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 THE PETITIONER

Kent Walker
Catherine Lacavera
Renny Hwang
GOOGLE LLC
1600 Amphitheatre Parkway
Mountain View, CA 94043

Lisa S. Blatt
David M. Krinsky
Sarah M. Harris
Meng Jia Yang
WILLIAMS & CONNOLLY LLP
725 Twelfth Street, N.W.
Washington, DC 20005

Robert A. Van Nest
Christa M. Anderson
Eugene M. Paige
Reid P. Mullen
KEKER, VAN NEST
& PETERS LLP
633 Battery Street
San Francisco, CA 94111

Thomas C. Goldstein
Counsel of Record
Sarah E. Harrington
Kevin K. Russell
Daniel Woofter
Erica Oleszczuk Evans
GOLDSTEIN & RUSSELL, P.C.
7475 Wisconsin Avenue
Suite 850
Bethesda, MD 20814
(202) 362-0636
tg@goldsteinrussell.com

Michael S. Kwun
KWUN BHANSALI LAZARUS LLP
555 Montgomery Street
Suite 750
San Francisco, CA 94111

Bruce W. Baber
Marisa C. Maleck
KING & SPALDING LLP
1180 Peachtree Street, N.E.
Atlanta, GA 30309

QUESTIONS PRESENTED

The Copyright Act provides that, while "original works of authorship" are generally eligible for copyright protection, 17 U.S.C. § 102(a), "[i]n no case does copyright protection for an original work of authorship extend to any idea, procedure, process, system, method of operation, concept, principle, or discovery, regardless of the form in which it is described, explained, illustrated, or embodied in such work," *id.* § 102(b). The Act also makes clear that "the fair use of a copyrighted work . . . is not an infringement of copyright." *Id.* § 107.

As is relevant here, software interfaces are lines of computer code that allow developers to operate prewritten libraries of code used to perform particular tasks. Since the earliest days of software development, developers have used interfaces to access essential tools for building new computer programs. Contravening that longstanding practice, the Federal Circuit in this case held both that a software interface is copyrightable and that petitioner's use of a software interface in a new computer program cannot constitute fair use as a matter of law.

The questions presented are:

1. Whether copyright protection extends to a software interface.

2. Whether, as the jury found, petitioner's use of a software interface in the context of creating a new computer program constitutes fair use.

RULE 29.6 STATEMENT

Petitioner Google LLC is an indirect subsidiary of Alphabet Inc., a publicly held company. Alphabet Inc. has no parent corporation, and no publicly held company owns 10% or more of its stock.

TABLE OF CONTENTS

TABLE OF AUTHORITIES

Cases

Statutes

Regulations

Rules

Other Authorities

BRIEF FOR THE PETITIONER

Petitioner Google respectfully requests that the judgment of the United States Court of Appeals for the Federal Circuit be reversed.

OPINIONS BELOW

The Federal Circuit's opinion regarding fair use (Pet. App. 1a-55a) is reported at 886 F.3d 1179. The District Court's orders denying respondent's motions for judgment as a matter of law (Pet. App. 92a-120a) and for a new trial (*id.* at 56a-91a) are unreported. The Federal Circuit's opinion regarding copyrightability (*id.* at 121a-192a) is reported at 750 F.3d 1339. The District Court's order granting petitioner's motion for judgment as a matter of law (Pet. App. 212a-272a) is reported at 872 F. Supp. 2d 974.

JURISDICTION

The court of appeals entered its judgment on March 27, 2018, and denied rehearing on August 28, 2018. Pet. App. 283a-284a. On October 23, 2018, the Chief Justice extended the time to file a petition for a writ of certiorari to January 25, 2019. The petition was filed on January 24, 2019, and was granted on November 15, 2019. The jurisdiction of this Court is invoked under 28 U.S.C. § 1254(1).

RELEVANT STATUTORY PROVISIONS

The relevant statutory provisions are reproduced as an Addendum to this brief.

STATEMENT OF THE CASE

To create an operating system for smartphones, petitioner Google wrote millions of lines of new computer code. Google also reused isolated instructions

from a work copyrighted by respondent Oracle, because those were the *only* instructions that could perform their functions. The instructions create an interface connecting the operating system to commands in applications written by software developers.

The Federal Circuit both reversed the District Court's holding that Oracle had no relevant copyright interest and overturned a jury's finding that Google's conduct was fair use. As reflected in the broad *amicus* support for Google, the Federal Circuit's rulings would upend the long-settled practice of the computer software industry of reusing software interfaces. That practice has spurred the rapid development of interoperable computer software and American technological progress in the digital era.

I. FACTUAL BACKGROUND

A. Android And The Java Programming Language

1. The Android operating system. Smartphones are now "such a pervasive and insistent part of daily life that the proverbial visitor from Mars might conclude that they were an important feature of human anatomy." *Riley v. California*, 573 U.S. 373, 385 (2014). But even recently, that was not true. Early mobile phones were much less useful, in part because many manufacturers used their own proprietary "operating systems"—*i.e.*, software that controls the phone—for which few useful applications were created. *See* JA138-139; JA219; JA225-226; Trial Exhibit (TX) 7238 at 13.

Google responded by creating Android, an "open source" operating system that worked with almost any smartphone. JA137-139; JA242; TX1 at 4-8; TX6 at

5-6, 23-24. Many phone manufacturers would adopt the operating system, Google recognized, leading many developers to write useful applications—such as programs for messaging, navigation, news, and social media. *See ibid.*

Google designed Android so that developers who knew the "free and open" Java programming language could write applications. Pet. App. 102a. Roughly six million developers already had Java programming expertise. JA228. This case arises from the requirements imposed by the Java language for Google to permit Java developers to write applications using thousands of common commands they already know.

2. Tools of the Java programming language: calls, methods, and declarations. Java developers depend on thousands of commands known as "calls." The developers become expert in using calls through formal training and years of experience. JA193-194. The developers have the unfettered right to use the calls, which are not copyrighted. Dist. Ct. Docket (Doc.) 853 at 5:20-22.

A developer uses a call to invoke—*i.e.*, use—separate, pre-written computer code called a "method." Pet. App. 217a-218a, 228a. Each method performs a basic function—for example, determining the larger of two integers. *See infra* at 5-6. The methods are thus "interoperable" with any application that uses the correct call.

This approach to programming—which is used by every major programming language—makes it unnecessary for developers to create their own versions of the method's pre-written computer code. In addition, when that code is later improved or adapted to work

with new technology, developers need not go back and change their applications.

Java developers use multiple calls to invoke methods in every application; the language will not work without them. *See, e.g.*, JA213-214; 2016 Trial Transcript (Tr.) 1463:3-1464:19; TX9223. Developers call numerous methods, using them as building blocks to create sophisticated applications. Pet. App. 228a; *id.* at 125a.

A call will not work correctly unless it corresponds precisely to instructions called "declarations." There are more than ten thousand calls in Java, each with corresponding declarations. The declarations perform two functions that are relevant here.[1]

First, declarations create the system that organizes the methods. Pet. App. 217a-218a. In Java, methods are organized and stored in "libraries."

The Java language requires that the libraries be organized hierarchically. There are three levels. The top is composed of "packages." Each package contains "classes." Each class file contains methods.

There is a declaration for every method, as well as for every class and package. Together, those declarations specify the method's place in the library. Pet. App. 216a, 227a-228a.

A library's hierarchy is thus a filing system for the methods. *See, e.g.*, 2012 Tr. 2205:3-2206:1. Each package is like a filing cabinet; each class like a drawer in that cabinet; and each method like a folder

---

[1] This brief uses the term "function" in the ordinary sense of what computer instructions do, as distinguished from the Java language's technical usage of "function" as a type of method.

in that drawer. The declarations are labels identifying each cabinet and drawer, as well as the folder containing the method.

Notably, the hierarchy is not itself a computer program. And the declarations for a library therefore do not appear together (as would a contiguous segment of a computer program); they are scattered throughout the library.

Second, and critically, declarations connect a Java developer's call to its corresponding method. Pet. App. 221a, 227a-228a. Together, the call and the declaration form the method's "interface." *See id.* at 226a-228a, 267a-268a; *see also* 2012 Tr. 2102:12-2103:6.[2]

In the Java language, calls take the form of the method's "fully qualified name." That is the combined name of each element in the hierarchy: *package.class.method*. JA34-35; JA38-39; *see also* Pet. App. 223a-225a. If the call does not correspond precisely to declarations, it will not work correctly.[3]

The computer code that performs a method's function is called "implementing code." Implementing code can be written many different ways.

3. An illustrative example. Here is an example of how calls, declarations, and methods work. A Java

---

[2] The brief uses the term "interface"—shorthand for "software interface"—in the ordinary sense of a means of connecting to, interacting with, or operating computer software, as distinguished from the Java language's technical usage of "interface" as a "reference type" for a class or method.

[3] In certain circumstances in which the developer has already specified the name of the package and/or class earlier in the application, the Java language may not require the developer to repeat those elements of the name again.

developer writes the call *java.lang.Math.max(5, 10)* to determine whether 5 or 10 is the larger number. That call invokes a method called *max*, which determines the larger of two integers. The call requires that the libraries contain precisely written declarations specifying the *max* method, within the *Math* class, within the *java.lang* package. The call and declarations together establish an interface that makes the Java application interoperable with the *max* method. *See* Pet. App. 222a.[4]

The call also identifies the numbers 5 and 10 as "arguments." The declarations pass the arguments to the implementing code. The Java language does not require that any particular implementing code be used to determine which is the larger number. The declarations then return the answer to the developer's application.

---

[4] The call *java.lang.Math.max(5, 10)* requires that the libraries set forth the following three declarations, in this order (albeit often separated by other instructions):

package java.lang;

...

public class Math {

...

public static int max(int a, int b) {

Pet. App. 224a. The declarations allow the Java language to recognize that call because they: (1) identify the *java.lang* package [*package java.lang*;], (2) create the *Math* class within that package [*public class Math* {], and (3) create the *max* method within that class [*public static int max(int a, int b)* {]. The last of the three declarations also specifies that the *max* method will receive two integers [(*int a, int b)*] and will return an integer [*int*].

