United States Copyright Office
Library of Congress · 101 Independence Avenue SE · Washington, DC 20559-6000 · www.copyright.gov

December 4, 2017

Professor Richard L. Revesz
Director, ALI
Ms. Stephanie A. Middleton
Deputy Director, ALI
Professor Christopher Jon Sprigman
Professor Daniel J. Gervais
Professor Lydia Pallas Loren
Professor R. Anthony Reese
Professor Molly S. Van Houweling
Reporters, ALI Restatement of the Law, Copyright

Re: Preliminary Draft No. 3

Dear Professor Revesz, Ms. Middleton, and Reporters:

The U.S. Copyright Office is responsible for administering significant portions the nation's copyright law and providing expert advice to Congress and federal agencies on copyright matters.[1] We have reviewed Preliminary Draft No. 3 of ALI's proposed Restatement of the Law of Copyright, which includes revised sections on the scope of copyright protection (§§ 2.01-2.05), a new section on copyright protection for designs of useful articles (§ 2.07), and a new chapter on copyright ownership issues (§§ 3.01-3.05).

We wish to reiterate the Copyright Office's view that the Restatement should play a limited role in analysis of the Copyright Act. It should not displace any of the "traditional tools of statutory construction," including examination of the statute's text, structure, purposes, and legislative history. *See INS v. Cardoza-Fonseca*, 480 U.S. 421, 446-450 (1987). Nor should the Restatement supplant the Copyright Office's rules and regulatory guidance that interpret and apply the text of the Copyright Act, which are entitled to ordinary and appropriate levels of judicial deference, even where they differ from views expressed in the Restatement.

In addition, we continue to urge the Reporters to avoid conflicts between the Restatement and Copyright Office regulations or other interpretive guidance, including the *Compendium of U.S. Copyright Office Practices (Third Edition)* ("*Compendium III*"). Such conflicts could produce significant confusion and adversely affect Copyright Office operations, especially to the extent they relate to issues the Office addresses in registering claims to copyright or administering statutory licenses under the Copyright Act.

[1] 17 U.S.C. § 701(a), (b).

The Reporters asked two specific questions when circulating the draft. The Copyright Office provides the following responses:

In § 3.05 (Works Made for Hire), we would be grateful for input regarding whether there are elements presently contained in the comments that should be included in the black letter. For example, in Comment f we discuss the writing requirement for specially ordered or commissioned works made for hire. In particular, we discuss the required timing of the writing. Should we include in the black letter of § 3.05 something about the required timing?

- The Copyright Office's recommendation is to not amend the black letter law section. Currently, the black letter section mirrors the statutory language of 17 U.S.C. §§ 101 and 201 fairly closely, which ought to be the objective of the black letter section. Even where there is judicial consensus on an issue, that additional information is properly left to the comments.

In § 3.04 (Ownership of a Joint Work*), we state in the Reporters' Note to Comment e that the better view is that agreements among co-owners of a joint work that alter initial ownership shares must be in writing. Should we promote this point to the Comment itself?*

- As discussed below, the Copyright Office agrees that agreements that alter initial ownership shares for a joint work should be in writing and signed. Accordingly, the Office believes that this point can be promoted to the Comment itself.

In addition, the Office has (in the short period permitted for review), identified the following points in the most recent draft (in the order they appear in the Preliminary Draft, rather than in order of importance).

