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 SYNOPSYS, INC.
AS AMICUS CURIAE
SUPPORTING RESPONDENT**

I. NEEL CHATTERJEE
GOODWIN PROCTER LLP
601 Marshall Street
Redwood City, CA 94063

WILLIAM M. JAY
Counsel of Record
ANDREW KIM
GOODWIN PROCTER LLP
1900 N Street, N.W.
Washington, DC 20036
wjay@goodwinlaw.com
(202) 346-4000

February 19, 2020

TABLE OF CONTENTS

PAGE

INTEREST OF THE AMICUS CURIAE 1

SUMMARY OF ARGUMENT 4

ARGUMENT 6

I. Oracle's declaring code is copyrightable expression. 6

A. The "statements and instructions" used in computer software are copyrightable as original, creative expression. 6

B. This is a case about creative expression used to achieve a particular function, not the function itself. 8

C. "Merger" does not apply simply because a popular and efficient expression of code is the only one that is currently in use. 11

D. The organization and structure of code are also protected by copyright. 16

II. Google's wholesale copying and use of Oracle's code was not fair use. 17

A. It is not fair use to employ an expression exactly as it was intended and change the platform in which the expression is made..... 19

B. Fair use looks to whether the copied code was "substantial," which requires more than a quantitative comparison.................... 24

C. Using a protected expression for a product introduced into a new market is not fair use. 25

CONCLUSION 28

TABLE OF AUTHORITIES

PAGE(S)

CASES

Am. Dental Ass'n v. Delta Dental Plans Ass'n, 126 F.3d 977 (7th Cir. 1997) 17

Applied Innovations, Inc. v. Regents of Univ. of Minn., 876 F.2d 626 (8th Cir. 1989) 9

ATC Distrib. Grp., Inc. v. Whatever It Takes Transmissions & Parts, Inc., 402 F.3d 700 (6th Cir. 2005) 17

Authors Guild v. Google, Inc., 804 F.3d 202 (2d Cir. 2015) 22

Baker v. Selden, 101 U.S. 99 (1879) 11, 12

Campbell v. Acuff-Rose Music, Inc., 510 U.S. 569 (1994) 19, 20

Cariou v. Prince, 714 F.3d 694 (2d Cir. 2013) 18

Computer Assocs. Int'l, Inc. v. Altai, Inc., 982 F.2d 693 (2d Cir. 1992) 13, 15, 16

Dun & Bradstreet Software Servs., Inc. v. Grace Consulting, Inc.,
307 F.3d 197 (3d Cir. 2002) 8, 16

Feist Publ'ns, Inc. v. Rural Tel. Serv. Co.,
499 U.S. 340 (1991)......................... 12, 16, 17, 27

Harper & Row, Publishers, Inc. v. Nation Enters.,
471 U.S. 539 (1985)......................... 17, 24, 25, 27

Herbert Rosenthal Jewelry Corp. v. Kalpakian,
446 F.2d 738 (9th Cir. 1971)............................. 11

Higgins v. Keuffel,
140 U.S. 428 (1891)... 9

Johnson Controls, Inc. v. Phoenix Control Sys., Inc.,
886 F.2d 1173 (9th Cir. 1989)............................. 8

Kienitz v. Sconnie Nation LLC,
766 F.3d 756 (7th Cir. 2014)....................... 20, 23

Mazer v. Stein,
347 U.S. 201 (1954)... 12

Sega Enters. Ltd. v. Accolade, Inc.,
977 F.2d 1510 (9th Cir. 1992)..................... 13, 23

Soc'y of Holy Transfiguration Monastery, Inc. v. Gregory,
689 F.3d 29 (1st Cir. 2012) 9

Sony Computer Entm't Inc. v.
Connectix Corp.,
203 F.3d 596 (9th Cir. 2000)............................ 23

TCA Television Corp. v. McCollum,
839 F.3d 168 (2d Cir. 2016) 20, 21, 23

Wall Data Inc. v. L.A. Cty. Sheriff's Dep't,
447 F.3d 769 (9th Cir. 2006)............................ 18

Whelan Assocs., Inc. v.
Jaslow Dental Lab., Inc.,
797 F.2d 1222 (3d Cir. 1986) 12, 16

Zomba Enters., Inc. v.
Panorama Records, Inc.,
491 F.3d 574 (6th Cir. 2007)............................ 20

CONSTITUTIONAL PROVISIONS AND STATUTES

U.S. Const. art. I, § 8, cl. 8...................................... 4

17 U.S.C. § 101 .... 7, 21

17 U.S.C. § 102 .... 6, 7, 9, 16

17 U.S.C. § 106 .... 23, 24

17 U.S.C. § 107 .... 17, 20, 24, 26

17 U.S.C. § 117 (Supp. IV 1980) 7

Act of Dec. 12, 1980,
Pub. L. No. 96-517, § 10, 94 Stat. 3028............. 7

Act of Dec. 31, 1974,
Pub. L. No. 93-573, § 201(b)(1)(A),
88 Stat. 1873 6

OTHER AUTHORITIES

Final Report of the National Commission on New Technological Uses of Copyrighted Works (1979).........................*passim*

H.R. Rep. No. 96-1307(II) (1980) 7

H.R. Rep. No. 94-1476 (1976)................................... 6

Brett N. Dorny & Michael K. Friedland, *Copyrighting "Look and Feel": Manufacturing Technologies v. CAMS*, 3 Harv. J. L. & Tech. 195 (1990) 26

