GOOGLE LLC v. ORACLE AMERICA, INC.

Supreme Court of the United States, 2021
141 S.Ct. 1183

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.

Justice BREYER delivered the opinion of the Court.

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.

I

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. 886 F.3d 1179, 1187 (C.A. Fed. 2018); App. 137–138, 242–243. 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. (For visual explanations of "platforms" and other somewhat specialized computer-related terms, you might want to look at the material in Appendix A, *infra*.)

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. *Oracle America, Inc. v. Google Inc.*, 872 F.Supp.2d 974, 978 (N.D. Cal. 2012); App. 111, 464. That vision required attracting a sizeable number of skilled programmers.

At that time, many software developers understood and wrote programs using the Java programming language, a language invented by Sun Microsystems (Oracle's predecessor). 872 F.Supp.2d at 975, 977. About six million programmers had spent considerable time learning, and then using, the Java language. App. 228. Many of those programmers used Sun's own popular Java SE platform to develop new programs primarily for use in desktop and laptop computers. *Id.*, at 151–152, 200. 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"). 872 F.Supp.2d at 977. Indeed, one of Sun's slogans was " 'write once, run anywhere.' " 886 F.3d at 1186.

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. *Oracle*, 872 F.Supp.2d at 978. But Google did not want to insist that all programs written on the Android platform be interoperable. 886 F.3d at 1187. 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," App. 659, 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.

The record indicates that roughly 100 Google engineers worked for more than three years to create Google's Android platform software. *Id.*, at 45, 117, 212. 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. *Id.,* at 197–198. 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. 886 F.3d at 1187. The copied lines of code are part of a tool called an Application Programming Interface, or API.

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." *Oracle America, Inc. v. Google, Inc.*, 750 F.3d 1339, 1349 (2014). 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.

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.

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). See *Oracle*, 872 F.Supp.2d at 979–980. 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.

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.

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. See *id.,* at 979–980. Without that declaring code, the method calls entered by the programmer would not call up the implementing code.

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.

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.”

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.

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.

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. (For an illustration of this technology, see Appendix B, *infra*.)

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. App. 212. 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. *Id*., at 106–107. 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.

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." *Oracle*, 872 F.Supp.2d at 982. 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.

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. *Id.,* at 978. As of 2015, Android sales produced more than $42 billion in revenue. 886 F.3d at 1187.

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

II

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. *Oracle*, 872 F.Supp.2d at 975. 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. *Ibid.*

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. *Id.,* at 976. 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. *Ibid.* 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. *Id.,* at 977 (citing 17 U.S.C. § 102(b)).

On appeal, the Federal Circuit reversed. That court held that both the API's declaring code and its organizational structure could be copyrighted. *Oracle*, 750 F.3d at 1354. 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. *Id.,* at 1361.

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.' " *Id.,* at 1372–1373. 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." *Id.,* at 1377. 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.

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 Java API packages that it copied "constitutes a 'fair use' under the Copyright Act?" App. 294. After three days of deliberation the jury answered the question in the affirmative. *Id.*, at 295. Google had shown fair use.

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. 886 F.3d at 1193. 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." *Id.,* at 1210. It remanded the case again, this time for a trial on damages.

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.

III

....

B

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.

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.

IV

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) (C.C. D.Mass. 1841). See *Campbell*, 510 U.S. at 576 (noting how "Justice Story's summary [of fair use considerations] is discernable" in § 107). 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 may be stronger where the copyrighted material is fiction, not fact, where it consists of a motion picture rather than a news broadcast, or where it serves an artistic rather than a utilitarian function. See, *e.g.*, *Stewart*, 495 U.S., at 237–238; *Harper & Row*, 471 U.S., at 563; see also 4 M. Nimmer & D. Nimmer, Copyright § 13.05[A] [2][a] (2019) (hereinafter Nimmer on Copyright) ("[C]opyright protection is narrower, and the corresponding application of the fair use defense greater, in the case of factual works than in the case of works of fiction or fantasy"). Similarly, courts have held that in some circumstances, say, where copyrightable material is bound up with uncopyrightable material, copyright protection is "thin." See *Feist*, 499 U.S., at 349 (noting that "the copyright in a factual compilation is thin"); see also *Experian Information Solutions, Inc. v. Nationwide Marketing Servs. Inc.*, 893 F.3d 1176, 1186 (C.A.9 2018) ("In the context of factual compilations, … there can be no infringement unless the works are virtually identical" (internal quotation marks omitted)).

Generically speaking, computer programs differ from books, films, and many other "literary works" in that such programs almost always serve functional purposes. These and other differences have led at least some judges to complain that "applying copyright law to computer programs is like assembling a jigsaw puzzle whose pieces do not quite fit." *Lotus Development Corp. v. Borland Int'l, Inc.*, 49 F.3d 807, 820 (C.A.1 1995) (BOUDIN, J., concurring). …

The upshot, in our view, is that fair use can play an important role in determining the lawful scope of a computer program copyright, such as the copyright at issue here. It can help to distinguish among technologies. It can

distinguish between expressive and functional features of computer code where those features are mixed. It can focus on the legitimate need to provide incentives to produce copyrighted material while examining the extent to which yet further protection creates unrelated or illegitimate harms in other markets or to the development of other products. In a word, it can carry out its basic purpose of providing a context-based check that can help to keep a copyright monopoly within its lawful bounds. See H. R. Rep. No. 94–1476, pp. 65–66 (1976) (explaining that courts are to "adapt the doctrine [of fair use] to particular situations on a case-by-case basis" and in light of "rapid technological change"); see, *e.g., Lexmark Int'l, Inc. v. Static Control Components, Inc.*, 387 F.3d 522, 543–545 (C.A.6 2004) (discussing fair use in the context of copying to preserve compatibility); *Sony Computer Entertainment, Inc. v. Connectix Corp.*, 203 F.3d 596, 603–608 (C.A.9 2000) (applying fair use to intermediate copying necessary to reverse engineer access to unprotected functional elements within a program); *Sega Enterprises Ltd. v. Accolade, Inc.*, 977 F.2d 1510, 1521–1527 (C.A.9 1992) (holding that wholesale copying of copyrighted code as a preliminary step to develop a competing product was a fair use).

Justice THOMAS' thoughtful dissent offers a very different view of how (and perhaps whether) fair use has any role to play for computer programs. We are told that no attempt to distinguish among computer code is tenable when considering "the nature of the work," see *post*, at 1215 – 12016, even though there are important distinctions in the ways that programs are used and designed, *post*, at 1220 ("The declaring code is what attracted programmers"). We are told that no reuse of code in a new program will ever have a valid "purpose and character," *post*, at 1218 – 1219, even though the reasons for copying computer code may vary greatly and differ from those applicable to other sorts of works, *ibid.* (accepting that copying as part of "reverse engineer[ing] a system to ensure compatibility" could be a valid purpose). And we are told that our fair use analysis must prioritize certain factors over others, *post*, at 1215, n. 5, even though our case law instructs that fair use depends on the context, see *Campbell*, 510 U.S., at 577–578.