B. Android's Reuse Of Declarations Necessary To Recognize Developers' Calls

Google's Android project took three years of work and more than 100 engineers to launch. JA45; JA116-117. Google's engineers created thousands of new methods designed for creating smartphone applications. *See* JA197-198. Google also designed Android to recognize the existing calls that Java developers would expect to use in developing smartphone applications. JA47-50; JA169-170; JA203; JA213-214.

The process of writing new software to perform certain functions of a legacy product is known as "reimplementation." Through reimplementation, the new entrant creates its own computer code to perform the functions, but reuses the limited number of instructions that are required to create the interface already known by the users. *See infra* at 26-27.

Android did not need to—and therefore did not—reimplement *every* method that Java developers already knew. Originally, the Java language was designed and principally used to write programs for servers and traditional desktop computers. JA151-152; JA228-229. Google determined that many of the Java methods were therefore inappropriate for smartphones. JA48-50; JA169-170; JA264-265.

Google's engineers wrote from scratch (or acquired from others) the millions of lines of implementing code that perform the functions of all the Android methods—both new methods and those that Android reimplemented. Pet. App. 213a, 218a-219a. The engineers tailored that implementing code to the constraints of smartphones, such as limited battery life

and computing power. JA158-160; JA167-168; JA197-198; JA204.

But as discussed, the Java language *would not permit* Google to write its own declarations for those methods that Android reimplemented, without requiring Java developers to learn thousands of new calls. The developers' calls only work with the methods' original declarations. As the District Court found: "Significantly, the rules of Java dictate the precise form of certain necessary lines of code called declarations, whose precise and necessary form explains why Android and Java *must be* identical when it comes to those particular lines of code." Pet. App. 221a.

Google's engineering team therefore reused the mandatory declarations that correspond to the calls for the Java methods reimplemented by Android. Pet. App. 218a-221a. Those methods, and the declarations, were originally set forth in the "Java SE libraries,"[5] which are in turn included in a work called "Java 2 SE" (Java SE). Respondent Oracle holds the copyright in Java SE through its acquisition of the creator, Sun Microsystems (Sun). *Id.* at 212a, 216a-218a.

Because Google created its own computer code whenever possible—including thousands of new methods and all the implementing code—Android reuses less than 0.5% of the 2.86 million lines of computer

---

[5] The lower courts sometimes referred to this collection of methods—including the implementing code—as the "Java SE API." This brief uses the term "Java SE libraries" instead. An API is an "application programming interface." As its name suggests, an API generally refers to the "interface" connecting "programs." It generally does not refer as well to the implementing code that performs the methods' functions.

code in the Java SE libraries. JA212. Those declarations amount to roughly 0.1% of the approximately 15 million lines of computer code in Android. *See ibid.*

Google released Android in 2007 as a free, open-source operating system for any smartphone manufacturer to use. Pet. App. 219a; JA117-118; JA138-139. Countless new developers learned the Java language to create Android applications. *See* JA212-213. Sun publicly offered Google its "heartfelt congratulations," because Android had "strapped another set of rockets to the [Java] community's momentum." JA130-133; JA148-149; TX2352. Independent developers using Java have created *millions* of Android applications used by more than a *billion* people.

II. PROCEDURAL HISTORY

Oracle brought this suit against Google for patent and copyright infringement. Doc. 1. Google prevailed on the patent claims, which Oracle abandoned on appeal. Based on Google's reuse of declarations from the Java SE libraries, Oracle has sought more than $8 billion in copyright damages.

Oracle does not assert a copyright in the Java language or the developers' calls. *See, e.g.*, JA276; Doc. 853 at 5:20-22. As to the latter, names and short phrases are not eligible for copyright protection. 37 C.F.R. § 202.1(a). Oracle nonetheless asserts that its copyright in the Java SE libraries gives it the same power to control the developers' use of their calls to create innovative applications, by indirection.

A. Copyrightability And Infringement

1. Original District Court proceedings. The District Court rejected Oracle's copyright claim, relying

principally on the "merger" doctrine. Section 102(b) of the Copyright Act prevents an author from securing the exclusive right to an idea or function itself, as opposed to the author's expression. 17 U.S.C. § 102(b). Merger precludes securing that right indirectly by asserting copyright when there are only a limited number of ways to express or embody the idea or function. Pet. App. 237a, 261a.

Here, the District Court concluded, Oracle asserts an exclusive right to the declarations' functions. Only the Java SE declarations can create the interface with the calls known to Java developers. The court reasoned that "when there is only one way to write something, the merger doctrine bars anyone from claiming exclusive copyright ownership of that expression. Therefore, there can be no copyright violation in using the identical declarations." Pet. App. 264a.

2. Oracle's first appeal to the Federal Circuit. The Federal Circuit had appellate jurisdiction because Oracle's complaint had included patent claims. That court held that the merger doctrine does not exclude the declarations from copyright protection. The court did not doubt that only the precise instructions that Google reused would perform the declarations' function of responding properly to the existing calls used by developers. But it reasoned that "nothing prevented Google from writing its own declaring code, along with its own implementing code, to achieve the same" general "result" as the Java SE libraries. Pet. App. 151a-152a.

The Federal Circuit also held that merger is inapplicable because Sun made numerous decisions regarding which methods to create, how to organize them, and what to name them. The court thus deemed

it irrelevant that the Java language gave Sun one way to *express* the choices it made. Pet. App. 150a-151a.

The court also recognized that the declarations principally consist of names, which are not protected by copyright. But it held that Google had infringed Oracle's interest in a "compilation" of the methods' names. Pet. App. 153a-155a. Relatedly, the court held that Oracle's copyright protected the non-literal "structure, sequence, and organization" (SSO) of the Java SE libraries—*i.e.*, the filing system into which Sun sorted methods of a certain type, apart from their literal contents. *Id.* at 158a-166a.

B. Fair Use

1. The jury verdict on remand. The Federal Circuit concluded that Google's fair-use defense could not be resolved as a matter of law, and remanded. Pet. App. 182a-184a. The District Court conducted a two-week trial. The jury heard sworn testimony from dozens of witnesses and received voluminous documentary evidence. The parties agreed to accept a general verdict. The jury found that Google's conduct was protected as fair use.

The District Court denied Oracle's motion for judgment as a matter of law, finding ample evidence to support the jury's verdict. The court explained that it had adopted Oracle's proposed instruction directing the jury to decide fair use holistically based on all the facts. Pet. App. 116a. The court also found that the evidence presented easily supported the verdict with respect to the four illustrative factors set forth in 17 U.S.C. § 107. The court gave several examples, but stressed that "many more variations and balancings

could have reasonably led to the same verdict." Pet. App. 117a.

First, regarding the purpose and character of the use, the court concluded that "though Google's use was commercial, which weighed against fair use, the jury could reasonably have found the open-source character [*i.e.*, the free distribution] of Android tempered Google's overall commercial goals." Pet. App. 108a. Further, the "jury could reasonably have found" Android to be "transformative," because it used only a subset of the Java SE declarations, in combination with Google's own implementing code and thousands of new methods, to create an innovative "mobile smartphone platform." *Id.* at 111a.

Second, regarding the nature of the work, the court recognized that Oracle had introduced evidence that its design of the Java SE libraries was to some degree creative. But "our jury could reasonably have gone the other way" and concluded that "functional considerations predominated in their design." Pet. App. 113a-114a.

Third, regarding the amount of the work used, "our jury could reasonably have found that Google duplicated the bare minimum" from the Java SE libraries, "thus finding that Google copied only so much as was reasonably necessary for a transformative use." Pet. App. 114a.

Fourth, with respect to the effect of the use on the market for the work, "our jury could reasonably have found that use of the declar[ations] (including their [structure]) in Android caused no harm to the market for the copyrighted works." Pet. App. 114a.

2. Oracle's renewed appeal. On Oracle's appeal, the same panel of the Federal Circuit this time deemed fair use to be a question of law that is reviewed *de novo*, treated the jury's verdict as "advisory," and reversed it. Pet. App. 19a, 23a-24a, 55a. The court agreed that the verdict established that the declarations are highly functional and have little expressive value. *Id.* at 42a. Nonetheless, it held as a matter of law that no reasonable jury could find that Google had engaged in fair use, principally because Google had supposedly used the declarations for the same purpose as had Oracle, in a commercial product, in a market in which Oracle said it wanted to compete. *Id.* at 53a-54a.

SUMMARY OF THE ARGUMENT

I. Oracle has no interest protected by copyright in the declarations of the Java SE libraries. This case is controlled by the merger doctrine, which holds that copyright protection does not apply when there are only a few ways to express or embody a particular function.

Over the course of three years, more than 100 Google engineers worked to create Android. Google had the right to reimplement methods from the Java SE libraries, in which Oracle holds no patent. To do that, Google wrote its own implementing code to perform the methods' functions.

Google reused declarations from the Java SE libraries because—and only because—no other option would recognize the calls used by Java developers. If it had been possible to create from scratch the minuscule portion of Android represented by the Java SE declarations, and still have those declarations function

as intended, Google's engineers would have done so. Those declarations were not beyond Google's capacity to create. The opposite is true: They were rote, de minimis instructions that easily fall within the rule that short phrases do not involve enough expression to receive copyright protection.

Because no other instructions can perform the declarations' function, merger excludes them from copyright protection. Any other result would impermissibly convert Oracle's copyright in the declarations' *expression* into an exclusive right to the declarations' *function*.

Google therefore had the right to reuse the declarations, free from Oracle's assertion of copyright. Oracle has no protected interest in the declarations individually or collectively. Nor can it invoke copyright on the basis of a supposed interest in a compilation of the declarations' names or in the libraries' structure, sequence, and organization. Holding that any of those is protected by copyright would produce the result that the merger doctrine prohibits: It would forbid Google from reusing the declarations and would give Oracle an exclusive right to their function.

