No. 18-956

In the Supreme Court of the United States

GOOGLE LLC, PETITIONER

v.

ORACLE AMERICA, INC.

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

BRIEF FOR THE UNITED STATES
AS AMICUS CURIAE SUPPORTING RESPONDENT

REGAN A. SMITH
General Counsel and Associate Register of Copyrights
KEVIN R. AMER
Deputy General Counsel
JORDANA S. RUBEL
Assistant General Counsel
United States Copyright Office
Washington, D.C. 20540

MICHAEL J. WALSH, JR.
Acting General Counsel
MEGAN HELLER
Associate Chief Counsel
Department of Commerce
Washington, D.C. 20230

NOEL J. FRANCISCO
Solicitor General
Counsel of Record
JOSEPH H. HUNT
Assistant Attorney General
MALCOLM L. STEWART
Deputy Solicitor General
MATTHEW GUARNIERI
Assistant to the Solicitor General
MARK R. FREEMAN
DANIEL TENNY
SONIA M. CARSON
Attorneys

Department of Justice
Washington, D.C. 20530-0001
SupremeCtBriefs@usdoj.gov
(202) 514-2217

QUESTIONS PRESENTED

The Copyright Act of 1976, 17 U.S.C. 101 *et seq.*, protects "original works of authorship," 17 U.S.C. 102(a), including "computer program[s]," 17 U.S.C. 101. The Act specifies, however, that copyright protection does not "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. 102(b). Under the "merger" doctrine, copyright protection also does not apply when an idea can be expressed in only a limited number of ways, such that the expression and idea "merge." Finally, the Copyright Act provides that "the fair use of a copyrighted work * * * is not an infringement of copyright." 17 U.S.C. 107.

The questions presented are as follows:

1. Whether Section 102(b) or the merger doctrine precludes copyright protection for respondent's original computer code, which defines and organizes a set of functions that are useful in writing computer programs.

2. Whether the court of appeals correctly held that petitioner's verbatim copying of respondent's original computer code into a competing commercial product was not fair use.

TABLE OF CONTENTS

TABLE OF AUTHORITIES

Cases:

Cases—Continued: Page

Statutes:

Statutes—Continued: Page

Miscellaneous:

Miscellaneous—Continued: Page

In the Supreme Court of the United States

No. 18-956

GOOGLE LLC, PETITIONER

v.

ORACLE AMERICA, INC.

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

BRIEF FOR THE UNITED STATES
AS AMICUS CURIAE SUPPORTING RESPONDENT

INTEREST OF THE UNITED STATES

This case presents questions concerning the copyrightability and use of a computer software program. Those issues implicate the expertise and responsibilities of several federal agencies and components. The Copyright Office is responsible for, among other things, determining whether a work is copyrightable before registering a copyright for the work, 17 U.S.C. 410(a), and advising Congress, agencies, the courts, and the public on copyright matters, 17 U.S.C. 701. The United States Patent and Trademark Office, through the Secretary of Commerce, advises the President on intellectual-property matters. 35 U.S.C. 2(b)(8) and (c)(5). At the invitation of the Court, the United States filed a brief as amicus curiae at the petition stage of this case.

STATEMENT

1. a. Under the Copyright Act of 1976, 17 U.S.C. 101 *et seq.*, "[c]opyright protection subsists * * * in original works of authorship." 17 U.S.C. 102(a). "[W]orks of authorship" include "literary works," which are "works, other than audiovisual works, expressed in words, numbers, or other verbal or numerical symbols or indicia." 17 U.S.C. 101, 102(a)(1). To be "original" in the relevant sense, a work must have been "independently created by the author (as opposed to copied from other works)" and must "possess[] at least some minimal degree of creativity." *Feist Publ'ns, Inc.* v. *Rural Tel. Serv. Co.*, 499 U.S. 340, 345 (1991). The copyright in an original literary work extends both to its literal aspects (*i.e.*, the actual text) and to its original non-literal aspects (such as the plot of a novel). See, *e.g.*, *Stewart* v. *Abend*, 495 U.S. 207, 238 (1990).

The Copyright Act defines and limits the rights that a copyright confers. As particularly relevant here, Section 102(b) states that copyright protection does not "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. 102(b). For example, a copyright for a book about a new surgical method would bar others from copying the book without authorization, but not from performing the surgical method. Section 102(b) codifies the longstanding common-law principle known as the "idea/expression dichotomy." *Golan* v. *Holder*, 565 U.S. 302, 328 (2012).

Other common-law doctrines limit the copyrightability of certain expressive works. Under the "'merger doctrine,'" if an idea "can only be expressed in a limited number of ways," those means of expression "cannot be

protected, lest one author own the idea itself." *Zalewski* v. *Cicero Builder Dev., Inc.*, 754 F.3d 95, 102-103 (2d Cir. 2014). In that circumstance, the idea and the expression are said to "merge." Under the doctrine of *scènes-à-faire*, elements of a work that are "standard, stock, or common to a topic," like cowboys and shootouts in stories of the American West, are not copyrightable. *Mitel, Inc.* v. *Iqtel, Inc.*, 124 F.3d 1366, 1374 (10th Cir. 1997); see *Zalewski*, 754 F.3d at 102.

b. A valid copyright gives the owner certain "exclusive rights," including the rights "to reproduce the copyrighted work" and "to prepare derivative works based upon the copyrighted work." 17 U.S.C. 106(1) and (2). But those rights are subject to exceptions and limitations, including the "fair use" doctrine, 17 U.S.C. 107, a "judge-made doctrine" that Congress codified in 1976. *Campbell* v. *Acuff-Rose Music, Inc.*, 510 U.S. 569, 576 (1994). The fair use doctrine permits certain uses of a copyrighted work when imposing infringement liability would "stifle the very creativity which [copyright] law is designed to foster." *Id.* at 577 (citation omitted). Section 107 identifies a nonexclusive list of factors that are relevant to whether a particular use of a copyrighted work constitutes "fair use": (1) "the purpose and character of the use," (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." 17 U.S.C. 107(1)-(4).

c. This case concerns the copyrightability of computer code. To induce a computer to perform a function, a person must give the computer written instructions. Typically, those instructions are written in "source code,"

which consists of words, numbers, and symbols in a particular "programming language," with its own unique syntax and semantics. The source code is then compiled into binary "object code"—ones and zeros—that is readable by the computer.