Sandra Gonzalez, *How Amazon's 'Man in the High Castle' Honors (and Deviates From) Its Source Material*, Mashable (Oct. 10, 2015), https://mashable.com/2015/10/10/man-in-the-high-castle-2/. 22

Mark A. Lemley, *Convergence in the Law of Software Copyright?* 10 High Tech. L.J. 1 (1995)........... 15

Ronald J. Mann, *Do Patents Facilitate Financing in the Software Industry?* 83 Tex. L. Rev. 961 (2005) 15

Stephen J. Mellor et al., *Why Systems-on-Chip Needs More UML Like a Hole in the Head*, *in UML for SOC Design* (Grant Martin ed., 2005).................................. 14

Oracle, The Java Tutorials: Defining Methods, https://docs.oracle.com/javase/tutorial/java/javaOO/methods.html................................ 9

William F. Patry, *Patry on Copyright* (2019).......................... 11, 21

Eric S. Raymond, *The New Hacker's Dictionary* (3d ed. 1996)... 15

Matthew J. Sottile et al., *Introduction to Concurrency in Programming Languages* (2009)...................... 14

INTEREST OF THE AMICUS CURIAE[1]

Synopsys, Inc. is one of the largest software companies in the world. The company is a leading provider of a suite of software products used for electronic design automation (EDA). Synopsys' EDA software is used to design and test computer chips used in all kinds of computing devices, ranging from computers to smartphones to automobiles to televisions. The company's customers include virtually all semiconductor companies in the world, including all of the top 20 semiconductor companies (such as Intel, Samsung, and Qualcomm).

Synopsys' software products are highly sophisticated computer programs that contain a great deal of creative expression. Synopsys calls its series of instructions within a particular product a "command set." A command set is what a circuit designer uses to communicate with Synopsys' highly complex computer programs. Synopsys' command sets are similar to the "declaring code" at issue in this appeal.

One software product line—the "synthesis" product—allows chip designers to create an abstract concept of what the chip's capabilities should be. From there, the product will transform the abstraction into a layout of the chip, which is a form of a "blueprint" often consisting of hundreds of millions of components. The software will then optimize this layout

[1] All parties have consented to the filing of this brief. No counsel for a party authored any part of this brief, and no such counsel or party made a monetary contribution intended to fund the preparation or submission of this brief. No person other than *amicus curiae* and its counsel made a monetary contribution to the brief's preparation or submission.

according to the desired goals of the chip designer, such as the size and speed of the chip, and how much power the chip consumes. Chip designers specify these detailed goals using a series of commands with parameters. The underlying code implements the instructions.

Another software product line—the "signoff" product—allows a chip designer to independently verify that a chip design will work as intended. This software evaluates how well a chip design will perform based on certain metrics, and whether a design may have vulnerabilities that were previously unknown and unanticipated. Much like the synthesis software, the signoff software has a set of Synopsys-created commands that chip designers use to carry out these functions. The underlying implementing code then carries out the instructions of the commands. The command sets are harmonized across products so that users familiar with the commands for one product line can use another. It is this approach—one that uses familiar and shared language choices—that gives Synopsys a competitive edge.

Synopsys invested hundreds of millions of dollars to develop its suite of products. Because chip design is a very complex process, the effectiveness and efficiency of the command set is important and has a large impact on what can be done with the tooling. Through painstaking research and development, Synopsys developed a comprehensive universe of commands, options to commands, parameters, objects, and attributes for these software products. Synopsys continues to refine its command sets with each new successive generation of technology. Engineers prefer Synopsys' tools in part because similar command sets can be used through different stages of chip development, and its commands have proven to be effective and successful on many chip design projects over time. As a result, customers become familiar with the "Synopsys" way of expressing commands and continue to use Synopsys. Synopsys' command set (and the expressions therein) were so popular and familiar, a competitor decided to copy them for its own software product. *Synopsys, Inc. v. Atoptech, Inc.*, No. 13-cv-2965 (N.D. Cal.). Synopsys was forced to engage in costly copyright litigation to protect the unique expressions used in its software. After three years, it successfully stopped the competitor's infringing products. But Synopsys lost sales to important customers during that time and was only able to regain it after spending millions of dollars to ensure the infringing products were taken off the market. Permanent Injunction and Disposition Order, ECF No. 929, *Synopsys, Inc. v. Atoptech, Inc.*, No. 13-cv-2965 (N.D. Cal. Dec. 19, 2016).

Synopsys submits this amicus brief to challenge the notion, offered by Google and its amici, that the copying of someone else's code is a mainstay of the computer programming world. It is simply not true that "everybody does it," and that software piracy allows for lawful innovative entrepreneurship, as Google suggests. Google's proposed re-interpretation of copyright law would strip software developers of their ability to protect their unique expressions.

Google further argues that computer code is purely functional with no originality or expression to protect. Worse, Google argues that to the extent there is any originality in code, its protection should be jettisoned in the spirit of "innovation"—essentially, "we can compete better through theft." As explained below, Google's arguments ask this Court to turn a blind eye to the plain text of the Copyright Act, as well as this Court's own caselaw.

Synopsys' own experience demonstrates that computer code contains significant originality and expressiveness which is deserving of copyright protection. Treating code as unprotectable, through merger or fair use, would stifle innovation and creativity—not promote it. In the world of computer programming, creativity and innovation come at a cost. Stripped of the assurance that creative code will be rewarded and protected, software developers may not spend the considerable time and resources necessary to create new and innovative software. Google's proposed rule would impede, not promote, "the Progress of Science and useful Arts." U.S. Const. art. I, § 8, cl. 8.

