Northwestern Journal of Technology and Intellectual Property

Volume 18 | Issue 2 Article 2

3-2021

REBOOTING BAKER V. SELDEN IN ORACLE V. GOOGLE

Ann Defranco
Northwestern Pritzker School of Law

NORTHWESTERN JOURNAL OF TECHNOLOGY AND INTELLECTUAL PROPERTY

REBOOTING BAKER V. SELDEN IN ORACLE V. GOOGLE

Ann Defranco

March 2021 VOL. 18, NO. 2

Volume 18, Number 2 (2021)
Northwestern Journal of Technology and Intellectual Property

REBOOTING BAKER V. SELDEN IN ORACLE V. GOOGLE

Ann Defranco

ABSTRACT— With the Supreme Court poised to rule on *Oracle v. Google*, the long-running suite of cases involving the copyrightability and fair use of a software interface called an API, the case typifies the difficult fit of copyright protection to software. This Note takes a close look at the code at issue and argues that the nature of software innovation is better suited to patent protection: object-oriented code, such as the Java language at issue in this case, evolves through a process of copying and tweaking, or in coding terms, modularity, abstraction, and inheritance. Thus, an IP regime which allows for such evolution (namely, patent) encourages such innovation whereas copyright, with its broad exclusive rights over derivative works, does not. The ill fit of the copyright regime is also exemplified by the carving out of copyright-free ("copyleft") spaces where coders and software innovation thrive. Nor do the concerns motivating copyright protection of protecting creative expression make sense in software development, where the goals are efficiency, productivity, and readability of the code. Thus, the Supreme Court should return to the principles of *Baker v. Selden* and plant the boundary marker keeping § 102(b) functionality on the patent side of copyright-patent boundary.

INTRODUCTION

In the last twenty years, advances in computer interoperability have given rise to a thriving API economy.[1] APIs (application programming interfaces) enable the owner of a digital asset in the form of data or services to provide access to that asset to a third party.[2] This creates opportunities for a variety of symbiotic relationships, such as consumer electronics manufacturer Samsung attracting buyers by adding a Netflix app to its "smart" televisions, which in turn grows Netflix's subscriber base, or Netflix outsourcing its data management to cloud service provider Amazon Web Services, or a start-up ride-sharing company like Lyft using Google Maps for navigation, Twilio for sign-up verification, and Stripe for payment.[3] Thus, the software-to-software interoperability of APIs enables broader access to customers, more efficient allocation of resources, and innovation enabled by third-party digital assets.

This API economy relies on computer interoperability. Copyright issues relating to the computer code that enables this type of functionality may have a tremendous impact on this industry if they create friction in the flow of information through these technology gates.[4] At issue in the *Oracle v. Google* suite of cases is whether an API is copyright-protected as an original expression, or whether it is a method of operation that is excluded from copyright protection.[5] Because APIs in use today rely on standardized

---

[1] Wendell Santos, *APIs Show Faster Growth Rate in 2019 than Previous Years*, PROGRAMMABLEWEB (July 17, 2019), https://www.programmableweb.com/news/apis-show-faster-growth-rate-2019-previous-years/research/2019/07/17 [https://perma.cc/TQK9-8LKJ].

[2] Michael Endler, *How API Management Accelerates Digital Business*, MEDIUM (Sept. 18, 2017), https://medium.com/apis-and-digital-transformation/how-api-management-accelerates-digital-business-4ccea9b302df [https://perma.cc/XZ6E-JUSP].

[3] Thomas H. Davenport & Bala Iyer, *Move Beyond Enterprise IT to an API Strategy*, HARV. BUS. REV. (Aug. 6, 2013), https://hbr.org/2013/08/move-beyond-enterprise-it-to-a [https://perma.cc/JK94-ZQCE]; *Netflix on AWS*, AWS, https://aws.amazon.com/solutions/case-studies/netflix/ [https://perma.cc/QTG6-8TMP]; Richard Yao, *What Is the "API Economy" and How Brands Can Benefit from It*, MEDIUM (May 31, 2018), https://medium.com/ipg-media-lab/what-is-the-api-economy-and-how-brands-can-benefit-from-it-b46210d0434d [https://perma.cc/A9ZX-Q545].

[4] Timothy B. Lee, *Google Asks Supreme Court to Overrule Disastrous Ruling on API Copyrights*, ARS TECHNICA (Jan. 25, 2019, 11:12 AM), https://arstechnica.com/tech-policy/2019/01/google-asks-supreme-court-to-overrule-disastrous-ruling-on-api-copyrights/ [https://web.archive.org/web/20190125194559/https://arstechnica.com/tech-policy/2019/01/google-asks-supreme-court-to-overrule-disastrous-ruling-on-api-copyrights/].

[5] Oracle Am., Inc. v. Google Inc., 872 F. Supp. 2d 974, 1002 (N.D. Cal. 2012), *rev'd*, 750 F.3d 1339 (Fed. Cir. 2014); Oracle Am., Inc. v. Google Inc., 750 F.3d 1339, 1381 (Fed. Cir. 2014); Oracle Am., Inc. v. Google Inc., No. C 10-03561, 2016 WL 3181206, at *13 (N.D. Cal. June 8, 2016), *rev'd sub nom*. Oracle Am., Inc. v. Google LLC, 886 F.3d 1179 (Fed. Cir. 2018), *cert. granted*, 140 S. Ct. 520 (2019);

software protocols that were assumed to be free of copyright protection, if Oracle's Java API is found to be protected by copyright, this may force companies to create all new proprietary APIs to avoid an onslaught of copyright infringement lawsuits, resulting in a massive overhaul of existing code.[6]

Computer programs "hover even more closely to the elusive boundary" between copyrightable expression and uncopyrightable ideas that Judge Learned Hand despaired of finding in *Nichols*.[7] Though computer programs are primarily functional sets of machine instructions, courts have found computer programs may contain copyrightable creative expression.[8] Such creativity is found in how a method or functionality is expressed, while the method or functionality itself is excluded from copyright protection.[9] For example, I can describe a method for estimating the mathematical constant pi as, "First, drop 427 one-inch pink needles onto a plane ruled with parallel purple lines spaced two inches apart. Second, divide 427 by the number of needles which intersect any of the purple lines."[10] Whatever "modicum of creativity" I may own in those sentences, I cannot exclude anyone from performing the procedure it describes.[11] This is the logical descendant of *Baker v. Selden*: a written expression may be copyright-protected, but the idea or functionality which it expresses may not.[12]

Because computer programs may have copyrightable elements, there is a risk of inadvertently creating a long-term monopoly on a functionality by declaring a functional or useful element to be copyright protectable.[13] This

Oracle Am., Inc. v. Google LLC, 886 F.3d 1179, 1211 (Fed. Cir. 2018), *cert. granted*, 140 S. Ct. 520 (2019).

