Google v. Oracle

United States Supreme Court

April 5, 2021

01 BREYER, J., delivered the opinion of the Court, in which ROBERTS, C. J., and SOTOMAYOR, KAGAN, GORSUCH, and KAVANAUGH, JJ., joined. THOMAS, J., filed a dissenting opinion, in which ALITO, J., joined. BARRETT, J., took no part in the consideration or decision of the case.

02 JUSTICE BREYER delivered the opinion of the Court.

03 Oracle America, Inc., is the current owner of a copyright in Java SE, a computer program that uses the popular Java computer programming language. Google, without permission, has copied a portion of that program, a portion that enables a programmer to call up prewritten software that, together with the computer's hardware, will carry out a large number of specific tasks. The lower courts have considered (1) whether Java SE's owner could copyright the portion that Google copied, and (2) if so, whether Google's copying nonetheless constituted a "fair use" of that material, thereby freeing Google from copyright liability. The Federal Circuit held in Oracle's favor (*i.e.*, that the portion is copyrightable and Google's copying did not constitute a "fair use"). In reviewing that decision, we assume, for argument's sake, that the material was copyrightable. But we hold that the copying here at issue nonetheless constituted a fair use. Hence, Google's copying did not violate the copyright law.

04 I

05 In 2005, Google acquired Android, Inc., a startup firm that hoped to become involved in smartphone software. Google sought, through Android, to develop a software platform for mobile devices like smartphones. A platform provides the necessary infrastructure for computer programmers to develop new programs and applications. One might think of a software platform as a kind of factory floor where computer programmers (analogous to autoworkers, designers, or manufacturers) might come, use sets of tools found there, and create new applications for use in, say, smartphones. …

06 Google envisioned an Android platform that was free and open, such that software developers could use the tools found there free of charge. Its idea was that more and more developers using its Android platform would develop ever more Android-based applications, all of which would make Google's Android-based smartphones more attractive to ultimate consumers. Consumers would then buy and use ever more of those phones. That vision required attracting a sizeable number of skilled programmers.

07 At that time, many software developers understood and wrote programs using

the Java programming language, a language invented by Sun Microsystems (Oracle's predecessor). About six million programmers had spent considerable time learning, and then using, the Java language. Many of those programmers used Sun's own popular Java SE platform to develop new programs primarily for use in desktop and laptop computers. That platform allowed developers using the Java language to write programs that were able to run on any desktop or laptop computer, regardless of the underlying hardware (*i.e.*, the programs were in large part "interoperable"). Indeed, one of Sun's slogans was "'write once, run anywhere.'"

08 Shortly after acquiring the Android firm, Google began talks with Sun about the possibility of licensing the entire Java platform for its new smartphone technology. But Google did not want to insist that all programs written on the Android platform be interoperable. As Android's founder explained, "[t]he whole idea about [an] open source [platform] is to have very, very few restrictions on what people can do with it," and Sun's interoperability policy would have undermined that free and open business model. Apparently, for reasons related to this disagreement, Google's negotiations with Sun broke down. Google then built its own platform.

09 The record indicates that roughly 100 Google engineers worked for more than three years to create Google's Android platform software. In doing so, Google tailored the Android platform to smartphone technology, which differs from desktop and laptop computers in important ways. A smartphone, for instance, may run on a more limited battery or take advantage of GPS technology. The Android platform offered programmers the ability to program for that environment. To build the platform, Google wrote millions of lines of new code. Because Google wanted millions of programmers, familiar with Java, to be able easily to work with its new Android platform, it also copied roughly 11,500 lines of code from the Java SE program. The copied lines of code are part of a tool called an Application Programming Interface, or API.

10 What is an API? The Federal Circuit described an API as a tool that "allow[s] programmers to use . . . prewritten code to build certain functions into their own programs, rather than write their own code to perform those functions from scratch." Through an API, a programmer can draw upon a vast library of prewritten code to carry out complex tasks. For lay persons, including judges, juries, and many others, some elaboration of this description may prove useful.

