On the WLF Legal Pulse

Timely commentary from WLF's blog

Washington Legal Foundation
Advocate for Freedom and Justice®
2009 Massachusetts Avenue, NW
Washington, DC 20036
202.588.0302 wlf.org

May 7, 2021

Google v. Oracle: A Copyrightability Decision Masquerading as Fair Use

by Devlin Hartline and Adam Mossoff

In what many have called the copyright case of the decade, the Supreme Court handed down its opinion in *Google v. Oracle* last month. The 6-2 decision by Justice Stephen Breyer was an unmitigated win for Google, as the Court held that Google's undisputed copying of nearly 11,500 lines of computer code to help launch its Android smartphone platform was "fair use." Justice Clarence Thomas, joined by Justice Samuel Alito, dissented, maintaining that Google's deliberate, commercial copying of Oracle's computer code did not qualify as "fair use."

The decision was surprising for numerous reasons. The impression many had after the oral argument was that Google was going to lose. The actual opinion was similarly surprising, as Justice Breyer skipped over the first question on the copyrightability of the computer code and addressed only the fair use question. We cannot speak to every issue raised by the *Google* decision in a brief essay, and so we will address the core of Justice Breyer's fair use analysis, explaining that it is a copyrightability decision masquerading as fair use.

As noted, the case arose from Google's undisputed copying of approximately 11,500 lines of computer code in the Java program created and owned by Sun Microsystems (a company Oracle later purchased). Google was unwilling to pay for one of the three paid licensing options, nor did it want the free, open-source licensing option because it planned to make its version of Java proprietary. After years of negotiating for a license, Google opted instead to copy the code to more quickly develop, release, and promote its new Android smartphone platform.

In short, Google engaged in the well-known practice of predatory infringement (what policy wonks call "efficient infringement"). Google decided that the commercial upside of copying the code outweighed the potential downside of being held liable for this infringement. As Justice Thomas pointed out in his dissent, it was a gamble that paid handsomely for Google, earning it tens of billions of dollars in revenue from the quick market adoption of its Android smartphone, while simultaneously destroying Oracle's market for licensing Java. The end result is that Oracle was forced to subsidize Google's commercial development of its own proprietary version of Java and the success of its Android smartphone—what the intellectual property laws forbid.

Devlin Hartline is an Assistant Professor of Law at Scalia Law and Director of Communications at the Center for the Protection of Intellectual Property. **Adam Mossoff** is Professor of Law at Antonin Scalia Law School, George Mason University, and Chair of the Forum for Intellectual Property and a Senior Fellow at the Hudson Institute.

In the decade-long lawsuit, Google now defended itself by arguing that the Java code was not copyrightable, or, in the alternative, that its copying of this code was "fair use." Justice Breyer's majority opinion skipped the copyrightability issue altogether, despite it being central to the analysis, and purported to presume that Oracle's computer code is copyrightable. Justice Thomas justifiably criticized the majority for bypassing a necessary, legal predicate: "By skipping over the copyrightability question, the majority disregards half the relevant statutory text and distorts its fair-use analysis." And he rightly pointed out that the majority's "fair-use analysis is wholly inconsistent with the substantial protection Congress gave to computer code."

The skipping of the copyrightability question is revealing, if only because much of the oral argument last October focused on this issue given that Google's arguments strained credibility. The text of the Copyright Software Protection Act of 1980 is clear: all code is copyrightable, and it is protected as such. In fact, Congress enacted this legislation precisely because courts had split on this issue in the 1960s and 1970s, and Congress resolved this judicial conflict by definitively stating that code is protectable expression under the Copyright Act. If Justice Breyer had addressed this issue, as opposed to assuming it for "the sake of argument," it would have required him to give full protection to the code, as Justice Thomas explained.

Justice Breyer's fair use analysis reveals a deep hostility to the idea that computer code is copyrightable. He adopted the functional approach urged by Google that ignores the originality of Oracle's expression and conflates it with the unprotected idea of the result that it brings about. This move allowed Justice Breyer to bring copyrightability in through the back door, as he limited the protectability of the elements that were copied by reasoning that they were allegedly different from other types of copyrighted code or text that receive full copyright protection. Justice Breyer then adopted a surprisingly broad view of "transformative" use that justifies copying as a fair use and conversely an unduly constricted view of substantiality and market harm that serve to cabin in fair use doctrine. In sum, Justice Breyer derisively dismissed the original, protectable expression in the code that accounted for its massive technical and commercial success, and thus he was able to assert that it deserved less protection under his unprecedented, massively expansive notion of "fair use."

Justice Breyer tipped his hand early in his fair use analysis by starting with the second fair use factor, which considers the nature of the copyrighted work and whether it lies close to the core of copyright protection because of its creative expression. Under factor two of the fair use analysis, Justice Breyer was able to do what he could not get away with under the straightforward copyrightability analysis of the Computer Software Protection Act: diminish to an infinitesimal point the amount of protection he was willing to afford to the stolen Java code.