The conclusion that the merger doctrine precludes Oracle's copyright claim is reinforced by the fact that Oracle's position is contrary to the essential purpose of copyright: to promote the creation of expressive works. Oracle's claim would upend the 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. Reimplementation encourages developers to create valuable new computer software that improves on legacy

products. It also encourages the development of software that is interoperable—*i.e.*, that freely shares information, so that multiple products work efficiently together.

Oracle's copyright theory would give software companies the power to block reimplementation and to inhibit the development of new and better software. This case is a perfect example. Google created Android, an innovative operating system for smartphones. Since then, innumerable developers have used the Java language to create millions of applications for Android, which is used by billions of people around the world. Oracle holds no rights to either the developers' applications (including the developers' calls) or to the Android platform (including the millions of lines of implementing code that Google tailored to work with smartphones). But it invokes a claim of copyright in the mandatory declarations, which could only ever be written one way, to seriously interfere with the development of both applications and platforms.

II. Even if the Court concludes that Oracle asserts an interest protected by copyright, there is no basis to overturn the jury's determination—based on the testimony of dozens of witnesses and hundreds of exhibits—that Google's reuse of the declarations was fair use. Oracle does not dispute that the jury was properly instructed. The Federal Circuit erroneously substituted its view of the evidence to hold that no reasonable jury could find in Google's favor.

For the reasons just given, the jury was entitled to conclude that Google's reuse of the declarations was consistent with the overarching purpose of the fair-use doctrine: avoiding the rigid application of copyright

that would stifle creativity. There was also plainly sufficient evidence to support the verdict with respect to the jury's weighing of each of the four illustrative fair-use factors specified by the Copyright Act.

First, the jury could reasonably have found that the purpose and character of the use weighed in Google's favor. Specifically, the evidence supported the conclusion that Google's reuse of a subset of the Java SE declarations, combined with Google's own vast implementing code, to create an innovative smartphone operating system, was transformative.

Second, the jury could reasonably have found that the nature of the copyrighted work supported a finding of fair use, because the declarations were highly functional, rather than expressive.

Third, the jury could reasonably have found that the amount and substantiality of the use in relation to the copyrighted work as a whole favored Google, which reused less than 0.5% of the Java SE libraries. Moreover, Google reused no more material than was absolutely necessary to allow Android to respond properly to the calls known to Java developers.

Fourth, the jury could reasonably have found that Google's fair-use defense was supported by the effect on the market for, or value of, the original work. There was substantial evidence that Android is not a substitute for Java SE, which is not suitable for smartphones. The Federal Circuit erred both by substituting its own view of the evidence and by holding that it was legally sufficient that Oracle merely *wanted* to compete with Android in the smartphone market.

The judgment accordingly should be reversed.

ARGUMENT

I. Google Did Not Commit Copyright Infringement.

A. Copyright Protection Does Not Extend To Any System Or Method Of Operation, Or To One Of Only A Few Ways To Express Or Embody The System Or Method.

Under Section 102(a) of the Copyright Act, "Copyright protection subsists . . . in original works of authorship," including "literary works." 17 U.S.C. § 102(a). The statute contemplates that literary works include a computer program, which the Act defines as "a set of statements or instructions to be used directly or indirectly in a computer in order to bring about a certain result." *Id.* § 101. But under Section 102(b), "[i]n no case does copyright protection for an original work of authorship extend" beyond the expression of the computer program "to any idea, procedure, process, system, method of operation, concept, principle, or discovery" that it "embodie[s]." *Id.* § 102(b).

Section 102(b) codifies the "idea/expression" dichotomy, which this Court embraced more than a century ago in *Baker v. Selden*, 101 U.S. 99 (1880)—*viz.*, the principle that copyright protection in a work extends only to expression, not to the idea that the expression conveys. *Baker* held that copyright protection did not apply to forms that embodied an accounting method. The Court reasoned:

> To give to the author of the book an exclusive property in the art described therein, when no examination of its novelty has ever been officially made, would be a surprise and a fraud

> upon the public. That is the province of letters-patent, not of copyright. The claim to an invention or discovery of an art or manufacture must be subjected to the examination of the Patent Office before an exclusive right therein can be obtained; and it can only be secured by a patent from the government.

Id. at 102.

In Section 102(b), Congress later specified that the exclusion from copyright protection extends beyond an "idea" to, *inter alia*, any "process, system, [or] method of operation" that the work describes or embodies. 17 U.S.C. § 102(b). "Due to this idea/expression distinction, every idea, theory, and fact in a copyrighted work becomes instantly available for public exploitation at the moment of publication; the author's expression alone gains copyright protection." *Golan v. Holder*, 565 U.S. 302, 328 (2012) (citations and internal alterations omitted).

In turn, the merger doctrine is an application of Section 102(b). Merger provides that an author cannot claim a copyright in an idea, system, or method indirectly, by copyrighting one of only a few possible means of expression. Granting the author an exclusive right to the expression would confer control over the idea, system, or method of operation itself. In such cases, the expression is not protected by copyright because it is said to "merge" with the non-expressive subject matter. *See generally* 4 Melville B. Nimmer & David Nimmer, *Nimmer on Copyright* § 13.03[B][3][b]-[c] (*Nimmer*); 1 Paul Goldstein, *Goldstein on Copyright* § 2.3.1.2 (*Goldstein*); 1 Howard B. Abrams & Tyler T. Ochoa, *The Law of Copyright* § 3.8.

B. Google's Reuse Of The Declarations Did Not Infringe Any Interest Protected By Copyright.

There are two related grounds on which this Court can hold that the declarations reused by Google from the Java SE libraries are excluded from copyright protection under Section 102(b). The Court can rule that the declarations are a "method of operation." 17 U.S.C. § 102(b). Instructions that respond to the calls used by Java developers are the equivalent of the forms in *Baker*. A declaration is an excluded "method of operation," because it allows a Java developer to invoke (*i.e.*, operate) the separate pre-written computer code.

The organizational system of the Java SE libraries similarly receives no copyright protection because it is entirely functional. Oracle is claiming the exclusive right to a functional filing system of a certain structure. The declarations direct the developer's application to the location of the relevant method. They are analogous to a collection of filing cabinets. Section 102(b) expressly excludes such "systems" from copyright protection.

Alternatively, the Court can decide the case more narrowly by applying the merger doctrine. That ruling would focus on the fact that the declarations can only be written one way to perform their function of responding to the calls already known to Java developers. For the reasons that follow, merger easily disposes of Oracle's copyright claim.

1. Google reused only the material that was required by the Java language to perform the function of responding to the developers' calls.

Oracle does not dispute the essential feature of this case: Only one precisely written set of declarations will perform the function of responding to the corresponding calls known to the developer. The two create an interface as an exclusively matched set: a key and an unpickable lock. If the declaration is changed, the call will fail. The developer's application will either not invoke the method or the method will not function properly. Each declaration that Google reused is thus the only instruction that performs the function of responding to the corresponding call in a certain, expected way.

Moreover, Google's engineers reused no more than what the Java language absolutely required. The engineers did not even duplicate the entirety of the declarations they reused. In one respect, developers' calls will respond properly to declarations that deviate from those in the Java SE libraries. Pet. App. 226a. Many calls and declarations include "arguments," which transfer information from the developer's application. The Java language does *not* require that a declaration use particular argument names.[6] And in reusing declarations from the Java SE libraries, the Google engineers regularly used their own argument names that

---

[6] From the earlier example, *see supra* at 6 n.4, recall that the declaration for the *max* method [*public static int max (int a, int b)*] anticipates receiving two integers as arguments, which are assigned the names *a* and *b* [(*int a, int b)*]. Instead of *a* and *b*, the declaration for *max* would still properly respond to the developer's call if it used, for example, *x* and *y*.

corresponded to their own implementing code. *See ibid.*

2. The merger doctrine provides that Google has the right to reuse the declarations.

Under the merger doctrine, an author cannot assert a copyright with respect to a function indirectly, by claiming an exclusive right to the only expression for the function. That is exactly Oracle's assertion here. It claims that no person can copy the only instructions that will respond correctly to the developers' calls. But only patent law provides rights of that sort. *See Eldred v. Ashcroft*, 537 U.S. 186, 217 (2003) ("A reader of an author's writing may make full use of any fact or idea she acquires from her reading. See § 102(b). The grant of a patent, on the other hand, does prevent full use by others of the inventor's knowledge."). The Federal Circuit, which has exclusive appellate jurisdiction over patent claims, granted Oracle rights akin to patent rights.

The application of the merger doctrine to the Java SE declarations is easily illustrated, in three ways.

First, if Oracle were not effectively claiming a copyright to the function of responding properly to the developers' calls, then it would be able to identify other Java instructions that would do the same thing. But it cannot, because none exist.

Second, look to the conduct of Google, which expended years of effort writing millions of lines of computer code. Google's engineering team reused the declarations only because it had no other choice.

Third, imagine that Oracle described how a particular declaration works in an English sentence. It might say something like: "The *max* method will

respond to a developer's call *java.lang.Math.max(x, y)* by accepting integers *x* and *y* and returning another integer." Under Section 102(b), Oracle plainly could not invoke copyright to assert an exclusive right to the function that sentence describes. The merger doctrine similarly provides that Oracle cannot assert copyright protection over the only declaration that can perform that function in the relevant programming language, which here is Java.

It also is significant that Oracle claims no copyright interest in the calls that Java developers use to invoke the methods. Under the Copyright Act, those developers have a significant interest in using those instructions to create their own valuable applications. It would be contrary to copyright's purpose to promote creative expression to block the developers from doing so by granting another author the exclusive right to the precise function that corresponds to the developers' expression.