11 Consider in more detail just what an API does. A computer can perform thousands, perhaps millions, of different tasks that a programmer may wish to use. These tasks range from the most basic to the enormously complex. Ask the computer, for example, to tell you which of two numbers is the higher number or to sort one thousand numbers in ascending order, and it will instantly give you the right answer. An API divides and organizes the world of computing tasks in a particular way. Programmers can then use the API to select the particular task that they need for their programs. In Sun's API (which we refer

to as the Sun Java API), each individual task is known as a "method." The API groups somewhat similar methods into larger "classes," and groups somewhat similar classes into larger "packages." This method-class-package organizational structure is referred to as the Sun Java API's "structure, sequence, and organization," or SSO.

12 For each task, there is computer code, known as "implementing code," that in effect tells the computer how to execute the particular task you have asked it to perform (such as telling you, of two numbers, which is the higher). The implementing code (which Google independently wrote) is not at issue here. For a single task, the implementing code may be hundreds of lines long. It would be difficult, perhaps impossible, for a programmer to create complex software programs without drawing on prewritten task-implementing programs to execute discrete tasks.

13 But how do you as the programmer tell the computer which of the implementing code programs it should choose, *i.e.*, which task it should carry out? You do so by entering into your own program a command that corresponds to the specific task and calls it up. Those commands, known as "method calls," help you carry out the task by choosing those programs written in implementing code that will do the trick, *i.e.*, that will instruct the computer so that your program will find the higher of two numbers. If a particular computer might perform, say, a million different tasks, different method calls will tell the computer which of those tasks to choose. Those familiar with the Java language already know countless method calls that allow them to invoke countless tasks.

14 And how does the method call (which a programmer types) actually locate and invoke the particular implementing code that it needs to instruct the computer how to carry out a particular task? It does so through another type of code, which the parties have labeled "declaring code." Declaring code is part of the API. For each task, the specific command entered by the programmer matches up with specific declaring code inside the API. That declaring code provides both the name for each task and the location of each task within the API's overall organizational system (*i.e.,* the placement of a method within a particular class and the placement of a class within a particular package). In this sense, the declaring code and the method call form a link, allowing the programmer to draw upon the thousands of prewritten tasks, written in implementing code. Without that declaring code, the method calls entered by the programmer would not call up the implementing code.

15 The declaring code therefore performs at least two important functions in the Sun Java API. The first, more obvious, function is that the declaring code enables a set of shortcuts for programmers. By connecting complex implementing code with method calls, it allows a programmer to pick out from the API's task library a particular task without having to learn anything more than a simple command. For example, a programmer building a new application for personal banking may wish to use various tasks to, say, calculate a user's balance

or authenticate a password. To do so, she need only learn the method calls associated with those tasks. In this way, the declaring code's shortcut function is similar to a gas pedal in a car that tells the car to move faster or the QWERTY keyboard on a typewriter that calls up a certain letter when you press a particular key. As those analogies demonstrate, one can think of the declaring code as part of an *interface* between human beings and a machine.

16 The second, less obvious, function is to reflect the way in which Java's creators have divided the potential world of different tasks into an actual world, *i.e.*, precisely which set of potentially millions of different tasks we want to have our Java-based computer systems perform and how we want those tasks arranged and grouped. In this sense, the declaring code performs an organizational function. It determines the structure of the task library that Java's creators have decided to build. To understand this organizational system, think of the Dewey Decimal System that categorizes books into an accessible system or a travel guide that arranges a city's attractions into different categories. Language itself provides a rough analogy to the declaring code's organizational feature, for language itself divides into sets of concepts a world that in certain respects other languages might have divided differently. The developers of Java, for example, decided to place a method called "draw image" inside of a class called "graphics."

