No. 18-956

IN THE
Supreme Court of the United States

GOOGLE LLC,
Petitioner,

v.

ORACLE AMERICA, INC.,
Respondent.

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

BRIEF OF THE MATHWORKS, INC. AS *AMICUS CURIAE* IN SUPPORT OF RESPONDENT

THOMAS M. SPERA
JONATHAN T. FOOT
The MathWorks, Inc.
3 Apple Hill Drive
Natick, MA 01760
(508) 647-7742

DAVID C. FREDERICK
Counsel of Record
KELLOGG, HANSEN, TODD, FIGEL & FREDERICK, P.L.L.C.
1615 M Street, N.W.
Suite 400
Washington, D.C. 20036
(202) 326-7900
(dfrederick@kellogghansen.com)

February 19, 2020

TABLE OF CONTENTS

TABLE OF AUTHORITIES

INTEREST OF *AMICUS CURIAE*[1]

Amicus The MathWorks, Inc. ("MathWorks") is a mid-sized software company founded in 1984, with more than 5,000 employees and $1 billion in annual revenues. MathWorks' flagship product is MATLAB®, a sophisticated computer program used by engineers and scientists worldwide to perform numeric calculations and visualizations, to solve complex research problems, and to design, create, and test new products. MATLAB is popular in a number of technical domains such as signal and image processing, communications, control design, test-and-measurement, financial modeling and analysis, and computational biology. MATLAB also is relied on throughout the aerospace, defense, automotive, communications, electronics, and industrial automation industries, among others, and is a staple of academic research and teaching at major universities worldwide.

MATLAB allows its users to develop better and more reliable products and to facilitate new discoveries. Cars, airplanes, smartphones, and even the flight control system for the F-35 Joint Strike fighter – to name just a few products – were designed in part using MATLAB and other MathWorks software products. For more than three decades, MathWorks has been improving MATLAB with each release.

---

[1] Pursuant to Supreme Court Rule 37.6, counsel for *amicus* represent that they authored this brief in its entirety and that none of the parties or their counsel, nor any other person or entity other than *amicus* or its counsel, made a monetary contribution intended to fund the preparation or submission of this brief. Pursuant to Rule 37.3(a), counsel for *amicus* also represent that all parties have consented to the filing of this brief.

Almost all of MathWorks' revenue comes from license fees customers pay to use its software products. This common license model is very different from how Google makes (and many of its *amici* make) money: deriving revenue through advertising and personal, behavioral, and other data collected from users.

This case can substantially affect the incentives and economic ability of firms such as MathWorks to develop sophisticated software products like MATLAB. Copyright protection for its products allows MathWorks to fund research and development costs and continue to build new and sophisticated software. If it is legal to simply copy and reimplement popular software products, in the way Google copied verbatim the Java declaring code, innovation will suffer. Software developers will have less incentive to create new works, knowing their work can simply be copied and used to compete against them. That worry is real: MathWorks has relied upon its copyright to protect against this kind of copying multiple times and has engaged in litigation (and prevailed). To promote the Copyright Clause interest in "Progress of Science and useful Arts" through the progress of software development, the Court should hold that declaring code, like other computer programming code, is copyrightable and that Google's copying of Java SE was not fair use.

STATEMENT

A. The Development Of MATLAB

In writing a computer program, the author faces myriad choices in expressing to the program user the various ideas, processes, and functionality the computer program can perform. Programs such as Java SE and MATLAB are large, complex, and sophisticated works. MATLAB contains more than 2,000 built-in functions for its users to do technical computing. Similar to Java, MATLAB enables users to combine MATLAB expressions in their own preferred order or sequence to solve problems by building on the functionality provided in MATLAB and creating their own functions or programs within MATLAB.

Authors of software programs such as MATLAB and Java SE must combine business and technical judgment to make the program understandable to the user in an appealing and memorable way so the user can reach the desired results efficiently. As Google's own witness admitted, there can be "creativity and artistry even in a single method declaration." Pet. App. 154a. MathWorks agrees. In MathWorks' experience, successfully designing MATLAB function signatures (analogous to a Java method's declaring code) requires creativity and business and technical judgments.

When designing function signatures (or declaring code), it is possible to choose code expressions that better "speak to" the program's user. Different audiences will have varied backgrounds, experience, and knowledge, with varied responses to different expressions. When done well, MathWorks' choice of function signatures helps its users learn products like MATLAB more quickly and helps them better remember, understand, and use the functionalities provided.

Once MathWorks decides to provide a certain functionality, its engineers consider a range of factors in choosing how to express MATLAB's functionality, including, among others:

- Is the name informative?
- Is the name easy to learn and remember?
- Does the name fit well with existing names and functionality?
- Is the name "catchy"? Memorable? Intuitive or counterintuitive?
- Should underscores be used in the name?
- What arguments should be included and in what order?
- Should there be default values if the user leaves certain inputs blank?