It is both "firmly established" and undisputed in this case that computer code can be copyrightable as a "literary work[]." 1 Melville B. Nimmer & David Nimmer, *Nimmer on Copyright* § 2A.10[B], at 2A-175 (2019). Section 101 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. And various Copyright Act provisions recognize that a person may own a copyright in a "computer program." See, *e.g.*, 17 U.S.C. 109(b)(1)(A), 117, 506(a)(3)(A).

2. a. In the 1990s, respondent's predecessor-in-interest, Sun Microsystems, Inc. (Sun), created a computer programming language called "Java," along with a variety of tools and software—known collectively as the "Java platform"—to assist software developers in writing and distributing computer programs in Java. Pet. App. 4a.

Like many programming languages, Java allows developers to use short, modular subprograms to create longer, more complex programs. In creating a video game, for example, a developer might create subprograms to perform tasks such as displaying text on the screen or playing a sound. In Java, these subprograms are called "methods." Pet. App. 125a. Sun created a library (referred to herein as the Java Standard Library) of thousands of pre-written methods, which Sun organized hierarchically into "classes" and "packages." *Ibid.* The version at issue here includes 166 packages,

comprising 3000 classes and more than 30,000 methods. *Id.* at 5a. A small fraction of those pre-written methods are necessary to write any program in the Java language. *Id.* at 45a, 102a. The rest are merely convenient building blocks that allow programmers to avoid having to "reinvent[] the wheel[]" by writing new code to perform certain functions. *Id.* at 228a.

Although respondent does not claim a copyright interest in the Java language itself, respondent owns the copyright in the Java Standard Library. Pet. App. 127a. Respondent makes the Java Standard Library available to developers under several different copyright licenses, including a royalty-free license. *Ibid.* For a commercial license, respondent requires the licensee to ensure that products it creates using the Java Standard Library remain compatible with the Java platform. *Id.* at 127a-128a. Respondent considers this rule essential to ensure compatibility of software created using the Java Standard Library with any device or computer that uses the Java platform, a concept known as "write once, run anywhere." *Id.* at 128a. Respondent's commercial licensees for various components of the Java Standard Library include IBM, Red Hat, Amazon (for the Kindle), Nokia, LG, Samsung, and RIM (for the Blackberry). 13-1021 C.A. App. 20,467-20,468, 20,550-20,554.

b. In general, to create a new Java method, a developer must write code that tells the computer both (i) what the method is, including its name, the circumstances in which it should be available to developers, what types of input data it should accept, what types of output data it should produce, and what types of errors it can generate; and (ii) how to perform the method, including steps for using the specified input data to produce the specified type of output data. The parties refer

to the first type of code as "declaring code" and to the second as "implementing code." Pet. App. 126a.

An example drawn from the district court's opinion illustrates the distinction. See Pet. App. 224a-225a (discussing "java.lang.Math.max" method). The following Java code defines a method named "max" that returns the larger of two integers, *x* and *y*:

```
Line 1:  public static int max (int x, int y) {
Line 2:  if (x > y) return x;
Line 3:  else return y;
Line 4:  }
```

See *ibid.* In this example, Line 1 is the "declaring code," which specifies the method's name (*max*); the circumstances in which the method is available to developers (*public* and *static*); the type of output data it produces (*int*, for integer); and the type and order of the input data it accepts (integer *x* and integer *y*). Lines 2-3 are the "implementing code," which specifies how to use the input data to produce the output data. See Resp. Br. 5-6 (giving additional example of declaring code).

Additional declaring code identifies the class and package to which each method belongs. The structure and organization of these groupings was not dictated by the technical features of the Java language. Instead, "the Oracle/Sun developers had a vast range of options" and made deliberate choices about which methods and classes should be grouped together. Pet. App. 140a-141a. Likewise, a software developer who eschewed the prewritten methods in the Java Standard Library, and instead wrote new code to perform the same functions, could arrange those new methods into different groupings of classes and packages. See *id.* at 215a.

Once a method has been written, a developer may invoke or "call" the method by typing a command consisting of the name of the method (which incorporates the method's package and class location) and the appropriate input data. Although that command is determined by the method's declaring code, it is not identical to the declaring code. See Pet. App. 100a-101a, 150a-151a, 228a. The developer does not need to see or understand how the pre-written methods in the Java Standard Library are actually implemented in order to use them as building blocks in other programs. The developer need only know (or look up) the name of the relevant method and the parameters established by its declaring code. *Id.* at 101a-102a.

c. Petitioner developed the Android operating system for mobile devices. Petitioner also created its own platform—*i.e.*, a set of programming tools—to assist others in developing applications for Android. The Android platform uses the Java programming language. Pet. App. 130a. As a result of petitioner's design choices, however, the platforms are not interoperable, *i.e.*, applications written for Android do not function on the Java platform, and vice versa. See *id.* at 46a n.11, 130a, 172a.

Like the Java platform, petitioner's Android platform contains a collection of pre-written methods organized into classes and packages. Petitioner created much of the Android library from scratch. For 37 of the 168 packages included in the Android library, however, petitioner copied respondent's declaring code verbatim, while writing its own implementing code. Pet. App. 129a. The copied packages contained the Java methods and classes that petitioner viewed as most useful for developing

smartphone applications. See Pet. 25. Petitioner asserts that it copied the declaring code so that developers familiar with the Java platform could write programs for the Android platform without learning new commands for invoking commonly used methods. See Pet. 25-26.

Petitioner copied 11,500 lines of respondent's copyrighted code, only 170 of which were necessary to use the Java programming language. Pet. App. 7a, 45a; see Resp. Br. 14 n.2 (respondent's infringement claims rest on the 11,330 non-essential lines). In so doing, petitioner also copied the complex architecture of the 37 packages at issue, including the names and specifications of the thousands of methods and classes in those packages and their hierarchical and interdependent relationships to each other. See Pet. App. 134a.

3. In August 2010, respondent sued petitioner in the Northern District of California, alleging that petitioner had infringed respondent's copyright in the Java Standard Library and had also infringed related patents. Pet. App. 1a-2a, 97a n.2. Respondent's claims of copyright infringement ultimately proceeded on two theories: (i) literal, verbatim copying of the declaring code; and (ii) nonliteral copying of the "structure, sequence, and organization" (SSO) of the Java Standard Library, which the declaring code establishes and reflects. See *id.* at 2a, 132a. The case proceeded to trial, and a jury found infringement but hung on fair use. *Id.* at 122a.