17 Consider a comprehensive, albeit farfetched, analogy that illustrates how the API is actually used by a programmer. Imagine that you can, via certain keystrokes, instruct a robot to move to a particular file cabinet, to open a certain drawer, and to pick out a specific recipe. With the proper recipe in hand, the robot then moves to your kitchen and gives it to a cook to prepare the dish. This example mirrors the API's task-related organizational system. Through your simple command, the robot locates the right recipe and hands it off to the cook. In the same way, typing in a method call prompts the API to locate the correct implementing code and hand it off to your computer. And importantly, to select the dish that you want for your meal, you do not need to know the recipe's contents, just as a programmer using an API does not need to learn the implementing code. In both situations, learning the simple command is enough.

18 Now let us consider the example that the District Court used to explain the precise technology here. *Id.,* at 980–981. A programmer wishes, as part of her program, to determine which of two integers is the larger. To do so in the Java language, she will first write **java.lang**. Those words (which we have put in bold type) refer to the "package" (or by analogy to the file cabinet). She will then write **Math**. That word refers to the "class" (or by analogy to the drawer). She will then write **max**. That word refers to the "method" (or by analogy to the recipe). She will then make two parentheses ( ). And, in between the parentheses she will put two integers, say 4 and 6, that she wishes to compare. The whole expression—the method call—will look like this: "**java.lang.Math.max(4, 6)**." The use of this expression will, by means of the API, call up a task-implementing program that will determine the higher number.

19 In writing this program, the programmer will use the very symbols we have placed in bold in the precise order we have placed them. But the symbols by themselves do nothing. She must also use software that connects the symbols to the equivalent of file cabinets, drawers, and files. The API is that software. It includes both the declaring code that links each part of the method call to the particular task-implementing program, and the implementing code that actually carries it out. …

20 Now we can return to the copying at issue in this case. Google did not copy the task-implementing programs, or implementing code, from the Sun Java API. It wrote its own task-implementing programs, such as those that would determine which of two integers is the greater or carry out any other desired (normally far more complex) task. This implementing code constitutes the vast majority of both the Sun Java API and the API that Google created for Android. For most of the packages in its new API, Google also wrote its own declaring code. For 37 packages, however, Google copied the declaring code from the Sun Java API. As just explained, that means that, for those 37 packages, Google necessarily copied both the names given to particular tasks and the grouping of those tasks into classes and packages.

21 In doing so, Google copied that portion of the Sun Java API that allowed programmers expert in the Java programming language to use the "task calling" system that they had already learned. As Google saw it, the 37 packages at issue included those tasks that were likely to prove most useful to programmers working on applications for mobile devices. In fact, "three of these packages were . . . fundamental to being able to use the Java language at all." By using the same declaring code for those packages, programmers using the Android platform can rely on the method calls that they are already familiar with to call up particular tasks (*e.g.*, determining which of two integers is the greater); but Google's own implementing programs carry out those tasks. Without that copying, programmers would need to learn an entirely new system to call up the same tasks.

22 We add that the Android platform has been successful. Within five years of its release in 2007, Android-based devices claimed a large share of the United States market. As of 2015, Android sales produced more than $42 billion in revenue.

23 In 2010 Oracle Corporation bought Sun. Soon thereafter Oracle brought this lawsuit in the United States District Court for the Northern District of California.

24 II

25 The case has a complex and lengthy history. At the outset Oracle complained that Google's use of the Sun Java API violated both copyright and patent laws. For its copyright claim, Oracle alleged that Google infringed its copyright by copying, for 37 packages, both the literal declaring code and the nonliteral

organizational structure (or SSO) of the API, *i.e.,* the grouping of certain methods into classes and certain classes into packages. For trial purposes the District Court organized three proceedings. The first would cover the copyright issues, the second would cover the patent issues, and the third would, if necessary, calculate damages. The court also determined that a judge should decide whether copyright law could protect an API and that the jury should decide whether Google's use of Oracle's API infringed its copyright and, if so, whether a fair use defense nonetheless applied.