With such factors in mind, MathWorks has expanded MATLAB over the past 35 years from fewer than 100 functions to more than 2,000 functions. Two examples – *tic/toc* and *linspace* – illustrate the creative choices inherent in their authorship.

tic toc. MathWorks developers wanted to provide MATLAB users the capability to measure the time a computer uses to execute any particular segment of MATLAB code. This function – measuring how long it takes the computer to execute certain code – is useful to engineers conducting various analyses in MATLAB. The relevant functionality is "determining how much time has elapsed," *see* https://www.mathworks.com/help/matlab/ref/toc.html, and that functionality is an idea or a concept and is not protected by copyright. Anyone can write code to perform that function. But how does the program author express or represent that idea and that functionality? There are many

possibilities, including referencing "elapsed time," "duration," or "starting and stopping."

With MATLAB, MathWorks decided to express and represent this functionality with the words *tic* and *toc*: by including *tic* at the beginning of the code the users want to time and *toc* at the end. *See* https://www.mathworks.com/help/matlab/ref/tic.html. MathWorks developers believed this *tic*/*toc* expression to be an artful, elegant, clever, and unusual way to represent that functionality. It is memorable and easy to use, reflecting the creativity involved in authoring it.

linspace(x1, x2, n). In another example, engineers or scientists often need to generate an evenly spaced range of numbers – for example, 2 6 10 14. The range can be hundreds or thousands of values over large or small ranges. To provide this functionality, MathWorks created the MATLAB function signature *linspace(x1, x2, n)*, which will generate a range of *n* values equally spaced between a beginning value *x1* and an ending value *x2*. So, for instance, when a user types a = *linspace*(0,1,5), MATLAB will generate five points evenly spaced between 0 and 1: *i.e.*, 0 0.25 0.5 0.75 1.

Here, the idea or functionality is to "generate a range of equally spaced numbers." Having a computer perform that task is not protected by copyright. Anyone is free to have a computer program that produces such a range of numbers, but copyright law does not permit others to copy MathWorks' original expression of that idea, here *linspace*, and MATLAB's instruction to the computer to perform that underlying functionality through separately written implementing code.

See CONTU Report[2] 22 ("[A]nyone is free to make a computer carry out any unpatented process, but not to misappropriate another's writing to do so."). Other ways to express that idea of "a range of equally spaced numbers" could include, for example, *range*, *eqspace*, *evenvector*, *equal_spaced_numbers*, and more.

linspace also reflects another aspect of creative expression. It is part of a small collection of original function signatures that are interrelated, such that, once a user understands the meaning of one, it is fairly easy to understand and remember the meaning of the others. The "lin" in *linspace* refers to "linearly" generating values. MATLAB also has functions called *logspace*, to generate a list of logarithmically spaced values, and *freqspace*, to generate a frequency range for equally spaced frequency responses. This sort of interrelated expression across an entire program is part of the creativity and originality involved in creating such complex computer programs as MATLAB or Java. *See also* Oracle Br. 7-10.

Importantly, there is a distinction between the function signature (*tic*/*toc*/*linspace*) code and the implementing code written to perform that function (calculating time for a computer to execute certain code; generating a range of numbers). These function signatures do not perform the function, but are instructions, similar to Java declaring code, that work with MATLAB's implementing code and the underlying computer operating system and computer hardware chips to instruct the computer to produce the intended result. To go with these function signatures, MathWorks also had to write the underlying implementing code, which are specific instructions for

---

[2] National Comm'n on New Technological Uses of Copyrighted Works, *Final Report* (July 31, 1978) ("CONTU Report").

how the computer should perform the intended functionality. And because a different function signature instruction, such as *start* and *stop*, or *range*, could be used to obtain the same result, this function signature expression is separate from the underlying ideas of time to run code and the range of numbers generated. In short, for any given functionality, a program author can choose a variety of different expressions that describe or reference the intended purpose.

MathWorks program authors had to make creative choices about the best way to express the underlying idea and functionality across the more than 2,000 function signatures in MATLAB today. MathWorks believes that its extensive collection of all these function signatures is a major asset of MATLAB. They have contributed to an ease of MATLAB's adoption in industry and academia and to its popularity, and is one basis on which MATLAB competes with other programs that have similar functionality.

MathWorks has faced very similar plagiarism to the type asserted against Google in this case by a foreign competitor that copied nearly 600 of MATLAB's most popular function signatures. By copying rather than authoring its own expressions, the competitor was able to produce and sell its competing product in a fraction of the time it took MathWorks and at a deep discount to MATLAB's price. MathWorks was able to bring and win a copyright suit to stop the copying. Google's proffered approach would thwart software authors that use a different but common business model to monetize their innovation – licensing fees – and would entrench its own dominant advertising model by exploiting the innovation of companies like Oracle.

B. The Software At Issue In This Case

This case arose because Google, after failing to reach agreement with Oracle for permission, copied verbatim 11,330 lines of Java code, comprising 37 Java platform packages. Pet. App. 7a.[3]

Google independently could have developed a mobile platform without copying Java SE code, just as Microsoft and Apple had done for their mobile phones. *Id.* at 149a n.5. Indeed, Google considered a number of alternative programming languages for the Android platform, including Objective-C, which was used for Apple's iOS operating system, but it rejected those alternatives as unsatisfactory and decided to use Java. JA478.