SUMMARY OF ARGUMENT

I. The Copyright Act protects "computer programs" as "literary works." This protection extends beyond the finished software or application. Copyright law also protects compositions of code that are used as part of software, when there are multiple ways to express the idea or function implemented by the code. Google does not dispute this, but it argues that Oracle is trying to copyright the underlying functionality. That is wrong. Oracle is seeking protection for a set of valuable declaring code and its creative word choices that name and describe the functions to be performed. There is more than one way to name and describe a function as well as a series of functions; the particular way that Oracle has selected in its declaring code is expressive and thus protectable. So too is the structure of Oracle's programming library: there is originality and creativity in how Oracle has structured its different pieces of component code.

II. Taking protected works of code and using them exactly as the copyright owner intended to create a competing commercial product is not "fair use." Google claims that its use is fair and "transformative" because Oracle's declarations are being used in two different products: Oracle's desktop software and Google's mobile device operating system. But changing the context in which a copyrighted work is used does not make a work "transformative." The relevant question for fair use is how *the copyrighted work* is used, not where it is used. If the code does exactly the same thing in Google's product as it does in Oracle's, then the "purpose and character" of the code has not changed. Moreover, Google's vision of fair use would allow a competitor to reverse-engineer original software, pilfer the best pieces of code, create new software based on the copied code, and use the stolen code to compete against the original creator. This, according to Google, promotes innovation in software development. But as the drafters of computer copyright laws have recognized and as Synopsys has recognized through its own experience, the promotion of innovation requires exactly the opposite: protecting the *original creator*'s ability to access the market so that the creator may reap the rewards of its innovation.

ARGUMENT

I. Oracle's declaring code is copyrightable expression.

A. The "statements and instructions" used in computer software are copyrightable as original, creative expression.

The Copyright Act of 1976 extends copyright protections to "literary works." 17 U.S.C. § 102(a)(1). When Congress enacted the Act, it understood "literary works" to "include[] computer data bases, and computer programs to the extent that they incorporate authorship in the programmer's expression of original ideas, as distinguished from the ideas themselves." H.R. Rep. No. 94-1476, at 54 (1976), *reprinted in* 1976 U.S.C.C.A.N. 5659, 5667.

The 1976 Act left a placeholder for future developments in computer copyright law. Those developments were to be guided by the National Commission on the New Technological Uses of Copyrighted Works (CONTU), which Congress formed to explore how copyright law could protect "works of authorship . . . in conjunction with automatic systems capable of storing, processing, retrieving, and transferring information." Act of Dec. 31, 1974, Pub. L. No. 93-573, § 201(b)(1)(A), 88 Stat. 1873.

CONTU, however, determined that computer programs required no additional protection, as they were already protected by coverage for other types of expressive works. CONTU, *Final Report of the National Commission on New Technological Uses of*

Copyrighted Works 1 (1979) ("CONTU Report") ("Works created by the use of computers should be afforded copyright protection if they are original works of authorship within [the 1976 Act]."). So when Congress followed up on CONTU's recommendations in 1980,[2] it did not see the need to provide special protections for "computer programs." *See* H.R. Rep. No. 96-1307(II), at 19 (1980), *reprinted in* 1980 U.S.C.C.A.N. 6492, 6509 (Act's purpose was to "clearly apply[] the 1976 law to computer programs"). Rather, it created a limited exception (not at issue here) to the exclusive right to copy a copyrighted "computer program." *See* 17 U.S.C. § 117 (Supp. IV 1980).

Congress accordingly had to define a copyrightable "computer program." It adopted a broad definition: 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." 17 U.S.C. § 101. So if "computer programs" are "literary works," *id.* § 102(a), the Copyright Act's protections extend to any arrangement of code if that arrangement includes more than one "statement[]" or instruction[]," and the arrangement "bring[s] about a certain result."

Like every other copyrightable work, a "computer program" is protectable only to the extent that it is an *expression*—the "idea, procedure, process, system, method of operation, concept, principle, or discovery" itself cannot be copyrighted. 17 U.S.C. § 102(b). CONTU acknowledged that copyright "protects the program so long as it remains fixed in a tangible medium of expression but does not protect the electromechanical functioning of the machine." CONTU

[2] Act of Dec. 12, 1980, Pub. L. No. 96-517, § 10, 94 Stat. 3028.

Report 20. If there were only one set of "specific instructions" that could carry out a particular idea, CONTU concluded it would be a "logical extension of the fundamental principle that copyright cannot protect ideas" to treat those instructions as uncopyrightable. *Id.* But "[w]hen other language *is* available," CONTU acknowledged that a program can be copyrighted; CONTU stated that other programmers could only "read copyrighted programs and use the ideas embodied in them" to "prepar[e] their own works." *Id.*

Because "computer program" is so broadly defined, the Copyright Act's protections for "computer programs" go beyond the end user's application software. Certain literal components within application software are protectable expression—for example, the commands that allow users to instruct the application on what to do and what task to perform. *E.g.*, *Dun & Bradstreet Software Servs., Inc. v. Grace Consulting, Inc.*, 307 F.3d 197, 214 (3d Cir. 2002) (holding that the commands used by a program for preparing employee W-2s are copyrightable). The non-literal components of a computer program, such as the "structure, sequence, and/or organization of [a] program" can also be expressive and copyrightable. *E.g.*, *Johnson Controls, Inc. v. Phoenix Control Sys., Inc.*, 886 F.2d 1173, 1175 (9th Cir. 1989).