26 After six weeks of hearing evidence, the jury rejected Oracle's patent claims (which have since dropped out of the case). It also found a limited copyright infringement. It deadlocked as to whether Google could successfully assert a fair use defense. The judge then decided that, regardless, the API's declaring code was not the kind of creation to which copyright law extended its protection. The court noted that Google had written its own implementing code, which constituted the vast majority of its API. It wrote that "anyone is free under the Copyright Act to write his or her own code to carry out exactly the same" tasks that the Sun Java API picks out or specifies. Google copied only the declaring code and organizational structure that was necessary for Java-trained programmers to activate familiar tasks (while, as we said, writing its own implementing code). Hence the copied material, in the judge's view, was a "system or method of operation," which copyright law specifically states cannot be copyrighted.

27 On appeal, the Federal Circuit reversed. That court held that both the API's declaring code and its organizational structure could be copyrighted. It pointed out that Google could have written its own declaring code just as it wrote its own implementing code. And because in principle Google might have created a whole new system of dividing and labeling tasks that could be called up by programmers, the declaring code (and the system) that made up the Sun Java API was copyrightable.

28 The Federal Circuit also rejected Oracle's plea that it decide whether Google had the right to use the Sun Java API because doing so was a "fair use," immune from copyright liability. The Circuit wrote that fair use "both permits and requires 'courts to avoid rigid application of the copyright statute when, on occasion, it would stifle the very creativity which that law is designed to foster.'" But, it added, this "is not a case in which the record contains sufficient factual findings upon which we could base a *de novo* assessment of Google's affirmative defense of fair use." And it remanded the case for another trial on that question. Google petitioned this Court for a writ of certiorari, seeking review of the Federal Circuit's copyrightability determination. We denied the petition.

29 On remand the District Court, sitting with a jury, heard evidence for a week. The court instructed the jury to answer one question: Has Google "shown by a preponderance of the evidence that its use in Android" of the declaring code and organizational structure contained in the 37 Sun JavaAPI packages that it copied

"constitutes a 'fair use' under the Copyright Act?" After three days of deliberation the jury answered the question in the affirmative. Google had shown fair use.

30 Oracle again appealed to the Federal Circuit. And the Circuit again reversed the District Court. The Federal Circuit assumed all factual questions in Google's favor. But, it said, the question whether those facts constitute a "fair use" is a question of law. Deciding that question of law, the court held that Google's use of the Sun Java API was not a fair use. It wrote that "[t]here is nothing fair about taking a copyrighted work verbatim and using it for the same purpose and function as the original in a competing platform." It remanded the case again, this time for a trial on damages.

31 Google then filed a petition for certiorari in this Court. It asked us to review the Federal Circuit's determinations as to both copyrightability and fair use. We granted its petition.

32 III

33 A

34 Copyright and patents, the Constitution says, are to "promote the Progress of Science and useful Arts, by securing for limited Times to Authors and Inventors the exclusive Right to their respective Writings and Discoveries." Art. I, §8, cl. 8. Copyright statutes and case law have made clear that copyright has practical objectives. It grants an author an exclusive right to produce his work (sometimes for a hundred years or more), not as a special reward, but in order to encourage the production of works that others might reproduce more cheaply. At the same time, copyright has negative features. Protection can raise prices to consumers. It can impose special costs, such as the cost of contacting owners to obtain reproduction permission. And the exclusive rights it awards can sometimes stand in the way of others exercising their own creative powers. See generally *Twentieth Century Music Corp.* v. *Aiken*, 422 U. S. 151, 156 (1975); *Mazer* v. *Stein*, 347 U. S. 201, 219 (1954).

35 Macaulay once said that the principle of copyright is a "tax on readers for the purpose of giving a bounty to writers." T. Macaulay, Speeches on Copyright 25 (E. Miller ed. 1913). Congress, weighing advantages and disadvantages, will determine the more specific nature of the tax, its boundaries and conditions, the existence of exceptions and exemptions, all by exercising its own constitutional power to write a copyright statute.

36 Four provisions of the current Copyright Act are of particular relevance in this case. First, a definitional provision sets forth three basic conditions for obtaining a copyright. There must be a "wor[k] of authorship," that work must be "original," and the work must be "fixed in any tangible medium of expression." 17 U. S. C. §102(a); see also *Feist Publications, Inc.* v. *Rural Telephone Service Co.*, 499

U. S. 340, 345 (1991) (explaining that copyright requires some original "creative spark" and therefore does not reach the facts that a particular expression describes).