We do not understand Congress, however, to have shielded computer programs from the ordinary application of copyright's limiting doctrines in this way. By defining computer programs in § 101, Congress chose to place this subject matter within the copyright regime. Like other protected works, that means that the owners of computer programs enjoy the exclusive rights set forth in the Act, including the right to "reproduce [a] copyrighted work" or to "prepare derivative works." 17 U.S.C. § 106. But that also means that exclusive rights in computer programs are limited like any other works. Just as fair use distinguishes among books and films, which are indisputably subjects of copyright, so too must it draw lines among computer programs. And just as fair use takes account of the market in which scripts and paintings are bought and sold, so too must it consider the realities of how technological works are created

and disseminated. We do not believe that an approach close to "all or nothing" would be faithful to the Copyright Act's overall design.

V

At the outset, Google argues that "fair use" is a question for a jury to decide; here the jury decided the question in Google's favor; and we should limit our review to determining whether "substantial evidence" justified the jury's decision. The Federal Circuit disagreed. It thought that the "fair use" question was a mixed question of fact and law; that reviewing courts should appropriately defer to the jury's findings of underlying facts; but that the ultimate question whether those facts showed a "fair use" is a legal question for judges to decide *de novo*.

We agree with the Federal Circuit's answer to this question. We have said, "[f]air use is a mixed question of law and fact." *Harper & Row*, 471 U.S., at 560. We have explained that a reviewing court should try to break such a question into its separate factual and legal parts, reviewing each according to the appropriate legal standard. But when a question can be reduced no further, we have added that "the standard of review for a mixed question all depends—on whether answering it entails primarily legal or factual work." *U. S. Bank N. A.* v. *Village at Lakeridge, LLC*, 583 U. S. ––––, ––––, 138 S.Ct. 960, 967 (2018). …

VI

We turn now to the basic legal question before us: Was Google's copying of the Sun Java API, specifically its use of the declaring code and organizational structure for 37 packages of that API, a "fair use." In answering this question, we shall consider the four factors set forth in the fair use statute as we find them applicable to the kind of computer programs before us. We have reproduced those four statutory factors *supra*, at 1196 – 1197. For expository purposes, we begin with the second.

A. "The Nature of the Copyrighted Work"

The Sun Java API is a "user interface." It provides a way through which users (here the programmers) can "manipulate and control" task-performing computer programs "via a series of menu commands." *Lotus Development Corp.*, 49 F.3d at 809. The API reflects Sun's division of possible tasks that a computer might perform into a set of actual tasks that certain kinds of computers actually will perform. Sun decided, for example, that its API would call up a task that compares one integer with another to see which is the larger. Sun's API (to our knowledge) will not call up the task of determining which great Arabic scholar decided to use Arabic numerals (rather than Roman numerals) to perform that "larger integer" task. No one claims that the decisions about what counts as a task are themselves copyrightable—although one might argue about decisions as to

how to label and organize such tasks (*e.g.*, the decision to name a certain task "max" or to place it in a class called "Math." Cf. *Baker v. Selden*, 101 U.S. 99 (1880)).

As discussed above and in Appendix B, *infra*, we can think of the technology as having three essential parts. First, the API includes "implementing code," which actually instructs the computer on the steps to follow to carry out each task. Google wrote its own programs (implementing programs) that would perform each one of the tasks that its API calls up.

Second, the Sun Java API associates a particular command, called a "method call," with the calling up of each task. The symbols **java.lang.**, for example, are part of the command that will call up the program (whether written by Sun or, as here, by Google) that instructs the computer to carry out the "larger number" operation. Oracle does not here argue that the use of these commands by programmers itself violates its copyrights.

Third, the Sun Java API contains computer code that will associate the writing of a method call with particular "places" in the computer that contain the needed implementing code. This is the declaring code. The declaring code both labels the particular tasks in the API and organizes those tasks, or "methods," into "packages" and "classes." We have referred to this organization, by way of rough analogy, as file cabinets, drawers, and files. Oracle does claim that Google's use of the Sun Java API's declaring code violates its copyrights.

The declaring code at issue here resembles other copyrighted works in that it is part of a computer program. Congress has specified that computer programs are subjects of copyright. It differs, however, from many other kinds of copyrightable computer code. It is inextricably bound together with a general system, the division of computing tasks, that no one claims is a proper subject of copyright. It is inextricably bound up with the idea of organizing tasks into what we have called cabinets, drawers, and files, an idea that is also not copyrightable. It is inextricably bound up with the use of specific commands known to programmers, known here as method calls (such as **java.lang.Math.max**, etc.), that Oracle does not here contest. And it is inextricably bound up with implementing code, which is copyrightable but was not copied.

Moreover, the copied declaring code and the uncopied implementing programs call for, and reflect, different kinds of capabilities. A single implementation may walk a computer through dozens of different steps. To write implementing programs, witnesses told the jury, requires balancing such considerations as how quickly a computer can execute a task or the likely size of the computer's memory. One witness described that creativity as "magic" practiced by an API developer when he or she worries "about things like power management" for devices that "run on a battery." App. 143; see also *id.*, at 147, 204. This is the very creativity that was needed to develop the Android software

for use not in laptops or desktops but in the very different context of smartphones.

The declaring code (inseparable from the programmer's method calls) embodies a different kind of creativity. Sun Java's creators, for example, tried to find declaring code names that would prove intuitively easy to remember. *Id.*, at 211. They wanted to attract programmers who would learn the system, help to develop it further, and prove reluctant to use another. See *post*, at 1215 ("Declaring code ... is user facing. It must be designed and organized in a way that is intuitive and understandable to developers so that they can invoke it"). Sun's business strategy originally emphasized the importance of using the API to attract programmers. It sought to make the API "open" and "then ... compete on implementations." App. 124–125. The testimony at trial was replete with examples of witnesses drawing this critical line between the user-centered declaratory code and the innovative implementing code. *Id.*, at 126–127, 159–160, 163–164, 187, 190–191.

These features mean that, as part of a user interface, the declaring code differs to some degree from the mine run of computer programs. Like other computer programs, it is functional in nature. But unlike many other programs, its use is inherently bound together with uncopyrightable ideas (general task division and organization) and new creative expression (Android's implementing code). Unlike many other programs, its value in significant part derives from the value that those who do not hold copyrights, namely, computer programmers, invest of their own time and effort to learn the API's system. And unlike many other programs, its value lies in its efforts to encourage programmers to learn and to use that system so that they will use (and continue to use) Sun-related implementing programs that Google did not copy.

Although copyrights protect many different kinds of writing, Leval 1116, we have emphasized the need to "recogni[ze] that some works are closer to the core of [copyright] than others," *Campbell*, 510 U.S., at 586. In our view, for the reasons just described, the declaring code is, if copyrightable at all, further than are most computer programs (such as the implementing code) from the core of copyright. That fact diminishes the fear, expressed by both the dissent and the Federal Circuit, that application of "fair use" here would seriously undermine the general copyright protection that Congress provided for computer programs. And it means that this factor, "the nature of the copyrighted work," points in the direction of fair use.