3. The merger doctrine applies to computer software interfaces designed to invoke the functions of a program.

The merger doctrine is critical in the field of computer science. The general rule—reflected in Section 102(b)—is that copyright protection does not extend to purely functional works, which are instead subject to the rules governing patents. *Baker v. Selden*, *supra*; *see also Incredible Techs., Inc. v. Virtual Techs., Inc.*, 400 F.3d 1007, 1012 (7th Cir. 2005) ("The exclusion of functional features from copyright protection grows out of the tension between copyright and patent laws. Functional features are generally within the domain of the patent laws."). Congress protected the

expression in computer programs as a limited departure from that principle. But it provided in Section 102(b) that copyright would not grant an exclusive right to the underlying function that the computer code embodies. In case of doubt, this Court's practice is to find no copyright violation, reasoning: "When technological change has rendered its literal terms ambiguous, the Copyright Act must be construed in light of [its] basic purpose" to "stimulate artistic creativity for the general public good." *Sony Corp. of Am. v. Universal City Studios, Inc.*, 464 U.S. 417, 432 (1984) (quoting *Twentieth Century Music Corp. v. Aiken*, 422 U.S. 151, 156 (1975)).

The merger doctrine's application to isolated instructions such as declarations comports with the Copyright Act's treatment of "computer programs." The Copyright Act defines a computer program as "a *set of statements or instructions* to be used directly or indirectly in a computer in order to bring about a certain result." 17 U.S.C. § 101 (emphasis added). A "set" of computer code can almost always be written in many different ways. For example, Google's engineers were able to create their own implementing code, which performs the function of each method.

By contrast, the Act notably does not extend copyright protection to an *isolated computer instruction*, which generally can be written only one way. Instructions are the building blocks of larger, creative computer programs. Here, for example, Oracle is invoking its supposed exclusive right to the declarations to prevent Java developers from using calls that they routinely combine in creative applications. Oracle's claim cannot be reconciled with the principle that "copyright assures authors the right to their original expression,

but *encourages* others to build freely upon the ideas and information conveyed by a work." *Feist Publ'ns, Inc. v. Rural Tel. Serv. Co.*, 499 U.S. 340, 349-350 (1991) (emphasis added); *see also Goldstein* § 2.3.2 ("[A]s a general rule, courts will be more inclined to find merger where the idea that the copyright owner seeks to constrain is a building block, a necessary foundation to further creative expression.").

The conclusion that merger applies to the function performed by individual instructions is reinforced by the report that Congress commissioned from the National Commission on New Technological Uses of Copyrighted Works. *See* Nat'l Comm'n on New Tech. Uses of Copyrighted Works, *Final Report* (1979) (CONTU Report). The Report laid the foundation for the Copyright Act's recognition of protection for computer programs. "Subsequent Congresses, the courts, and commentators have regarded the CONTU Report as the authoritative guide to congressional intent." *Sega Enters. Ltd. v. Accolade, Inc.*, 977 F.2d 1510, 1520 n.5 (9th Cir. 1993). The Report explained that the copyright in a computer program would not give the author any exclusive right to "the electromechanical functioning of a machine." CONTU Report 20. A later software developer would remain free to make the computer "perform *any* conceivable process," *ibid.* (emphasis added), including obviously the process performed by a single instruction.

4. The number of declarations that Google reused does not change the merger analysis.

Oracle stresses that Google's engineers reused thousands of declarations. But under the merger doctrine, none of those declarations is protected by

copyright. If a book contains thousands of unprotected facts, it is not copyright infringement to copy either one fact or one thousand facts. *See Feist*, 499 U.S. at 361. The amount of the work copied may be relevant to the question of fair use, but if and only if copyright protection applies. *See infra* at 46-47.

Oracle stresses the enormous effort that Sun put into developing the Java SE libraries. But copyright does not reward effort. *Feist*, 499 U.S. at 359-360; *Meshwerks, Inc. v. Toyota Motor Sales U.S.A., Inc.*, 528 F.3d 1258, 1268 (10th Cir. 2008). It rewards creative expression. Here, the creativity is reflected in the implementing code, which performs the methods' functions and which could be written in numerous different ways. Oracle can recover the value of its work through its exclusive right to the implementing code.

Oracle's argument is also something of an optical illusion. It is designed to suggest that Google's engineers reused thousands of related commands, as if they comprised a large section of a computer program—the implication being that the program could be written in many ways. That is not correct. The Java SE libraries are simply a large collection of methods. No application uses more than a very small proportion of them. The declarations, in turn, do not appear together but instead are sprinkled throughout the Java SE libraries; most are functionally unrelated to each other.

In any event, even viewed as a collective whole, the declarations are excluded from copyright protection by the merger doctrine. The declarations are a "system," because they collectively respond to thousands of developers' commands. *See* 17 U.S.C. § 102(b). But only one collection of instructions—the

precise declarations that Google reused from the Java SE libraries—will perform the function of that system. Merger accordingly excludes it from copyright protection. Any exclusive right to such a system can only be provided under the rules governing software patents. The Federal Circuit's decisions thus threaten to undermine the limits on software patents recognized by this Court. *See, e.g.*, *Alice Corp. v. CLS Bank Int'l*, 573 U.S. 208 (2014).

In the context of computer science, the merger doctrine is essential to the ability of innovators to improve on numerous functions of older software. Building on existing conventions and knowledge is a foundational practice in computer science. *See* JA245-246. New entrants often create software that improves on a legacy product. Any successful new product must be as compatible as possible with the relevant existing skills and experience of the users it seeks to support. Software Innovators Cert. Br. 4-13; Mozilla Cert. Br. 10-15.

New entrants into a software market therefore "reimplement" existing tools. They write the extensive computer code that performs the relevant *functions* from the legacy product. But they reuse the more limited code that is required—because it cannot be written any other way—to allow users to use commands they already know from the legacy product. The code that recognizes the users' commands is part of the interface.

As the most distinguished technology companies and computer scientists in the world have explained, the decades-long understanding in the software industry has been that software functions may be freely reimplemented—and that such reimplementation

"unleashed the personal computer revolution, created popular operating systems and programming languages, and established the foundation upon which the Internet and cloud computing depend." 78 Computer Scientists Cert. Br. 4; *see* Mozilla Cert. Br. 10-11; Developers Alliance Cert. Br. 5-8; *see also* JA114-115; JA141; JA153-154; JA157; JA164-165; JA245-246. (Notably, Java SE itself "reimplements" interfaces from earlier programming languages. JA154-157; *see also* JA211.) This common practice provided the backdrop to Congress's express recognition of computer programs in the Copyright Act.

A ruling by this Court that copyright prohibits that reimplementation would allow the authors of older software to hold their users hostage, lest the skills that the users have built up over long periods of time become worthless when they move to a new environment. *Lotus Dev. Corp. v. Borland Int'l, Inc.*, 49 F.3d 807, 821 (1st Cir. 1995) (Boudin, J., concurring), *aff'd by an equally divided Court*, 516 U.S. 233 (1996) (per curiam). Because new entrants would have difficulty attracting users, they would be less likely to create innovative applications. That is the opposite of what copyright intends.

This case illustrates the point. Oracle claims the right to erect serious obstacles to both the creation of an innovative platform like Android and the creation of applications for it. Oracle would require Java developers to learn thousands of new calls to replace those they already know, with no benefit to anyone. That is effectively the equivalent of requiring the developers to learn an entirely new programming language, simply to invoke the same functions using different labels.

At bottom, Oracle's position is a bait and switch. Like Sun before it, Oracle touted Java as a "free and open" programming language. Pet. App. 5a, 9a (citation omitted). Millions of developers collectively invested incalculable effort in learning how to use thousands of non-copyrighted calls to create applications. Now Oracle says that it has the power to effectively lock those developers into programming for only its platform.

More broadly, interfaces such as declarations are critical to compatibility not just between platforms and applications, but also between and among programs, platforms, and software-enabled devices in general. As explained by amicus Microsoft (a company that both relies on copyright protection to develop new products and relies on compatibility with competitors' products to innovate), the Federal Circuit's decision "threatens the viability of the interconnected software ecosystem." Microsoft Cert. Br. 21 (capitalization altered). As Microsoft's brief points out: "Consumers already expect to be able to take a photo on their Apple phone, save it onto Google's cloud servers, and edit it on their [Microsoft] Surface tablets." *Id.* at 22-23.

5. The Federal Circuit's rationales for rejecting the merger doctrine lack merit.

The Federal Circuit held that it was sufficient that Google could have written different declarations that still would have allowed the *methods* to perform the same general functions. That reasoning misapprehends the nature of Oracle's copyright claim, and therefore asks the wrong question. Oracle is asserting that Google infringed its copyright with respect to the declarations. The question is therefore whether the

declarations receive copyright protection. Merger turns on the fact that the declarations that Google reused are the only instructions that can perform their intended function.

The Federal Circuit also held that the merger doctrine determines copyrightability, and therefore should be assessed from the perspective of the author at the time the work is created, not based on the range of options available to a later software developer. Here, Sun (the original creator of Java) had an array of conceptual options in naming and organizing the declarations.

Even on that view, merger still applies. With respect to any given declaration, Sun made a variety of conceptual choices, which are ideas. But it made only one *expressive* choice: what names to use for the packages, classes, and methods. Names receive no copyright protection. 37 C.F.R. § 202.1(a). The Java language determined every other element of each declaration. Having made the conceptual choices about the function a method should perform—*i.e.*, having chosen an *idea*—Sun was compelled to write the declarations just as it did.

This case is analogous to *Southco, Inc. v. Kanebridge Corp.*, 390 F.3d 276 (3d Cir. 2004) (en banc), which rejected the plaintiff's claim to copyright protection for a system of part numbers. The court reasoned that the part numbers were purely functional and were determined by the "'mechanical application'" of a "rigidly dictated" system. *Id.* at 282 (citation omitted). In this case, the declarations were rigidly determined by the rules of the Java language. (Just as Sun picked names in the system of declarations, the plaintiff in *Southco* picked the numbers used

in the system of parts.) It was dispositive that the system determined the actual expression.