Relying on the First Circuit's opinion in *Lotus v. Borland*, Justice Breyer began his analysis by characterizing Oracle's Java code merely as a "user interface" that allows programmers to control computer tasks "via a series of menu commands." The cite to *Lotus* is the classic tell in poker: this case was specifically about copyrightability, not fair use. In *Lotus*, the First Circuit held that the now-ubiquitous drop-down menu in the then-famous Lotus 1-2-3 spreadsheet program was an unprotectible "method of operation" since it was "the means by which a person operates . . . a computer." Even more important, *Lotus* is inapplicable to *Google* both as a matter of fact and law: Borland did copy Lotus' hierarchy of commands in the drop-down menu, but Borland did *not* copy any of the underlying computer code. Borland wrote its own code. In fact, it did not need to copy the code to recreate the menu hierarchy.

The comparison made by Justice Breyer to an uncopyrightable menu hierarchy is significant, as he used it to differentiate between the declaring code from the implementing code in Oracle's Java program. (The declaring code is the software code directly used by programmers, whereas the implementing code is the software code that actually operates the computer.) The declaring code, Justice Breyer stated, is different "from many other kinds of copyrightable computer code," like the implementing code, because it is "inextricably bound" with the idea of dividing and organizing tasks and commands; in sum, according to Justice Breyer, the declaring code is more like the unprotectable menu hierarchy in *Lotus* because it cannot be separated from the functionality that it creates.

None of this makes much sense if Justice Breyer was truly presuming copyrightability. All creative expression is "inextricably bound" with the idea that it expresses. As Justice Thomas pointed out, a copyrighted novel is "inextricably bound" with the abstract ideas of a protagonist, plot, chapter divisions, etc., and yet courts have long recognized the valuable text of a novel is fully protected by the copyright laws. Equally important, courts have easily been able to discern for centuries that the undisputed copying of thousands of lines of text in a novel is copyright infringement.

Justice Breyer then shifted gears and distinguished between the declaring and implementing codes. The creativity behind the implementing code is like "magic," Justice Breyer explained, because it is dictated by things like "how quickly a computer can execute a task or the likely size of the computer's memory." Oracle's declaring code, on the other hand, "embodies a different kind of creativity" because its creation was influenced by considerations of making it "intuitively easy to remember" in order to "attract programmers." Of course, the user-centricity of the declaring code embodies the creative choices of the Java developers just as does its implementing code, and, if anything, it embodies *more* creativity as it was not limited by the mechanical considerations of the implementing code.

Thus, Justice Breyer harped on the attractiveness of Oracle's declaring code to programmers while downplaying the significant, creative efforts that went into making this declaring code of value to programmers in taking the time and effort to learn in the first place. Moreover, he makes no mention of the fact that, as with the implementing code, these programmers do not memorize or even type the declaring code. They simply structure their inputs to comply with the formatting that the declaring code demands; it is the simplicity of calling up the methods that makes the declaring code valuable. Curiously, Justice Breyer also states that the declaring code is also "inextricably bound up with the implementing code, which is copyrightable but was not copied." But if the implementing code is protected expression, then so is the declaring code, for it is only when both work together that the method performs the particular function. Both are "inextricably bound" with the uncopyrightable ideas of task division and organization. The differences identified by Justice Breyer between the two types of code at work in Java are a distinction without a difference, and he erroneously conflates the functionality of the declaring code with its expression.

This is a classic case where, as the Supreme Court put it in *Campbell*, the copyist merely used the expression "to get attention or to avoid the drudgery in working up something fresh." Having expressed serious doubts about the copyrightability of Oracle's computer code and concluded that at best it should receive minimal protection, Justice Breyer went through the rest of the fair use factors in an almost perfunctory fashion, ignoring many longstanding precedents (or directly contradicting them). He argued that Google's copying is "transformative" (despite the copying of the code for the exact same use), insubstantial (despite it being the heart of Oracle's work), and unharmful (despite it being used in a directly competing product that generated billions of dollars

in revenues while causing billions of dollars in losses for the owner of the copyrighted work).

The expansive application of fair use in this case is nothing short of breathtaking. The *Google* Court allows a commercial firm to copy a copyrighted work in a commercial, proprietary product that competes commercially with the work of the original copyright-owner. After this case, it is next to impossible to see how any unauthorized copying of an API program like Java will not be immunized from a charge of copyright infringement by qualifying as fair use. To wit, *Google* is a copyrightability decision masquerading as fair use.

Perhaps the best thing that can be said is that the decision, by its own terms, has very limited reach. Justice Breyer expressly noted that "We do not overturn or modify our earlier cases involving fair use." He had to say this, if only because one cannot read this opinion without trying to square the circle of all the contradictions with prior fair use decisions, such as the Supreme Court's 1985 decision in *Harper & Row v. The Nation Enterprises*.

Even with this express limitation, the opinion still creates a messy goulash of copyright law and policy. It claims to presume copyrightability while showing in its analysis that it does not; it is inconsistent with existing precedent but then says it is not overturning these prior decisions; it paints a false dichotomy between copyrightability and innovation while ignoring the importance of licensing of copyrighted works as a driver of innovation; and it pretends like Google would not have been able to further the goal of promoting new expression without copying Oracle's work.

One wonders if this decision might be Justice Breyer's swansong on copyrightability and fair use. If so, it certainly seems to reflect his long-held policy view, expressed over fifty years ago in then-law professor Breyer's famous 1970 article in the *Harvard Law Review*: "Computer programs should not receive copyright protection at the present time."