After acquiring Android, Inc., Google negotiated for months with Sun Microsystems about the possibility of taking a license to adapt Java SE for mobile devices. Those negotiations broke down because Google refused to maintain Java's "write once/run everywhere" philosophy, which was a central part of Java's appeal. Pet. App. 127a-128a. *See also* Oracle Br. 13-14.

Google claims that the merger doctrine grants it the right to copy this code because the established declarations are the "only . . . way to perform their function of responding to the calls already known to Java developers." Google Br. 19; *see also id.* at 20 ("Only one precisely written set of declarations will perform the function *of responding to the corresponding calls known to the developer*.") (emphasis added), 21 ("function of responding properly to the developers' calls"), 21-22 ("method will respond to a developer's

---

[3] Oracle places the lines of copied code at 11,330, after removing 170 lines arguably needed to make meaningful use of the Java language. Oracle Br. 14 n.2.

call *java.lang.Math.max(x, y)*"). Copying those declarations was important because Google wanted to attract Java users to create apps for Android devices.

Google makes much of the fact that experienced Java users would have familiarity with the method declarations that it ultimately copied. *See id.* at 20, 23, 27. That is true and reflects the success of the Java authors in creating understandable and memorable declaring code. But a programmer's skill set is far broader than familiarity with declaring code. In MathWorks' experience, analyzing problems, designing solutions, and debugging (finding errors) are all more fundamental programming skills than knowing Java syntax and method names.

Moreover, Google ignores that new programming platforms are regularly created and that users continue to learn them, and that users in fact had to learn all the new, non-infringing declarations in Android. For example, Dart and Swift are newer programming platforms released by Google and Apple in 2011 and 2014, respectively, and the users of those platforms also had to learn pertinent new, non-infringing declarations. Similarly, if Google had also written replacement declarations for the Java code it copied, developers readily could have learned those new declarations (assuming they were as memorable as Java's).

SUMMARY OF ARGUMENT

I. Declaring code is copyrightable source code under Section 102 of the Copyright Act because it is the author's work. Permitting competitors to copy or plagiarize source code represents the kind of impingement to creativity and technical knowledge that copyright law is intended to guard against. The Copyright Act does not support Google's broad contention that all declaring code, even when original, is unprotected either as a method of operation or under the merger doctrine.

A. Section 101 of the Copyright Act protects computer program code. The Java SE declaring code and implementing code together constitute human readable source code that 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. Declaring code can reflect "creativity and artistry" and requires judgment in how best to express the underlying functionality the declaring code can initiate via the implementing code. Contrary to Google's contention, declaring code is not an unprotected "method of operation" under Section 102(b). Declaring code can be written in different ways and still work with implementing code to have the computer produce the same output. As such, the declaring code does not extend to the method of operation for a computer's electromechanical functioning of the machine, which is not copyright protected. The legislative history supports that construction.

B. Google's reliance on the merger doctrine is misplaced. That doctrine posits that, where there is essentially only one way to express a given idea, the expression merges with the idea and receives no copyright protection. That principle does not apply here

because Sun Microsystems could have written the 11,330 lines of declaring code at issue many different ways, and Google could have independently written its own code to achieve the same functionality. Android did not need to copy Java but chose to do so because it wanted to take advantage of Java's popularity with users.

C. Removing all declaring code from copyright protection, as Google proposes, would have significant economic consequences. The declaring or signature codes are the most visible part of many programs for users. Protecting that code against copying is important because it ensures that innovation will be rewarded.

D. Upholding the copyrightability of declaring code will not harm interoperability or innovation, as Google erroneously asserts. Interoperability occurs when two programs or a program and a piece of hardware work together or communicate and share data. Android does not interoperate with Java SE; as such, applying copyright to the Java declaring code will not affect interoperability. Moreover, there are independent market interests and competitive pressures for programs to interoperate, and the copyright holder should be the one to decide when and how to do so. Holding declaring code eligible for copyright protection will not change that or the wide-ranging interoperability that exists today.

II. Copying declaring code to develop a competitive commercial product is not protected under the fair use doctrine.

A. Google did not copy Java code for criticism, commentary, or any of the other uses in Section 107. Rather, Google copied Java for monetary gain. Google

undertook no transformative change with new expression; rather, it copied thousands of lines of Java code verbatim and did so to reference exactly the same functions as Java and to take advantage of the success of the code it copied.

B. Google copied the declaring code of 37 packages in their entirety. No reasonable jury could conclude that Google's copying was insubstantial. First, this factor encompasses the quality and importance of the materials used. A substantial amount of creativity is necessary to make declaring code memorable and understandable. Second, the sheer volume of Google's copying far exceeds what is reasonable fair use.

C. Google's harm to Oracle in the marketplace is similar to harms by other copiers that have harmed other software developers like MathWorks. Considering the copier's effects on the marketplace is important. Ruling for Google in this case would produce less innovation and less competition by encouraging knockoff products promising familiar programming expressions.

ARGUMENT

I. DECLARING CODE EMBODIES CREATIVE DECISIONMAKING AND MERITS COPYRIGHT PROTECTION

The entire source code (consisting of both the declaring code and the implementing code) is copyrightable under Section 102 of the Copyright Act because all of it comprises the author's work. Crafting user-friendly declaring code – and organizing it logically – requires creativity, technical knowledge, and business judgment. *See supra* pp. 3-6 (discussing *tic/toc* and *linspace*). Permitting competitors simply to steal the source code fruits of that labor would seriously harm the future development of software products.