37 Second, the statute lists certain kinds of works that copyright can protect. They include "literary," "musical," "dramatic," "motion pictur[e]," "architectural," and certain other works. §102(a). In 1980, Congress expanded the reach of the Copyright Act to include computer programs. And it defined "computer program" as "'a set of statements or instructions to be used directly or indirectly in a computer in order to bring about a certain result.'" §10, 94 Stat.3028 (codified at 17 U. S. C. §101).

38 Third, the statute sets forth limitations on the works that can be copyrighted, including works that the definitional provisions might otherwise include. It says, for example, that copyright protection cannot be extended to "any idea, procedure, process, system, method of operation, concept, principle, or discovery . . . ." §102(b). These limitations, along with the need to "fix" a work in a "tangible medium of expression," have often led courts to say, in shorthand form, that, unlike patents, which protect novel and useful ideas, copyrights protect "expression" but not the "ideas" that lie behind it. See *Sheldon* v. *Metro-Goldwyn Pictures Corp.*, 81 F. 2d 49, 54 (CA2 1936) (Hand, J.); B. Kaplan, An Unhurried View of Copyright 46–52 (1967).

39 Fourth, Congress, together with the courts, has imposed limitations upon the scope of copyright protection even in respect to works that are entitled to a copyright. For example, the Copyright Act limits an author's exclusive rights in performances and displays, §110, or to performances of sound recordings, §114. And directly relevant here, a copyright holder cannot prevent another person from making a "fair use" of copyrighted material. §107.

40 We have described the "fair use" doctrine, originating in the courts, as an "equitable rule of reason" that "permits courts to avoid rigid application of the copyright statute when, on occasion, it would stifle the very creativity which that law is designed to foster." *Stewart* v. *Abend*, 495 U. S. 207, 236 (1990). The statutory provision that embodies the doctrine indicates, rather than dictates, how courts should apply it. The provision says:

41 > "[T]he fair use of a copyrighted work, . . . for purposes such as criticism, comment, news reporting, teaching . . . scholarship, or research, is not an infringement of copyright. In determining whether the use made of a work in any particular case is a fair use the factors to be considered shall include—

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

43 > "(2) the nature of the copyrighted work;

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

45 "(4) the effect of the use upon the potential market for or value of the copyrighted work." §107.

46 In applying this provision, we, like other courts, have understood that the provision's list of factors is not exhaustive (note the words "include" and "including"), that the examples it sets forth do not exclude other examples (note the words "such as"), and that some factors may prove more important in some contexts than in others. See *Campbell* v. *Acuff-Rose Music, Inc.*, 510 U. S. 569, 577 (1994); *Harper & Row, Publishers, Inc.* v. *Nation Enterprises*, 471 U. S. 539, 560 (1985); see also Leval, Toward a Fair Use Standard, 103 Harv. L. Rev 1105, 1110 (1990) (Leval). In a word, we have understood the provision to set forth general principles, the application of which requires judicial balancing, depending upon relevant circumstances, including "significant changes in technology." *Sony Corp. of America* v. *Universal City Studios, Inc.*, 464 U. S. 417, 430 (1984); see also *Aiken*, 422 U. S., at 156 ("When technological change has rendered its literal terms ambiguous, the Copyright Act must be construed in light of its basic purpose").

47 B

48 Google's petition for certiorari poses two questions. The first asks whether Java's API is copyrightable. It asks us to examine two of the statutory provisions just mentioned, one that permits copyrighting computer programs and the other that forbids copyrighting, *e.g.*, "process[es]," "system[s]," and "method[s] of operation." Pet. for Cert. 12. Google believes that the API's declaring code and organization fall into these latter categories and are expressly excluded from copyright protection. The second question asks us to determine whether Google's use of the API was a "fair use." Google believes that it was.