B. "The Purpose and Character of the Use"

In the context of fair use, we have considered whether the copier's use "adds something new, with a further purpose or different character, altering" the copyrighted work "with new expression, meaning or message." *Id.*, at 579. Commentators have put the matter more broadly, asking whether the copier's use

"fulfill[s] the objective of copyright law to stimulate creativity for public illumination." Leval 1111. In answering this question, we have used the word "transformative" to describe a copying use that adds something new and important. *Campbell*, 510 U.S., at 579. An " 'artistic painting' " might, for example, fall within the scope of fair use even though it precisely replicates a copyrighted " 'advertising logo to make a comment about consumerism.' " 4 Nimmer on Copyright § 13.05[A][1][b] (quoting Netanel, Making Sense of Fair Use, 15 Lewis & Clark L. Rev. 715, 746 (2011)). Or, as we held in *Campbell*, a parody can be transformative because it comments on the original or criticizes it, for "[p]arody needs to mimic an original to make its point." 510 U.S., at 580–581.

Google copied portions of the Sun Java API precisely, and it did so in part for the same reason that Sun created those portions, namely, to enable programmers to call up implementing programs that would accomplish particular tasks. But since virtually any unauthorized use of a copyrighted computer program (say, for teaching or research) would do the same, to stop here would severely limit the scope of fair use in the functional context of computer programs. Rather, in determining whether a use is "transformative," we must go further and examine the copying's more specifically described "purpose[s]" and "character." 17 U.S.C. § 107(1).

Here Google's use of the Sun Java API seeks to create new products. It seeks to expand the use and usefulness of Android-based smartphones. Its new product offers programmers a highly creative and innovative tool for a smartphone environment. To the extent that Google used parts of the Sun Java API to create a new platform that could be readily used by programmers, its use was consistent with that creative "progress" that is the basic constitutional objective of copyright itself. Cf. *Feist*, 499 U.S., at 349–350 ("The primary objective of copyright is not to reward the labor of authors, but '[t]o promote the Progress of Science and useful Arts' " (quoting U. S. Const., Art. I, § 8, cl. 8)).

The jury heard that Google limited its use of the Sun Java API to tasks and specific programming demands related to Android. It copied the API (which Sun created for use in desktop and laptop computers) only insofar as needed to include tasks that would be useful in smartphone programs. App. 169–170. And it did so only insofar as needed to allow programmers to call upon those tasks without discarding a portion of a familiar programming language and learning a new one. *Id.*, at 139–140. To repeat, Google, through Android, provided a new collection of tasks operating in a distinct and different computing environment. Those tasks were carried out through the use of new implementing code (that Google wrote) designed to operate within that new environment. Some of the *amici* refer to what Google did as "reimplementation," defined as the "building of a system ... that repurposes the same words and syntaxes" of an existing system—in this case so that programmers who had learned an existing system could put their basic skills to use in a new one. Brief for R Street Institute et al. as *Amici Curiae* 2.

The record here demonstrates the numerous ways in which reimplementing an interface can further the development of computer programs. The jury heard that shared interfaces are necessary for different programs to speak to each other. App. 125 ("We have to agree on the APIs so that the application I write to show a movie runs on your device"). It heard that the reimplementation of interfaces is necessary if programmers are to be able to use their acquired skills. *Id.*, at 191 ("If the API labels change, then either the software wouldn't continue to work anymore or the developer ... would have to learn a whole new language to be able to use these API labels"). It heard that the reuse of APIs is common in the industry. *Id.*, at 115, 155, 663. It heard that Sun itself had used pre-existing interfaces in creating Java. *Id.*, at 664. And it heard that Sun executives thought that widespread use of the Java programming language, including use on a smartphone platform, would benefit the company. *Id.*, at 130–133. ....

These and related facts convince us that the "purpose and character" of Google's copying was transformative—to the point where this factor too weighs in favor of fair use.

There are two other considerations that are often taken up under the first factor: commerciality and good faith. The text of § 107 includes various noncommercial uses, such as teaching and scholarship, as paradigmatic examples of privileged copying. There is no doubt that a finding that copying was not commercial in nature tips the scales in favor of fair use. But the inverse is not necessarily true, as many common fair uses are indisputably commercial. For instance, the text of § 107 includes examples like "news reporting," which is often done for commercial profit. So even though Google's use was a commercial endeavor—a fact no party disputed, see 886 F.3d at 1197—that is not dispositive of the first factor, particularly in light of the inherently transformative role that the reimplementation played in the new Android system.

As for bad faith, our decision in *Campbell* expressed some skepticism about whether bad faith has any role in a fair use analysis. 510 U.S., at 585, n. 18. We find this skepticism justifiable, as "[c]opyright is not a privilege reserved for the well-behaved." Leval 1126. We have no occasion here to say whether good faith is as a general matter a helpful inquiry. We simply note that given the strength of the other factors pointing toward fair use and the jury finding in Google's favor on hotly contested evidence, that factbound consideration is not determinative in this context.

C. "The Amount and Substantiality of the Portion Used"

If one considers the declaring code in isolation, the quantitative amount of what Google copied was large. Google copied the declaring code for 37 packages of the Sun Java API, totaling approximately 11,500 lines of code. Those

lines of code amount to virtually all the declaring code needed to call up hundreds of different tasks. On the other hand, if one considers the entire set of software material in the Sun Java API, the quantitative amount copied was small. The total set of Sun Java API computer code, including implementing code, amounted to 2.86 million lines, of which the copied 11,500 lines were only 0.4 percent. App. 212.

The question here is whether those 11,500 lines of code should be viewed in isolation or as one part of the considerably greater whole. We have said that even a small amount of copying may fall outside of the scope of fair use where the excerpt copied consists of the " 'heart' " of the original work's creative expression. *Harper & Row*, 471 U.S., at 564–565. On the other hand, copying a larger amount of material can fall within the scope of fair use where the material copied captures little of the material's creative expression or is central to a copier's valid purpose. See, *e.g.*, *Campbell*, 510 U.S., at 588; *New Era Publications Int'l, ApS v. Carol Publishing Group*, 904 F.2d 152, 158 (C.A.2 1990). If a defendant had copied one sentence in a novel, that copying may well be insubstantial. But if that single sentence set forth one of the world's shortest short stories—"When he awoke, the dinosaur was still there."—the question looks much different, as the copied material constitutes a small part of the novel but the entire short story. See A. Monterroso, El Dinosaurio, in Complete Works & Other Stories 42 (E. Grossman transl. 1995). (In the original Spanish, the story reads: "Cuando despertó, el dinosaurio todavía estaba allí.")