The district court set aside the infringement verdict on the ground that respondent did not possess a valid copyright in the copied material. Pet. App. 212a-272a. The court held that, under Section 102(b), the SSO is ineligible for copyright protection because it constitutes a "method of operation" or "system" for using the pre-

written methods included in the Java platform. *Id.* at 267a. The court also held that the merger doctrine rendered the declaring code uncopyrightable. *Id.* at 264a.

The Federal Circuit, which had appellate jurisdiction because of respondent's since-abandoned patent claims, affirmed in part and reversed in part. Pet. App. 121a-192a. The court held that the declaring code and the SSO were both "entitled to copyright protection," despite their functional character. *Id.* at 123a. With respect to Section 102(b), the court explained that computer programs are "by definition functional," and that the functional character of computer code does not by itself make the code a "'system or method of operation.'" *Id.* at 162a (citation omitted). The court also rejected petitioner's merger argument, explaining that the record "showed that [respondent] had 'unlimited options as to the selection and arrangement'" of the code petitioner copied. *Id.* at 150a (quoting 13-1021 Resp. C.A. Br. 50). Finally, the court remanded for a new trial on fair use. *Id.* at 191a.

This Court denied petitioner's request for review. 135 S. Ct. 2887.

4. On remand, a jury found that petitioner's copying was fair use, and the district court denied respondent's post-trial motions for judgment as a matter of law. Pet. App. 9a; see *id.* at 57a, 92a-120a.

The Federal Circuit reversed. Pet. App. 1a-55a. The court first determined that it was required to defer to the jury's implicit findings "relating to any relevant historical facts," but that the question "whether the use at issue is ultimately a fair one" is a legal determination reviewed de novo. *Id.* at 19a. Applying that framework, the court found that the first and fourth statutory fair use factors—the "purpose and character of the use" and

its "effect * * * upon the potential market for or value of the copyrighted work," 17 U.S.C. 107(1) and (4)—both "weigh[ed] heavily against a finding of fair use." Pet. App. 53a. The court observed that petitioner's copying was "overwhelmingly commercial," *id.* at 28a, and that petitioner's use of the copied material was not "transformative," *id.* at 31a; see *id.* at 33a-35a. The court also found "'overwhelming'" evidence that petitioner's copying had inflicted "actual and potential harm" on the market for respondent's work, including by enabling one of respondent's customers to use the existence of petitioner's Android platform as leverage "to negotiate a steep discount" for continuing to license the Java platform. *Id.* at 50a-51a (citation omitted).

By contrast, the court of appeals viewed the "nature of the copyrighted work," 17 U.S.C. 107(2), as supporting the jury's fair use finding. The court explained that, although writing the declaring code and the SSO had "involved some level of creativity," a reasonable jury "could have concluded that functional considerations were both substantial and important." Pet. App. 42a. The court viewed the remaining fair use factor—the "amount and substantiality of the portion used in relation to the copyrighted work as a whole," 17 U.S.C. 107(3)—as "at best[] neutral." Pet. App. 47a; see *id.* at 45a (noting that petitioner had "copied 11,500 lines of code," and that "only 170 lines of code were necessary to write in the Java language"). Weighing all those factors together, the court held that petitioner's "use of the declaring code and SSO * * * was not fair as a matter of law." *Id.* at 53a.

SUMMARY OF ARGUMENT

I. Respondent holds a valid copyright in the Java Standard Library.

A. The Copyright Act makes clear that computer code may be copyrighted. Petitioner has conceded that the Act's originality requirement, 17 U.S.C. 102(a), is satisfied for both the 11,330 lines of declaring code that it copied and the SSO of the Java Standard Library.

B. Petitioner contends (Br. 19) that the declaring code that it copied is a "method of operation" and therefore uncopyrightable under 17 U.S.C. 102(b). Section 102(b) codifies the idea/expression dichotomy, under which copyright law protects the means of expressing ideas or concepts, but does not give the copyright holder exclusive rights in the ideas or concepts themselves. Section 102(b) does not foreclose copyright protection for respondent's work. The declaring code could be described as a "method of operation" only in the same sense that any computer program could be so described, *i.e.*, that it induces a computer to perform various functions. The Copyright Act as a whole makes clear that computer programs are copyrightable despite that functional character.

C. Petitioner contends (Br. 20-34) that the merger doctrine precludes copyright protection for respondent's work. The merger doctrine reinforces the idea/expression dichotomy by precluding copyright when an idea can be expressed in only a limited number of ways. In the computer-programming context, merger principles apply if code must be written in a specific way (or in one of a small number of ways) to induce a computer to perform a particular function. That doctrine does not

apply here because Sun had "unlimited" expressive options when it designed the Java Standard Library. Pet. App. 150a (citation omitted).

Petitioner's merger argument rests on the assertion that the declaring code it copied is the only way to perform the "function of responding to the calls already known to Java developers." Pet. Br. 19. Petitioner thus asks the Court to perform its merger analysis based on the circumstances that existed when petitioner's copying occurred. But copyrightability is determined as of the time when a work is created. The calls that developers would use did not constrain Sun's options when it wrote the declarations, and those calls became known to developers only after the Java Standard Library was publicly accessible. And because developers' familiarity with the relevant calls is directly attributable to the Library's marketplace success, it would be particularly destructive of sound copyright policy to treat that familiarity as a ground for divesting respondent's work of copyright protection.

D. Petitioner's policy arguments are unpersuasive. Petitioner has not identified any industry understanding that software "interfaces" are per se uncopyrightable, and concerns about the interaction of copyright and emerging technology do not justify such an atextual rule. Petitioner's policy concerns about interoperability are irrelevant here. Petitioner designed its Android platform in a manner that made it *incompatible* with respondent's Java platform.

II. Petitioner's verbatim copying of respondent's original computer code into a competing commercial product was not fair use. The equitable doctrine of fair use, codified in 17 U.S.C. 107, limits the exclusive rights that a copyright confers. Section 107 identifies four

non-exclusive factors to consider in assessing whether a particular use is "fair." The court of appeals correctly held that the first and fourth Section 107 factors weigh so decisively against fair use that the second and third factors cannot tip the balance in petitioner's favor, even with all appropriate deference to the jury's implicit factual findings.