B. This is a case about creative expression used to achieve a particular function, not the function itself.

Google does not dispute that copyright protections should extend to code. Google Br. 25 (arguing that implementing code is protectable creative expression). But it claims that Oracle's declarations *cannot* be expressive because they are meant to do one thing: they *function* as a means of calling up implementing code. *Id.* at 20. Because declarations serve a function, Google argues, they are uncopyrightable "methods of operation" under § 102(b). *Id.*

Google's argument misses the point: Oracle's declarations are not the functions themselves; rather, they are Oracle's creative expressions of how to carry out particular functions. The purpose of each portion of declaring code is to provide, among other things, a name for the method, the names of the variables to be used, and a description of when the function can or should be used. Oracle, The Java Tutorials: Defining Methods, https://docs.oracle.com/javase/tutorial/java/javaOO/methods.html. The words are used together in a unique and meaningful sentence, and each set of declaring code uniquely expresses a set of activities. And the set of declaring code collectively creates multiple expressions for multiple activities. Oracle's code goes beyond "mere labels": each declaration is a composition that is "intended to instruct" on the function to be carried out. *Higgins v. Keuffel*, 140 U.S. 428, 431-32 (1891). An individual work of declaring code has creativity in it that can be considered protectable creative expression. *See Applied Innovations, Inc. v. Regents of Univ. of Minn.*, 876 F.2d 626, 634-35 (8th Cir. 1989) (concluding that 550 "short, simple declarative sentences" are copyrightable); *Soc'y of Holy Transfiguration Monastery, Inc. v. Gregory*, 689 F.3d 29, 50-52 (1st Cir. 2012) ("not all short phrases will automatically be deemed uncopyrightable"). There is creativity involved in crafting almost every part of a declaration. Take one aspect of Google's example of the declaring code for determining the larger of two integers: the name of the declaration *java.lang.Math.max*. Google Br. 6. Oracle decided that the best name for the function is *max*. Java developers, including Google's programmers, grew accustomed to using that nomenclature. But Oracle—or Google—could have called that function *LargeInt* (larger integer), *GreaterInt* (greater of the two integers), *HighestNumber*, *MaxNumber*, or a variety of other conventions to accomplish the same thing. Oracle is not trying to copyright the concept of taking the larger of two integers—it is instead seeking to copyright how that function is expressed, including the name that Oracle has selected for that function.

In Synopsys' own experience, users value thoughtfulness and creativity in coding, even for relatively minute aspects such as the names of commands. Synopsys expended considerable resources to develop commands that are familiar to chip designers and efficiently expedite chip development. For example, in Synopsys' PrimeTime timing analysis software, users conducting timing tests can instruct the software to make a certain assumption about a change in the voltage rate over time (the slew rate) if one variable (the drive resistance) is lower than the other (network impedance). The command to call up that assumption is:

rc_degrade_min_slew_when_rd_less_than_rnet.

Here, "degrade_min_slew" refers to the function to be performed (an assumption about a change in the slew rate), and "rd" and "rnet" refer to the two variables, drive resistance and network impedance. "Less than" is a description of when the assumption should be made, *i.e.*, when one variable is lower than the other.

Synopsys did not need to assign this particular name to this particular function. It could have named the command (1) *threshold_resistance_ratio_drivenet*, (2) *dnr_ratio_thresh*, (3) *rndr_adjust_limit*, or (4) *accuracy_threshold_ratio*, among other expressions. But Synopsys named this particular function in a way it believed struck the right balance between accurate description and efficiency. That is a creative expression deserving of copyright protection.

C. "Merger" does not apply simply because a popular and efficient expression of code is the only one that is currently in use.

1. Google's copyrightability argument is ultimately about efficiency: if a developer creates an expression for a function, the expression enters common usage, and no one else develops an alternative expression, the expression becomes uncopyrightable because programmers should not be required to start over and create new expressions from scratch. Google Br. 27, 29 (Sun "made only one expressive choice," and protecting that choice against unlicensed copying would force "Java developers to learn thousands of new calls to replace those they already know"). Google claims that, under the doctrine of merger,[3] the lone

[3] At the outset, there is some doubt as to whether there is a "merger" doctrine at all. Citing this Court's decision in *Baker v. Selden*, 101 U.S. 99 (1879), courts have crafted a "merger" rule based on the idea/expression distinction: when an idea and expression are "inseparable," the two merge together, and the expression become uncopyrightable. *Herbert Rosenthal Jewelry Corp. v. Kalpakian*, 446 F.2d 738, 742 (9th Cir. 1971). But the problem with this so-called doctrine is that *Baker* never mentions it—that case was about whether a copyright in a book describing an accounting system extends protections for the system itself. William F. Patry, 2 *Patry on Copyright* § 4:46 (2019) (casting merger as

expression is uncopyrightable because there is no other way of expressing the function. *Id.* at 30-31. But nothing in copyright law, not even "merger," allows the creative laziness of copiers to determine the copyrightability of works that otherwise should be protected.

"Merger" requires Google to establish that no one *can* develop another expression, which Google has failed to demonstrate. *Whelan Assocs., Inc. v. Jaslow Dental Lab., Inc.*, 797 F.2d 1222, 1236 (3d Cir. 1986) ("everything that is not necessary to the purpose or function" of a computer program is "part of the expression of the idea"). All Google has argued is that no one *has* developed another expression. As the Federal Circuit correctly observed, "nothing prevented Google from writing its own declaring code, along with its own implementing code, to achieve the same result [as Oracle's declaring code]." Pet App. 151-152a. "Because alternative expressions were available, there is no merger." *Id.* at 151a (citation omitted).