[6] Lee, *supra* note 4.

[7] Comput. Assocs. Int'l, Inc. v. Altai, Inc., 982 F.2d 693, 704 (2d Cir. 1992); Nichols v. Universal Pictures Corp., 45 F.2d 119, 121 (2d Cir. 1930).

[8] 17 U.S.C. § 101; NAT'L COMM'N ON NEW TECH. USES OF COPYRIGHTED WORKS, FINAL REPORT OF THE NATIONAL COMMISSION ON NEW TECHNOLOGICAL USES OF COPYRIGHTED WORKS 9–10 (1978) [hereinafter CONTU], http://digital-law-online.info/CONTU/PDF/Chapter3.pdf [https://perma.cc/8MQW-DXWJ]; *see Altai*, 982 F.2d at 702.

[9] CONTU, *supra* note 8, at 19–20 (citing 1 MELVILLE B. NIMMER & DAVID NIMMER, NIMMER ON COPYRIGHT § 37.83 (1976)); *see also* Baker v. Selden, 101 U.S. 99 (1879).

[10] This is Buffon's Needle problem. The number of needles (427) is arbitrary.

[11] Feist Publ'ns, Inc. v. Rural Tel. Serv. Co., 499 U.S. 340, 346–47 (1991).

[12] *Baker*, 101 U.S. at 103.

[13] *Feist Publ'ns, Inc.*, 499 U.S. at 346–47; 17 U.S.C. § 102; Satava v. Lowry, 323 F.3d 805, 812 n.5 (9th Cir. 2003); Pamela Samuelson, *Strategies for Discerning the Boundaries of Copyright and Patent Protections*, 92 NOTRE DAME L. REV. 1493, 1495 (2017); *see also Baker*, 101 U.S. at 102 ("To give to the author of the book an exclusive property in the art described therein, when no examination of its novelty has ever been officially made, would be a surprise and a fraud upon the public."); Sony Comput. Ent., Inc. v. Connectix Corp., 203 F.3d 596, 605 (9th Cir. 2000) ("If Sony wishes to obtain a lawful monopoly on the functional concepts in its software, it must satisfy the more stringent standards of the patent laws.").

risk is heightened by the fact that copyright protection is much faster and easier to obtain than patent protection.[14] Moreover, patents have a much shorter term to encourage more rapid turnover of ideas and are limited in scope to encourage innovation.[15] All these factors create an incentive for software developers to seek copyright protection instead of patent protection.[16]

However, patent law is a better fit for software than copyright because of how innovation in software development occurs.[17] New software is often created by copying and adapting or improving already existing software.[18] Thus, software's incremental innovation is incentivized by patent law but discouraged by copyright's broad protections concerning derivative works.[19] Therefore, an ill-placed boundary runs the risk of impeding software innovation, the consequences of which may be amplified in the $4.06 trillion software development industry.[20]

In overturning the 2012 decision in *Oracle v. Google*, the Federal Circuit prioritized original expression over functionality in finding that the "structure, sequence, and organization" of the thirty-seven packages of the Java API are protected by copyright.[21] This is inapt for the practice of computer programming for two reasons: first, creativity in programming takes the form of creative problem-solving rather than original or creative expression, and second, creative expression itself is antithetical to coding practice.

The dueling 2012 and 2014 *Oracle v. Google* rulings exemplify the circuit split concerning how copyright law, or specifically § 102(b) of the Copyright Act, should be applied to nonliteral elements of computer programs. One side of the split recognizes that under § 102(b), copyright should not protect processes or methods of operation. This is the side where the 2012 ruling falls, holding that while a system or method of command may be creative and original, it "does not change its character as a method

---

[14] Samuelson, *supra* note 13, at 1495.

[15] *Id*.

[16] *Id*.

[17] Dennis S. Karjala, *Distinguishing Patent and Copyright Subject Matter*, 35 CONN. L. REV. 439, 453 (2003).

[18] Clark D. Asay, *Software's Copyright Anticommons*, 66 EMORY L.J. 265, 281 (2017).

[19] Karjala, *supra* note 17, at 453–54 (functional works amenable to "incremental improvement" would constitute infringement under copyright's substantial similarity test).

[20] As of November 12, 2020. *Software*, FIDELITY (Nov. 12, 2020), https://eresearch.fidelity.com/eresearch/markets_sectors/sectors/industries.jhtml?tab=learn&industry=451030 [https://perma.cc/AT5Y-YMN7].

[21] Oracle Am., Inc. v. Google Inc., 750 F.3d 1339, 1381 (Fed. Cir. 2014).

of operation."[22] The other side of the split, where the 2014 decision falls, says that the structure of the computer program elements are expressions that embody creative choices that are separable from their functionality, thus falling under § 102(a)'s scope of protection, protection that is not to be denied by § 102(b).[23]

Because copyright law has an uncomfortable fit with computer programs, any analysis requires understanding how programming languages and coding practice work. The analysis should also examine the coding culture that uses copyright law to protect the right *to* copy rather than protecting *against* it. Part I of this Note discusses how traditional copyright doctrine has been applied in computer program infringement cases and how conflicting interpretations of § 102(b) of the Copyright Act have led to a circuit split. Part II provides a brief technical primer on the Java API and on APIs generally. Part III explains how coding practices point to patent protection as being better suited for software innovation than copyright. Part IV examines the 2012 and 2014 *Oracle v. Google* decisions in light of these considerations.

This Note concludes there may be no right or wrong application of copyright doctrine to computer programs. In that case, this Note recommends practical factors that courts should consider in deciding how copyright law should be applied to these highly utilitarian literary works to protect the vital and thriving software industry.

I. COPYRIGHT LAW AND COMPUTER PROGRAMS

The *Baker v. Selden* decision of 1879 is a landmark on the boundary between copyright and patent, holding that while the expression of an idea or useful article is protected by copyright, the idea itself is not.[24] The 1976 Copyright Act defined computer programs to be "literary works" and codified the idea-expression dichotomy in § 102(b), distinguishing ideas, processes, and methods of operation as uncopyrightable.[25] The 1978 Commission on New Technological Uses of Copyrighted Works (CONTU) report recognized that fixing the boundary between the copyrightable program and the uncopyrightable process would be difficult, and leaving it

[22] Oracle Am., Inc. v. Google Inc., 872 F. Supp. 2d 974, 999–1000 (N.D. Cal. 2012), *rev'd*, 750 F.3d 1339 (Fed. Cir. 2014).

[23] *Oracle*, 750 F.3d at 1368, 1369 ("[W]e conclude that Section 102(b) does not bar the [computer program] packages from copyright protection just because they also perform functions."). The court also found literal copying of the program elements. *Id.* at 1356.

