.g.*, *Country Shindig Opry, Inc. v. Cessna Aircraft Co.*, 780 F.2d 1408, 1413 (8th Cir. 1986); *see* JA283.

That does not mean a copyright owner may lull others into infringing by creating settled expectations that copying is permissible. GB14. Implied license and waiver of copyright protection address those situations—not fair use. The district court rejected those defenses, Pet. App. 273a-276a; *see also* JA356-367, 532-534, 538 (Sun's CEO). Google never appealed. It cannot now smuggle them into fair use.

Google's argument that a settled practice would confirm Google's "good faith" is also legally flawed. GB38-39. At common law, and under the Copyright Act, good faith does not "operate[] as a legal defense." Patry, *Fair Use*, *supra*, 11 & n.22; *see* 4 Nimmer §13.08[B][1] ("the innocent intent of the defendant constitutes no defense to liability."); *accord* Leval, *supra*, 1126-27 (good faith irrelevant to fair use).

Compatibility & lock-in. Google is wrong to argue that rejecting fair use would "empower Oracle … to prevent anyone from developing a product compatible with [its] software interfaces." GB40. First, that interest could not justify *Google's* copying because Google designed Android to be *in*compatible with Java SE. *Supra* 14. Second, claims of interoperability cannot excuse Google's copying to create a market substitute. Fair use always precludes a superseding use. *Supra* 36. Moreover, Congress decided that conduct promoting interoperability can be "infringement," §1201(f), and did not give special treatment to copying for interoperability, which is particularly probative because Congress granted special treatment to other copying of software, *e.g.*, §117.

Moreover, Oracle did not "prevent anyone from developing … compatible" products. For all Google's talk about "lock-in," GB30, 40, and "deter[ring] competition," GB40, it presented no evidence of either below. That is because the very notion is not only false, but downright hypocritical. Oracle liberally licensed its work to platform developers, including its competitors, so long as they maintained compatibility—the exact opposite of lock-in. Google could have taken a *free* license, but it did not want to "give back" any improvements it and its customers made. Google could have taken a license just to the declaring code and organization, for a fraction of the billions it has made on Android. But Google *refused* the non-negotiable condition Sun required for the benefit of app developers and the public: to maintain cross-platform compatibility. *Supra* 13.

In any event, fair use is not the doctrine for addressing concerns about abusive licensing practices, "deter[ring] competition, creating … insurmountable barriers," etc. GB40. Copyright misuse and antitrust address such practices. *Chamberlain Grp., Inc. v. Skylink Techs., Inc.*, 381 F.3d 1178, 1201 (Fed. Cir. 2004).

Developer knowledge & expression. Google also reframes in fair-use terms its argument that it copied to enable app developers to use "their existing knowledge." GB40. But that is simply another way of saying Google copied the code for the same purpose Oracle created it, which is not fair. *Supra* 39-43. Copyright law does not excuse copying based on a desire to appeal to what the audience knows about the orig-

inal, such as a concert conductor who wants to perform songs his orchestra already knows (at 32) or an author who wants to write a *Seinfeld* trivia book based on familiar dialogue (at 34).

As all these examples illustrate, Google has to defend *its own* unlicensed copying—that is the "use" at issue. §107(1), (4); *see Infinity Broad. Corp. v. Kirkwood*, 150 F.3d 104, 108 (2d Cir. 1998). It may not justify its copying based on what it believes would appeal to the target audience. That is the epitome of using another work "to get attention" and "avoid the drudgery in working up something fresh." *Campbell*, 510 U.S. at 580.

In any event, Google cannot base its fair-use case on an imperative to protect app developers from having "to learn thousands of new calls." GB27. Google replaced entire packages that app developers had learned with "thousands of new methods." GB8.

Google does not change that result by characterizing Oracle's position as a "bait and switch." GB28. Part of Java SE's appeal to developers was the "write once, run anywhere" promise. That meant run anywhere in the Java SE universe—a universe built on license agreements requiring compliance with Oracle's compatibility terms. *Supra* 11-12. It was never an assurance that developers should expect their knowledge to transfer when they write programs for platforms outside the universe, like for Apple's iOS. It certainly was not a promise that knowledge would transfer seamlessly to an unlicensed platform "specifically designed" to break the write once, run anywhere promise. Pet. App. 46a n.11.

Google also does not change the result by insisting that its copying "unleashe[d] enormous innovation." GB40. Java SE was already unleashing torrents of innovation before Android. Google presented no evidence that its copying unleashed expression that would not have otherwise materialized—particularly if Google had taken a license or written its own code. Meanwhile, Google broke write once, run anywhere, which was designed to aid app developers—and, ultimately, consumers.

Expression *at the expense* of markets for the original is not the sort of "creativity which th[e] law is designed to foster." GB37 (quotation marks omitted). Releasing a pirated copy of Adobe Photoshop would unleash innovation. Yet no one would consider that fair use. This Court has rejected the notion "that fair use be imposed whenever the social value of dissemination … outweighs any detriment to the artist" because that view negates the original incentive for creation. *Harper & Row*, 471 U.S. at 559 (quotation marks and brackets omitted). This Court has also condemned any notion of "judicially imposing, a 'compulsory license.'" *Id.* at 569. The Copyright Act does not condone stifling the original author's incentives just to make it more convenient for others to create.

III. Google's Policy Arguments Are Misplaced And Misguided.

Ultimately, Google tries to justify rewriting the Copyright Act based on unsupported dire predictions about the future of "interoperable computer software and American technological progress." GB2. Congress, the body responsible for weighing such policy

concerns, already had this debate. It made the considered judgment to treat software as a "literary work" and not to create a carveout for interoperability. Settled law was—and is—that "interfaces are treated no differently from other program features for copyright purposes." Miller, *supra*, 1032.