But in any event, the authoritative CONTU Report makes clear that, at least in the context of computer software, merger is evaluated at the time material is reused. The Report explained how merger would apply "[i]n the computer context": "[W]hen specific instructions, even though previously copyrighted, are the only and essential means of accomplishing a given task, their later use by another will not amount to an infringement." CONTU Report 20. As a result, the extension of copyright to computer programs would not "block the use of . . . program language previously developed by others when that use is necessary to achieve a certain result." *Ibid.*

That must be true. On Oracle's contrary view, a software developer could write computer code that is the only way to perform a function, then effectively use copyright to claim the exclusive right to that function. Here, Google would be prohibited from creating software with declarations that recognized Java developers' instructions, and Oracle could lock developers into the Java SE libraries as the only system that would recognize their calls. Such a right to own an entire function can only be conferred by patent law.

Baker v. Selden is on point. There, Selden could have chosen from a vast array of potential accounting systems. But he selected one and disclosed it in his book. At that point, Baker had the right to perform the method, and thus the right to publish forms like Selden's. It made no difference that Baker could have conceived of other forms of accounting that would have achieved the same general result; if he was going to perform Selden's method, he needed to use forms like

Selden's. Likewise, Oracle cannot avoid the application of merger by relying on the fact that Sun could have originally chosen different declarations, or the fact that Google later could have created a different organization of methods that responded to different calls. The relevant point is that once Sun released Java SE, Google had to use the declarations from the Java SE libraries to respond properly to the existing calls that the developers then knew.

6. The merger doctrine also resolves Oracle's claims that copyright protection applies to the structure created by the declarations.

Merger resolves not only Oracle's assertion that the declarations are protected by copyright, but also its claim that it has a protected interest in the structure of the Java SE libraries. Merger provides that copyright will not prevent Google from reusing the declarations, because Google has the right to create its own software that performs their functions. But Oracle's "structural" copyright claims would do just that. If Oracle prevailed, Google could not reuse those very declarations.

It is thus critical that Oracle's claims related to the libraries' structure arise *only* from Google's reuse of the declarations themselves, and that Google did *not* duplicate any elements of the structure that were not compelled by the declarations. The declarations create the hierarchical organization into which methods are sorted, including what Oracle describes as the structure, sequence, and organization. *Only* the declarations do so. The structure does not exist apart from the declarations. Google in turn duplicated the

structure of the Java SE libraries only as the inevitable and unavoidable consequence of reusing the declarations.

Moreover, Google duplicated the structure of the Java SE libraries only to the extent absolutely required. Whenever it could depart from that structure, it did so. That is true in two respects.

First, as noted, methods are stored in the computer files of their respective classes. In the filing cabinet analogy, they are folders in drawers. For example, the *max* method is stored in the *Math* class computer file. The Google engineers could not store the methods (and thus their declarations) in class files that were different from those in the Java SE libraries, because the developers' calls would no longer have worked.

But the Java language does not dictate the *order* in which the methods (and their corresponding declarations) are stored in each class file. Google therefore did not duplicate the ordering that Sun had used within the class files of the Java SE libraries. Instead, the Android declarations regularly appear in a different order. Again, using the file-cabinet analogy, while the Java language required Google to use the same file cabinets, drawers, and file labels as the Java SE libraries, Android organizes many of its file folders in a different order within each drawer.

Second, the libraries have a logical structure based on connections between the methods that are created by the implementing code. Each method is an individual building block that performs a common function. 2012 Tr. 288:21-290:7, 729:13-25. Therefore, one Java method's implementation may invoke

another method, which in turn may invoke yet another, and so on. Doc. 955 at 13-14. For example, the implementing code of a method that performs a complicated mathematical function may need to determine the sums of multiple sets of numbers, then sort those results in ascending order. Those individual subsidiary functions may be performed by calling other methods. Here, Google wrote its own implementing code. The methods in Android therefore contain their own distinct calls to other subsidiary methods. The logical structure of methods in Android is therefore very different from the structure of the Java SE libraries. *Ibid.*; *see also* 2012 Tr. 2183:21-2184:21.[7]

---

[7] The lower courts that have recognized a copyright interest in a computer program's SSO thus would find no infringement of that interest here. *See, e.g.*, *Computer Assocs. Int'l, Inc. v. Altai, Inc.*, 982 F.2d 693, 697, 702, 715 (2d Cir. 1992) (affirming denial of claim based on infringement of SSO, described as including "non-literal components such as general flow charts" that "map the interactions between modules that achieve the program's end goal," as "well as the more specific organization of inter-modular relationships, parameter lists, and macros"); *see also Softel, Inc. v. Dragon Med. & Sci. Commc'ns, Inc.*, 118 F.3d 955, 967 (2d Cir. 1997); *Gates Rubber Co. v. Bando Chem. Indus., Ltd.*, 9 F.3d 823, 835 (10th Cir. 1993). Oracle's SSO claim instead relates to the hierarchical scheme it chose to organize largely unrelated segments of computer code—*i.e.*, the methods. No other court has recognized such a claim.

There also is no merit to the Federal Circuit's view that the Java SE libraries are a compilation of names. The Copyright Act does protect a compilation, defined as a "work formed by the collection and assembling of preexisting materials or of data that are selected, coordinated, or arranged in such a way that the resulting work as a whole constitutes an original work of authorship." 17 U.S.C. § 101. The Java SE libraries are a

The District Court therefore correctly held that Google's reuse of declarations from the Java SE libraries did not violate any interest protected by copyright.

II. There Is No Basis To Overturn The Jury's Finding That Google's Reuse Of The Declarations From The Java SE Libraries Was Fair Use.

Even if this Court concludes that Google's reuse of the declarations infringed an element of the Java SE libraries that was protected by copyright, there is no basis to overturn the jury's finding of fair use.

A. The Jury's Verdict Is Reviewed For Substantial Evidence.

Following an extensive trial, covering numerous hotly contested factual issues, the jury reached a general verdict finding fair use. Every fair-use case "must be decided on its own facts" and cannot be captured by a "generally applicable definition." *Harper & Row, Publishers, Inc. v. Nation Enters.*, 471 U.S. 539, 560 (1985) (quoting H.R. Rep. No. 94-1476, at 65 (1976) (House Report)); *Sony Corp. of Am. v. Universal City Studios, Inc.*, 464 U.S. 417, 448 n.31 (1983).

Although the Federal Circuit concluded in the first appeal that it could not decide fair use as a matter of law because of the fact-intensive nature of the

---

"compilation" in the sense that they are an organized collection of materials. But Oracle did not select and arrange the *names*; instead, it systematically organized the *methods* by their functionality, filing those methods according to what they do, not by what they are named. 2012 Tr. 588:12-592:10, 628:22-629:6. Such a "system" is expressly excluded from copyright protection by Section 102(b).

inquiry, Pet. App. 174a-184a, the court took the extraordinary step in the second appeal of reversing the jury's verdict based on the court's own weighing of the evidence—an unprecedented action by an appellate court. In so doing, the court inverted the proper substantial-evidence standard of review.

The jury was correctly instructed on the legal rules governing fair use—and Oracle did not challenge the instructions on appeal. Because the jury rendered a general verdict of fair use, all reasonable inferences from the evidence presented at trial must be construed in favor of the jury's verdict. *Sentilles v. Inter-Caribbean Shipping Corp.*, 361 U.S. 107, 110 (1959). Emphasizing that the "trial presented a series of credibility calls for our jury," Pet. App. 92a, the District Court correctly held that there were "myriad ways that the jury could reasonably have balanced the statutory factors and found in favor of fair use," *id.* at 117a.

The question before the Federal Circuit was therefore whether the evidence was sufficient to allow *any* reasonable jury to find facts that would support the ultimate conclusion that Google's conduct was fair use. Consistent with the Seventh Amendment's admonition that it is the responsibility of juries to find facts, the Federal Circuit had no authority to overturn the jury verdict if that verdict was based on substantial evidence, *i.e.*, evidence sufficient to support the jury's conclusion even when it would also support a contrary conclusion. *See Biestek v. Berryhill*, 139 S. Ct. 1148, 1154 (2019).

"[D]ecisionmaking in fact-intensive disputes" such as this "necessarily requires" a jury to make "judgment calls." *Hana Fin., Inc. v. Hana Bank*, 574 U.S. 418, 425 (2015). Those judgment calls—including

resolution of competing evidence and drawing of inferences from testimony and documents—are reflected in the jury's general verdict and are entitled to deference on appeal, not *de novo* review. *See* Fed. R. Civ. P. 50(a)(1). There are no express findings of fact to review, given the general verdict to which the parties agreed. The Federal Circuit was therefore required to assume that the jury resolved all factual disputes in favor of its ultimate verdict. Those implicit factual determinations were entitled to deference—that is, they should have been viewed "with a serious thumb on the scale for the" factfinder (here, the jury)—on appeal. *U.S. Bank Nat'l Ass'n v. Village at Lakeridge, LLC*, 138 S. Ct. 960, 966-967 (2018). Instead, the Federal Circuit did the opposite, expressly treating the jury's verdict as "advisory only," Pet. App. 24a, and assigning to *itself* the role of making and weighing inferences from evidence "de novo" while conducting its own factfinding.

The Federal Circuit justified its lack of deference by relying on Ninth Circuit decisions governing review of district courts' *summary judgment* holdings. Pet. App. 18a, 22a-24a (citing *SOFA Entm't, Inc. v. Dodger Prods., Inc.*, 709 F.3d 1273, 1277 (9th Cir. 2013); *Fisher v. Dees*, 794 F.2d 432, 434 (9th Cir. 1986)). That was obviously wrong, because the very fact that a court granted summary judgment means that there are *no* material disputed factual questions, whereas here there was a lengthy trial precisely because the relevant facts are disputed.

As discussed below, the evidence was more than sufficient to support the jury's verdict.