[24] Baker v. Selden, 101 U.S. 99, 105 (1879).

[25] *See* H.R. REP. NO. 94-1476, at 54, 56–57 (1976), *as reprinted in* 1976 U.S.C.C.A.N. 5659, 5667, 5670.

to the judiciary to determine on a "case-by-case basis."[26] Now, forty-two years after the CONTU report and 142 years after *Baker v. Selden*, the Supreme Court has granted certiorari on the issues of copyrightability of the software and fair use in *Oracle v. Google*, perhaps to plant a new marker on the copyright-patent boundary.[27]

A. Copyright Doctrines Applied to Computer Programs

The 1992 *Altai* decision introduced the application of the abstraction-filtration-comparison (AFC) test of substantial similarity to determine whether the copyright on elements of a computer program had been infringed.[28] The legal principle of abstraction recognizes that a higher-level function "conceptually replaces" the implementations of lower-level modules of code, and this type of replacement is repeated at progressively higher levels of abstraction until one reaches the ultimate function of the program at the highest level.[29] At each level of abstraction, the AFC test filters out from copyright protection elements of code whose design is dictated by efficiency and by external factors along with elements taken from the public domain.[30] Thus, the AFC test implements two traditional copyright doctrines: efficient coding gives rise to merger, and external requirements are the programming world's version of *scene à faire*.[31]

In computer programs, merger occurs when specific parts of a code "are the only and essential means of accomplishing a given task," and "their later use [is not] infringement."[32] However, if there are alternative ways to complete the task, then merger does not apply.[33] An example of merger can be seen in a sample of the Java implementation codes created by Sun developers and independently recreated by Google developers.[34]

---

[26] CONTU, *supra* note 8, at 22–23.

[27] Google LLC v. Oracle Am., Inc., 140 S. Ct. 520 (2019).

[28] Comput. Assocs. Int'l, Inc. v. Altai, Inc., 982 F.2d 693, 706 (2d Cir. 1992).

[29] *Id.* at 706–07 (first quoting Nichols v. Universal Pictures Corp., 45 F.2d 119, 121 (2d Cir. 1930); and then quoting Steven R. Englund, Note, *Idea, Process, or Protected Expression?: Determining the Scope of Copyright Protection of the Structure of Computer Programs*, 88 MICH. L. REV. 866, 867–73 (1990)).

[30] *Id.* at 707–11.

[31] *Id.*

[32] CONTU, *supra* note 8, at 20.

[33] *Id.*

[34] Oracle Am., Inc. v. Google, Inc., 872 F. Supp. 2d 974, 978 (N.D. Cal. 2012), *rev'd*, 750 F.3d 1339 (Fed. Cir. 2014). The implementation code is not at issue in *Oracle v. Google*.

```
237:   /**
238:    * Return whichever argument is larger.
239:    *
240:    * @param a the first number
241:    * @param b a second number
242:    * @return the larger of the two numbers
243:    */
244:   public static int max(int a, int b)
245:   {
246:     return (a > b) ? a : b;
247:   }
```

Fig. 1. Java method definition for java.lang.Math.max().[35]

```
1274    /**
1275     * Returns the greater of two {@code int} values. That is, the
1276     * result is the argument closer to the value of
1277     * {@link Integer#MAX_VALUE}. If the arguments have the same value,
1278     * the result is that same value.
1279     *
1280     * @param   a   an argument.
1281     * @param   b   another argument.
1282     * @return  the larger of {@code a} and {@code b}.
1283     */
1284    public static int max(int a, int b) {
1285        return (a >= b) ? a : b;
1286    }
```

Fig. 2. Android method definition for java.lang.Math.max().[36]

[35] *Source for java.lang.Math*, GNU CLASSPATH, http://developer.classpath.org/doc/java/lang/Math-source.html [https://perma.cc/PLG7-JHCP]. Lines 237–243 are non-functional comments. Line 244 contains the "header" or source code declaration of the method. Note that "public," "static," and "int" are *keywords* with specifically defined uses in Java and many other programming languages. *See Oracle Am., Inc.*, 872 F. Supp. 2d at 979, 981. Lines 245–247 are the implementation code which performs the computational task. Note that implementation code in a Java method definition is always encased within curly brackets.

[36] *Math.java*, GOOGLE GIT, https://android.googlesource.com/platform/libcore/+/refs/heads/master/ojluni/src/main/java/java/lang/Math.java [https://perma.cc/QSA7-V6GK]. Lines 1274–1283 are non-functional comments. Line 1284 is the "header" or source code declaration of the method up to the "{" separator character. The implementation code of the method begins with the "{" on line 1284 and runs through line 1286.

Although line 246 of Figure 1 and line 1285 of Figure 2 were created independently, they are nearly identical.[37] They are the most succinct and efficient way to code the process of finding the greater of two given variables. Under *Altai*, therefore, these lines of code have merged with that process.[38]

Scene à faire in traditional literary works excludes common storyline elements or stock literary devices from copyright protection.[39] In computer programs, *scene à faire* takes the form of external factors such as compatibility requirements, software development standards, and "widely accepted programming practices"—things that are necessary or fundamental to the utility of the code.[40] For example, the *Altai* decision notes that "compatibility requirements of other programs with which a program is designed to operate in conjunction" may "circumscribe a programmer's freedom of design choice."[41] Subsequent to the *Altai* decision, the *Gates Rubber* decision, in applying the AFC test in an infringement case, suggested that *scene à faire* may also include programming elements necessary for interfacing.[42]

B. The Circuit Split

The *Oracle v. Google* cases involve computer interfacing or interoperability, or the means by which a computer code interacts with other programs.[43] A survey of the most important copyright cases involving nonliteral copying of computer programs is an odyssey that reveals a fracture in how § 102(b) has been interpreted or applied to computer interoperability or interface issues.

One set of decisions found that elements of a computer program that are necessary for interoperability should be excluded from copyright protection due to their functional nature.[44] Of these, the 1995 *Lotus v.*

---

[37] The choice of ">" or "≥" is functionally immaterial.

[38] *See* Comput. Assocs. Int'l, Inc. v. Altai, Inc., 982 F.2d 693, 708 (2d Cir. 1992) (the more efficient the code, the more closely the code approximates the idea).

[39] Hoehling v. Universal City Studios, Inc., 618 F.2d 972, 979 (2d Cir. 1980).

[40] *Altai*, 982 F.2d at 710 (citing 3 MELVILLE B. NIMMER & DAVID NIMMER, NIMMER ON COPYRIGHT § 13.03[F][3] (1991)).

[41] *Id.* at 709–10.