Google and its *amici* ask this Court to hold, as a matter of law, that no declaring code is ever copyrightable, even when original. *See* Google Br. 22 ("The merger doctrine applies to computer software interfaces designed to invoke the functions of a program."). Google argues broadly (at 19) that declaring code, here in the form of the copied Java declaring code, is unprotected either (1) as a method of operation under Section 102(b), or (2) under the merger doctrine. That broad position finds no support in the Copyright Act. Nor is that position supported as a matter of computer programming; Google is incorrect that declaring code is "rote" and "de minimis," with all of the author's creativity residing in the implementing code. *See id.* at 14, 25, 44-45.

A. The Copyright Act Protects All The Source Code (Both Declaring Code And Implementing Code), And The Declaring Code Is Not A Method Of Operation

Google seeks to remove all copyright protection for certain types of source code but not others – namely, declaring code, but not implementing code. For several reasons, Google's artificial distinction between declaring code and implementing code is fundamentally flawed, and declaring code is not an unprotected method of operation, nor a process or idea.

Both declaring code and implementing code are human readable source code, well recognized as protected by copyright. *See* 17 U.S.C. §§ 101, 109(b)(1)(A), 117, 506(a)(3)(A). And both declaring code and implementing code fall within the Copyright Act's definition of a protected computer program: "a set of statements or instructions to be used directly or indirectly in a computer in order to bring about a certain result." *Id.* § 101. They reside together as

one whole. A declaration without any implementing code cannot instruct a computer to perform any underlying function. Conversely, implementing code without a declaration provides no way for the user to instruct the computer to perform the underlying functionality. The Java SE authors created the entire Java method source code together, both declaring code and implementing code, to allow others to use the preprogrammed functionality.

It is undisputed that the Java SE declaring code is original, the result of a creative and expressive work by its authors, and is properly copyrightable under the rule of *Feist Publications, Inc. v. Rural Telephone Service Co.*, 499 U.S. 340, 345, 358 (1991). *See* 17 U.S.C. § 102(a). Instead, Google asserts (at 19), though with almost no argument, that any such copyright protection for the declaring code is removed by Section 102(b). That assertion is inconsistent with Section 102(b), which states: "In no case does copyright protection for an original work of authorship extend to any idea, procedure, process, system, method of operation, concept, principle, or discovery, regardless of the form in which it is described, explained, illustrated, or embodied in such work." 17 U.S.C. § 102(b).

In MathWorks' view, the MATLAB function signatures discussed *supra* pp. 4-7 (and the comparable Java declaring code) are ***not*** a "method of operation." The computer operations of "measuring the time needed to execute code," and generating a set of "equally spaced range of numbers," are separate from the written expressions *tic/toc* and *linspace,* which their authors created to describe or represent that functionality in MATLAB. MathWorks could have selected other names and syntax and obtained the same result, just as Google could have created its own

declaring code for what it copied from Java SE – the underlying computer operations would be unchanged. Only the words and symbols chosen to express those operations would change.

The "method of operation" for these functions is what executes in the computer – actual calculations and computing work done to produce the intended result. As the CONTU Report recognized (at 20) in the context of copyright protecting published game rules but not the ability of others to actually play the game, "one is always free to make a machine perform any conceivable process (in the absence of a patent), but one is not free to take another's program." *See also id.* ("Copyright . . . protects the program so long as it remains fixed in a tangible medium or expression but *does not protect the electromechanical functioning of a machine*.") (emphasis added); *id.* at 22 ("The movement of electrons through the wires and components of a computer is precisely that process over which copyright has no control. Thus, copyright leads to the result that anyone is free to make a computer carry out any unpatented process, but not to misappropriate another's writing to do so."). Section 102(b) does not mean, as Google claims, that one part of the source code (declaring code) loses copyright protection, while implementing code is protected.

The pertinent House report also cuts against Google's reading of Section 102(b):

> Some concern has been expressed lest copyright in computer programs should extend protection to the methodology or processes adopted by the programmer, rather than merely to the "writing" expressing his ideas. Section 102(b) is intended, among other things, to make clear that the expres-

> sion adopted by the programmer is the copyrightable element in a computer program, and that the actual processes or methods embodied in the program are not within the scope of the copyright law.

H.R. Rep. No. 94-1476, at 57 (1976); S. Rep. No. 94-473, at 54 (1975). Thus, the actual processes or methods performed on the computer hardware to produce the result is the unprotected functionality (or idea) (e.g., producing a range of equally spaced numbers), while the programmer's writing is the protected source code (e.g., MATLAB's *linspace* function signature).