B. Substantial Evidence Supported The Jury's Overall Finding That Google's Conduct Was Fair Use.

The Copyright Act identifies four factors that must be considered in determining fair use. 17 U.S.C. § 107. Those factors are "not meant to be exclusive," *Harper & Row*, 471 U.S. at 560, and must be considered holistically, "in light of the purposes of copyright," *Campbell v. Acuff-Rose Music, Inc.*, 510 U.S. 569, 578 (1994). The purpose of the doctrine is "to avoid rigid application of the copyright statute when, on occasion, it would stifle the very creativity which that law is designed to foster." *Harper & Row*, 471 U.S. at 550 n.3 (citation omitted).

Particularly relevant here, fair use is designed to adapt to changing technology and to account for the nature of the copyrighted material. House Report 66. In all instances—and particularly when applied in a new technological environment—the fair-use doctrine "must be construed in light of [its] basic purpose." *Sony*, 464 U.S. at 432 (citation omitted); House Report 66 (explaining that Congress did not intend "to freeze the doctrine in the statute, especially during a period of rapid technological change"). The jury was entitled to find that Google's reuse of the declarations furthered that essential purpose.

1. The long-standing practice of the software industry and reasonable software owners. In determining fair use, the jury was entitled to consider the widespread, longstanding practice of software companies and developers of reimplementing declarations and other software interfaces that operate pre-written computer code—just as Google reimplemented the declarations. At common law, the question of fair use was

viewed as whether a reasonable author "would have consented to the use, i.e., where the custom or public policy at the time would have defined the use as reasonable." *Wall Data Inc. v. L.A. Cty. Sheriff's Dep't*, 447 F.3d 769, 778 (9th Cir. 2006) (internal quotation marks omitted); *accord Harper & Row*, 471 U.S. at 549. The jury was entitled to conclude that Sun (the original author and the owner of Java SE at the time Android was launched) did not consider Google's reuse of the declarations to be infringement—and in fact that Sun *celebrated* the launch of Android, which it said had "strapped another set of rockets" to the Java ecosystem by expanding the developer community interested in programming in the Java language. JA119-120; JA124-125; JA129; JA132; JA148-149; TX2352.

The jury also heard testimony from former Sun executives that, consistent with that industry practice, Sun never considered the declarations to be proprietary and allowed anyone to use them without a license. *See, e.g.*, JA119-120; JA124-125; JA129; TX2352. The District Court recognized that the weight to give the testimony of the former Sun executives was the "classic role of a jury to resolve." Pet. App. 107a-108a.

The jury heard substantial supporting evidence that reasonable software copyright owners would permit Google's conduct and that Google acted in good-faith reliance on the industry understanding that interfaces may be reused. JA114-115; JA153-154; JA157; JA164-166; JA245-246. Indeed, Java SE *itself* "reimplements" functionality from earlier programming languages. JA154-157; *see also* JA211. The District Court correctly concluded that the "jury could reasonably have concluded that Google's use of parts

of the Java [SE libraries] as an accelerant was undertaken based on a good faith belief that at least the declaring code and SSO were free to use," "while a license was necessary for the implementing code." Pet. App. 107a.

2. The expression made possible by reuse of software interfaces. Because "[c]opyright seeks to maximize the creation and publication of socially useful material," Pierre N. Leval, *Toward a Fair Use Standard*, 103 Harv. L. Rev. 1105, 1126 (1990), any fair-use analysis must weigh the expressive value of the copyrighted material against that of the new product, *Stewart v. Abend*, 495 U.S. 207, 236-237 (1990); *Harper & Row*, 471 U.S. at 550 n.3. Fair use prevents copyright from prohibiting the reuse of a small amount of low-value expression when reuse would unleash a large amount of high-value expression. *See Harper & Row*, 471 U.S. at 549. The jury was thus instructed that "[t]he policy behind the right of fair use is to encourage and allow the development of new ideas that build on earlier ones, thus providing a counterbalance to the copyright policy to protect creative works." JA279.

The District Court explained that the "jury could reasonably have found that, while the [declarations] and SSO were creative enough to qualify for copyright protection, functional considerations predominated in their design." Pet. App. 114a; *id.* at 113a (noting that the jury "could reasonably have" "concluded that the declaring code was not highly creative"). The Federal Circuit itself agreed that fact supported a finding of fair use. *Id.* at 42a.

Conversely, there was substantial evidence that prohibiting reuse of the declarations would seriously

inhibit a large amount of creative expression by other parties in two ways. *First*, reusing the declarations unleashes enormous innovation and creativity by enabling developers to use their existing knowledge of the free and open Java language to create innovative programs that can run on new platforms such as Android. *See supra* at 26-27; JA139-140; JA187; JA190-191; Pet. App. 196a.

Second, reuse of the declarations prevents Oracle from locking Java developers into Oracle-approved platforms. *See supra* at 27-28; *Sega*, 977 F.2d at 1523-1524 ("[A]n attempt to monopolize the market by making it impossible for others to compete runs counter to the statutory purpose of promoting creative expression and cannot constitute a strong equitable basis for resisting the invocation of the fair use doctrine.").

The net effect of Oracle's novel approach would be to empower Oracle (and other software companies) to prevent anyone from developing a product compatible with their software interfaces. Although Oracle trumpets Java's "write once, run anywhere" marketing slogan, its cramped reading of fair use would result in a world of "write once, run only on Oracle-approved platforms." That would deter competition, creating often insurmountable barriers to new market entrants and start-ups, leaving consumers with fewer choices, and stifling technological innovation. Software Innovators Cert. Br. 4-13; Mozilla Cert. Br. 10-15. For decades, computer scientists have built on each others' work, drawing on shared foundations of knowledge and the compatibility of interfaces that allow developers to create applications available on a variety of devices and platforms. The decision below would make the industry less efficient, reduce consumer choice, and limit

creative expression. These "practical problems . . . are too serious, too extensive, and too likely to come about for [the Court] to dismiss them as insignificant" in interpreting the Copyright Act. *Kirtsaeng v. John Wiley & Sons, Inc.*, 568 U.S. 519, 545 (2013).

Although the Solicitor General now sides with Oracle, the U.S. Copyright Office has publicly committed to the view that the reuse of "appropriately limited amounts of code," including software interfaces like the declarations at issue here, should be considered fair use when the reuse is "for purposes of compatibility and interoperability." U.S. Copyright Office, *Software-Enabled Consumer Products* 52, 57 (Dec. 2016). Where a reuse "is simply to 'permit . . . functionality'" of a new product, the Office has explained, "literal copying of code may be favored" under the fair-use doctrine. *Id.* at 58 (quoting *Lexmark Int'l, Inc. v. Static Control Components, Inc.*, 387 F.3d 522, 544 (6th Cir. 2004)) (ellipses in original). That describes precisely the reuse at issue here—Google's reuse of the declarations and SSO made the new Android platform accessible to and compatible with Java developers using existing code libraries and Java-language skills.

C. Substantial Evidence Supported The Verdict With Respect To Each Of The Illustrative Statutory Fair-Use Factors.

Section 107 recognizes the common-law doctrine of fair use without "chang[ing], narrow[ing], or enlarg[ing] it in any way." *Harper & Row*, 471 U.S. at 549 (quoting House Report 66). The provision lists four "criteria which, though in no case definitive or determinative, provide some gauge for balancing the equities." House Report 65. The District Court carefully

reviewed each factor and found substantial evidence to support the verdict. That conclusion is reinforced by the fact that the jury's role is not only to find the relevant facts but also to then weigh the different factors (as well as others it finds relevant) to reach the ultimate determination of fair use. The Federal Circuit erred in reversing the verdict by applying erroneous legal rules and reweighing the evidence.

1. Factor one: The purpose and character of the use

a. *New, innovative, and socially valuable use.* The first statutory fair-use factor is "the purpose and character of the use, including whether such use is of a commercial nature or is for nonprofit educational purposes." 17 U.S.C. § 107(1). "The central purpose of this investigation is to see, in Justice Story's words, whether the new work merely 'supersedes the objects' of the original creation," "or instead adds something new, with a further purpose or a different character, altering the first with new expression, meaning, or message; it asks, in other words, whether and to what extent the new work is 'transformative.'" *Campbell*, 510 U.S. at 579 (citations and brackets omitted). Here, the jury was properly instructed on that legal standard. JA281.

The District Court concluded that the "jury could reasonably have found" that Google's reuse of the declarations from "37 out of 166 Java API packages," "reimplemented with new implementing code adapted to the constrained operating environment of mobile smartphone devices with small batteries," and "combined with brand new methods, classes, and packages written by Google for the mobile smartphone platform"

gave "new expression, meaning, or message to the duplicated code." Pet. App. 111a.

The jury heard testimony that, prior to Android, there had been little innovation in the smartphone space because of the difficulties of aggregating the necessary components to make a modern smartphone work. JA138-139; *see also* JA111; JA242. Using Java SE would not have solved the problems because it was built to run on servers and desktop computers, JA98; JA195; JA252; JA255, and was not suitable for the serious constraints of a smartphone platform, JA264-265. Google built a revolutionary open-source smartphone platform that was "completely different from any other approach" in the market, including all new implementing code to perform the functions of both the reimplemented methods and the methods unique to that constrained environment. JA111-112; *see* JA168-169; TX7805. The jury heard that "mobile devices have really different constraints from those servers and desktops for which Java 2 SE was written," including differences in power supply, memory availability, and the power of the processing chips. JA159.