[42] Gates Rubber Co. v. Bando Chem. Indus., Ltd., 9 F.3d 823, 838 & n.14 (10th Cir. 1993) (*scene à faire* exclusions in the filtration step of the *Altai* abstraction-filtration-comparison test may include computer interfacing functionality).

[43] Oracle Am., Inc. v. Google Inc., 872 F. Supp. 2d 974, 975 (N.D. Cal. 2012), *rev'd*, 750 F.3d 1339 (Fed. Cir. 2014).

[44] Sega Enters. Ltd. v. Accolade, Inc., 977 F.2d 1510, 1526 (9th Cir. 1992) (copyright protection of interface procedures which enable compatibility would give the copyright owner a *de facto* monopoly over functionality in violation of § 102(b)); Lotus Dev. Corp. v. Borland Int'l, Inc., 49 F.3d 807, 815 (1st

Borland decision by the First Circuit deals with user interface elements which have a creative structure or taxonomy.[45] The *Lotus* decision found that although the original creator, Lotus, had made creative or "expressive" choices in the design of the software menu command hierarchy, it was nevertheless an uncopyrightable "method of operation."[46] Ultimately, the Supreme Court's 4-4 ruling failed to dislodge the finding that such a functional taxonomy was not copyright protected.[47] Notably, in his concurring opinion in *Lotus*, Judge Boudin recognized that although the "form" of computer programs is text-based like traditional literary works, their "substance" is more akin to the subject matter of patents.[48]

Another set of decisions found that when nonliteral elements of computer programs are the result of creative choices by developers, those elements should be protected by copyright.[49] These cases involved user or software interfacing and turned on whether the original program designers made creative choices in designing the interfaces. Of note in this group is the precedent set by the Ninth Circuit's decision in *Johnson Controls* which found that the "structure, sequence and organization and user interface" of the program at issue were expressive choices on the basis that these nonliteral components were "customized to the needs of the purchaser."[50] Thus, this latter set of decisions prioritizes the creative choices that occur in the creation of the literary works over the dominant functional nature particular to computer programs. But, as noted IP scholar Professor Pamela Samuelson argues, conventional literary works such as novels and plays have no functional nature that must be teased out, making this problem unique to utilitarian yet creative works such as computer programs.[51] Thus, when courts apply copyright doctrine to utilitarian works just as they do to

Cir. 1995), *aff'd*, 516 U.S. 233 (1996) (a hierarchical menu structure is functional and therefore not copyright protectable); *Gates*, 9 F.3d at 838 & n.14 (*scene à faire* exclusions in the filtration step of the *Altai* abstraction-filtration-comparison test may include computer interfacing functionality); Lexmark Int'l, Inc. v. Static Control Components, Inc., 387 F.3d 522, 543 (6th Cir. 2004) (program to produce a unique unlock code for interoperability is not copyright-protected due to merger).

[45] *Lotus*, 49 F.3d at 815 (a hierarchical menu structure is functional and therefore not copyright protectable).

[46] *Oracle*, 872 F. Supp. 2d at 991 (citing *Lotus*, 49 F.3d at 815).

[47] *Lotus*, 49 F.3d at 819.

[48] *Id.* at 820 (Boudin, J., concurring).

[49] Atari Games Corp. v. Nintendo of Am. Inc., 975 F.2d 832, 840 (Fed. Cir. 1992) (creative organization and sequencing of code designed to unlock gaming console was arbitrary and creative in design and therefore is copyright protectable); Johnson Controls, Inc. v. Phoenix Control Sys., Inc., 886 F.2d 1173, 1175–76 (9th Cir. 1989) (the adaptability of a computer program to the specific needs of each customer indicates "individualized expression" and is thus copyright protectable).

[50] *Johnson Controls*, 886 F.2d at 1175–76.

[51] Pamela Samuelson, *Functionality and Expression in Computer Programs: Refining the Tests for Software Copyright Infringement*, 31 BERKELEY TECH. L.J. 1215, 1272 (2016).

conventional literature, this overlooks the "deeply functional nature of programs."[52] When it comes to computer programs, perhaps copyright protection should be the exception rather than the rule.[53]

II. THE JAVA LANGUAGE AND JAVA API STRUCTURE

To understand the key copyright issue in *Oracle v. Google*, it is helpful to first look at what Google is *not* accused of infringing.

In 2005, Google chose Java to be the primary coding language for its newly acquired Android software development platform for mobile devices.[54] By 2004, Java had come to be the most popular programming language in the U.S.[55] Google was free to use Java because, as everyone agrees, the Java programming language is neither patent nor copyright protected.[56] However, given the memory constraints of mobile devices, Google wanted to limit its support of Java within Android to just an essential subset of its functionality, that is, to just thirty-seven of the 166 packages of code that comprise the Java language.[57] Oracle disagreed. By not fully supporting Java, Android would violate Sun/Oracle's design philosophy of platform independence behind Java: "Write Once, Run Anywhere."[58] This meant Java programs written for Android would not run everywhere, nor would Java programs written for other platforms necessarily run on Android. In essence, Google's limited implementation of Java "forked" the Java development community.[59]

Java packages of code provide the functionality by which Java programmers can build new, more complex computer programs without having to re-write everything from scratch. The Java language began in 1996 with just eight packages; by 2008 there were 166 packages available to Java

---

[52] *Id.*

[53] *Lotus*, 49 F.3d at 820 (Boudin, J., concurring) (IP issues around computer programs are more related to patent law except "in those rare cases where copyright law has confronted industrially useful expressions.").

[54] Oracle Am., Inc. v. Google Inc., 872 F. Supp. 2d 974, 978 (N.D. Cal. 2012), *rev'd*, 750 F.3d 1339 (Fed. Cir. 2014).

[55] *TIOBE Index for November 2020*, TIOBE, https://www.tiobe.com/tiobe-index/ [https://perma.cc/M6XS-SPJT].

[56] *Oracle*, 872 F. Supp. 2d at 997.

[57] Oracle Am., Inc. v. Google Inc., No. C 10-03561, 2016 WL 3181206, at *9 (N.D. Cal. June 8, 2016), *rev'd sub nom*. Oracle Am., Inc. v. Google LLC, 886 F.3d 1179 (Fed. Cir. 2018), *cert. granted*, 140 S. Ct. 520 (2019).

[58] *How Will Java Technology Change My Life?*, ORACLE JAVA DOCUMENTATION, https://docs.oracle.com/javase/tutorial/getStarted/intro/changemylife.html [https://perma.cc/P9EB-MPLY].