49 A holding for Google on either question presented would dispense with Oracle's copyright claims. Given the rapidly changing technological, economic, and business-related circumstances, we believe we should not answer more than is necessary to resolve the parties' dispute. We shall assume, but purely for argument's sake, that the entire Sun Java API falls within the definition of that which can be copyrighted. We shall ask instead whether Google's use of part of that API was a "fair use." Unlike the Federal Circuit, we conclude that it was.

50 IV

51 The language of §107, the "fair use" provision, reflects its judge-made origins. It is similar to that used by Justice Story in *Folsom* v. *Marsh*, 9 F. Cas. 342, 348 (No. 4,901) (CCMass. 1841). That background, as well as modern courts' use of the doctrine, makes clear that the concept is flexible, that courts must apply it in light of the sometimes conflicting aims of copyright law, and that its application may well vary depending upon context. Thus, copyright's protection

others to compete runs counter to the statutory purpose of promoting creative expression"); *Lexmark Int'l*, 387 F. 3d, at 544 (noting that where a subsequent user copied a computer program to foster functionality, it was not exploiting the programs "commercial value *as a copyrighted work*" (emphasis in original)). After all, "copyright supplies the economic incentive to [both] create and disseminate ideas," *Harper & Row*, 471 U. S., at 558, and the reimplementation of a user interface allows creative new computer code to more easily enter the market.

103 The uncertain nature of Sun's ability to compete in Android's market place, the sources of its lost revenue, and the risk of creativity-related harms to the public, when taken together, convince that this fourth factor—market effects—also weighs in favor of fair use.

104 The fact that computer programs are primarily functional makes it difficult to apply traditional copyright concepts in that technological world. See *Lotus Development Corp.*, 49 F. 3d, at 820 (Boudin, J., concurring). In doing so here, we have not changed the nature of those concepts. We do not overturn or modify our earlier cases involving fair use—cases, for example, that involve "knockoff" products, journalistic writings, and parodies. Rather, we here recognize that application of a copyright doctrine such as fair use has long proved a cooperative effort of Legislatures and courts, and that Congress, in our view, intended that it so continue. As such, we have looked to the principles set forth in the fair use statute, §107, and set forth in our earlier cases, and applied them to this different kind of copyrighted work.

105 We reach the conclusion that in this case, where Google reimplemented a user interface, taking only what was needed to allow users to put their accrued talents to work in a new and transformative program, Google's copying of the Sun Java API was a fair use of that material as a matter of law. The Federal Circuit's contrary judgment is reversed, and the case is remanded for further proceedings in conformity with this opinion.

106 *It is so ordered.*

107 JUSTICE THOMAS, with whom JUSTICE ALITO joins, dissenting.

108 Oracle spent years developing a programming library that successfully attracted software developers, thus enhancing the value of Oracle's products. Google sought a license to use the library in Android, the operating system it was developing for mobile phones. But when the companies could not agree on terms, Google simply copied verbatim 11,500 lines of code from the library. As a result, it erased 97.5% of the value of Oracle's partnership with Amazon, made tens of billions of dollars, and established its position as the owner of the largest mobile operating system in the world. Despite this, the majority holds that this copying was fair use.

109 The Court reaches this unlikely result in large part because it bypasses the antecedent question clearly before us: Is the software code at issue here protected by the Copyright Act? The majority purports to assume, without deciding, that the code is protected. But its fair-use analysis is wholly inconsistent with the substantial protection Congress gave to computer code. By skipping over the copyrightability question, the majority disregards half the relevant statutory text and distorts its fair-use analysis. Properly considering that statutory text, Oracle's code at issue here is copyrightable, and Google's use of that copyrighted code was anything but fair.

110 I

111 In the 1990s, Oracle created a programming language called Java. Like many programming languages, Java allows developers to prewrite small subprograms called "methods." Methods form the building blocks of more complex programs. This process is not unlike what legislatures do with statutes. To save space and time, legislatures define terms and then use those definitions as a shorthand. For example, the legal definition for "refugee" is more than 300 words long. Rather than repeat all those words every time they are relevant, the U. S. Code encapsulates them all with a single term that it then inserts into each relevant section. Java methods work similarly. Once a method has been defined, a developer need only type a few characters (the method name and relevant inputs) to invoke everything contained in the subprogram. A programmer familiar with prewritten methods can string many of them together to quickly develop complicated programs without having to write from scratch all the basic subprograms.