Although Google's creation of Android was a commercial endeavor, that does not preclude a finding that the first factor favors a finding of fair use. To the extent the declarations derive any commercial advantage from creative expression, they do so through the libraries of pre-written implementing code (which Google did not reuse). *See* JA124-125 (Java SE declarations were never sold or licensed separately from the language). The jury heard testimony that Sun's business model was to *share* the declarations freely and to compete with respect to the implementing code. JA123-129. The SSO similarly has neither function

nor economic value separate from the declarations. Oracle therefore had no reasonable expectation of financial reward from the SSO *qua* SSO. Because the nature of Google's reuse was so transformative, moreover, its commercial purpose was less significant. *See Campbell*, 510 U.S. at 580, 584-585. The jury could also have concluded that Google's reuse had some noncommercial purposes (such as promoting software innovation) based on the undisputed fact that Android is an open-source initiative that benefits hundreds of device manufacturers, millions of developers, and more than a billion consumers around the world. Pet. App. 107a-108a.

b. *The Federal Circuit's legally erroneous approach to computer code.* In overturning the jury's fair-use verdict, the Federal Circuit concluded that Google's reuse of the declarations could not have been transformative because the declarations supposedly served the "same purpose" in Java SE and in Android. Pet. App. 31a, 33a-35a. That legal rule would dramatically limit any fair use of computer code, and would ironically provide more rather than less protection to a work because of its functional nature.

The declarations that Google reused—indeed, any declarations—are by definition purely functional. Their function is to point a computer to the creative implementing code; the declarations do not instruct the computer how to execute any task. It is principally the *implementing* code that contains the "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 (definition of copyrightable "computer program"). The declarations are *designed* to be used as a means of accessing and operating the creative

implementing code that the computer will execute when running a program. Because the declarations are purely functional, they inevitably will serve "the same purpose," Pet. App. 31a, in the sense of performing the same function in every computing environment. It is therefore impossible to reuse declarations in a software environment for a different function—reusing a declaration necessarily replicates its function. But that is a reason to give them *less* copyright protection, not more.

The Federal Circuit's approach to the transformativeness inquiry is also fundamentally flawed because it focuses on whether the *reused material* itself was transformed rather than on whether the *new work as a whole* transformed the use of the borrowed elements. But a fair-use inquiry must focus on whether "the new work" "adds something new, with a further purpose." *Campbell*, 510 U.S. at 579. Here, the new work is the Android platform, which undoubtedly added something new to the computing world, with a further purpose: It enabled Java developers to unleash their creativity on a new and widely adopted (smartphone) platform, which they could not do while using Oracle's copyrighted work (Java SE).

2. Factor two: The nature of the copyrighted work

The second statutory fair-use factor is "the nature of the copyrighted work." 17 U.S.C. § 107(2). The jury was correctly instructed that it should "focus" "on how close the used material is to the core values of copyright protection" because "[t]he less the used material implicates the core values of copyright protection, the more viable will be fair use and vice versa." JA284; *accord Campbell*, 510 U.S. at 586; Leval 1116. "Under

this factor, the more creative a work, the more protection it should be accorded from copying; correlatively, the more informational or functional the plaintiff's work, the broader should be the scope of the fair use defense." *Nimmer* § 13.05[A][2][a] (footnotes omitted).

Both the District Court and the Federal Circuit recognized that there was substantial evidence that the declarations were functional, not creative, supporting a finding of fair use. *See supra* at 12-13. The declarations serve two relevant functions: connecting the developers' applications to the methods' implementing code and defining the organizational structure in which the methods are stored. But they are minimally expressive because the Java language mandates all or nearly all of the content of each declaration, as well as the organizational structure that the declarations reflect. The jury therefore had ample basis to find that the declarations are entitled to, at best, minimal copyright protection.

3. Factor three: The amount and substantiality of the reuse

The third statutory fair-use factor is "the amount and substantiality of the portion used in relation to the copyrighted work as a whole." 17 U.S.C. § 107. As the jury was properly instructed, JA286-287, this factor (1) measures the amount of material that was reused *relative to* the entire copyrighted work and (2) assesses the importance of the reused portions to the copyrighted work as a whole. The jury was entitled to conclude that those considerations weighed in favor of finding fair use.

The jury heard testimony that the *total* amount of code Google reused comprised less than 0.5% of the

code in the Java SE libraries, which are themselves only a subset of the Java SE work as a whole. JA212. Moreover, there was evidence that although the declarations are necessary to allow code written in the Java language to work on Android, they have no value independent of the implementing code (which Google did not copy) and were not viewed by Sun as proprietary. *See* JA124-125; JA129.

The reused declarations are short and scattered within the copyrighted work. Google's engineers did not reuse anything resembling a contiguous block of computer code; they reused only the highly functional declarations. In holding that "no reasonable jury could conclude that what was copied was qualitatively insignificant," Pet. App. 46a, moreover, the Federal Circuit erroneously applied a legal rule that placed no weight on evidence that the declarations are less important because it is Google's new method implementations that actually carry out the functionality called by the declarations. JA159-160; JA203.

When "no more was taken than necessary" for the new work, this Court has explained, it is difficult "to see how the copying can be excessive in relation" to the purpose of the reuse. *Campbell*, 510 U.S. at 589 (quotation marks omitted). Here, the "jury could reasonably have found that Google's engineering team duplicated the bare minimum of" Java SE, "just enough to preserve inter-system consistency in usage" and "only so much as was reasonably necessary for a transformative use." Pet. App. 114a. The jury thus heard extensive testimony that the engineers reused only the declarations that were necessary to allow developers to use their Java language skills to create creative content for Android or to reuse their own existing code

already written in the Java language. JA169-170; JA200-203; JA213-214; *see* JA139-140.

4. Factor four: The effect on the market for, or value of, the original work

The final statutory fair-use factor is "the effect of the use upon the potential market for or value of the copyrighted work." 17 U.S.C. § 107. The jury was correctly instructed that "[t]his factor considers whether the accused work is offered or used as a substitute for the original copyrighted work" and assesses "the harm, if any, to the potential market for or value of the copyrighted work itself and to its licensing value for it and its derivative works." JA287-288.

The District Court correctly concluded that the "jury could reasonably have found that use of the declaring lines of code (including their SSO) in Android caused no harm to the market for the copyrighted works," which worked on an entirely different class of devices. Pet. App. 114a. The jury heard ample evidence that Android did not supplant or supersede the market for Java SE—and that Android and Java SE did not compete—because Java SE, which was designed for servers and desktop computers, is not suitable for the modern smartphone market. Unlike Android, it did not include the features and functionalities needed for a modern smartphone. JA144-145; JA197-198; JA255-256; *see* JA98-102; JA134-136; JA173-174; JA176-178; JA200; JA227; JA235; JA254; TX7362 at 1; TX2052 at 17.

In concluding to the contrary that "the fourth factor weighs heavily in favor of Oracle," Pet. App. 53a, the Federal Circuit impermissibly relied on its *own* resolution of disputed factual questions about real and

potential market harm, ignoring the voluminous evidence that Java SE had not succeeded in modern smartphones because it was not suited to that environment.

The jury also heard that Oracle's Java SE business was growing well in its intended market of servers and desktop computers, even after the introduction of Android. JA252; TX7788 at 277:24-278:7; *see* TX7787 at 151:10-152:1.

The Federal Circuit's ruling also assigns no weight to the respects in which Android *benefitted* Oracle. The evidence showed that Android's popularity led massive numbers of developers to use the Java language, as Sun had anticipated when Android was announced. JA212-213; JA228; TX2352. Sun itself celebrated Android's release for expanding the Java community, which inevitably created more customers for Oracle's Java-based products. *See supra* at 9.

Further, as the District Court explained, the jury heard evidence that "before Android was released, Sun made all of the Java [SE libraries] available"—including "the very same 37 packages as wound up in Android with the very same SSO"—"as free and open source under the name OpenJDK, subject only to" an easily available license. Pet. App. 115a. The jury was entitled to conclude "that Android's impact on the market for the copyrighted works paralleled what Sun already expected via its OpenJDK." *Ibid.*

The Federal Circuit also erred as a matter of law in holding that Oracle's mere *wish* to enter the smartphone market weighed against a finding of fair use. Pet. App. 51a-52a. If a platform owner can block new entrants from making fair use of its interfaces

merely by *wanting* to compete with those entrants, there would be little hope for the fair-use defense in software copyright cases. For example, if Oracle could lock down any market where a declaration could be reused, it could control any technological development that requires compatibility with the Java language—and could extract licensing fees from every platform developer based on the value of the downstream innovations. It need only express an *interest* in *possibly* entering potential markets in the future. Such a low bar to a finding of market harm would gut the fair-use doctrine and chill a large amount of innovative expression. That is why the Copyright Office has endorsed the view that reusing limited amounts of code to create a "legitimate competitor" is fair use. *Software-Enabled Consumer Products* 59 (citation omitted).

CONCLUSION

The Federal Circuit's rulings are inconsistent with both basic copyright principles and the role of the jury in our justice system. They would disrupt the ongoing development of modern, interoperable computer software. The judgment of the court of appeals should be reversed.

Respectfully submitted,

Kent Walker
Catherine Lacavera
Renny Hwang
GOOGLE LLC
1600 Amphitheatre Parkway
Mountain View, CA 94043

Lisa S. Blatt
David M. Krinsky
Sarah M. Harris
Meng Jia Yang
WILLIAMS & CONNOLLY LLP
725 Twelfth Street, N.W.
Washington, DC 20005

Robert A. Van Nest
Christa M. Anderson
Eugene M. Paige
Reid P. Mullen
KEKER, VAN NEST
& PETERS LLP
633 Battery Street
San Francisco, CA 94111

Thomas C. Goldstein
Counsel of Record
Sarah E. Harrington
Kevin K. Russell
Daniel Woofter
Erica Oleszczuk Evans
GOLDSTEIN & RUSSELL, P.C.
7475 Wisconsin Avenue
Suite 850
Bethesda, MD 20814
(202) 362-0636
tg@goldsteinrussell.com

Michael S. Kwun
KWUN BHANSALI LAZARUS LLP
555 Montgomery Street
Suite 750
San Francisco, CA 94111

Bruce W. Baber
Marisa C. Maleck
KING & SPALDING LLP
1180 Peachtree Street, N.E.
Atlanta, GA 30309

January 6, 2020

ADDENDUM

17 U.S.C. § 101 provides in relevant part:

§ 101. Definitions

Except as otherwise provided in this title, as used in this title, the following terms and their variant forms mean the following:

* * *

A "computer program" is a set of statements or instructions to be used directly or indirectly in a computer in order to bring about a certain result.