[59] Testimony of Plaintiff's Witness, Safra Catz, Oracle Am., Inc. v. Google Inc., No. C 10-03561, 2016 WL 3181206 (N.D. Cal. June 8, 2016).

developers.[60] These packages are what make up the Java library, also known as the Java API.[61]

As the Java API has grown, other libraries of Java code have sprung up to accommodate the vast universe of coding endeavors, many of which are open-source and free to use.[62] For example, a Java programmer performing a probability analysis who seeks to use the mathematical error function would not find that function in the Java API. Instead, the programmer can compute the error function (universally denoted in programming as "erf") in a Java program by invoking the command "org.apache.commons.math4.special.Erf.erf()" provided and supported by the free and open-source Apache Commons Mathematical Library.[63]

Failing to reach an agreement with Oracle, Google moved forward with using the Java language in Android and wrote its own compiling software to run Java programs in Android.[64] Everyone agrees that Google was free to do this.[65] But Google supported only the thirty-seven packages of the Java API that it deemed "key to mobile devices" and that experienced Java programmers would rely on.[66] In doing so, Google developers did not simply copy and paste those thirty-seven coding packages into the Android version of the Java API. Instead, Google wrote its own "clean room" versions of each of those packages using the same organization or taxonomy of the originals.[67] Using exactly the same taxonomy is necessary to being able to use those packages, and it is the duplication of the taxonomy which is at the heart of *Oracle v. Google*.[68]

A. Java Programming Language

What is a Java package? A Java package comprises one or more subsets of computer code called "classes." These classes are logically grouped into packages according to their functionality. Within the classes, methods are defined. The methods are functions that a programmer can call to perform a

---

[60] *Oracle*, 872 F. Supp. 2d at 982.

[61] *Id.*

[62] *See, e.g.*, *10 Useful Third-Party Java Libraries*, CODECONDO (Feb. 9, 2016), https://codecondo.com/10-useful-third-party-java-libraries/ [https://perma.cc/5M2U-PT8Y].

[63] *5 Special Functions*, APACHE COMMONS, http://commons.apache.org/proper/commons-math/userguide/special.html [https://perma.cc/PH9E-QRTM] (last updated Aug. 28, 2016).

[64] *Oracle*, 872 F. Supp. 2d at 978.

[65] *Id.*

[66] *Id.*; *see* Asay, *supra* note 18, at 304.

[67] *Oracle*, 872 F. Supp. 2d at 978.

[68] *Id.* at 978–79; *but see id*. at 1000 ("Oracle has made much of this [issue of fragmenting Java], at times almost leaving the impression that if only Google had replicated *all* 166 Java API packages, Oracle would not have sued.").

specific task, such as generating a pseudorandom number, drawing a rectangle of a given width and length, or as described above, computing the error function for a given value. In the earlier example, the package org.apache.commons.math4.special contains the class Erf, which in turn contains many functions related to the error function including erf().[69] Similarly, the function max() (to which both the 2012 and 2014 rulings referred) resides in the class Math in the package java.lang. One benefit of organizing code into methods, classes, and packages is that Java is modular. A user can import the necessary class or package to call its methods (e.g., "max()" or "erf()"). Alternatively, a programmer can invoke a method without importing code by calling the method by its full name: "java.lang.Math.max()."

The method calls embody the hierarchical package-class-method structure, but what they do not show is the underlying implementation code that carries out the computational work. The implementation codes of the Java and Android versions of the max() method are shown in Figures 1 and 2 enclosed by curly brackets. This is the coding world's version of *abstraction*: "the process of hiding certain details and showing only essential information to the user."[70] A method call is the interface by which a user implements the functionality.[71] Abstracting out every instance of a functionality in a program and replacing it with a method call saves human labor and computational time and makes computer programs easier to read and debug.[72]

The process of creating a new Java code is based on another coding principle called *inheritance*.[73] A programmer will create a new class of code by inheriting the properties of a pre-existing superclass and adding new methods or modifying the inherited methods.[74] Inheritance, then, enables programming by incremental innovation: a new class is created by, in essence, copying and modifying an already existing class.[75] The Java language itself embodies this idea in its design: every class (or self-contained

[69] *5 Special Functions*, *supra* note 63.

[70] *Java Abstraction*, W3SCHOOLS.COM, https://www.w3schools.com/java/java_abstract.asp [https://perma.cc/2HCD-FC36]; *Abstraction (Computer Science)*, WIKIPEDIA, https://en.wikipedia.org/wiki/Abstraction_(computer_science)#Abstraction_in_object_oriented_programming [https://perma.cc/NWT4-3UEB] (last updated Nov. 10, 2020).

[71] Oracle Am., Inc. v. Google Inc., No. C 10-03561, 2016 WL 3181206, at *4 (N.D. Cal. June 8, 2016) ("[A]ll that the Java programmer need master are the declarations. The implementing code remains a 'black box' to the programmer."), *rev'd sub nom*. Oracle Am., Inc. v. Google LLC, 886 F.3d 1179 (Fed. Cir. 2018).

[72] ANDREW HUNT & DAVID THOMAS, THE PRAGMATIC PROGRAMMER 26–27 (1999).

[73] *Oracle*, 872 F. Supp. 2d at 980.

[74] *Id*.

[75] *Id*.

body of code) in the Java language is created by inheriting properties of an already existing class and then modifying that class, typically by adding methods to it, meaning all Java classes are descendants from the first class called "java.lang.Object."[76] This is the mechanism by which the original eight Java packages in the Java API grew to 166 as more and more functionality was developed.[77] Of these, three packages are necessary to every Java program.[78] Indeed, the trial court found that the Java API is so integral to the Java language as to be inseparable.[79]

To be clear, Oracle claims that the taxonomy of the thirty-seven Java packages that Sun Microsystems developed, such as putting "max()" in a class called "Math" in a package called "java.lang," or, in other words, "java.lang.Math.max()," is protected by copyright.[80] Had Google devised a novel taxonomy for the substance of those packages, there would be no issue.[81] Thus, it is not the "package.Class.method()" format of the taxonomy at issue, it is the particular arrangement of the thirty-seven packages to which Oracle claims (and currently has) copyright protection.[82]

B. APIs and the API Economy

Application programming interfaces or APIs promote interconnectivity which in turn have fueled the growth of third-party application development.[83] As of this writing, there are nearly 23,000 APIs listed in the ProgrammableWeb API directory.[84] An API specifies the inputs and defines a set of outputs that are available, while the implementation code that performs the data handling is hidden from the API user.[85] If a client (say, a cellphone app) provides a specified input, the remote server will respond by

---

[76] *The Java™ Tutorials*, ORACLE JAVA DOCUMENTATION, https://docs.oracle.com/javase/tutorial/java/IandI/objectclass.html [https://perma.cc/EZD8-V5EZ].

[77] *Oracle*, 872 F. Supp. 2d at 982.

[78] *Id.*

[79] *Id.* ("Contrary to Oracle, there is no bright line between the language and the API.").

[80] *Id.* at 978.

[81] *Id.* at 1000.

[82] *Id*.