Several features of Google's copying suggest that the better way to look at the numbers is to take into account the several million lines that Google did not copy. For one thing, the Sun Java API is inseparably bound to those task-implementing lines. Its purpose is to call them up. For another, Google copied those lines not because of their creativity, their beauty, or even (in a sense) because of their purpose. It copied them because programmers had already learned to work with the Sun Java API's system, and it would have been difficult, perhaps prohibitively so, to attract programmers to build its Android smartphone system without them. Further, Google's basic purpose was to create a different task-related system for a different computing environment (smartphones) and to create a platform—the Android platform—that would help achieve and popularize that objective. The "substantiality" factor will generally weigh in favor of fair use where, as here, the amount of copying was tethered to a valid, and transformative, purpose. *Supra*, at 1203 – 1204; see *Campbell*, 510 U.S., at 586–587 (explaining that the factor three "enquiry will harken back to the first of the statutory factors, for ... the extent of permissible copying varies with the purpose and character of the use").

We do not agree with the Federal Circuit's conclusion that Google could have achieved its Java-compatibility objective by copying only the 170 lines of code that are "necessary to write in the Java language." 886 F.3d at 1206. In our view, that conclusion views Google's legitimate objectives too narrowly. Google's

basic objective was not simply to make the Java programming language usable on its Android systems. It was to permit programmers to make use of their knowledge and experience using the Sun Java API when they wrote new programs for smartphones with the Android platform. In principle, Google might have created its own, different system of declaring code. But the jury could have found that its doing so would not have achieved that basic objective. In a sense, the declaring code was the key that it needed to unlock the programmers' creative energies. And it needed those energies to create and to improve its own innovative Android systems.

We consequently believe that this "substantiality" factor weighs in favor of fair use.

D. Market Effects

The fourth statutory factor focuses upon the "effect" of the copying in the "market for or value of the copyrighted work." 17 U.S.C. § 107(4). Consideration of this factor, at least where computer programs are at issue, can prove more complex than at first it may seem. It can require a court to consider the amount of money that the copyright owner might lose. As we pointed out in *Campbell*, "verbatim copying of the original in its entirety for commercial purposes" may well produce a market substitute for an author's work. 510 U.S., at 591. Making a film of an author's book may similarly mean potential or presumed losses to the copyright owner. Those losses normally conflict with copyright's basic objective: providing authors with exclusive rights that will spur creative expression.

But a potential loss of revenue is not the whole story. We here must consider not just the amount but also the source of the loss. As we pointed out in *Campbell*, a "lethal parody, like a scathing theatre review," may "kil[l] demand for the original." *Id.*, at 591–592. Yet this kind of harm, even if directly translated into foregone dollars, is not "cognizable under the Copyright Act." *Id.,* at 592.

Further, we must take into account the public benefits the copying will likely produce. Are those benefits, for example, related to copyright's concern for the creative production of new expression? Are they comparatively important, or unimportant, when compared with dollar amounts likely lost (taking into account as well the nature of the source of the loss)? Cf. *MCA, INC. v. Wilson*, 677 F.2d 180, 183 (C.A.2 1981) (calling for a balancing of public benefits and losses to copyright owner under this factor).

We do not say that these questions are always relevant to the application of fair use, not even in the world of computer programs. Nor do we say that these questions are the only questions a court might ask. But we do find them relevant here in helping to determine the likely market effects of Google's reimplementation.

As to the likely amount of loss, the jury could have found that Android did not harm the actual or potential markets for Java SE. And it could have found that Sun itself (now Oracle) would not have been able to enter those markets successfully whether Google did, or did not, copy a part of its API. First, evidence at trial demonstrated that, regardless of Android's smartphone technology, Sun was poorly positioned to succeed in the mobile phone market. The jury heard ample evidence that Java SE's primary market was laptops and desktops. App. 99, 200. It also heard that Sun's many efforts to move into the mobile phone market had proved unsuccessful. *Id.,* at 135, 235, 671. As far back as 2006, prior to Android's release, Sun's executives projected declining revenue for mobile phones because of emerging smartphone technology. *Id.,* at 240. When Sun's former CEO was asked directly whether Sun's failure to build a smartphone was attributable to Google's development of Android, he answered that it was not. *Id.,* at 650. Given the evidence showing that Sun was beset by business challenges in developing a mobile phone product, the jury was entitled to agree with that assessment.

Second, the jury was repeatedly told that devices using Google's Android platform were different in kind from those that licensed Sun's technology. For instance, witnesses explained that the broader industry distinguished between smartphones and simpler "feature phones." *Id.,* at 237. As to the specific devices that used Sun-created software, the jury heard that one of these phones lacked a touchscreen, *id.*, at 359–360, while another did not have a QWERTY keyboard, *id.*, at 672. For other mobile devices, the evidence showed that simpler products, like the Kindle, used Java software, *id.*, at 396, while more advanced technology, like the Kindle Fire, were built on the Android operating system, *id.*, at 206. This record evidence demonstrates that, rather than just "repurposing [Sun's] code from larger computers to smaller computers," *post*, at 16, Google's Android platform was part of a distinct (and more advanced) market than Java software.

Looking to these important differences, Google's economic expert told the jury that Android was not a market substitute for Java's software. As he explained, "the two products are on very different devices," and the Android platform, which offers "an entire mobile operating stack," is a "very different typ[e] of produc[t]" than Java SE, which is "just an applications programming framework." App. 256; see also *id.*, at 172–174. Taken together, the evidence showed that Sun's mobile phone business was declining, while the market increasingly demanded a new form of smartphone technology that Sun was never able to offer.

Finally, the jury also heard evidence that Sun foresaw a benefit from the broader use of the Java programming language in a new platform like Android, as it would further expand the network of Java-trained programmers. *Id.*, at 131–133; see also *id.*, at 153 ("Once an API starts getting reimplemented, you know it has succeeded"). In other words, the jury could have understood Android and Java SE as operating in two distinct markets. And because there are two markets

at issue, programmers learning the Java language to work in one market (smartphones) are then able to bring those talents to the other market (laptops). See 4 Nimmer on Copyright § 13.05[A][4] (explaining that factor four asks what the impact of "widespread conduct of the sort engaged in by the defendant" would be on the market for the present work).

Oracle presented evidence to the contrary. Indeed, the Federal Circuit held that the "market effects" factor militated against fair use in part because Sun had tried to enter the Android market. 886 F.3d at 1209 (Sun sought licensing agreement with Google). But those licensing negotiations concerned much more than 37 packages of declaring code, covering topics like "the implementation of [Java's] code" and "branding and cooperation" between the firms. App. 245; see also 4 Nimmer on Copyright § 13.05[A][4] (cautioning against the "danger of circularity posed" by considering unrealized licensing opportunities because "it is a given in every fair use case that plaintiff suffers a loss of a *potential* market if that potential is defined as the theoretical market for licensing the very use at bar"). In any event, the jury's fair use determination means that neither Sun's effort to obtain a license nor Oracle's conflicting evidence can overcome evidence indicating that, at a minimum, it would have been difficult for Sun to enter the smartphone market, even had Google not used portions of the Sun Java API.