The first statutory factor—the "purpose and character of the use, including whether such use is of a commercial nature," 17 U.S.C. 107(1)—strongly favors respondent. Petitioner's use was commercial and not transformative. Indeed, petitioner used respondent's declaring code for the same purpose for which it was created. The reuse of code can be transformative in other contexts, and the lower courts have appropriately found certain instances of copying to be fair use. But this is not such a case.

The fourth statutory factor—the "effect of the use upon the potential market for or value of the copyrighted work," 17 U.S.C. 107(4)—also heavily favors respondent. The record demonstrated that petitioner's copying harmed the market for respondent's work by (among other things) enabling respondent's customers to use petitioner's Android platform as leverage for discounts. Pet. App. 50a.

Petitioner's remaining fair use arguments lack merit. Petitioner's copying did not serve the essential purposes of copyright, nor was it necessary to foster innovation. Other companies, including Apple, developed successful mobile operating systems without copying respondent's work.

ARGUMENT

I. THE JAVA STANDARD LIBRARY IS COPYRIGHTABLE

The court of appeals correctly held that respondent possesses a valid copyright in the work that petitioner copied. The court also correctly held that respondent's work is not an uncopyrightable "system" or "method of operation," 17 U.S.C. 102(b), and that the merger doctrine does not apply here.

A. The Copyright Act Makes Clear That Computer Code Is Copyrightable Despite Its Functional Character

1. Under the Copyright Act of 1976, computer code is a "'[l]iterary work[]'" because it is a work "expressed in words, numbers, or other verbal or numerical symbols or indicia." 17 U.S.C. 101; see H.R. Rep. No. 1476, 94th Cong., 2d Sess. 54 (1976) (House Report). In 1980, at the recommendation of the National Commission on New Technological Uses of Copyrighted Works (CONTU), Congress amended the Act to confirm explicitly that "computer program[s]" may receive copyright protection. Act of Dec. 12, 1980, Pub. L. No. 96-517, § 10(a), 94 Stat. 3028; see CONTU, *Final Report* 1 (1978) (CONTU Report). The Act now defines the term "'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, and various other provisions recognize that computer programs are copyrightable, *e.g.*, 17 U.S.C. 117.

Copyright law protects the non-literal elements of a literary work, such as the plot of a novel. See *Stewart* v. *Abend*, 495 U.S. 207, 238 (1990) (copyright protection extends to a work's "unique setting, characters, plot, and sequence of events"). That principle applies to computer programs because the Copyright Act makes them

eligible for copyright protection on the same terms as other literary works. Copyright protection thus extends both to the text of a computer program and to the "non-literal components of [the] program, including [its] structure, sequence and organization" (SSO). *Johnson Controls, Inc.* v. *Phoenix Control Sys., Inc.*, 886 F.2d 1173, 1175 (9th Cir. 1989); see *Computer Assocs. Int'l, Inc.* v. *Altai, Inc.*, 982 F.2d 693, 702-710 (2d Cir. 1992); *Whelan Assocs., Inc.* v. *Jaslow Dental Lab., Inc.*, 797 F.2d 1222, 1236 (3d Cir. 1986), cert. denied, 479 U.S. 1031 (1987).

To receive copyright protection, any particular computer code must meet the basic requirements of copyright law, including originality, see 17 U.S.C. 102(a). With respect to originality, "[t]he vast majority of works make the grade quite easily, as they possess some creative spark, 'no matter how crude, humble or obvious' it might be." *Feist Publ'ns, Inc.* v. *Rural Tel. Serv. Co.*, 499 U.S. 340, 345 (1991) (citation omitted).

2. Petitioner copied verbatim 11,330 lines of respondent's computer code. Pet. App. 7a. The 11,330 copied lines were the "declarations" or "declaring code" for thousands of methods (pre-written modular subprograms) in respondent's Java Standard Library. The declaring code specifies certain parameters for each method, including the class and package to which the method belongs. See pp. 5-6, *supra*. Petitioner wrote its own implementing code for each method. By copying the declaring code, however, petitioner also necessarily copied into its own work the structure and organization of respondent's work; the methods for which petitioner wrote its own implementing code appear with the same

names in the same "elaborately organized taxonomy" of classes and packages. Pet. App. 129a.[*]

"The testimony at trial revealed that designing" the complex architecture of the Java Standard Library "was a creative process and that the Sun/Oracle developers had a vast range of options for the structure and organization." Pet. App. 140a-141a; see, *e.g.*, 13-1021 C.A. App. 20,788. Consistent with that evidence, petitioner conceded below that both the 11,330 lines of declaring code that it copied and the SSO of the Java Standard Library satisfy the originality requirement for copyright protection. Pet. App. 140a-141a.

Petitioner suggests (Br. 23) that respondent's declaring code is ineligible for copyright protection because a single declaration is not "a set of statements or instructions" under the Copyright Act's definition of "'computer program,'" 17 U.S.C. 101. The copyrighted work at issue here, however, is the Java Standard Library, from which petitioner copied 11,330 lines of code verbatim. And for purposes of copyrightability under Section 102(a), the whole can be greater than its parts. "By analogy, the opening of Charles Dickens' *A Tale of Two Cities* is nothing but a string of short phrases. Yet no one could contend that this portion of Dickens' work

---

[*] Petitioner refers (Br. 5 & n.2) to the code it copied as the "interface" between the commands typed by developers to invoke a method and the code implementing that method. The "overall set of declarations" in the Java Standard Library is also sometimes called the "Java Application Program[ming] Interface or Java API." Pet. App. 101a-102a; see *id.* at 121a-122a, 193a-196a. This brief generally eschews those terms, which the district court described as "slippery" (*id.* at 197a) because they can be used at varying degrees of generality. Cf. *A Dictionary of Computer Science* 278 (7th ed. 2016) (defining an "interface" broadly as a "[s]pecification of the communication between two program units").

is unworthy of copyright protection because it can be broken into those shorter constituent components." Pet. App. 154a.