[83] *See* David Berlind, *How Web and Browser APIs Fuel the API Economy*, PROGRAMMABLEWEB (Dec. 3, 2015), https://www.programmableweb.com/news/how-web-and-browser-apis-fuel-api-economy/analysis/2015/12/03 [https://perma.cc/QF2B-URD4].

[84] *Search the Largest API Directory on the Web*, PROGRAMMABLEWEB, https://www.programmableweb.com/apis/directory [https://perma.cc/9K5E-V5BK].

[85] Peter S. Menell, *Rise of the API Copyright Dead?: An Updated Epitaph for Copyright Protection of Network and Functional Features of Computer Software*, 31 HARV. J.L. & TECH. 305, 444 (2018); Endler, *supra* note 2.

providing information or by initiating an action in a specified manner.[86] At the device- or consumer-level, this networking capability or interoperability is appealing to (and increasingly expected by) consumers seeking to use automation to simplify their lives.[87]

In the commercial space, APIs are the primary mechanism driving "inter-organizational collaboration and information exchange" giving rise to the "API economy."[88] Innovation is outsourced when parties external to an organization find new ways to use that organization's data, while both share in the revenue stream.[89] Twitter, for example, offers developers three tiers of pricing for API access, the lowest level being free.[90] eBay's APIs allow third-parties to list its auctions on their websites accounting for 60% of eBay's revenue.[91] Nor are the benefits strictly economic. More importantly, APIs will streamline access to and integration of critical information between hospitals, doctors, and insurance companies, breaking through the difficulty of having patients' clinical information contained in "disconnected data silos."[92]

The general fear following the 2014 ruling is that granting copyright protection to the Java API will place a gate at programming interfaces which will thwart interoperability and impede software innovation. Technical writer Timothy Lee describes the problem:

It's quite common for software developers to clone the functionality of established software platforms and standards in order to make sure their new products are compatible with what's already out there. Sometimes this compatible software is then packaged into open source libraries that become free for others to use, and it can be bundled together with other programs to produce larger software packages. Because it has been widely assumed that API's can't be copyrighted—or at least that the copyrights aren't likely to be

---

[86] MARK L. BRAUNSTEIN, HEALTH INFORMATICS ON FHIR: HOW HL7'S NEW API IS TRANSFORMING HEALTHCARE 9 (2018).

[87] Asay, *supra* note 18, at 288; *Internet of Things: Consumer Expectations Increase with Each Smart Home Device Purchase*, PARKS ASSOCS. (Sept. 22, 2014), https://www.parksassociates.com/blog/article/pr-sept2014-iot-webcast [https://perma.cc/5MMP-PRVV].

[88] Davenport & Iyer, *supra* note 3.

[89] *Id.*

[90] *Getting Started*, TWITTER DEV., https://developer.twitter.com/en/docs/basics/getting-started [https://perma.cc/T8YD-WMZ5].

[91] Bala Iyer & Mohan Subramaniam, *The Strategic Value of APIs*, HARV. BUS. REV. (Jan. 7, 2015), https://hbr.org/2015/01/the-strategic-value-of-apis [https://perma.cc/D5L8-2SGB].

[92] Bill Siwicki, *What You Need to Know About Healthcare APIs and Interoperability*, HEALTHCARE IT NEWS (Apr. 11, 2019, 12:41 PM), https://www.healthcareitnews.com/news/what-you-need-know-about-healthcare-apis-and-interoperability [https://perma.cc/P3PP-2C5H]; *see also* Iyer & Subramaniam, *supra* note 91.

enforced—companies haven't worried about using libraries that take advantage of third-party APIs that might belong to someone else.[93]

Thus, finding that Google infringed Oracle's copyright on the Java API "threatens the continued vitality of software innovation."[94]

III. CODING PRACTICE AND THE "ELUSIVE BOUNDARY"

Much has been written about the coding cultural norms of sharing and collaboration, where new software is created by copying someone else's code and tinkering with it.[95] These norms have driven the coding community to carve out a knowledge commons where the creation of derivative works is encouraged and protected by copyleft licenses.[96] Copyleft licenses turn copyright protections on their head by enforcing the rights of downstream users by perpetuating the right to share, use, and modify copyleft software.[97] The success of this type of licensing is borne out by the sheer size and number of free and open-source software projects and repositories: Linux, Debian, GitHub, Apache, to name a few, as well as OpenJDK, the Java open-source software project that operates in parallel with the commercial Java JDK.[98] Thus, the copyright domain can be made conducive to the coding cultural norms of sharing and collaboration.[99]

But there are other aspects inherent to the practice of coding that support patent protection as the more appropriate domain than copyright. First, patent law may provide a better fit than copyright law for the way innovation occurs in software development. For example, if a person holds

---

[93] Lee, *supra* note 4.

[94] *Id.* (quoting copyright scholar and software developer Professor James Grimmelmann on the 2018 decision denying Google's defense of fair use).

[95] *See* Asay, *supra* note 18, at 280–85; Lawrence Lessig, *Free, as in Beer*, WIRED (Sept. 1, 2006, 12:00 PM), https://www.wired.com/2006/09/free-as-in-beer/ [https://perma.cc/PUL5-3GFH] (quoting Richard Stallman: "You can charge whatever you want for free software. But what you can't do is lock up the knowledge that makes it run. Others must be allowed to learn from and tinker with it. No one is permitted a monopoly on the teaching that stands behind it."); Michael J. Madison, Brett M. Frischmann & Katherine J. Strandburg, *Constructing Commons in the Cultural Environment*, 95 CORNELL L. REV. 657, 661 (2010).

[96] Madison et al., *supra* note 95; *What is GNU?*, GNU OPERATING SYS., https://www.gnu.org [https://perma.cc/6Q8J-RNEK] (last updated Feb. 15, 2021); *Welcome to Apache Commons*, APACHE COMMONS, https://commons.apache.org [https://perma.cc/SKB6-8Z9L] (last updated Aug. 7, 2020).

[97] *See*, *e.g.*, *About CC Licenses*, CREATIVE COMMONS, https://creativecommons.org/use-remix/cc-licenses [https://perma.cc/UU5V-4TP5].

[98] Madison et al., *supra* note 95; *About Debian*, DEBIAN, https://www.debian.org/intro/about [https://perma.cc/9Z3B-YNXD]; *How GitHub Secures Open Source Software*, GITHUB (May 23, 2019), https://resources.github.com/whitepapers/How-GitHub-secures-open-source-software/ [https://perma.cc/CD6G-ZCLZ]; APACHE, https://www.apache.org [https://perma.cc/78YS-THL2]; *OpenJDK FAQ*, OPENJDK (Dec. 18, 2010), https://openjdk.java.net/faq/ [https://perma.cc/UWV4-6VLK].