On the other hand, Google's copying helped Google make a vast amount of money from its Android platform. And enforcement of the Sun Java API copyright might give Oracle a significant share of these funds. It is important, however, to consider why and how Oracle might have become entitled to this money. When a new interface, like an API or a spreadsheet program, first comes on the market, it may attract new users because of its expressive qualities, such as a better visual screen or because of its superior functionality. As time passes, however, it may be valuable for a different reason, namely, because users, including programmers, are just used to it. They have already learned how to work with it. See *Lotus Development Corp.*, 49 F.3d at 821 (BOUDIN, J., concurring).

The record here is filled with evidence that this factor accounts for Google's desire to use the Sun Java API. See, *e.g.,* App. 169–170, 213–214. This source of Android's profitability has much to do with third parties' (say, programmers') investment in Sun Java programs. It has correspondingly less to do with Sun's investment in creating the Sun Java API. We have no reason to believe that the Copyright Act seeks to protect third parties' investment in learning how to operate a created work.

Finally, given programmers' investment in learning the Sun Java API, to allow enforcement of Oracle's copyright here would risk harm to the public. Given the costs and difficulties of producing alternative APIs with similar appeal to programmers, allowing enforcement here would make of the Sun Java API's declaring code a lock limiting the future creativity of new programs. Oracle alone

would hold the key. The result could well prove highly profitable to Oracle (or other firms holding a copyright in computer interfaces). But those profits could well flow from creative improvements, new applications, and new uses developed by users who have learned to work with that interface. To that extent, the lock would interfere with, not further, copyright's basic creativity objectives. See *Connectix Corp.*, 203 F.3d at 607; see also *Sega Enterprises*, 977 F.2d at 1523–1524 ("An attempt to monopolize the market by making it impossible for 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.

The uncertain nature of Sun's ability to compete in Android's marketplace, 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.

* * *

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.

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.

It is so ordered.

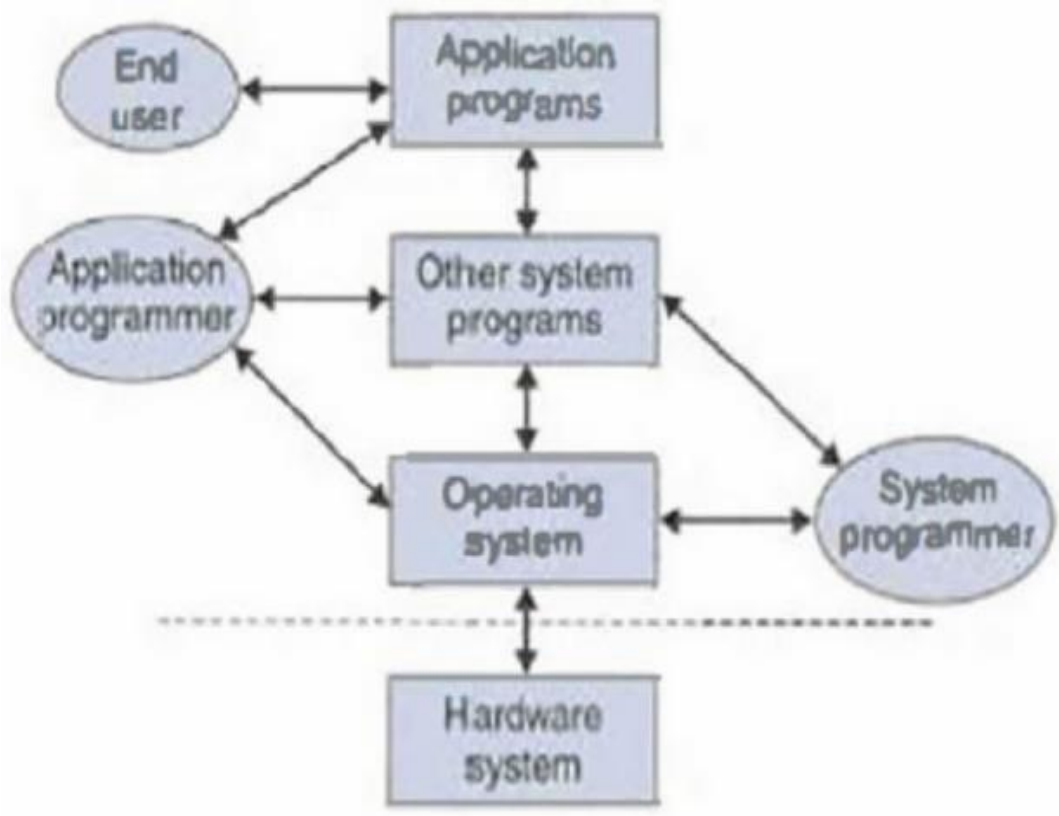
Justice BARRETT took no part in the consideration or decision of this case.

APPENDIX A

Computer System Diagram

Some readers might find it helpful to start with an explanation of what a "software platform" is. Put simply, a software platform collects all of the software tools that a programmer may need to build computer programs. The Android platform, for instance, includes an "operating system," "core libraries," and a "virtual machine," among other tools. App. 197–198.

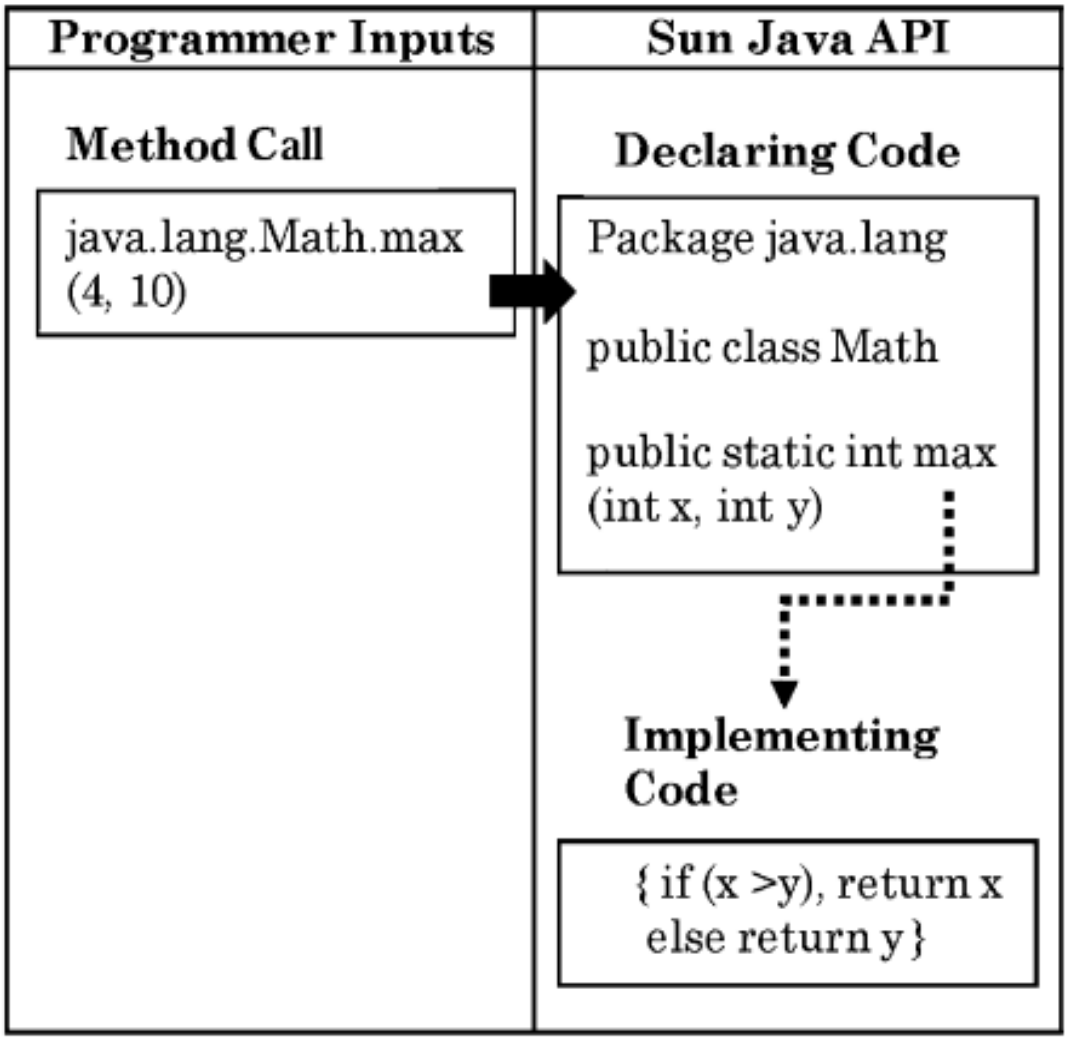
The diagram below illustrates the general features of a standard computer system, with the dotted line reflecting the division between a computer's hardware and its software. (It is not intended to reflect any specific technology at issue in this case.)