112 To create Java methods, developers use two kinds of code. The first, "declaring code," names the method, defines what information it can process, and defines what kind of data it can output. It is like the defined term in a statute. The second, "implementing code," includes the step-by-step instructions that make those methods run.[1] It is like the detailed definition in a statute.

113 Oracle's declaring code was central to its business model. Oracle profited financially by encouraging developers to create programs written in Java and then charging manufacturers a fee to embed in their devices the Java software

[1] Consider what the relevant text of a simple method—designed to return the largest of three integers—might look like:

```
public static int MaxNum (int x, int y, int z) {
if (x >= y && x >= z) return x;
else if (y >= x && y >= z) return y;
else return z;
}
```

The first line is declaring code that defines the method, including what inputs (integers x, y, and z) it can process and what it can output (an integer). The remainder is implementing code that checks which of the inputs is largest and returns the result. Once this code is written, a programmer could invoke it by typing, for example, "MaxNum (4, 12, 9)."

platform needed to run those programs. To this end, Oracle created a work called Java 2 Platform, Standard Edition, which included a highly organized library containing about 30,000 methods. Oracle gave developers free access to these methods to encourage them to write programs for the Java platform. In return, developers were required to make their programs compatible with the Java platform on any device. Developers were encouraged to make improvements to the platform, but they were required to release beneficial modifications to the public. If a company wanted to customize the platform and keep those customizations secret for business purposes, it had to pay for a separate license.

114 By 2005, many companies were racing to develop operating systems for what would become modern smartphones. Oracle's strategy had successfully encouraged millions of programmers to learn Java. As a result, Java software platforms were in the vast majority of mobile phones. Google wanted to attract those programmers to Android by including in Android the declaring code with which they were now familiar. But the founder of Android, Andrew Rubin, understood that the declaring code was copyrighted, so Google sought a custom license from Oracle. At least four times between 2005 and 2006, the two companies attempted to negotiate a license, but they were unsuccessful, in part because of "trust issues."

115 When those negotiations broke down, Google simply decided to use Oracle's code anyway. Instead of creating its own declaring code—as Apple and Microsoft chose to do—Google copied verbatim 11,500 lines of Oracle's declaring code and arranged that code exactly as Oracle had done. It then advertised Android to device manufacturers as containing "Core Java Libraries." Oracle predictably responded by suing Google for copyright infringement. The Federal Circuit ruled that Oracle's declaring code is copyrightable and that Google's copying of it was not fair use.

116 II

117 The Court wrongly sidesteps the principal question that we were asked to answer: Is declaring code protected by copyright? I would hold that it is.

118 Computer code occupies a unique space in intellectual property. Copyright law generally protects works of authorship. Patent law generally protects inventions or discoveries. A library of code straddles these two categories. It is highly functional like an invention; yet as a writing, it is also a work of authorship. Faced with something that could fit in either space, Congress chose copyright, and it included declaring code in that protection.

119 The Copyright Act expressly protects computer code. It recognizes that a "computer program" is protected by copyright. See 17 U. S. C. §§109(b), 117, 506(a). And it defines "'computer program'" as "a set of statements or instructions to be used directly or indirectly in a computer in order to bring about a certain result." §101. That definition clearly covers declaring

code—sets of statements that indirectly perform computer functions by triggering prewritten implementing code.

120 Even without that express language, declaring code would satisfy the general test for copyrightability. "Copyright protection subsists . . . in original works of authorship fixed in any tangible medium of expression." §102(a). "Works of authorship include . . . literary works," which are "works . . . expressed in words, numbers, or other verbal or numerical symbols." §§101, 102(a). And a work is "original" if it is "independently created by the author" and "possesses at least some minimal degree of creativity." *Feist Publications Inc.,* v. *Rural Telephone Service Co.*, 499 U. S. 340, 345 (1991). The lines of declaring code in the Java platform readily satisfy this "extremely low" threshold. First, they are expressed in "words, numbers, or other verbal or numerical symbols" and are thus works of authorship. Second, as Google concedes, the lines of declaring code are original because Oracle could have created them any number of ways.