Even if that judgment required updating, only Congress has the institutional capacity and expertise to do so. This Court should reject Google's bid to launch an era of judicial line-drawing to decide what code qualifies as an "interface" and how to adapt the governing rules.

There is, however, no problem to fix. By adopting the same approach to software as other literary works, Congress preserved the flexibility inherent in copyright doctrine, which protects authors' original expression, regardless of format, while allowing later uses of existing works in ways that do not supersede the objects of the original. It is under the *current* rules and incentives that the U.S. software industry has enjoyed its meteoric rise. *See* SAS Inst. Br. §III. The six years since the Court of Appeals' copyrightability decision have brought new bursts of innovation—cloud computing, machine learning, artificial intelligence, autonomous vehicles, and 5G. In contrast, neither Google nor its amici cite a single real-world example of any innovation being chilled.

Why then do Google's amici express concerns about innovation?

Some incorrectly think that copyright protection will undermine open-source licenses. In fact, such licenses *depend* on copyright protection. *Jacobsen v. Katzer*, 535 F.3d 1373, 1378-79 (Fed. Cir. 2008).

Others overread the opinions below as holding that it is impermissible to copy anything one might term an "interface" (whether original or not) or that any copying of code is unfair, regardless of the use. The Court of Appeals' opinions, however, focused narrowly on Google's verbatim copying of a concededly original work into a competing commercial product. Pet. App. 53a-54a. Copying unoriginal elements or copying for research purposes—to figure out how a program works and to create an interoperable product that contains no copied code are unaffected by the opinions. *E.g.*, *Sony Comput.*, 203 F.3d at 606-07 (distinguishing such uses). The opinions preserve this existing flexibility to adapt for new innovations and new uses.

Licensing agreements can and do meet industry demands for platforms that are free to reuse. Miller, *supra*, 1031; *cf. Harper & Row*, 471 U.S. at 566 n.9 ("permitting 'fair use' to displace normal copyright channels disrupts the copyright market without a commensurate public benefit"). Developers offer open-source licenses because it is in their business interest. Market forces likewise foster interoperability. Consumers demand products that work together, so software vendors "wall off" their products at their peril. *See* MathWorks Br. §I.D. Those market forces confirm the wisdom of Congress's judgment that innovation is best served by *not* letting plagiarists "tell copyright

holders the best way for them to exploit their copyrights." *Sony Corp.*, 464 U.S. at 446 n.28.

In sharp contrast to the unfounded fears of Google's amici, this case is an object lesson on software innovation. Oracle's groundbreaking work flourished due to its elegance and cross-platform compatibility. Manufacturers and platform developers respected Oracle's copyright and licensed the platform, building innovation on innovation. Google had that option. Google also had the option of innovating, as Apple and Microsoft did. It tried, but ran out of time.

No company will make the enormous investment required to launch a groundbreaking work like Java SE if this Court declares that a competitor may copy it precisely because it has become so popular, or because it is functional—like all computer code. *See* Synopsys Br. §III. The only winners under that regime will be monopolists and corporate giants with resources so vast that they can always beat the startup to release their superseding version, and achieve market penetration so deep as to occupy the terrain before the original ever catches up. *See, e.g.*, Daisuke Wakabayashi, *Prime Leverage: How Amazon Wields Power in the Technology World*, N.Y. Times, Dec. 15, 2019, https://tinyurl.com/vsjhwc3. Therein lies a deep irony. Historically, fair use protected the public, and those with limited power, from the overwhelming dominance of monopolies. Google's view of fair use shifts all the power in the other direction. *See* Copyright Alliance Br. §V; IP Profs. Br. §III.C.

Only by ignoring the record here and the past 40 years of empirical market history could Google even suggest that strong copyright protection for software threatens U.S. leadership in software innovation. Diminishing that protection on the false premise of some settled expectation that everyone is free to copy the innovation of others will only encourage piracy, here and abroad. Rather, it is *because* our legal system rewards innovation with robust copyright protection that the United States is the world leader. We cannot credibly insist on strong protections abroad while abandoning them at home. Taking a cue from the Framers, Congress's view has always been that the United States wins by rewarding authors' creativity with "the exclusive Right to their respective Writings." U.S. Const. art. I, §8, cl. 8. Neither Congress nor the courts have ever rewarded the party that plagiarizes because it was too desperate to innovate itself.

CONCLUSION

The Court should affirm the judgment of the Court of Appeals.

Respectfully submitted,

Dorian E. Daley
Deborah K. Miller
Matthew M. Sarboraria
Andrew C. Temkin
ORACLE AMERICA, INC.
500 Oracle Parkway
Redwood Shores, CA
94065

Dale M. Cendali
Joshua L. Simmons
Jordan M. Romanoff
Ari E. Lipsitz
KIRKLAND & ELLIS LLP
601 Lexington Avenue
New York, NY 10022

E. Joshua Rosenkranz
Counsel of Record
Annette L. Hurst
Peter Bicks
Lisa T. Simpson
Andrew D. Silverman
Matthew R. Shahabian
Jeremy Peterman
Hannah Garden-Monheit
Geoffrey Moss
ORRICK, HERRINGTON &
SUTCLIFFE LLP
51 West 52nd Street
New York, NY 10019
(212) 506-5000
jrosenkranz@orrick.com

Date: February 12, 2020