J. Garrido & R. Schlesinger, Principles of Modern Operating Systems 8 (2008) ("Figure 1.4. An External View of a Computer System").

APPENDIX B

Sun Java API Diagram



This image depicts the connection between the three parts of the Sun Java API technology at issue, using the District Court's example. *Oracle*, 872 F.Supp.2d at 980–981. The programmer enters a method call to invoke a task from within the API (the solid arrow). The precise symbols in the method call correspond to a single task, which is located within a particular class. That class is located within a particular package. All of the lines of code that provide that organization and name the methods, classes, and packages are "declaring code." For each method, the declaring code is associated with particular lines of

implementing code (the dotted arrow). It is that implementing code (which Google wrote for its Android API) that actually instructs the computer in the programmer's application.

Justice THOMAS, with whom Justice ALITO joins, dissenting.

Oracle spent years developing a programming library that successfully attracted software developers, thus enhancing the value of Oracle's products.[1] 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.

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.

I

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. 8 U.S.C. § 1101(42). 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.

[1] A different company, Sun, created the library. But because Oracle later purchased Sun, for simplicity I refer to both companies as Oracle.

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.[2] It is like the detailed definition in a statute.

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 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.

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." App. 657.

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." *Id.*, at 600. Oracle predictably responded by suing Google for copyright infringement. The

---

[2] 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)."

Federal Circuit ruled that Oracle's declaring code is copyrightable and that Google's copying of it was not fair use.

II

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.

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.

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.

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. *Ibid.* First, they are expressed in "words, numbers, or other verbal or numerical symbols" and are thus works of authorship. § 101. Second, as Google concedes, the lines of declaring code are original because Oracle could have created them any number of ways.

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. *Id.*, at 101–102. 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.

That argument fails. As the majority correctly recognizes, declaring code and implementing code are "inextricably bound" together. *Ante*, at 1201. 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.

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.' " *RadLAX Gateway Hotel, LLC v. Amalgamated Bank*, 566 U.S. 639, 645 (2012).

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." *United States v. Williams*, 553 U.S. 285, 294 (2008). 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. …

III

The Court inexplicably declines to address copyrightability. Its sole stated reason is that "technological, economic, and business-related circumstances" are

"rapidly changing." *Ante*, at 1197 – 1198. That, of course, has been a constant where computers are concerned.

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. *Ante,* at 1199. 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.

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 and inferences in Google's favor and ask whether the evidence was sufficient as a matter of law to support the jury's verdict. See Fed. Rule Civ. Proc. 50(b). But whether a statutory fair-use factor favors one side or the other is a legal question reviewed *de novo*. Congress has established four statutory fair-use factors for courts to weigh.[4] Three decisively favor Oracle. And even assuming that the remaining factor favors Google, that factor, without more, cannot legally establish fair use in this context.

The majority holds otherwise—concluding that *every* factor favors Google—by relying, in large part, on a distinction it draws between declaring and implementing code, a distinction that the statute rejects. Tellingly, the majority evaluates the factors neither in sequential order nor in order of importance (at least two factors are more important under our precedent[5]). Instead, it starts with the second factor: the nature of the copyrighted work. It proceeds in this manner in order to create a distinction between declaring and implementing code that renders the former less worthy of protection than the latter. Because the majority's mistaken analysis rests so heavily on this factor, I begin with it as well.

A. The Nature of the Copyrighted Work

This factor requires courts to assess the level of creativity or functionality in the original work. It generally favors fair use when a copyrighted work is more

[5] The fourth factor—the effect of Google's copying on the potential market for Oracle's work—is "undoubtedly the single most important element of fair use." *Harper & Row, Publishers, Inc. v. Nation Enterprises*, 471 U.S. 539, 566 (1985). The first factor—the purpose and character of the use, including whether the use is commercial—is the second-most important because it can prove dispositive. See *id*., at 550 ("[In general,] the fair use doctrine has always precluded a use that 'supersede[s] the use of the original' ").

"informational or functional" than "creative." 4 M. Nimmer & D. Nimmer, Copyright § 13.05[A][2][a] (2019). Because code is predominantly functional, this factor will often favor copying when the original work is computer code. But because Congress determined that declaring and implementing code are copyrightable, this factor alone cannot support a finding of fair use.

The majority, however, uses this factor to create a distinction between declaring and implementing code that in effect removes copyright protection from declaring code. It concludes that, unlike implementing code, declaring code is far "from the core of copyright" because it becomes valuable only when third parties (computer programmers) value it and because it is "inherently bound together with uncopyrightable ideas." *Ante*, at 1202 – 1203.

Congress, however, rejected this sort of categorical distinction that would make declaring code less worthy of protection. The Copyright Act protects code that operates "in a computer in order to bring about a certain result" both "directly" (implementing code) and "indirectly" (declaring code). § 101. And if anything, declaring code is *closer* to the "core of copyright." *Ante*, at 1202 – 1203. Developers cannot even see implementing code. *Oracle Am., Inc. v. Google Inc.*, 2016 WL 3181206, *4 (ND Cal., June 8, 2016); see also *ante*, at 1201 – 1202 (declaring code is "user-centered"). Implementing code thus conveys *no* expression to developers. Declaring code, in contrast, is user facing. It must be designed and organized in a way that is intuitive and understandable to developers so that they can invoke it.