B. The Merger Doctrine Does Not Eliminate Copyright Protection For Declaring Code

Google principally relies upon the merger doctrine, which holds that, where there is essentially only one way to express a given idea, the expression merges with the idea and receives no copyright protection. *See*, *e.g.*, *Computer Assocs. Int'l, Inc. v. Altai, Inc.*, 982 F.2d 693, 707-08 (2d Cir. 1992). Thus, in *Baker v. Selden*, 101 U.S. 99 (1880), because Selden's forms (or similar forms) were "necessary incidents" to his accounting system, they received no copyright protection, even though the body of the book describing the system was protected. *Id.* at 103.

"In the computer context, [the merger doctrine] means that when specific instructions, even though previously copyrighted, are the only and essential means of accomplishing a given task, their later use by another will not amount to an infringement." CONTU Report 20. But that principle has no application here because Sun Microsystems could have written the 11,330 lines of declaring code at issue in many different ways, not to mention the "unlimited options"

for how to arrange all of the methods into classes and packages. Pet. App. 150a & n.6. Sun's creative choices were not the "only and essential means" of accomplishing the functionality the code supported.

Google could have written alternative declaring code, along with its new implementing code, to provide the same underlying functionality as the copied Java declaring code. *Id.* For instance, instead of copying java.lang.Math.max(), Google could have written an Android method called *maximum*() or *larger*() to perform the same function, and could have placed it in any Android package and class, reflecting its own organization. *See*, *e.g.*, *id.* at 215a ("the Android method and class names could have been different from the names of their counterparts in Java and still have worked"). This is similar to MathWorks deciding to use *linspace* to represent the idea of creating a range of numbers, but someone else could have chosen a different way to express or initiate that underlying functionality. And the opportunity to do original work is only greater across hundreds and thousands of lines of code, far beyond this one simple example.

Because Google cannot dispute that alternative code could have fully substituted for the copied Java code, it cannot argue credibly that no other computer code would achieve Java's underlying functionality. Instead, its merger argument is that, once Sun/Oracle created 'java.lang.Math.max,' for example, the only way to "perform the function of responding to the corresponding call[] known to the developer" is to use the same declaring code as Java did. Google Br. 20. This is circular reasoning: The call is just a reflection or derivative of the declaring code, *after* the declaring code is created. Thus, Android copied Java not because no other expression was possible – there was – but

only because Java had achieved significant popularity with users (meaning, *if Google was going to take advantage of Java's success,* no other expression was possible).

Google's argument turns copyright on its head and would make successful, expressive code that is popular less protected because of its success. Google's argument would mean that once a program becomes popular it loses its copyright protection because customers will want to be able to use the same declaring code expressions in a different program. That would never fly for any other type of copyrighted work and simply cannot be the law. *Cf. Harper & Row Publishers, Inc. v. Nation Enters.*, 471 U.S. 539, 559 (1985) ("It is fundamentally at odds with the scheme of copyright to accord lesser rights in those works that are of greatest importance to the public.").

Moreover, as the Federal Circuit persuasively explained, Google's argument fails because the merger analysis considers "the options that were available to Sun/Oracle at the time it created the API packages," not the options available to Google years later, when it decided to create a new platform for smartphones. Pet. App. 151a. Whether idea and expression merge is an inherent quality of the original work and does not depend on analyzing the subsequent actions of a plagiarist.

This point is clear from *Baker v. Selden*. Selden's forms were not eligible for copyright protection because "no one can use [his] system without using substantially the same ruled lines and headings which he has appended to his books in illustration of it." 101 U.S. at 101. That fact was true at the moment he published his book, regardless of any later popularity or copying. For Baker to use Selden's accounting system, he had

to use similar forms to Selden's, because no other expression of Selden's system was possible – the forms were necessary to his system.

By contrast, Google was not required by merger to copy Java declaring code; rather, Google chose to copy Java to speed up (and popularize) Android's adoption. Google could have written its own declaring code to represent the same underlying functionality without copying Java, just as Microsoft and Apple did. And Google should have written its own declaring code for all the Java declarations it copied, just as it wrote its own particular declaring code for all the declarations it did not copy.

C. Permitting Copying Of Declaring Code Allows Knockoff Software Products And Diminishes The Incentives To Create Original Software

Google's proposal to remove copyright protections from all declaring code would have profound economic effects on MathWorks and similar software companies. "By establishing a marketable right to the use of one's expression, copyright supplies the economic incentive to create and disseminate ideas." *Harper & Row*, 471 U.S. at 558. Where that protection is weakened, authors will naturally have diminished incentive to create new expression.

As a commercial matter, protection for MATLAB function signatures – that part of MATLAB closest in nature to declaring code – is crucial. Although implementing code is also important, the signatures are the most visible part of MATLAB to the users. After all, those are the instructions the user actually writes to use MATLAB. The whole point of having preprogrammed functions is to make it easier for the

user to write his own programs without needing to rewrite implementing code.

The value of protecting those MATLAB expressions was made clear 15 years ago when the Swedish company COMSOL developed and started marketing a software product called COMSOL Script. That program copied nearly 600 of MATLAB's most popular expressions, including the *linspace* and *tic/toc* function signatures. In less than a year, with a small number of developers, COMSOL was able to write implementing code for these 600 functions, even though it had taken MathWorks decades to create MATLAB.