* * *

"Literary works" are works, other than audiovisual works, expressed in words, numbers, or other verbal or numerical symbols or indicia, regardless of the nature of the material objects, such as books, periodicals, manuscripts, phonorecords, film, tapes, disks, or cards, in which they are embodied.

* * *

17 U.S.C. § 102 provides:

§ 102. Subject matter of copyright: In general

(a) Copyright protection subsists, in accordance with this title, in original works of authorship fixed in any tangible medium of expression, now known or later developed, from which they can be perceived, reproduced, or otherwise communicated, either directly or with the aid of a machine or device. Works of authorship include the following categories:

(1) literary works;

(2) musical works, including any accompanying words;

(3) dramatic works, including any accompanying music;

(4) pantomimes and choreographic works;

(5) pictorial, graphic, and sculptural works;

(6) motion pictures and other audiovisual works;

(7) sound recordings; and

(8) architectural works.

(b) In no case does copyright protection for an original work of authorship extend to any idea, procedure, process, system, method of operation, concept, principle, or discovery, regardless of the form in which it is described, explained, illustrated, or embodied in such work.

17 U.S.C. § 106 provides in relevant part:

§ 106. Exclusive rights in copyrighted works

Subject to sections 107 through 122, the owner of copyright under this title has the exclusive rights to do and to authorize any of the following:

(1) to reproduce the copyrighted work in copies or phonorecords;

(2) to prepare derivative works based upon the copyrighted work;

* * *

17 U.S.C. § 107 provides:

§ 107. Limitations on exclusive rights: Fair use

Notwithstanding the provisions of sections 106 and 106A, the fair use of a copyrighted work, including such use by reproduction in copies or phonorecords or by any other means specified by that section, for purposes such as criticism, comment, news reporting, teaching (including multiple copies for classroom use), scholarship, or research, is not an infringement of copyright. In determining whether the use made of a work in any particular case is a fair use the factors to be considered shall include—

(1) the purpose and character of the use, including whether such use is of a commercial nature or is for nonprofit educational purposes;

(2) the nature of the copyrighted work;

(3) the amount and substantiality of the portion used in relation to the copyrighted work as a whole; and

(4) the effect of the use upon the potential market for or value of the copyrighted work.

The fact that a work is unpublished shall not itself bar a finding of fair use if such finding is made upon consideration of all the above factors.

17 U.S.C. § 109 provides in relevant part:

§ 109. Limitations on exclusive rights: Effect of transfer of particular copy or phonorecord

* * *

(b)(1)(A) Notwithstanding the provisions of subsection (a), unless authorized by the owners of copyright in the sound recording or the owner of copyright in a computer program (including any tape, disk, or other medium embodying such program), and in the case of a sound recording in the musical works embodied therein, neither the owner of a particular phonorecord nor any person in possession of a particular copy of a computer program (including any tape, disk, or other medium embodying such program), may, for the purposes of direct or indirect commercial advantage, dispose of, or authorize the disposal of, the possession of that phonorecord or computer program (including any tape, disk, or other medium embodying such program) by rental, lease, or lending, or by any other act or practice in the nature of rental, lease, or lending. Nothing in the preceding sentence shall apply to the rental, lease, or lending of a phonorecord for nonprofit purposes by a nonprofit library or nonprofit educational institution. The transfer of possession of a lawfully made copy of a computer program by a nonprofit educational institution to another nonprofit educational institution or to faculty, staff, and students does not constitute rental, lease, or lending for direct or indirect commercial purposes under this subsection.

* * *

17 U.S.C. § 117 provides:

§ 117. Limitations on exclusive rights: Computer programs

(a) MAKING OF ADDITIONAL COPY OR ADAPTATION BY OWNER OF COPY.—Notwithstanding the provisions of section 106, it is not an infringement for the owner of a copy of a computer program to make or authorize the making of another copy or adaptation of that computer program provided:

(1) that such a new copy or adaptation is created as an essential step in the utilization of the computer program in conjunction with a machine and that it is used in no other manner, or

(2) that such new copy or adaptation is for archival purposes only and that all archival copies are destroyed in the event that continued possession of the computer program should cease to be rightful.

(b) LEASE, SALE, OR OTHER TRANSFER OF ADDITIONAL COPY OR ADAPTATION.—Any exact copies prepared in accordance with the provisions of this section may be leased, sold, or otherwise transferred, along with the copy from which such copies were prepared, only as part of the lease, sale, or other transfer of all rights in the program. Adaptations so prepared may be transferred only with the authorization of the copyright owner.

(c) MACHINE MAINTENANCE OR REPAIR.—Notwithstanding the provisions of section 106, it is not an infringement for the owner or lessee of a machine to make or authorize the making of a copy of a computer program if such copy is made solely by virtue of the activation of a machine that lawfully contains an

authorized copy of the computer program, for purposes only of maintenance or repair of that machine, if—

(1) such new copy is used in no other manner and is destroyed immediately after the maintenance or repair is completed; and

(2) with respect to any computer program or part thereof that is not necessary for that machine to be activated, such program or part thereof is not accessed or used other than to make such new copy by virtue of the activation of the machine.

(d) DEFINITIONS.—For purposes of this section—

(1) the "maintenance" of a machine is the servicing of the machine in order to make it work in accordance with its original specifications and any changes to those specifications authorized for that machine; and

(2) the "repair" of a machine is the restoring of the machine to the state of working in accordance with its original specifications and any changes to those specifications authorized for that machine.

17 U.S.C. § 302 provides:

§ 302. Duration of copyright: Works created on or after January 1, 1978

(a) IN GENERAL.—Copyright in a work created on or after January 1, 1978, subsists from its creation and, except as provided by the following subsections, endures for a term consisting of the life of the author and 70 years after the author's death.

(b) JOINT WORKS.—In the case of a joint work prepared by two or more authors who did not work for

hire, the copyright endures for a term consisting of the life of the last surviving author and 70 years after such last surviving author's death.

(c) ANONYMOUS WORKS, PSEUDONYMOUS WORKS, AND WORKS MADE FOR HIRE.—In the case of an anonymous work, a pseudonymous work, or a work made for hire, the copyright endures for a term of 95 years from the year of its first publication, or a term of 120 years from the year of its creation, whichever expires first. If, before the end of such term, the identity of one or more of the authors of an anonymous or pseudonymous work is revealed in the records of a registration made for that work under subsections (a) or (d) of section 408, or in the records provided by this subsection, the copyright in the work endures for the term specified by subsection (a) or (b), based on the life of the author or authors whose identity has been revealed. Any person having an interest in the copyright in an anonymous or pseudonymous work may at any time record, in records to be maintained by the Copyright Office for that purpose, a statement identifying one or more authors of the work; the statement shall also identify the person filing it, the nature of that person's interest, the source of the information recorded, and the particular work affected, and shall comply in form and content with requirements that the Register of Copyrights shall prescribe by regulation.

(d) RECORDS RELATING TO DEATH OF AUTHORS.—Any person having an interest in a copyright may at any time record in the Copyright Office a statement of the date of death of the author of the copyrighted work, or a statement that the author is still living on a particular date. The statement shall identify the person filing it, the nature of that person's interest, and the source

of the information recorded, and shall comply in form and content with requirements that the Register of Copyrights shall prescribe by regulation. The Register shall maintain current records of information relating to the death of authors of copyrighted works, based on such recorded statements and, to the extent the Register considers practicable, on data contained in any of the records of the Copyright Office or in other reference sources.

(e) PRESUMPTION AS TO AUTHOR'S DEATH.—After a period of 95 years from the year of first publication of a work, or a period of 120 years from the year of its creation, whichever expires first, any person who obtains from the Copyright Office a certified report that the records provided by subsection (d) disclose nothing to indicate that the author of the work is living, or died less than 70 years before, is entitled to the benefits of a presumption that the author has been dead for at least 70 years. Reliance in good faith upon this presumption shall be a complete defense to any action for infringement under this title.

17 U.S.C. § 506 provides in relevant part:

§ 506. Criminal offenses

(a) CRIMINAL INFRINGEMENT.—

* * *

(3) DEFINITION.—In this subsection, the term "work being prepared for commercial distribution" means—

(A) a computer program, a musical work, a motion picture or other audiovisual work, or a sound

recording, if, at the time of unauthorized distribution—

(i) the copyright owner has a reasonable expectation of commercial distribution; and

(ii) the copies or phonorecords of the work have not been commercially distributed; or

* * *

17 U.S.C. § 1201 provides in relevant part:

§ 1201. Circumvention of copyright protection systems

* * *

(f) REVERSE ENGINEERING.—(1) Notwithstanding the provisions of subsection (a)(1)(A), a person who has lawfully obtained the right to use a copy of a computer program may circumvent a technological measure that effectively controls access to a particular portion of that program for the sole purpose of identifying and analyzing those elements of the program that are necessary to achieve interoperability of an independently created computer program with other programs, and that have not previously been readily available to the person engaging in the circumvention, to the extent any such acts of identification and analysis do not constitute infringement under this title.

(2) Notwithstanding the provisions of subsections (a)(2) and (b), a person may develop and employ technological means to circumvent a technological measure, or to circumvent protection afforded by a technological measure, in order to enable the identification and analysis under paragraph (1), or for the purpose

of enabling interoperability of an independently created computer program with other programs, if such means are necessary to achieve such interoperability, to the extent that doing so does not constitute infringement under this title.

(3) The information acquired through the acts permitted under paragraph (1), and the means permitted under paragraph (2), may be made available to others if the person referred to in paragraph (1) or (2), as the case may be, provides such information or means solely for the purpose of enabling interoperability of an independently created computer program with other programs, and to the extent that doing so does not constitute infringement under this title or violate applicable law other than this section.

(4) For purposes of this subsection, the term "interoperability" means the ability of computer programs to exchange information, and of such programs mutually to use the information which has been exchanged.

* * *