"merely a judgment that there is a lack of originality"). CONTU itself recognized that *Baker*'s holding "is often misconstrued as imposing a limit on the copyrightability of works which express ideas, systems, or processes." CONTU Report 19. According to CONTU, *Baker* "properly stands for the proposition that *using* the system does not infringe the copyright in the description." *Id.* (emphasis added). CONTU believed "the rationale for the doctrine of *Baker v. Selden* in no event justifies the denial of copyrightability to any work." *Id.* (quoting 1 *Nimmer on Copyright* § 37.31 (1976)). This accords with how this Court has described *Baker*'s holding. *See Feist Publ'ns, Inc. v. Rural Tel. Serv. Co.*, 499 U.S. 340, 350 (1991); *Mazer v. Stein*, 347 U.S. 201, 217 (1954). And to the extent that *Baker* speaks to originality, Google has conceded that issue.

2. Google contends that "merger" should apply because it would be inefficient to require programmers to rewrite existing tools from scratch. Google Br. 26. But efficiency should play no role in determining whether there is only one expression of an idea. Google's misplaced focus on efficiency—which is not part of the copyrightability analysis—may stem from decisions from the courts of appeals that discuss the "industry-wide goal" of efficiency in computer programming as part of the merger analysis. *See, e.g.*, *Computer Assocs. Int'l, Inc. v. Altai, Inc.*, 982 F.2d 693, 708 (2d Cir. 1992); *see also Sega Enters. Ltd. v. Accolade, Inc.*, 977 F.2d 1510, 1525 (9th Cir. 1992) (as computer programs are "essentially utilitarian," "many aspects of the program are not protected by copyright"). *Altai* and cases that follow it reason that "there may be only a limited number of efficient implementations for any given program tasks," so "multiple programmers, working independently, will design the identical method employed in the allegedly infringed work." 982 F.2d at 708. Because "efficiency concerns may so narrow the practical range of choice as to make only one or two forms of expression workable options," the one or two forms of expression that are typically used are considered by these courts to be uncopyrightable under the merger doctrine. *Id.*

To be clear, even *Altai* would not save Google from an infringement claim here: *Altai* merely observes that programmers may *independently* arrive at the same expression, while Google advocates for a copyrightability standard by which it can deliberately steal other programmers' code. But even the Second Circuit's efficiency-driven analysis is unrealistic—in particular, the assumption that there is only one efficient way to code something. There are many aspects of computer software that can be expressed in multiple ways. Naming a command that performs a specific function is just one example. *See supra* pp. 9-11. Implementing code provides another. An instruction, for example, that is designed to detect whether a letter is a vowel can be written as:

```
    if (letter == ‘a’ | | letter == ‘e’ | |letter == ‘i’ [] letter
== ‘o’ | | letter == ‘u’) {
        puts(“That letter is a vowel.”);
    } else {
        puts(“That letter is not a vowel.”);
    }
```

But the same instruction can be written as:

```
    switch (letter) {
    case ‘a’ : case ‘e’: case ‘i’:  case ‘o’:  case ‘u’:
        puts(“That letter is a vowel.”);
        break;
    default:
        puts(“That letter is not a vowel.”);
    }
```

Indeed, there are different schools of thought on how code should be expressed. Proponents of “syntactic sugar” favor the use of programming shorthand, which presents a function in a shorter, more aesthetically pleasing format. *See* Matthew J. Sottile et al., *Introduction to Concurrency in Programming Languages* 151 (2009); *see also* Stephen J. Mellor et al., *Why Systems-on-Chip Needs More UML Like a Hole in* *the Head*, *in UML for SOC Design* 23 (Grant Martin ed., 2005) (describing syntactic sugar as "a sweeter way of expressing the same thing," and noting that discerning "which one is syntactic sugar, and which the one true way of expressing the statement is often a matter of heated debate"). Advocates for "syntactic salt," by contrast, favor the "showing of work." Instead of using shorthand for a certain action, the programmer spells out everything that is necessary for a script to be understood and carried out, so as to avoid the possibility of writing "bad code." Eric S. Raymond, *The New Hacker's Dictionary* 431 (3d ed. 1996). These two schools of thought suggest that for many programmed functions, there is more than one way of expressing the function and that there is no uniform view as to what is truly "efficient." As with all literary works, the quality is subjective based upon the artistic choices of the coder.

The Second Circuit's obsolete understanding of how computer programming works led to its creation of a confusing merger test that has severely limited the universe of protectable code.[4] *See* Mark A. Lemley, *Convergence in the Law of Software Copyright?* 10 High Tech. L.J. 1, 14 (1995) (recounting two lawyers' descriptions of the Second Circuit's test as a "legal Chernobyl" that "complicated the simple and confounded the complex"); Ronald J. Mann, *Do Patents*

[4] Under this test, "a court would first break down the allegedly infringed program into its constituent structural parts. Then, by examining each of these parts for such things as incorporated ideas, expression that is necessarily incidental to those ideas, and elements taken from the public domain, a court would be able to sift out all non-protectable material." *Computer Assocs. Int'l, Inc. v. Altai*, 982 F.2d 693, 706 (2d Cir. 1992).

Facilitate Financing in the Software Industry? 83 Tex. L. Rev. 961, 971 (2005) (*Altai* "made it difficult to obtain copyright protection for the broader structural features of programs"). Although Google's merger arguments fail under any understanding of the "doctrine," this Court should reject any framework that has a restrictive view of whether code can be "expressive." All that the Copyright Act requires is that there be more than one expression of an idea. Thus, "everything that is not necessary to the purpose or function" should be treated as protectable expression. *Whelan Assocs.*, 797 F.2d at 1236; *see also Dun & Bradstreet*, 307 F.3d at 216 (rejecting *Altai*'s "doctrine of externalities").