COMSOL's action demonstrated the truism that copying an innovator's work is the easiest way to compete. COMSOL Script was marketed as a replacement for MATLAB at 50% of the price. Facing this serious and unfair threat to its business, MathWorks brought suit for copyright infringement. *See The MathWorks, Inc. v. COMSOL AB*, No. 6:06-CV-00335 (E.D. Tex.). After a jury verdict in its favor, MathWorks received monetary compensation and a permanent injunction prohibiting COMSOL from marketing its infringing software product.

Had COMSOL prevailed on its legal argument that MATLAB expressions are uncopyrightable, using arguments similar to those being made by Google, MathWorks would have faced ongoing competition from a copy of its own flagship computer program. The copy would have looked and worked just like MATLAB and offered the expressions that MathWorks had created and developed over decades.

The COMSOL episode illustrates clearly the dangers of Google's position. Any successful software product with published declaring code, such as Java SE, would be at risk of being copied by a domestic or foreign

competitor. By copying the declarations, the competitor could develop an inexpensive knockoff version and undersell the original author. That regime would significantly disincentivize research and development of new software products, thus impeding the progress of science and useful arts.

D. Affirming The Copyrightability Of Declaring Code Will Not Harm Interoperability Or Innovation

Google and its *amici* tell doomsday tales about the death of innovation and interoperability should the Java declaring code be held copyrightable. These concerns are misplaced. A decision confirming that the Java declaring code is copyrightable will not harm interoperability.

To be clear, interoperability is best understood as two programs or a program and a piece of hardware working together or communicating and sharing data. *See* 17 U.S.C. § 1201(f)(4) ("the ability of computer programs to exchange information, and of such programs mutually to use the information which has been exchanged"). MATLAB, for example, runs on (or interoperates with) the Microsoft Windows operating system, which runs on millions of computers made by numerous companies. The two programs – MATLAB and Windows – share information and data to work with the computer, but do so in a manner that respects the intellectual property of each part of the interoperation.

MATLAB also interoperates with other programs, not just operating systems like Windows. This interoperability with other programs and devices is a feature MATLAB promotes. Specific function signatures in MATLAB are created with the purpose of allowing other programs to communicate with MATLAB, and

vice versa. Java SE and many other programs have similar capabilities. For instance, someone using Java SE could call out to MATLAB to perform an analysis at which MATLAB is very good, then return the result to Java for further processing or analysis. The reverse is also true.

That is not what happened here with Android and Java SE. "Google specifically designed Android to be *incompatible* with the Java platform and not allow for interoperability with Java programs." Pet. App. 46a n.11. Instead, Android is a replacement for Java.

In that way, Google acts contrary to and ignores the market incentives that exist for copyright holders to make their declaring code available to others who wish to make devices and software interoperate. Indeed, many companies license relevant declaring code (or interfaces) or make it freely available to encourage people to interoperate with (not replace) their program.[4] Microsoft adopted that strategy for its operating system to encourage developers to write applications that could run on Windows. In addition, other interfaces (or APIs) are maintained and published by standards-setting bodies, such as the Internet Engineering Task Force, which are then available for companies to use and communicate across products and programs. As such, interoperability will not be impaired by a decision of this Court that the Java

---

[4] In some well-established circumstances, someone can reverse engineer a product to enable interoperation. *See Sega Enters. Ltd. v. Accolade, Inc.*, 977 F.2d 1510 (9th Cir. 1992). There, the court found it was fair use to engage in intermediate copying of certain program code to understand unprotected underlying functionality needed for interoperability. But, in that case, the alleged infringer did not copy code and use it as a replacement for the actual copied code, as Google did with Java SE.

SE declaring code can be protected by copyright to stop a competitor from copying.[5]

The point is that each author should be allowed to decide how to license its code, including declaring code. This freedom of choice helps set the market and is part of market competition. If an author chooses to license or not license interface code, the market for that program will be affected by the chosen approach. Google is seeking to remove that market discipline and be able to appropriate anyone's declaring code for itself, for any purpose.

Under Google's approach, no declaring code (whether original or not) could ever be protected by copyright. Thus, if a small development team comes up with an innovative new way to express its declaring code, once it starts to gain traction and appeal in the market, copyright would not impede a large company like Google from just taking the declaring code for its own, rewriting the implementing code, and distributing the package as its own, in direct competition with the original. As with MathWorks' experience with COMSOL, such a rule would create

---

[5] The suggestion by some *amici* supporting Google that "everyone" understood and accepted that interfaces could be freely copied, without restriction, is refuted by at least several facts: (i) Sun/Oracle had for years offered a specific license for the declaring code, which allowed the licensee to write its own implementing code, but required strict testing to maintain write once/run everywhere, and lots of companies took that license; (ii) Sun sued Microsoft when it violated that license by modifying the Java declaring code, and won a favorable settlement, demonstrating the declaring code was not free for the taking; (iii) a well-recognized Eleventh Circuit decision rejected the argument that interfaces are not copyrightable as a matter of law, which is what Google is arguing for, *Bateman v. Mnemonics, Inc.*, 79 F.3d 1532, 1547 (11th Cir. 1996); and (iv) that was not MathWorks' understanding.

a huge disincentive for businesses to develop new declaring code.