Even setting those concerns aside, the majority's distinction is untenable. True, declaring code is "inherently bound together with uncopyrightable ideas." *Ante,* at 1201 – 1203. Is anything not? Books are inherently bound with uncopyrightable ideas—the use of chapters, having a plot, or including dialogue or footnotes. This does not place books far "from the core of copyright." And implementing code, which the majority concedes is copyrightable, is inherently bound up with "the division of computing tasks" that cannot be copyrighted.[6] *Ante*, at 1201. We have not discounted a work of authorship simply because it is associated with noncopyrightable ideas. While ideas cannot be copyrighted, expressions of those ideas can. *Golan*, 565 U.S., at 328.

Similarly, it makes no difference that the value of declaring code depends on how much time third parties invest in learning it. Many other copyrighted works depend on the same. A Broadway musical script needs actors and singers to invest time learning and rehearsing it. But a theater cannot copy a script—the

[6] The majority also belittles declaring code by suggesting it is simply a way to organize implementing code. *Ante*, at 1201 – 1202. Not so. Declaring code *defines* subprograms of implementing code, including by controlling what inputs they can process. Similarly, the majority is wrong to suggest that the purpose of declaring code is to connect pre-existing method calls to implementing code. *Ante,* at 1192. Declaring code *creates* the method calls.

rights to which are held by a smaller theater—simply because it wants to entice actors to switch theaters and because copying the script is more efficient than requiring the actors to learn a new one.

What the majority says is true of declaring code is no less true of implementing code. Declaring code is how programmers access prewritten implementing code. The value of that implementing code thus is directly proportional to how much programmers value the associated declaring code. The majority correctly recognizes that declaring code "is inextricably bound up with implementing code," *ante*, at 1201 – 1202, but it overlooks the implications of its own conclusion.

Only after wrongly concluding that the nature of declaring code makes that code generally unworthy of protection does the Court move on to consider the other factors. This opening mistake taints the Court's entire analysis.

B. Market Effects

"[U]ndoubtedly the single most important element of fair use" is the effect of Google's copying " 'upon the potential market for or value of [Oracle's] copyrighted work.' " *Harper & Row, Publishers, Inc. v. Nation Enterprises*, 471 U.S. 539, 566 (1985). As the Federal Circuit correctly determined, "evidence of actual and potential harm stemming from Google's copying was 'overwhelming.' " 886 F.3d 1179, 1209 (2018). By copying Oracle's code to develop and release Android, Google ruined Oracle's potential market in at least two ways.

First, Google eliminated the reason manufacturers were willing to pay to install the Java platform. Google's business model differed from Oracle's. While Oracle earned revenue by charging device manufacturers to install the Java platform, Google obtained revenue primarily through ad sales. Its strategy was to release Android to device manufacturers for free and then use Android as a vehicle to collect data on consumers and deliver behavioral ads. With a free product available that included much of Oracle's code (and thus with similar programming potential), device manufacturers no longer saw much reason to pay to embed the Java platform.

For example, before Google released Android, Amazon paid for a license to embed the Java platform in Kindle devices. But after Google released Android, Amazon used the cost-free availability of Android to negotiate a 97.5% discount on its license fee with Oracle. Evidence at trial similarly showed that right after Google released Android, Samsung's contract with Oracle dropped from $40 million to about $1 million. Google contests none of this except to say that Amazon used a different Java platform, Java Micro Edition instead of Java Standard Edition. That difference is inconsequential because the former was simply a smaller subset of the latter. Google copied code found in both platforms. The majority does not dispute—or even mention—this enormous harm.

Second, Google interfered with opportunities for Oracle to license the Java platform to developers of smartphone operating systems. Before Google copied Oracle's code, nearly every mobile phone on the market contained the Java platform. Oracle's code was extraordinarily valuable to anybody who wanted to develop smartphones, which explains why Google tried no fewer than four times to license it. The majority's remark that Google also sought other licenses from Oracle, *ante*, at 1207 – 1208, does not change this central fact. Both parties agreed that Oracle could enter Google's current market by licensing its declaring code. But by copying the code and releasing Android, Google eliminated Oracle's opportunity to license its code for that use.

The majority writes off this harm by saying that the jury could have found that Oracle might not have been able to enter the modern smartphone market successfully.[7] *Ante*, at 1206 – 1207. But whether Oracle could itself enter that market is only half the picture. We look at not only the potential market "that creators of original works would in general develop" but also those potential markets the copyright holder might "license others to develop." *Campbell v. Acuff-Rose Music, Inc.*, 510 U.S. 569, 592 (1994). A book author need not be able to personally convert a book into a film so long as he can license someone else to do so. That Oracle could have licensed its code for use in Android is undisputed.

Unable to seriously dispute that Google's actions had a disastrous effect on Oracle's potential market, the majority changes course and asserts that enforcing copyright protection could harm the public by giving Oracle the power to "limi[t] the future creativity" of programs on Android. *Ante*, at 1208. But this case concerns only versions of Android released through November 2014. Order in No. 3:10–cv–3561 (ND Cal., Feb. 5, 2016), Doc. 1479, p. 2 (identifying versions through Android Lollipop 5.0). Google has released six major versions since then. Only about 7.7% of active Android devices still run the versions at issue.[8][8] The majority's concern about a lock-in effect might carry more weight if this suit concerned versions of Android widely in use or that will be widely in use. It makes little sense in a suit about versions that are close to obsolete.

The majority's concern about a lock-in effect also is speculation belied by history. First, Oracle never had lock-in power. The majority (again) overlooks that Apple and Microsoft created mobile operating systems without using Oracle's declaring code. Second, Oracle always made its declaring code freely available to programmers. There is little reason to suspect Oracle might harm programmers by stopping now. And third, the majority simply assumes that the

[7] It also suggests that Oracle may have received some incidental benefit from Android. *Ante*, at 1206 – 1208. But even assuming that is true, it would go to the question of damages, not fair use. And there is no evidence that any benefit came even close to offsetting Oracle's enormous loss.

[8] Rahman, Android Version Distribution Statistics Will Now Only Be Available in Android Studio (Apr. 10, 2020), https://www.xda-developers.com/android-version-distribution-statistics-android-studio.

jury, in a future suit over current Android versions, would give Oracle control of Android instead of just awarding damages or perpetual royalties.

If the majority is going to speculate about what Oracle *might* do, it at least should consider what Google *has* done. The majority expresses concern that Oracle might abuse its copyright protection (on outdated Android versions) and "'attempt to monopolize the market.' " *Ante*, at 1208 – 1209. But it is Google that recently was fined a record $5 billion for abusing Android to violate antitrust laws. Case AT.40099, *Google Android*, July 18, 2018 (Eur. Comm'n-Competition); European Comm'n Press Release, Commission Fines Google €4.34 Billion for Illegal Practices Regarding Android Mobile Devices to Strengthen Dominance of Google's Search Engine, July 18, 2018. Google controls the most widely used mobile operating system in the world. And if companies may now freely copy libraries of declaring code whenever it is more convenient than writing their own, others will likely hesitate to spend the resources Oracle did to create intuitive, well-organized libraries that attract programmers and could compete with Android. If the majority is worried about monopolization, it ought to consider whether Google is the greater threat.