D. The organization and structure of code are also protected by copyright.

Google also challenges the copyrightability of "[t]he organizational system of the Java SE libraries." Google Br. 19. How Oracle chooses to arrange its code, Google contends, is merely a "system" that is expressly left unprotected under § 102(b). *Id.*

Google's argument continues to confuse the idea—here, the "system"—with an expression of that idea. *See Feist Publ'ns, Inc. v. Rural Tel. Serv. Co.*, 499 U.S. 340, 356 (1991) (section 102(b) merely "restate[s] [] the basic dichotomy between expression and idea," and does not "enlarge[] or contract[] the scope of copyright protection" (citation omitted)). As the Federal Circuit explained, the Java library is like a protectable taxonomy, a method of organizing "over six thousand commands to carry out pre-assigned functions." Pet. App. 162a. Oracle's code will not organize itself, and there is more than one way of organizing it.

"Classification is a creative endeavor," *Am. Dental Ass'n v. Delta Dental Plans Ass'n*, 126 F.3d 977, 979 (7th Cir. 1997), and there is "creative thought" that goes into an "underlying taxonomy." *ATC Distrib. Grp., Inc. v. Whatever It Takes Transmissions & Parts, Inc.*, 402 F.3d 700, 708 (6th Cir. 2005); *see also Feist*, 499 U.S. at 348 (noting that there are creative "choices as to selection and arrangement"). That Oracle's library collects and organizes expressions does not make it into a "system," and it also does not change the answer to the copyrightability question—there is still creativity in how those expressions are to be organized. *See Am. Dental Ass'n*, 126 F.3d at 980-81 (observing that a word processor is not a "system" "just because it has a command structure for producing paragraphs").

II. Google's wholesale copying and use of Oracle's code was not fair use.

Commercial plagiarism is the worst form of piracy, and fair use does not excuse it. Whether the use of a work is "fair" depends on four nonexclusive factors: "(1) the purpose and character of the use; (2) the nature of the copyrighted work; (3) the substantiality of the portion used in relation to the copyrighted work as a whole; [and] (4) the effect on the potential market for or value of the copyrighted work." *Harper & Row, Publishers, Inc. v. Nation Enters.*, 471 U.S. 539, 560-61 (1985).

While the four factors may be nonexclusive, they are expressly in the fair use statute. 17 U.S.C. § 107. What is not in the statute are two additional considerations that Google seeks to include as part of the fair use inquiry: (1) whether "the custom or public policy at the time would have defined the use as reasonable," Google Br. 38 (citation omitted); and (2) whether "the reuse of a small amount of low-value expression . . . would unleash a large amount of high-value expression," *id.* at 39. Since the codification of the fair-use factors, the first consideration has never been the basis of any fair-use decision.[5] And the comparative judgments of the second consideration—that the use of a copied expression in a secondary work is more "valuable" than the use of the expression in the original work—has never been part of the fair-use inquiry at all. The value added by a secondary use has only been relevant insofar as it demonstrates that the secondary work has "a different manner or [] a different purpose from the original." *Cariou v. Prince*, 714 F.3d 694, 706 (2d Cir. 2013).

Google offers three reasons why its use of Oracle's code was fair: (1) Oracle intended for the code to be used for desktop software, while Google is using it for mobile devices; (2) Google took only code that was necessary for its product; and (3) Oracle's code allowed Google to introduce an "innovative" new product into the smartphone market, a market that was untapped by Oracle. But none of these reasons is statutory *fair use* that excuses Google's copying of Oracle's declaring code: (1) the change of format from desktops to mobile devices does not change the "purpose" of Oracle's copyrighted work; (2) Google's use of Oracle's code was

[5] One court has stated *in dicta* that courts should "bear in mind" the "custom or public policy" associated with the copyrighted work—but that court never looked to an actual custom or policy to discern fair use (and even held there was no fair use in that case). *Wall Data Inc. v. L.A. Cty. Sheriff's Dep't*, 447 F.3d 769, 778 (9th Cir. 2006).

not "insubstantial"; and (3) fair use should never allow a work to be copied for the purpose of being used as part of a competing product.

A. It is not fair use to employ an expression exactly as it was intended and change the platform in which the expression is made.

Google claims that its use of Oracle's code was fair because the work that it created by copying that code—the Android operating system—was "innovative," "revolutionary," and unlike any other use of Java. Google Br. 43-44. But as the Federal Circuit pointed out, whether the end result is "innovative" has nothing to do with whether the use of the *copyrighted work* was fair. Pet. App. 33a-36a. Because Google conceded here that its code and Oracle's code "serve[d] the same function," *id.* at 30a, Google's use of Oracle's code was derivative, not fair. *Id.* at 35a.

A copied work can be "fairly" used if the secondary work created from that use is "transformative." To be "transformative," the secondary work must do more than "merely supersede[] the objects of the original creation"—it must "add[] something new, with a further purpose or character, altering the first with a new expression, meaning, or message." *Campbell v. Acuff-Rose Music, Inc.*, 510 U.S. 569, 579 (1994). By contrast, a work is derivative (and, thus, not transformative) if it uses the copyrighted work exactly as it was originally used, even if more is added to the original expression.