That negative impact will also adversely affect companies and open source communities that offer open source programs. Those programs often have a license restriction requiring the licensee to contribute back to the community any improvements and additions the licensee makes to the programs. That "give back" requirement can be a disincentive for a commercial user that wants to own and control its intellectual property. When that restriction is included, the copyright holder may also offer a version of the software for a fee, but omit the "give back" requirement, which is an incentive to license the version with the fee. Regardless, if not copyrightable, original declaring code of these open source programs would no longer enjoy the protections of their open source licenses. A company like Google, or anyone, would be permitted to copy the declaring code of such open source program, prepare its own implementing code, and then remove the "give back" requirement, and distribute the program on any terms it wants.

As such, Google's approach would undercut the chosen licensing model of the open source project or company and make it difficult for that company either to sustain its own business model to support the programs or to generate community interest in improving the open source programs.[6] That, in fact, is what Google did here, given that Sun/Oracle offered just such an open source licensing option with its OpenJDK license. Oracle Br. 12-13.

---

[6] *See*, *e.g.*, Frederic Lardinois, *AWS gives open source the middle finger*, TechCrunch (Jan. 9, 2019), https://techcrunch.com/2019/01/09/aws-gives-open-source-the-middle-finger/.

II. COPYING DECLARING CODE TO DEVELOP A COMPETING COMMERCIAL PRODUCT IS NOT FAIR USE

"The purpose of copyright is to create incentives for creative effort." *Sony Corp. of Am. v. Universal City Studios, Inc.*, 464 U.S. 417, 450 (1984). Permitting Google's copying of Java SE as fair use – when such copying served a non-transformative commercial goal and seriously harmed Oracle in the marketplace – would undermine that purpose and ultimately hurt the public.

A. The Indisputably Commercial Nature Of Google's Copying Weighs Against A Finding Of Fair Use

Google did not copy Java code for "criticism, comment, news reporting, teaching . . . , scholarship, or research." 17 U.S.C. § 107. Instead, it copied Java to create a competing product as part of a concededly "commercial endeavor." Google Br. 43. Contrary to Google's arguments, that copying was anything but transformative; rather, Google intended to provide familiar, unchanged declaring code representing the same underlying functionality in Android as in Java. No reasonable jury could conclude otherwise. This first statutory factor plainly weighs against Google's fair use defense.

"The crux of the profit/nonprofit distinction is not whether the *sole motive* of the use is monetary gain but whether the user stands to profit from exploitation of the copyrighted material without paying the customary price." *Harper & Row*, 471 U.S. at 562 (emphasis added). Here, no reasonable jury could conclude that Google did not intend to profit from

developing Android by copying part of the Java platform.[7]

Nor was Google's copying transformative. Instead, Android copied verbatim what was in Java and did *not* "alter[] the first [work] with new expression, meaning, or message." *Campbell v. Acuff-Rose Music, Inc.*, 510 U.S. 569, 579 (1994). Quite the opposite, it copied Java so that the Android declaring code would have the *same* expression, *same* meaning, and *same* message in Android as in Java. Indeed, Google was counting on the fact that, e.g., java.lang.Math.max in the Java platform means "here is a method to compute the larger of two numbers," and that it means *the exact same thing* in Android, so as to make it easier to entice Java users to Android. That was true for all of the 11,330 lines of declaring code Google copied, the entirety of 37 packages of Java.

Google's argument that Android's smartphone setting makes its copying transformative lacks merit. Smartphones are simply smaller computers, and it is not transformative to copy the declaring code to a different size or type of computer. Developers migrate software across platforms all the time. For 35 years, MathWorks has migrated MATLAB to different hardware devices as the hardware technology and market demand change, including to much smaller and less

---

[7] To the extent it has been suggested that, because Android is distributed with an open source license and without a license fee, it could be found to be less than fully commercial, *see* Google Br. 12, 44, that is unpersuasive. The absence of a licensing fee is irrelevant to the fair use analysis, which inquires of the "commercial nature" of the use, in contrast to nonprofit educational use. *See*, *e.g.*, Pet. App. 25a-26a. Open source software can be and is commercial, and it is used by businesses around the world. Indeed, MathWorks sees open source software in various competitive sales situations.

powerful devices. A key to MathWorks' success is that, as the author of MATLAB, it has been able to decide which devices to move to and when and how to recover the revenue in those potential markets, including how to modify the program for those different devices. Oracle, as the copyright holder of Java SE, should have the choice of when and how to move its software to other devices and what modifications to make in the process. *See* 17 U.S.C. § 106.

In short, Google's copying provides no new expression and simply "avoid[s] the drudgery in working up something fresh." *Campbell*, 510 U.S. at 580. It chose to plagiarize Java, whereas both Microsoft and Apple independently created successful mobile platforms without copying others' work.[8]

B. Google Copied A Substantial Portion Of The Java Platform

Google copied the declaring code of 37 packages in their entirety, far more than the three packages arguably necessary to make meaningful use of the Java language. For two main reasons, no reasonable jury could conclude that Google did not copy a substantial amount of Java.

