SUPREME COURT OF THE UNITED STATES

No. 18–956

GOOGLE LLC, PETITIONER *v.*
ORACLE AMERICA, INC.

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

[April 5, 2021]

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 copy-

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

rightability 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.

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.

---

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

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

---

```
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)."

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 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 22. 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.

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

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 15. 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 18. 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 circum-

stance 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.[3] 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

---

[3] I would not, however, definitively resolve Google's argument that the Seventh Amendment commits the question of fair use to a jury. I tend to agree with the Court that fair use was not "itself necessarily a jury issue" when the Constitution was ratified. *Markman* v. *Westview Instruments*, *Inc.*, 517 U. S. 370, 376–377 (1996). Google cites cases about "fair abridgment," but Congress has since made clear that copyright holders have "exclusive rights" over any "abridgment." 17 U. S. C. §§101, 106. And in any event, judges often declined to refer these issues to juries. See, *e.g.*, *Gyles* v. *Wilcox*, 2 Atk. 141, 144, 26 Eng. Rep. 489, 490–491 (Ch. 1740); *Folsom* v. *Marsh*, 9 F. Cas. 342, 345–349 (No. 4,901) (CC Mass. 1841) (Story, J). Still, we should not so casually decide this question when the parties barely addressed it.

[4] The factors are: "(1) the purpose and character of the use, including whether such use is of a commercial nature or is for nonprofit educational purposes; (2) the nature of the copyrighted work; (3) the amount and substantiality of the portion used in relation to the copyrighted work as a whole; and (4) the effect of the use upon the potential market for or value of the copyrighted work." §§107(1)–(4).

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 "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 23–24.

---

[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'").

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 24. 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 23 (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 23–24. 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 22. 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

---

[6]The majority also belittles declaring code by suggesting it is simply a way to organize implementing code. *Ante*, at 22–23. 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 5. Declaring code *creates* the method calls.

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 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 22–23, 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 33, 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 31–32. 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 34. 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] 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

---

[7] It also suggests that Oracle may have received some incidental benefit from Android. *Ante*, at 32–33. 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.

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 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 34–35. 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.

The majority attempts to dismiss this overwhelming commercial use by noting that commercial use does "not necessarily" weigh against fair use. *Ante*, at 27. 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

---

[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 copyright holder's market.

Regardless, Google fairs 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 25. 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 26 (Google's copying "can further the development of computer programs").

---

[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).

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 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.

---

[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.

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 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.[12]

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.

[12] To be sure, these factors are not necessarily exclusive, but they are "especially relevant," *Harper*, 471 U. S., at 560; the majority identifies no other relevant factors; and I can think of none that could overcome the overwhelming weight of these key factors.