By copying Oracle's work, Google decimated Oracle's market and created a mobile operating system now in over 2.5 billion actively used devices, earning tens of billions of dollars every year. If these effects on Oracle's potential market *favor* Google, something is very wrong with our fair-use analysis.

C. The Purpose and Character of the Use

The second-most important factor—"the purpose and character of the use, including whether such use is of a commercial nature or is for nonprofit educational purposes," § 107(1)—requires us to consider whether use was "commercial" and whether it was "transformative." *Campbell*, 510 U.S., at 578–579. Both aspects heavily favor Oracle.

Begin with the overwhelming commercial nature of Google's copying. In 2015 alone, the year before the fair-use trial, Google earned $18 billion from Android. That number has no doubt dramatically increased as Android has grown to dominate the global market share.[9] On this scale, Google's use of Oracle's declaring code weighs heavily—if not decisively—against fair use.

[9] The real value also may be much higher because Android indirectly boosts other sources of revenue. For years Google has set its search engine as the default engine on Android. Google can use that engine to collect reams of data used to deliver behavioral advertisements to consumers on desktops. Using control over Android to choose a default search engine may seem trivial, but Google certainly does not think so. According to a Goldman Sachs analysis, Google paid Apple *$12 billion* to be the default search engine for Safari, Apple's web browser, for just one year. Leswing, Apple Makes Billions From Google's Dominance in Search—And It's a Bigger Business Than iCloud or Apple Music, Business Insider, Sept. 29, 2018. Google does not appear to have disputed this figure.

The majority attempts to dismiss this overwhelming commercial use by noting that commercial use does "not necessarily" weigh against fair use. *Ante*, at 1204. True enough. Commercial use sometimes can be overcome by use that is sufficiently "transformative." *Campbell*, 510 U.S., at 579. But "we cannot ignore [Google's] *intended purpose* of supplanting [Oracle's] commercially valuable" platform with its own. *Harper*, 471 U.S., at 562 (emphasis in original). Even if we could, we have never found fair use for copying that reaches into the tens of billions of dollars and wrecks the copyright holder's market.

Regardless, Google fares no better on transformative use. A court generally cannot find fair use unless the copier's use is transformative.[10] A work is "transformative" if it "adds something new, with a further purpose or different character, altering the first with new expression, meaning, or message." *Campbell*, 510 U.S., at 579. This question is "guided by the examples [of fair use] given in the preamble to § 107." *Id.*, at 578. Those examples include: "criticism, comment, news reporting, teaching ..., scholarship, or research." § 107. Although these examples are not exclusive, they are illustrative, and Google's repurposing of Java code from larger computers to smaller computers resembles none of them. Google did not use Oracle's code to teach or reverse engineer a system to ensure compatibility. Instead, to "avoid the drudgery in working up something fresh," *id.*, at 580, Google used the declaring code for the same exact purpose Oracle did. As the Federal Circuit correctly determined, "[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." 886 F.3d at 1210.

The majority acknowledges that Google used the copied declaring code "for the same reason" Oracle did. *Ante*, at 1203. So, by turns, the majority transforms the definition of "transformative." Now, we are told, "transformative" simply means—at least for computer code—a use that will help others "create new products." *Ibid;* accord, *ante*, at 1203 (Google's copying "can further the development of computer programs").

That new definition eviscerates copyright. A movie studio that converts a book into a film without permission not only creates a new product (the film) but enables others to "create products"—film reviews, merchandise, YouTube highlight reels, late night television interviews, and the like. Nearly every computer program, once copied, can be used to create new products. Surely the

[10] Although "transformative use is not *absolutely* necessary" every time, *Campbell v. Acuff-Rose Music, Inc.*, 510 U.S. 569, 579, and n. 11 (1994) (emphasis added), as a general matter "the fair use doctrine has always precluded a use that 'supersedes the use of the original,' " *Harper*, 471 U.S., at 550 (brackets omitted).

majority would not say that an author can pirate the next version of Microsoft Word simply because he can use it to create new manuscripts.[11]

Ultimately, the majority wrongly conflates transformative use with derivative use. To be transformative, a work must do something fundamentally different from the original. A work that simply serves the same purpose in a new context—which the majority concedes is true here—is derivative, not transformative. Congress made clear that Oracle holds "the exclusive rights ... to prepare derivative works." § 106(2). Rather than create a transformative product, Google "profit[ed] from exploitation of the copyrighted material without paying the customary price." *Harper*, 471 U.S., at 562.

D. The Amount and Substantiality of the Portion Used

The statutory fair-use factors also instruct us to consider "the amount and substantiality of the portion used in relation to the copyrighted work as a whole." § 107(3). In general, the greater the amount of use, the more likely the copying is unfair. *Ibid.* But even if the copier takes only a small amount, copying the " 'heart' " or "focal points" of a work weighs against fair use, *Harper*, 471 U.S., at 565–566, unless " 'no more was taken than necessary' " for the copier to achieve transformative use, *Campbell*, 510 U.S., at 589.

Google does not dispute the Federal Circuit's conclusion that it copied the heart or focal points of Oracle's work. 886 F.3d at 1207. The declaring code is what attracted programmers to the Java platform and why Google was so interested in that code. And Google copied that code "verbatim," which weighs against fair use. *Harper*, 471 U.S., at 565. The majority does not disagree. Instead, it concludes that Google took no more than necessary to create new products. That analysis fails because Google's use is not transformative. *Campbell*, 510 U.S., at 586 (recognizing that this fourth factor "will harken back to the [purpose-and-character] statutory facto[r]"). This factor thus weighs against Google.

Even if Google's use were transformative, the majority is wrong to conclude that Google copied only a small portion of the original work. The majority points out that the 11,500 lines of declaring code—enough to fill about 600 pages in an appendix, Tr. of Oral Arg. 57—were just a fraction of the code in the Java platform. But the proper denominator is *declaring code*, not all code. A copied work is quantitatively substantial if it could "serve as a market substitute for the original" work or "potentially licensed derivatives" of that work. *Campbell*, 510 U.S., at 587. The declaring code is what attracted programmers. And it is what made Android a "market substitute" for "potentially licensed derivatives" of

[11] Because the majority's reasoning would undermine copyright protection for so many products long understood to be protected, I understand the majority's holding as a good-for-declaring-code-only precedent.

Oracle's Java platform. Google's copying was both qualitatively and quantitatively substantial.

* * *

In sum, three of the four statutory fair-use factors weigh decidedly against Google. The nature of the copyrighted work—the sole factor possibly favoring Google—cannot by itself support a determination of fair use because holding otherwise would improperly override Congress' determination that declaring code is copyrightable.

IV

The majority purports to save for another day the question whether declaring code is copyrightable. The only apparent reason for doing so is because the majority cannot square its fundamentally flawed fair-use analysis with a finding that declaring code is copyrightable. The majority has used fair use to eviscerate Congress' considered policy judgment. I respectfully dissent.