1. Insofar as "transformation" is concerned, Google's focus on the Android operating system as a whole is misplaced. The Federal Circuit correctly

recognized that changing the environment in which code is used does not make the use of the code "transformative." Pet. App. 35a. The focus should be on the copyrighted work itself—to determine whether "the original creation" has taken on a "further purpose or different character." *Campbell*, 510 U.S. at 579; *TCA Television Corp. v. McCollum*, 839 F.3d 168, 180 (2d Cir. 2016) ("critical inquiry" is "whether the new work uses the copyrighted material itself for a new purpose"); *Zomba Enters., Inc. v. Panorama Records, Inc.*, 491 F.3d 574, 582 (6th Cir. 2007) ("the end-user's utilization of the product is largely irrelevant").

It is classic infringement, not fair use, to take a copyrighted work and use it as-is, even if there is new expression surrounding the secondary work's use of the copied expression. "The fair-use privilege under § 107 is not designed to protect lazy appropriators." *Kienitz v. Sconnie Nation LLC*, 766 F.3d 756, 759 (7th Cir. 2014). For example, in *TCA*, the Second Circuit considered whether a contemporary play's use of Abbott and Costello sketch "Who's on First" was sufficiently transformative so as to have a new purpose or character. The play used the sketch exactly as it was delivered by Abbott and Costello—but instead having the bit performed by two men engaging in quick verbal wordplay, a teen and his sock puppet alter ego delivered it instead. 839 F.3d at 176-77.

The play's creators argued that the purpose of the overall work—the play itself—was different from "Who's on First": the former was intended to be a "dark comedy" about Bible Belt values, the latter was famed "vaudevillian humor." *Id.* at 170. The court of appeals rejected that argument, explaining that "the focus of inquiry is not simply on the new work, i.e., on whether that work serves a purpose or conveys an overall expression, meaning, or message different from the copyrighted material it appropriates." *Id.* at 180 (citing *Campbell*, 510 U.S. at 579). The "critical inquiry," the court observed, was "whether the new work uses the copyrighted material itself for a purpose, or imbues it with a character, different from that for which it was created." *Id.* The play used "Who's on First" exactly as Abbott and Costello intended: a comedic exchange that takes advantage of clever wordplay. That is copying, not fair use.

2. The fact that Google has added more to Oracle's code does not make its use of that code "transformative" or "fair." Google is still relying on Oracle's original expression—it needs Oracle's declaring code to serve the same purpose that it serves as part of Oracle's software. Because of that reliance, Google's use of Oracle's work is derivative—it has simply "adapted" the declaring code by adding more to it, while relying on the original expression. 17 U.S.C. § 101 ("derivative work"); William F. Patry, 4 *Patry on Copyright* § 10:21 (2019) (derivative works "result in complementary synergy to the original," and "[w]here the secondary works are, in essence, the original but in a different version, the availability of fair use may be questioned"). Google took Oracle's coded expressions, *used them exactly as intended*, and put together a new, finished work using Oracle's expressions. That is classic derivativeness.

Conceptually, Google's use of Oracle's work is no different than converting a novel into a television series that builds on the novel's plotlines. There is certainly unique creative expression in Amazon's *The Man in the High Castle*, for example. The show presents the themes, major plot points, and characters of Philip K. Dick's novel in a new format. But it also tells new storylines (and introduces new visual aesthetics) that were not in the original book. Sandra Gonzalez, *How Amazon's 'Man in the High Castle' Honors (and Deviates From) Its Source Material*, Mashable (Oct. 10, 2015), https://mashable.com/2015/10/10/man-in-the-high-castle-2/. Yet the television show is clearly a derivative of Dick's novel, as it presents (and builds on) Dick's protected expressions in a changed form. *See Authors Guild v. Google, Inc.*, 804 F.3d 202, 215-16 (2d Cir. 2015) (explaining that derivatives, like "the adaptation of a novel into a movie or play," "generally involve transformations in the nature of changes of form," and thus do not fall under fair use).

The same is true here: Google presented Oracle's code in a different form (code for a mobile operating system); yet, in that form, Oracle's expressions did exactly the same thing as it did in the original work. For Google's use of Oracle's work to be truly transformative, the code had to serve an entirely new function. Suppose, for example, an artist seeking to portray how computers are taking over the world decided to create a piece of artwork displaying lines of Oracle's JAVA SE code with the caption "write once, use anywhere." And perhaps that artist flashed lines of declaring code that changed over time. That would truly be putting Oracle's code to a different, potentially transformative use.

Google's use in this case is considerably different from a transformative one, because Google's "copying [was] verbatim, for an identical function or purpose, and there [were] no changes to the expressive content or message," the fact that the surrounding *format* had changed was not enough to make Google's use a fair one. Pet. App. 37a. If the copied code serves the same purpose as the original, there is no transformation, for "there is 'nothing transformative' about using an original work 'in the manner it was made to be' used." *TCA*, 839 F.3d at 182-83 (quoting *On Davis v. Gap, Inc.*, 246 F.3d 152, 174 (2d Cir. 2001)).

3. Google argues that fair use requires that programmers be able to take existing code and use their own "enormous innovation and creativity" to develop new applications for new platforms, and to free developers from the constraints of "Oracle-approved platforms." Google Br. 40. Missing from Google's brief is any explanation as to why that is an acceptable "purpose." It is true that some courts have blessed the copying of code in the name of "interoperability," *i.e.*, for the development of "new products" for use on "new platforms." *Sony Computer Entm't Inc. v. Connectix Corp.*, 203 F.3d 596, 606 (9th Cir. 2000); *Sega*, 977 F.2d at 1514-15. But as the Federal Circuit correctly pointed out, none of those courts allowed the copied code to be used in a finished, competing product—the copies were only "intermediate" ones. *Sega*, 977 F.2d at 1522; *see also* Pet. App. 32a. Had *Sony* and *Sega* involved fact patterns where the copied code was used in the finished secondary work, those decisions would have impermissibly opened the door for all derivative works to be treated as transformative ones, defeating the whole point of extending copyright protections to derivative works yet to be created. 17 U.S.C. § 106(2); *see also Kienitz*, 766 F.3d at 758 ("To say that a new use transforms the work is precisely to say that it is

derivative and thus, one might suppose, protected under § 106(2).").