121 Google contends that declaring code is a "method of operation" and thus excluded from protection by §102(b). That subsection excludes from copyright protection "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." This provision codifies the "idea/expression dichotomy" that copyright protection covers only the "the author's expression" of an idea, not the idea itself. *Golan* v. *Holder*, 565 U. S. 302, 328 (2012). A property right in the idea itself "can only be secured, if it can be secured at all, by letters-patent." *Baker* v. *Selden*, 101 U. S. 99, 105 (1880). Thus, for example, a "method of book-keeping" is not protected by copyright, but the expression describing that accounting method is. So too, a person who writes a book inventing the idea of declaring code has a copyright protection in the expression in the book, but not in the idea of declaring code itself. Google acknowledges that implementing code is protected by the Copyright Act, but it contends that declaring code is much more functional and thus is a "method of operation" outside the scope of protection.

122 That argument fails. As the majority correctly recognizes, declaring code and implementing code are "inextricably bound" together. Declaring code defines the scope of a set of implementing code and gives a programmer a way to use it by shortcut. Because declaring code incorporates implementing code, it has no function on its own. Implementing code is similar. Absent declaring code, developers would have to write every program from scratch, making complex programs prohibitively time consuming to create. The functionality of both declaring code and implementing code will thus typically rise and fall together.

123 Google's argument also cannot account for Congress' decision to define protected computer code as "a set of statements or instructions to be used *directly or indirectly* in a computer in order to bring about a certain result." §101 (emphasis added). Hence, Congress rejected any categorical distinction between declaring and implementing code. Implementing code

orders a computer operation directly. Declaring code does so indirectly by incorporating implementing code. When faced with general language barring protection for "methods of operation" and specific language protecting declaring code, the "'specific governs the general.'"

124 This context makes clear that the phrase "method of operation" in §102(b) does not remove protection from declaring code simply because it is functional. That interpretation does not, however, render "method of operation" meaningless. It is "given more precise content by the neighboring words with which it is associated." Other terms in the same subsection such as "idea," "principle," and "concept" suggest that "method of operation" covers the functions and ideas implemented by computer code—such as math functions, accounting methods, or the idea of declaring code—not the specific expression Oracle created. Oracle cannot copyright the idea of using declaring code, but it can copyright the specific expression of that idea found in its library.

125 Google also contends that declaring code is not copyrightable because the "merger doctrine" bars copyright protection when there is only one way to express an idea. That argument fails for the same reasons Google's §102(b) argument fails. Even if the doctrine exists, Google admits that it is merely an application of §102(b). And, in any event, there may have been only one way for Google to copy the lines of declaring code, but there were innumerable ways for Oracle to write them. Certainly, Apple and Microsoft managed to create their own declaring code.

126 III

127 The Court inexplicably declines to address copyrightability. Its sole stated reason is that "technological, economic, and business-related circumstances" are "rapidly changing." That, of course, has been a constant where computers are concerned.

128 Rather than address this principal question, the Court simply assumes that declaring code is protected and then concludes that every fair-use factor favors Google. I agree with the majority that Congress did not "shiel[d] computer programs from the ordinary application" of fair use. But the majority's application of fair use is far from ordinary. By skipping copyrightability, the majority gets the methodology backward, causing the Court to sidestep a key conclusion that ineluctably affects the fair-use analysis: Congress rejected categorical distinctions between declaring and implementing code. But the majority creates just such a distinction. The result of this distorting analysis is an opinion that makes it difficult to imagine any circumstance in which declaring code will remain protected by copyright.

129 I agree with the majority that, under our precedent, fair use is a mixed question of fact and law and that questions of law predominate. Because the jury issued a finding of fair use in favor of Google, we must construe all factual disputes