First, this factor requires consideration "not only about the quantity of the materials used, but about

---

[8] As to the second fair use factor, MathWorks believes the creative nature of the Java declaring code is undervalued. While it is true that a computer program is functional, Congress knew that when it decided to expressly provide copyright protection. *See* 17 U.S.C. § 101 ("computer program" defined). As such, the nature of the work should be viewed in that context and, unlike code copied in *Lexmark International, Inc. v. Static Control Components, Inc.*, 387 F.3d 522, 540-41 (6th Cir. 2004), the Java declaring code is quite complex, intricate, and sophisticated. Thus, at most, this factor should be neutral on fair use.

their quality and importance, too." *Campbell*, 510 U.S. at 587. Declaring code provides the structure of the platform and is among the most important code in Java SE, as it is the code the user actually types to use the program. As discussed above and in the Oracle brief, a substantial amount of creative investment goes into making sure the declarations are elegant, understandable, and memorable.

Second, with respect to quantity, Google copied a massive amount of code: 11,330 lines, 600 pages worth. Oracle Br. 5, 14. That far exceeds, for example, the small amount of code at issue in the *Lexmark* ink cartridge case. *See Lexmark*, 387 F.3d at 529-30.

Google's copying is an even more extreme violation of copyright than when a magazine copied 300 words out of President Ford's unpublished memoir. This Court noted that, "[i]n absolute terms, the words actually quoted were an insubstantial portion of" the memoir, but, qualitatively, the magazine had taken "essentially the heart of the book" – precisely the passages that "embodied Ford's distinctive expression." *Harper & Row*, 471 U.S. at 564-65.

As the Copyright Office emphasized, "it is more important to focus on whether the [copied] use is principally for the purpose of exploiting the creativity of the original author of the code, or for some purpose 'unrelated to copyright protection.'" U.S. Copyright Office, *Software-Enabled Consumer Products* 57 (Dec. 2016) (footnote and citation omitted), https://www.copyright.gov/policy/software/software-full-report.pdf.[9]

---

[9] Google selectively quotes this report to argue that the Copyright Office has approved verbatim copying of code where a reuse "'is simply to "permit . . . functionality" of a new product.'" Google Br. 41 (ellipses in original). But that ignores the second half of the sentence, which says literal copying is *not* favored if

Unlike in *Lexmark*, where the competing ink cartridges were reproducing the code snippet to allow the functional interaction between the cartridge and the printer, Android verbatim copied thousands of lines of source code very much to exploit the creativity of the original Java authors.

C. Google's Copying Significantly Harmed Oracle In The Marketplace Just As COMSOL Script Threatened To Devastate The Sale Of MATLAB

Market harm "is undoubtedly the single most important element of fair use," *Harper & Row*, 471 U.S. at 566, and it weighs overwhelmingly in Oracle's favor. No reasonable jury could conclude that Oracle had not suffered market harm. The evidence showed that many of its customers switched to Android, and even those who stayed demanded discounts by pointing to the availability of Android. Pet. App. 7a.

Moreover, uses like Google's will harm software companies in the aggregate. If permitted, this sort of copying will encourage knockoff products promising familiar programming expressions. As noted previously, COMSOL appropriated vast sections of MATLAB's function signatures (*i.e.*, declaring code), thereby saving the trouble of deciding on the actual expression to use, and then sold a reimplemented copy at a deeply discounted price. Had MathWorks not prevailed on its copyright claim, COMSOL would still be using MathWorks code and MathWorks would still be losing customers to the knockoff version.

This use is clearly disfavored under factor four, which "requires courts to consider not only the extent

---

the purpose is "to exploit the creativity of the original author," *Software-Enabled Consumer Products* 58, as Google did here.

of market harm caused by the particular actions of the alleged infringer, but also whether unrestricted and widespread conduct of the sort engaged in by the defendant . . . would result in a substantially adverse impact on the potential market." *Campbell*, 510 U.S. at 590 (citation omitted, ellipsis in original); *see also Los Angeles News Serv. v. Reuters Television Int'l, Ltd.*, 149 F.3d 987, 994 (9th Cir. 1998) (fair use inapplicable where defendant's conduct, "if permitted[,] would result in a substantially adverse impact on the potential market for the original works").

With respect to both Java and MATLAB, the copier intended to supply a "market replacement" for the original software product, "making it likely that cognizable market harm to the original will occur." *Campbell*, 510 U.S. at 591. If Google prevails, it and other large companies will be able to use their massive resources to simply appropriate for their own business new and creative expressions of smaller, promising start-up companies, or even mid-size companies like MathWorks. Larger companies could then reimplement and undersell the smaller company's original product. That will produce only more consolidation, less innovation, and less competition.

CONCLUSION

The court of appeals' judgment should be affirmed.

Respectfully submitted,

THOMAS M. SPERA
JONATHAN T. FOOT
The MathWorks, Inc.
3 Apple Hill Drive
Natick, MA 01760
(508) 647-7742

DAVID C. FREDERICK
Counsel of Record
KELLOGG, HANSEN, TODD,
FIGEL & FREDERICK,
P.L.L.C.
1615 M Street, N.W.
Suite 400
Washington, D.C. 20036
(202) 326-7900
(dfrederick@kellogghansen.com)

February 19, 2020