B. Fair use looks to whether the copied code was "substantial," which requires more than a quantitative comparison.

Fair use depends on "the amount and substantiality of the portion used in relation to the copyrighted work as a whole." 17 U.S.C. § 107(3). Google says (at 46-47) that its use of Oracle's code was fair because it took "less than 0.5% of code in the Java SE libraries." That goes to "amount," but not "substantiality." Google's use of Oracle's code was small but substantial, which is a further reason it was not "fair."

The amount of copying alone has never determined whether the use of a work is "fair." Taking fifty-five seconds out of a film that lasts an hour and 29 minutes (1.5%) can still be so substantial as to not be "fair." *Harper & Row*, 471 U.S. at 565. Instead, the proper focus is on "the qualitative nature of the taking." *Id.* Imagine missing an entire whodunit film except for the last three minutes where the killer is revealed. The last three minutes may not be "substantial," but still be highly significant as a qualitative matter.

On this point, Google's argument defeats itself: it admits that Oracle's declaring code was "necessary." Google Br. 47. Declaring instructions, Google explains, "are the building blocks of larger, creative computer programs," such as Oracle's Java SE software. *Id.* at 23. Java SE would not function without the instructions. The declarations must therefore be *substantial*, even if Google used only a handful of them. It is not "fair" to take even a small fraction of what lies at the "heart" of a protected work to create a potentially competing work. *Harper & Row*, 471 U.S. at 564-66.

C. Using a protected expression for a product introduced into a new market is not fair use.

The effect that a secondary work has on the market for the original copyrighted work is "undoubtedly the single most important element of fair use." *Id.* at 566. The market inquiry takes into account "not only [] harm to the original but also [] harm to the market for derivative works." *Id.* at 568.

Google's vision of fair use would always favor a new application over an original work, so long as the new application is "innovative" in some way. It contends that the marketability of original works must yield to the development of new works—to have it any other way would "threaten[] the viability of the interconnected software ecosystem." Google Br. 28 (quoting Microsoft Cert. Br. 21); *see also id.* at 49-50 (considering the impact on an original work's potential markets would "chill a large amount of innovative expression").

But Google fails to recognize that reducing copyright protection for original computer programs also has the effect of killing innovation. As CONTU itself recognized, "[t]he cost of developing computer programs is far greater than the cost of their duplication." CONTU Report 11. As a result, "some form of protection is necessary to encourage the creation and broad distribution of computer programs in a competitive market." *Id.*

Setting aside the implication for potential markets, Google would extend its innovation rationale to *actual* markets (and to direct competition). Google states that companies *should* be able to copy code "to create a 'legitimate competitor.'" Google Br. 50. In other words, a competitor can take an original work, strip out the parts that are most attractive to consumers, and use those parts to create a new application. If a competitor purports to offer a better version of popular original software, consumers will inevitably shift their product preferences, leaving the original creator with a reduced share of the market and the lion's share of the development costs. Brett N. Dorny & Michael K. Friedland, *Copyrighting "Look and Feel": Manufacturing Technologies v. CAMS*, 3 Harv. J. L. & Tech. 195, 196 (1990) (describing the software copycat "free rider" problem").

That was certainly not what Congress intended, given that the fair use statute expressly instructs courts to consider "the potential market for . . . the copyrighted work." 17 U.S.C. § 107(4). Google's vision is also irreconcilable with CONTU's—CONTU believed protection was necessary to encourage "broad distribution" of the original work. CONTU Report 11. "Broad distribution" would not be possible if fair use allows a competitor to steal substantial parts of an original work and dress up the stolen code as its own.

Synopsys has already had a taste of Google's dim view of copyright protection. When a competitor launched a product that copied the commands used in Synopsys' software products, Synopsys lost over $100 million in sales. The competitor had clearly captured a portion of Synopsys' market by using Synopsys' code: the competing software took in $129 million in revenue. Synopsys eventually defeated the competitor's fair-use claim and recovered most of its lost share of the market. But under Google's "innovation first" approach to fair use, it would not have been able to do so—the competitor could have simply claimed that its "new application" was an innovative necessity, even if the new application depended on a significant portion of Synopsys' code.

Google fails to explain how the creation of a "legitimate competitor" can be a fair use when the "substantial potential for damage to the marketability of" the original work weighs *against* fair use. *Harper & Row*, 471 U.S. at 569; *see also id.* at 562 (no fair use purpose where the infringing work had "the intended purpose of supplanting the copyright holder's commercially valuable right"). If copycats can run roughshod over copyright protection for an original work by claiming "competitive fair use," there will be little incentive for developers to innovate, for their work will quickly get stolen. While innovation is important, it cannot come at the cost of an author's right to its "original expression." *See Feist*, 499 U.S. at 349.

CONCLUSION

The judgment of the court of appeals should be affirmed.

Respectfully submitted.

I. NEEL CHATTERJEE	WILLIAM M. JAY
GOODWIN PROCTER LLP	*Counsel of Record*
601 Marshall Street	ANDREW KIM
Redwood City, CA 94063	GOODWIN PROCTER LLP
	1900 N Street, N.W.
	Washington, DC 20036
	wjay@goodwinlaw.com
	(202) 346-4000

February 19, 2020