B. Section 102(b) Does Not Foreclose Copyright Protection For Respondent's Work

Petitioner contends (Br. 19, 25-26) that respondent's declaring code is an uncopyrightable "method of operation" for using the pre-written functions in the Java Standard Library. 17 U.S.C. 102(b). Despite twice seeking this Court's review to address the application of Section 102(b) to computer software (Pet. 12-14; 14-410 Pet. 13-18), petitioner now presents that argument only in a perfunctory manner. In any event, petitioner's Section 102(b) argument lacks merit.

Section 102(b) codifies the "idea/expression dichotomy," *Golan* v. *Holder*, 565 U.S. 302, 328 (2012), under which a copyright in an "original work[] of authorship," 17 U.S.C. 102(a), covers only the expressive work itself—not the underlying ideas or methods of operation that are "described, explained, illustrated, or embodied" in the work, 17 U.S.C. 102(b). Although a book on how to build a bicycle may be eligible for copyright protection, that copyright does not include any exclusive right to practice the bicycle-building method that the book explains. See *Baker* v. *Selden*, 101 U.S. 99, 105 (1880).

If the Copyright Act contained no explicit references to computer programs, one might reasonably conclude that such programs are not protectable expression. Computer code differs in a fundamental way from many traditional means of literary expression, in that it is the actual means by which a computer is induced to perform desired functions. It therefore would not be unnatural to describe computer code as a "system" or "method of operation" for a computer.

The Copyright Act as a whole makes clear, however, that computer programs are potentially copyrightable. See p. 14, *supra*; Pet. App. 163a. Respondent's declarations are a "system" or "method of operation" only in the same sense that computer programs in general could be described as such—*i.e.*, they enable users of the program to induce a computer to perform certain functions. If that functional character were sufficient to bring respondent's code within Section 102(b), no computer code would qualify for copyright protection.

Read in light of the larger statutory context, Section 102(b) is best understood to foreclose copyright protection only for the underlying ideas or processes implemented in a computer program. See House Report 57; CONTU Report 18-20; see also U.S. Copyright Office, *Software-Enabled Consumer Products* 14 (Dec. 2016) (2016 Copyright Office Report). For example, a copyright in a computer program that checks the accuracy of citations in a legal brief would prevent others from copying the particular code of the program, but not from writing new code to perform the same function. Respondent's assertion of copyright here is consistent with that understanding. Without infringing any copyright, petitioner could and did write its own code in the Java programming language to implement the same processes or methods for which Sun/Oracle had previously written implementing code in the Java Standard Library. But petitioner also copied 11,330 lines of respondent's declaring code into Android, thereby replicating in its own program respondent's creative expression in the text and SSO of the Java Standard Library. Section 102(b) does not support petitioner's challenge to the copyrightability of that work.

C. The Merger Doctrine Is Inapplicable

Petitioner's principal argument on copyrightability (Br. 20-34) is that the materials it copied are covered by the merger doctrine. The court of appeals correctly rejected that contention. Pet. App. 147a-153a.

1. The merger doctrine reinforces the idea/expression dichotomy by precluding copyright when an idea can be expressed in only a limited number of ways. In that circumstance, the idea and its expression "merge," and the expression is uncopyrightable. *Zalewski* v. *Cicero Builder Dev., Inc.*, 754 F.3d 95, 103 (2d Cir. 2014). That rule ensures that no author can use copyright law to obtain a practical monopoly on the idea itself.

A similar principle applies to computer code. If, within a given technological environment, code must be drafted in a specific way (or in one of a limited number of ways) to induce a computer to perform a particular function (*e.g.*, identifying the larger of two integers, see p. 6, *supra*), then the expression and the function "merge," and the code is uncopyrightable. See, *e.g.*, *Computer Assocs.*, 982 F.2d at 708 (observing that "efficiency concerns" in a given programming context may "narrow the practical range of choice as to make only one or two forms of expression workable options," thus potentially giving rise to merger). "If, however, there are multiple ways to carry out [a] process, the merger doctrine would not apply and the author could claim copyright in the expression used to capture the ideas even though the idea itself remained a public good." 2016 Copyright Office Report 15.

When Sun developed the Java Standard Library, it "had 'unlimited' options as to the selection and arrangement" of the methods. Pet. App. 150a (citation omitted). The court of appeals used the "'java.text'" package as an

example, explaining that "Java's creators had to determine whether to include a java.text package in the first place, how long the package would be, what elements to include, how to organize that package, and how it would relate to other packages." *Id.* at 150a-151a n.6 (citation omitted). Accordingly, the text of the declaring code and the complex structure and organization that is expressed in it are creative aspects of respondent's work, not uncopyrightable ideas that could have been expressed in only a limited number of ways.

2. At the time that petitioner copied respondent's declaring code for incorporation into the Android platform, developers who had written programs in the Java language were already familiar with a large number of commands ("calls") that were necessary to invoke commonly used methods. In arguing that the merger doctrine applies here, petitioner emphasizes (Br. 19) that the declarations it copied "can only be written one way to perform their function of responding to the calls already known to Java developers." Petitioner thus invokes the (correct) general rule that, in the programming context, the merger doctrine applies if code must be written in a particular way in order to induce the computer to perform a specified function. For purposes of that general rule, however, petitioner would treat the "function" of declaring code not as (for example) inducing a computer to identify the larger of two integers, but as inducing a computer to identify the larger of two integers *in response to a particular pre-existing call*. That approach is unsound.

a. Until Sun wrote its declaring code, there was no way to identify the "calls" that would invoke the corresponding methods. "Of course, once Sun/Oracle created 'java.lang.Math.max,' programmers who want to use

that particular package have to call it by that name." Pet. App. 151a. More generally, "[i]n writing his or her own Java program, a programmer may only invoke a method with a statement using the precise form defined by the declaring code for the method." *Id.* at 100a. Because the code a developer types to "call[]" a particular method is "defined by the declaring code" (*ibid.*), not the reverse, the calls that developers would use to invoke the various methods in the Java Standard Library did not constrain Sun's range of options in drafting the declarations. From Sun's perspective at the time the Java platform was created, it therefore would get the matter backwards to speak of declarations "be[ing] written one way to perform their function of responding to the calls." Pet. Br. 19.