[99] Asay, *supra* note 18, at 283.

a patent on a cup, another may obtain an improvement patent for adding a handle to the cup. In the coding world, the "modularity and self-contained nature" of object-oriented programming languages like Java enables developers to take existing blocks of code from multiple sources and combine them to create new programs.[100] Inheritance takes this one step further: while modularity enables code sharing and reuse, inheritance enables new code to evolve from existing code. This manner of code creation also boosts productivity.[101] These are the kinds of incremental innovation that patent law contemplates and encourages, but which would be thwarted by copyright's broad exclusivity protections concerning derivative works.[102]

Moreover, taking a computer program and modifying it for a new or different use could properly be examined for a nonobvious inventive concept in order to protect truly novel functionality. In the patent world, the nonobvious requirement under 35 U.S.C. § 103 bars from patent protection an invention or innovation that would have been obvious to "a person having ordinary skill in the art."[103] This ensures that only improvements with an "inventive concept" receive patent protection, creating a bar to trivial and/or obvious innovations and improvements.[104]

The second principle that indicates patent protection may be more appropriate for software innovation is that creativity in coding takes the form of creative problem-solving, rather than original expression. Coders are insatiable problem-solvers, and they do not want to have to reinvent the wheel when there are far more interesting problems to solve.[105] Why write your own pseudorandom number generator for a gaming application when there are hundreds freely available and ready to plug in? As author Gabriella Coleman describes the hacking community: "Indeed, overcoming resistance and solving problems, some of them quite baffling, is central to the sense of accomplishment and pride that hackers routinely experience."[106] Coleman recounts an example of originality in programming: a clever coder accomplished in a functionally-laden single line of code what would

[100] *Id.* at 281.

[101] *See* Berlind, *supra* note 83.

[102] Karjala, *supra* note 17 (functional works amenable to "incremental improvement" would constitute infringement under copyright's substantial similarity test); 17 U.S.C. §§ 103, 107.

[103] 35 U.S.C. § 103.

[104] Parker v. Flook, 437 U.S. 584, 594 (1978); Alice Corp. v. CLS Bank Int'l, 573 U.S. 208, 222 (2014).

[105] Jeff Atwood, *The Best Code Is No Code at All*, CODING HORROR (May 30, 2007), https://blog.codinghorror.com/the-best-code-is-no-code-at-all/ [https://perma.cc/LK9F-PK7B] ("We never met a problem we couldn't solve with some duct tape, a jury-rigged coat hanger, and a pinch of code.").

[106] E. GABRIELLA COLEMAN, CODING FREEDOM: THE ETHICS AND AESTHETICS OF HACKING 12–13 (2013).

otherwise be accomplished (in a more obvious, more typical, and less thoughtful approach) in several functionally simpler lines.[107] But while this distillation of the code to a single line is a notable and arguably creative accomplishment, the initial intent was for the *functional* purposes of efficiency and performance—generally speaking, a single line of code is going to be computationally faster for a computer to process than multiple lines of code performing the same task.[108] Functional improvements of this kind are more accurately classified as creative problem-solving than original expression. Thus, where creativity is tied to functionality, such as in creative problem-solving, there is a strong suggestion that such subject matter belongs in the patent domain.[109]

So, too, do coding best practices eschew creativity in expression by recommending the use of naming standards. Early on in the development of Java, Sun developers prescribed naming standards and conventions to promote consistency and readability.[110] For example, Java developers are exhorted to name packages in all lowercase to distinguish them from classes.[111] Method names should be verbs in lowercase, and multi-word method names should be specified in "lowerCamelCase."[112] Consistency in naming enables other coders to quickly grasp the purpose of the package, class, or method.[113] Readability makes modifying or debugging the code easier for future coders.[114] Consistent or standardized naming also aids coders in finding the method to perform a particular task, saving them the labor of having to write that method themselves. To quote Oracle on

[107] *Id.* at 93–94 (a six-line snippet of code written in Perl to "count the number of stars in the sky" is distilled to a single line of code, decipherable and appreciable only to those who understand the language).

[108] On the other hand, a single line of code that is readily understood only by experienced Perl coders works against the goal of readability, another important aspect of functionality.

[109] *See* Karjala, *supra* note 17, at 448 ("[P]atent protects creative but functional invention, while copyright protects creative but nonfunctional authorship."); Asay, *supra* note 18, at 274 (patent law is traditionally viewed as the appropriate body of law for utilitarian solutions).

[110] *See 9 – Naming Conventions*, ORACLE (Apr. 20, 1999), https://www.oracle.com/technetwork/java/codeconventions-135099.html [https://perma.cc/Y3WU-QB6N].

[111] *Naming a Package*, *The Java™ Tutorials*, ORACLE JAVA DOCUMENTATION, https://docs.oracle.com/javase/tutorial/java/package/namingpkgs.html [https://perma.cc/F4U6-HTKZ].

[112] *Defining Methods*, *The Java™ Tutorials*, ORACLE JAVA DOCUMENTATION, https://docs.oracle.com/javase/tutorial/java/javaOO/methods.html [https://perma.cc/HQR7-VALX].

[113] *See* DEREK M. JONES, THE NEW C STANDARD (IDENTIFIERS): AN ECONOMIC AND CULTURAL COMMENTARY 304 (2008) (ebook).

[114] *See id.* at 372–73.

enforcing consistency in coding style: "Consistency is *vitally important* in making an API easy to learn and use."[115]

Thus, for coding practice, choosing the most meaningful name is more important than choosing from among many creative synonyms. Yet, as discussed below, the 2014 ruling's finding that the creators of the Java API had available to them multiple forms of expression was of paramount importance in deciding the issue of copyrightability, more so than the issue of functionality, and in contrast to the practical reality that creativity took a backseat to functional considerations when the API was designed.[116]

IV. THE 2012 AND 2014 *ORACLE V. GOOGLE* DECISIONS

This Section examines the 2012 and 2014 rulings in light of the preceding legal and technical considerations concerning the Java API focusing on the idea-expression dichotomy, merger, and *scene à faire*.[117]

The 2012 ruling recognized the idea-expression dichotomy in the Java API method definitions: the method calls are ideas, while the bracketed implementation code articulates those ideas, thus synchronizing the coding and legal principles of abstraction.[118] Having framed the Java methods in this way, the 2012 ruling found that under § 102(b), the method calls were uncopyrightable ideas, and "[n]o one may monopolize [those] *idea[s]*."[119] Moreover, in accordance with *Lotus*, the 2012 ruling found that the functionality of the API taxonomy was dispositive on the issue of copyrightability.[120] The decision recognized the originality and creativity by Sun developers in its design, but pointed out that copyright exclusivity is not meant to reward the "sweat of the brow."[121]

Merger is a key point of conflict between the 2012 and 2014 rulings regarding the idea-expression dichotomy. The 2012 ruling found that Google could not have called its recreated Java packages by any other name or in any other structure in order to attain the compatibility it needed for interoperability, and based on that, the source code declarations had merged

---

[115] Richard Bair & Kevin Rushforth, *Code Style Rules*, OPENJDKWIKI, https://wiki.openjdk.java.net/display/OpenJFX/Code+Style+Rules [https://perma.cc/DFP7-BWPG] (last updated Oct. 1, 2019).