More generally, when two segments of code are created contemporaneously and are specifically designed to work in tandem, it may often be the case that, so long as one remains unchanged, the other must be held constant as well in order for the two to operate together. Petitioner's argument would suggest that, in that circumstance, the merger doctrine would preclude copyright protection for either segment, on the theory that each could be written in only one way if it was to perform its "function of working with" the other. That approach would unduly constrict the availability of copyright protection for computer code.

b. In arguing that the relevant "function" here is that of "responding to the calls *already known to Java developers*," Pet. Br. 19 (emphasis added), petitioner urges the Court to focus on the state of affairs that existed when petitioner copied the declarations and incorporated them into the Android platform, rather than on the range of options that were available to Sun when it

wrote the declarations several years earlier. The court of appeals correctly rejected that argument, finding it "well-established that copyrightability" should be "evaluated at the time of creation." Pet. App. 151a. To make a work's copyrightability turn on events that occurred years after its creation would be contrary to the basic design of the Copyright Act, under which copyright protection "subsists from [the] creation" of a work through the prescribed statutory term. 17 U.S.C. 302(a). It would also be inconsistent with the CONTU Report (cited at Pet. Br. 30), which recognized that copyright "protect[s] all works of authorship from the moment of their fixation in any tangible medium of expression." CONTU Report 21.

Petitioner's approach is especially misguided because the particular post-creation changed circumstance on which it relies—*i.e.*, developers' acquired familiarity with the calls used to invoke various methods in the Java Standard Library—is a direct result of the Library's marketplace success. "By establishing a marketable right to the use of one's expression, copyright supplies the economic incentive to create and disseminate ideas." *Harper & Row, Publishers, Inc.* v. *Nation Enters.*, 471 U.S. 539, 558 (1985) (citing *Mazer* v. *Stein*, 347 U.S. 201, 209 (1954)). The efficacy of that incentive obviously depends on potential authors' confidence that they will be allowed to reap the benefits if their works attract a following. It therefore would be particularly destructive of copyright principles to treat the popularity of respondent's Java platform, and developers' consequent familiarity with the calls needed to invoke the various methods, as a ground for retroactively divesting respondent's declarations of copyright protection.

3. Petitioner's reliance (Br. 17-18, 22, 30-31) on *Baker* v. *Selden*, *supra*, is misplaced. *Baker* involved a copyrighted book that explained a system of accounting and included forms that could be used to implement the system. The Court held that the copyright in the book itself did not bar others from using substantially similar forms to practice the accounting method that the book described, because the forms were necessary to practice the method and the method itself was not copyrightable. 101 U.S. at 101, 104-105. Petitioner identifies nothing comparable here. Respondent does not claim any right to exclude others from writing their own code to implement any of the functions or processes implemented in the Java Standard Library.

4. Although petitioner claims to have copied only "functional" aspects of respondent's work, the structure and organization of the declaring code that petitioner copied is also expressive. The authors of the Java Standard Library made creative decisions to render it appealing to developers, such as by designing packages to make them "easy and intuitive" to learn. C.A. App. 51,472; see *id.* at 51,459-51,464, 51,472-51,473. Petitioner therefore is wrong in contending (Br. 29) that the only expressive choice made by Sun was "what names to use for the packages, classes, and methods." Petitioner disregards entirely the expression inherent in the SSO of the work.

Aspects of a computer program that are designed to appeal to human developers by being well-organized and easy to learn are not uncopyrightable "functional" aspects of the program. The functional aspect of a computer program that cannot be copyrighted is the underlying idea, process, or method, such as identifying the

larger of two integers, that the program induces a machine to perform. See p. 18, *supra*. By contrast, the characteristics of being readable and understandable to developers are much more analogous to the communicative function of a traditional literary work.

D. Petitioner's Policy Arguments Are Unpersuasive

Petitioner argues (Br. 26-28) that software developers must be free to "reuse" declarations in order to build on existing products, and that "reimplementation" is a common and desirable industry practice that fosters innovation and compatibility. Policy arguments could not justify adopting a rule that "interfaces" (Pet. Br. i) are per se uncopyrightable, where the Copyright Act articulates no such rule. See, *e.g.*, *Azar* v. *Allina Health Servs.*, 139 S. Ct. 1804, 1815 (2019). And even on their own terms, petitioner's policy arguments are unpersuasive.

In positing an industry practice of reusing code, petitioner and its amici do not distinguish between licensed and unlicensed copying. Cf. IBM & Red Hat Amicus Br. 13-18; 83 Computer Scientists Amicus Br. 17-22. Software is "increasingly being distributed under a variety of 'open-source' licenses" that permit the reuse of existing code, subject to licensing conditions. 2016 Copyright Office Report 61. Those practices, however, depend in part on the copyrightability of the licensed code. As amicus Microsoft explained below, in arguing that respondent's work is copyrightable, "[t]he chief legal reason why users must abide by the terms of * * * any open-source license[] is that failing to do so exposes the violator to potential copyright liability." 13-1021 Microsoft C.A. Amicus Br. 16 (emphasis omitted). Because the reuse of code in those circumstances occurs pursuant to a license, the prevalence of such

practices does not support petitioner's proposed rule that "interfaces" are uncopyrightable.

A ruling that declarations and other "interfaces" are categorically ineligible for copyright protection would also be unnecessary to address the concerns that petitioner's amici identify. Rather, other tools are better suited to that task. The fair use doctrine, unlike merger, can take account of the conditions that exist at the time of copying. Copying computer code to achieve interoperability between established and newly developed products can be fair use in some circumstances. See pp. 28-29, *infra*; cf. *Lexmark Int'l, Inc.* v. *Static Control Components, Inc.*, 387 F.3d 522, 544-545 (6th Cir. 2004); *Sony Computer Entm't, Inc.* v. *Connectix Corp.*, 203 F.3d 596, 602-605 (9th Cir.), cert. denied, 531 U.S. 871 (2000). The doctrine of *scènes-à-faire* may also limit or eliminate copyright protection "for elements of a program that are dictated by external factors," such as the "compatibility requirements of other programs with which the program is intended to operate." 2016 Copyright Office Report 16; see *Lexmark*, 387 F.3d at 535-536; *Mitel, Inc.* v. *Iqtel, Inc.*, 124 F.3d 1366, 1374-1375 (10th Cir. 1997). Petitioner invoked that doctrine below but failed to support it. Pet. App. 156a-157a.

In any event, petitioner's policy arguments are inapplicable to this case. Petitioner designed its Android platform in a manner that made it *incompatible* with the Java platform. Pet. App. 46a n.11. Petitioner thus is not seeking to ensure that its new products are compatible with a "legacy product" (Pet. Br. 26). Petitioner instead created a competing platform and copied thousands of lines of code from the Java Standard Library in order to attract software developers familiar with respondent's work. No technical requirement compelled

that copying; indeed, competitors like Apple and Microsoft created their own mobile operating systems without copying respondent's work. Pet. App. 149a n.5.

Petitioner is also wrong in suggesting (Br. 27-28) that its copying was necessary to prevent software developers from becoming locked into the Java platform by virtue of their familiarity with common commands for invoking methods in the Java Standard Library. Petitioner's Android platform required developers to learn many new commands anyway, and developers regularly learn new commands when a new programming language or platform is introduced. See Pet. Br. 7-9. And as explained above (p. 22, *supra*), the core purpose of copyright law is to create appropriate economic incentives for creative expression, by assuring potential authors that they will reap the benefits of any marketplace success their works ultimately achieve. It therefore would be antithetical to sound copyright policy to treat the popularity of the Java platform among developers as a ground for deeming respondent's declaring code uncopyrightable.

II. PETITIONER'S COPYING WAS NOT FAIR USE

The court of appeals correctly held that petitioner's verbatim copying of respondent's original computer code into a competing commercial product was not fair use. Pet. App. 3a. The equitable doctrine of fair use limits the exclusive rights that a copyright otherwise confers. The doctrine permits courts to consider whether "rigid application of the copyright statute" in a particular case "would stifle the very creativity which that law is designed to foster." *Campbell* v. *Acuff-Rose Music, Inc.*, 510 U.S. 569, 577 (1994) (citation omitted). But the fair use doctrine does not permit a new market entrant to

copy valuable parts of an established work simply to attract fans to its own competing commercial product. To the contrary, copying "to get attention or to avoid the drudgery in working up something fresh" actively disserves copyright's goals. *Id.* at 580.

A. Petitioner's Commercial Copying Harmed The Market For Respondent's Work And Was Not Transformative

Section 107 identifies four non-exclusive factors that courts evaluating all forms of work "shall" consider in assessing whether a particular use is fair: (1) "the purpose and character of the use, including whether such use is of a commercial nature"; (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." 17 U.S.C. 107(1)-(4). A court must evaluate all four factors in light of "the goal of copyright, to promote science and the arts." *Campbell*, 510 U.S. at 579. The court of appeals correctly held that, on this record, the first and fourth Section 107 factors weigh so decisively against fair use that the second and third factors cannot tip the balance in petitioner's favor. Pet. App. 53a.

1. Petitioner contends (Br. 34-36) that the court of appeals did not faithfully apply the applicable standard of review. That criticism is unfounded. The court assumed that the jury had resolved any disputed factual issues in petitioner's favor, Pet. App. 23a, and it acknowledged its obligation to "defer[]" to those findings, *id.* at 19a. The court also correctly identified several specific bases for its conclusion that "no reasonable jury" could have ruled for petitioner on fair use, *id.* at

35a; see *id.* at 42a, 46a, 51a, and it determined that petitioner's verbatim copying of 11,330 lines of respondent's code "was not fair as a matter of law," *id.* at 54a.

2. Under the first statutory factor, a court examines whether the defendant's use is commercial, *Campbell*, 510 U.S. at 578, and whether it "adds something new, with a further purpose or different character," *id.* at 579. A "transformative" use furthers the goals of copyright and is more likely to be deemed fair use. *Ibid.* If a work is highly transformative, other factors that may weigh against fair use have "less * * * significance." *Ibid.*

Petitioner did not transform respondent's code by incorporating a verbatim copy into the Android platform. Pet. App. 35a-37a. Just as a copier does not ordinarily give a copyrighted poem a "further purpose or different character" by including it in his own book of poetry, *Campbell*, 510 U.S. at 579, simply copying code from one computer program into the "new technological environment" of another computer program (Pet. Br. 37) is not transformative. Petitioner used respondent's declaring code for the same purpose for which that code had originally been created, without changing its expression, meaning, or message. Cf. *TCA Television Corp.* v. *McCollum*, 839 F.3d 168, 181-183 (2d Cir. 2016) (not transformative use to copy a comedic routine, without altering its meaning, into the new context of a dramatic play), cert. denied, 137 S. Ct. 2175 (2017).

Petitioner contends (Br. 44) that this understanding of the first factor would "dramatically limit any fair use of computer code." But computer code *can* be used in transformative ways—for example, by excerpting code in a textbook to illustrate a coding technique, or by running code through a program to check for plagiarism

(*e.g.*, in a computer-science class). Lower courts have also confronted issues, not presented here, about whether making temporary copies of existing code to "reverse engineer" a system, in order to create compatible works that do not incorporate the pre-existing code, constitutes fair use. See *Sony Computer Entm't*, 203 F.3d at 603-605; *Sega Enters. Ltd.* v. *Accolade, Inc.*, 977 F.2d 1510, 1525-1526 (9th Cir. 1993); *Atari Games Corp.* v. *Nintendo of Am. Inc.*, 975 F.2d 832, 844-845 (Fed. Cir. 1992); see also 2016 Copyright Office Report 54-59. Courts have generally found that copying code to discern how an existing product works, in order to ensure that a new (non-infringing) product is interoperable with the existing product, is a transformative use. But petitioner copied lines of code verbatim from a rival software platform, inserted them into a competing, incompatible platform, and then marketed the infringing product.

Petitioner also asserts (Br. 44-45) that the court of appeals failed to take account of the "functional nature" of computer code. In fact, the court held that the second fair use factor favored petitioner because the jury could reasonably have concluded that "functional considerations were both substantial and important" in creating the declaring code and SSO. Pet. App. 42a. But the court correctly recognized that the incorporation of respondent's declaring code into a new computer program was not a transformative use of that code. See *id.* at 37a.

Petitioner also faults the court of appeals (Br. 45) for purportedly giving undue attention to the code petitioner copied, rather than examining the Android platform as a whole. Of course, "no plagiarist can excuse the wrong by showing how much of his work he did not

pirate." *Harper & Row*, 471 U.S. at 565 (citation omitted). The court of appeals properly analyzed "what [declaring code] does in Java and in Android, how the audience of computer developers perceives it, how much [petitioner] took and added, [and] what the added code does." Pet. App. 33a. The court then correctly recognized that petitioner did not use respondent's code in a novel way. Rather, the copied code still specifies the same methods, in the same packages and classes, for use by developers. Simply surrounding copied material with new material is not a transformative use.