[116] *See* Oracle Am., Inc. v. Google Inc., 750 F.3d 1339, 1361 (Fed. Cir. 2014).

[117] Another point of conflict between the two rulings is the application of the names and short phrases doctrine, with the 2014 ruling overturning the 2012 court's ruling in finding that this doctrine also denied copyrightability.

[118] Oracle Am., Inc. v. Google Inc., 872 F. Supp. 2d 974, 997–98 (N.D. Cal. 2012), *rev'd*, 750 F.3d 1339 (Fed. Cir. 2014).

[119] *Id.* at 998.

[120] *See id.* at 999–1000.

[121] *Id.* at 992 (quoting Feist Publ'ns, Inc. v. Rural Tel. Serv. Co., 499 U.S. 340, 353 (1991)).

with their underlying processes.[122] The 2014 ruling, however, found that the 2012 ruling assessed merger in the wrong time frame.[123] Merger, the 2014 decision points out, should be determined at the time the code was created, and not at the time of infringement.[124] At the time the code was created, the Sun developers could have named and organized their Java packages in many different ways.[125] Therefore, the source code declarations did not merge with their underlying processes, finding that when an expression adopted by a programmer is "separable" from its function, that expression "is entitled to protection."[126] On that basis, and consistent with the binding precedent of *Johnson Controls*, the 2014 ruling held that "Section 102(b) does not . . . automatically deny copyright protection to elements of a computer program that are functional."[127]

In the 2014 ruling on this issue of merger, we see how traditional copyright law and the court's attempt to apply the concept of multiple forms of expression onto the computer program design is a remarkably bad fit.[128] In the 2014 ruling, in a bout of copyright tunnel vision, the court posited a rather preposterous alternative that Sun developers could have chosen to name the Math.max() method—Arith.larger()—to show that Sun developers made original and creative choices worthy of copyright protection.[129] Thus, the 2014 opinion ignores the coding culture that prizes practicality over creativity, imposing upon it a value more at home with fine literature which esteems aesthetics such as a lyrical turn of phrase.[130]

Finally, on the issue of *scene à faire*, which was not examined in the 2012 ruling but was raised by Google on appeal, the 2014 decision found that while compatibility is necessary for the API to be functional, this is a defense to infringement, not copyrightability, and, like the doctrine of merger, it must be assessed at the time of creation, not at infringement.[131]

---

[122] *See id.* at 1000.

[123] *See* Oracle Am., Inc. v. Google Inc., 750 F.3d 1339, 1361 (Fed. Cir. 2014).

[124] *Id.*

[125] *Id.*

[126] *Id.* at 1361, 1367.

[127] *Id.* at 1367; Johnson Controls, Inc. v. Phoenix Control Sys., Inc., 886 F.2d 1173, 1175–76 (9th Cir. 1989).

[128] *Cf.* Lotus Dev. Corp. v. Borland Int'l, Inc., 49 F.3d 807, 820 (1st Cir. 1995) (Boudin, J., concurring) ("Applying copyright law to computer programs is like assembling a jigsaw puzzle whose pieces do not quite fit.").

[129] *Oracle*, 750 F.3d at 1361.

[130] Fine literature also appreciates literary devices such as alliteration and assonance; Arith.larger() loses to Math.max() on those as well.

[131] *Oracle*, 750 F.3d at 1364 (citing Mitel, Inc. v. Iqtel, Inc., 124 F.3d 1366, 1375 (10th Cir. 1997)).

CONCLUSION

Both the 2012 and 2014 *Oracle v. Google* decisions find support in copyright doctrine and in precedent as evidenced by the circuit split. There are plausible arguments on both sides. But no decision can or should be made without understanding the nature of the code at issue, how that code is used, and what were and are the priorities of the code creators. These considerations get at the difference between highly utilitarian literary works that are computer programs and traditional literary works that are solely the embodiment of creative choices. Simply stamping copyright doctrine on the issues ignores those aspects of coding practice that have given rise to the $12.1 trillion information technology sector of the U.S. economy.[132]

There are a number of factors that must be considered when deciding whether the Java APIs should have copyright protection. First, at the doctrinal level, allowing functionality into the domain of copyright protection conflicts with § 102(b) of the Copyright Act and creates the possibility that innovation that rightly belongs in the more stringently policed patent sphere will open the door to monopolizing functionality.[133]

Second, coding values consistency and readability over creativity, so the 2014 rulings' emphasis on whether there are "multiple ways to express the underlying idea" should be tempered with practical considerations when copyrightability issues are to be decided for software innovations.[134] In other words, proving multiple forms of expression should not be a way to bypass the § 102(b) exclusion of methods of operation from protection when it is of little or no practical importance in coding.

Third, the networking capability of APIs has given rise to innovation and economic growth through "inter-organizational collaboration and information sharing,"[135] and this should signal to the courts that interoperability is a critical *function* of computer code that would be hindered by finding APIs to be copyright protectable.

Finally, innovation in software development is better suited to patent protection. Coding best practices encourage creating new code that derives from existing code and discourages reinventing the wheel. The coding principle of inheritance and indeed the Java language itself embody the

---

[132] As of November 11, 2020. *Sectors & Industries Overview*, FIDELITY, https://eresearch.fidelity.com/eresearch/markets_sectors/sectors/sectors_in_market.jhtml [https://perma.cc/3GAN-4F2Q].

[133] *See* Oracle Am., Inc. v. Google Inc., 872 F. Supp. 2d 974, 985 (N.D. Cal. 2012) (finding that section 102(b) "codified a *Baker*-like limitation" that precludes copyright protection of ideas and methods of operation), *rev'd*, 750 F.3d 1339 (Fed. Cir. 2014).

[134] *Oracle*, 750 F.3d at 1367.

[135] Davenport & Iyer, *supra* note 3.

practice of incremental innovation that patent law incentivizes and which is better suited for examination under patent law's nonobviousness requirement. Moreover, the type of creativity involved in computer programming—creative problem-solving—is driven by the need for practical improvements like greater efficiency. For these reasons, the Supreme Court should reboot *Baker v. Selden*[136] and deny copyright protection to the Java API at issue in *Oracle v. Google*.

[136] 101 U.S. 99, 102 (1879).