Petitioner suggests (Br. 43-44) that it was entitled to copy 11,330 lines of respondent's declaring code because that code standing alone is not commercially valuable. If that approach were sound, a developer could steal half of another developer's program and finish it herself, so long as the stolen half did not function on its own. By its own account, moreover, petitioner copied respondent's code to make the Android platform more appealing to respondent's fans (Pet. 25-26); and petitioner's primary copyrightability argument (Br. 19-21, 30) is that merger divests the declaring code of copyright protection *because* the code is valuable to developers.

3. As to the fourth factor—the "effect of the use upon the potential market for or value of the copyrighted work," 17 U.S.C. 107(4)—copying that usurps the original work discourages authors from investing the effort that creative expression entails. A defendant therefore "would have difficulty carrying the burden of demonstrating fair use without favorable evidence about relevant markets." *Campbell*, 510 U.S. at 590. Relevant markets include "the market for derivative works," *Harper & Row,* 471 U.S. at 568, which copyright holders enjoy the exclusive right to create and license,

17 U.S.C. 106(2). The fourth factor weighs heavily because "the licensing of derivatives is an important economic incentive to the creation of originals." *Campbell*, 510 U.S. at 593.

Petitioner asserts (Br. 48) that Android and Java "did not compete" because Java "was designed for servers and desktop computers" and "is not suitable for the modern smartphone market." But the record contained "'overwhelming'" evidence that petitioner's copying harmed the market for the Java platform. Pet. App. 50a (citation omitted). That included undisputed evidence that respondent's customer Amazon had used the existence of the Android platform as leverage "to negotiate a steep discount" for continuing to license respondent's Java platform for use in Kindle tablets. *Id.* at 51a. Undisputed evidence also showed that mobile phones used the Java platform and that respondent licensed its work to smartphone manufacturers. *Id.* at 35a. Petitioner's argument also cannot be reconciled with the fourth factor's role in protecting an author's broad right to authorize derivative works, such as a version of the Java Standard Library tailored to "modern smartphones." See 17 U.S.C. 101 (defining a "'derivative work'" to include "any * * * form in which a work may be recast, transformed, or adapted").

Petitioner also observes that respondent made some of its work available "as free and open source under the name OpenJDK, subject only to an easily available license." Pet. Br. 49 (citation and quotation marks omitted). But petitioner declined to take that or any other license, despite "lengthy licensing negotiations" with respondent. Pet. App. 51a. Instead, petitioner simply appropriated the material it wanted. And petitioner's assertion (Br. 37-39) that the Android system as a whole

(including use of the Java language, which was freely available without a license) was beneficial to respondent may be relevant to damages, but it cannot justify petitioner's unauthorized copying. The fact that a wide range of entities have licensed various elements of respondent's work further undermines petitioner's claim (Br. 27-28) that respondent's copyright "erect[s] serious obstacles" to innovation. See 13-1021 C.A. App. 20,467-20,468, 20,550-20,554 (IBM, Red Hat, SAP, Sony, Panasonic, Cisco Systems, Amazon, Nokia, LG, General Electric, eBay, Visa, Samsung, and RIM).

B. Petitioner's Remaining Fair Use Arguments Lack Merit

Petitioner makes several other fair use arguments that are not directly tied to specific Section 107 factors. Each lacks merit.

Petitioner contends (Br. 38-39) that it relied on an "industry understanding that interfaces may be reused," and that the jury could have concluded that petitioner acted in good faith. But any industry practice of reusing code to achieve interoperability would not help petitioner, because petitioner designed its Android platform in a way that made it incompatible with the Java platform. An application written for petitioner's Android platform therefore will not function on respondent's Java platform, and vice versa.

Contrary to petitioner's contention (Br. 41), the Copyright Office has never endorsed the kind of copying in which petitioner engaged. The Office has stated that, "in many cases, copying of appropriately limited amounts of code from one software-enabled product into a competitive one for purposes of compatibility and interoperability should * * * be found to be a fair use." 2016 Copyright Office Report 57; see, *e.g.*, *Sega Enters.*, 977 F.2d at 1523. But enabling developers for Android

to draw on their preexisting knowledge of commands used on the Java platform does not constitute "interoperability" as that term is defined in the Copyright Act or discussed in any judicial decision or Copyright Office publication. See 17 U.S.C. 1201(f) (defining "'interoperability'" for purposes of Section 1201(f) to mean "the ability of computer programs to exchange information, and of such programs mutually to use the information which has been exchanged").

Finally, petitioner asserts (Br. 39) that its copying was justified because the 11,330 lines of respondent's code that petitioner copied verbatim were "a small amount of low-value expression," while the Android platform into which petitioner incorporated the code represents "a large amount of high-value expression." Petitioner does not attempt to explain why, if the expression had such low value, petitioner felt compelled to copy it to attract software developers. In any event, nothing in this Court's precedent or the Copyright Act supports injecting into the evaluation of fair use a subjective, normative assessment of whether particular expression is "low" or "high" value—an inquiry that would force the Court into a role it has repeatedly disclaimed. See, *e.g.*, *Campbell*, 510 U.S. at 582 (observing that "it would be a dangerous undertaking for persons trained only to the law to constitute themselves final judges of the worth of a work") (quoting *Bleistein* v. *Donaldson Lithographing Co.*, 188 U.S. 239, 251 (1903)) (brackets omitted).

CONCLUSION

The judgment of the court of appeals should be affirmed.

Respectfully submitted.

REGAN A. SMITH
General Counsel and Associate Register of Copyrights
KEVIN R. AMER
Deputy General Counsel
JORDANA S. RUBEL
Assistant General Counsel
United States Copyright Office

MICHAEL J. WALSH, JR.
Acting General Counsel
MEGAN HELLER
Associate Chief Counsel
Department of Commerce

NOEL J. FRANCISCO
Solicitor General
JOSEPH H. HUNT
Assistant Attorney General
MALCOLM L. STEWART
Deputy Solicitor General
MATTHEW GUARNIERI
Assistant to the Solicitor General
MARK R. FREEMAN
DANIEL TENNY
SONIA M. CARSON
Attorneys

FEBRUARY 2020