CHAPTER 2

Section 2.03

Section 2.03 is includes many of the same problems as the ALI's previous draft. In particular, section 2.03, Comments *d* and *e*, and the associated Reporters Notes, continue to reflect significant and substantial errors of law. Comment *d* and the associated Reporters' Note are premised on the view that section 102(b) can serve as an independent bar on copyrightability of otherwise expressive works of authorship.[2] Comment *e*, in turn, states that expressive works can

[2] One relatively minor point: Section 2.03, Comment *d*, states that "a 'method of operation,' as that term is employed in § 102(b), refers to the means by which a person operates something, whether it be a car, a food processor, or a computer." We think that there is at least some evidence that supports a different understanding of that particular phrase. As far as we can tell, the phrase "method of operation" makes its first appearance in relation to copyright law in the *Baker v. Selden* decision, and was used to refer specifically to the abstract algorithms that a mathematician sets forth in his proofs: "The copyright of a work on mathematical science cannot give to the author an exclusive right to the *methods of operation* which he propounds, or to the diagrams which he employs to explain them, so as to prevent an engineer from using them whenever occasion requires." 101 U.S. 99, 103 (1879). Given that Congress drew from *Baker* in crafting § 102(b), it seems likely that the specific phrase "method of operation" is better understood as a reference to an "operation" in the mathematical or theoretical sense—something that is "propound[ed]"—rather than a method by which a person operates something. That having been said, we think that other phrases in section 102(b)—"procedure," "process," or "system"—cover the same ground.

be excluded from copyright protection even in circumstances where there are multiple ways of expressing or describing a particular "system" or "method of operation." And Reporters' Note to Comment *e* continues to posit that "[t]ypically, an API is composed of code and structural elements that are, by design, essentially functional and therefore should be excluded from protection under § 102(b) regardless of whether they contain expression."[3]

The Copyright Office reiterates its disagreement with these points. As we explained in our previous letter, they directly conflict with the positions taken by the United States government in litigation before the Supreme Court. As the government has explained to the Supreme Court, "[i]f a work constitutes expression (and if it is original), it is copyrightable under Section 102(a)."[4] Section 102(b) "is not a limitation on what kinds of expressive works may be protected by a copyright."[5] Instead, "it is a limitation on how broadly the copyright extends."[6]

This understanding is based on a straightforward reading of the text of section 102(b) itself: "In no case does copyright protection *for* an original work of authorship *extend to* 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 *in such work*."[7] Section 102(b) thus unambiguously defines the *scope* of copyright protection "for" something that has already been determined to be "an original work of authorship" under section 102(a), excluding from that scope only the non-expressive ideas, procedures, processes, etc. embodied "in such work." In other words, section 102(b) "merely excludes from copyright protection the subject matter explained or described in the expressive work."[8] Under this understanding, the "underlying computer function triggered by the written code—for example, an algorithm that the computer executes to sort a data set[—would be unprotected]. The code itself, however, [would be] eligible for copyright protection."[9]

Although the merger doctrine can serve to deprive otherwise expressive works of copyright protection, that doctrine requires a finding that there were only a limited number of ways for the original author to express some underlying idea or function.[10] Similarly, the *scènes à faire* doctrine requires a finding that some expression is stock or standard, or dictated by external constraints, and thus was not original to the author.[11]

Comments *d* and *e* rest heavily on the First Circuit's decision in *Lotus Development Corp. v. Borland International, Inc.*[12] That is an extraordinarily unstable foundation for the reasoning of

---

[3] DRAFT NO. 3 at 21.
[4] Brief for the United States as Amicus Curiae at 13, Google Inc. v. Oracle Am., Inc., 135 S. Ct. 2887 (2015) (No. 14-410) ("U.S. Google v. Oracle Br.").
[5] *Id.* at 12.
[6] *Id.*
[7] 17 U.S.C. § 102(b) (emphases added).
[8] U.S. Google v. Oracle Br. at 13.
[9] *Id.* at 14.
[10] *Id.* ("If within a given technological environment, code must be drafted in a specific way in order to induce the computer to carry out a particular function, then the expression would 'merge' with the function, and the code would be uncopyrightable.").
[11] *Id.* ("[S]ome computer routines may be so standard in the programming industry that the *scènes à faire* doctrine deprives them of copyright protection. But computer code is not an uncopyrightable 'method of operation' or 'system' under Section 102(b) simply because it causes a computer to function.").
[12] 49 F.3d 807, 815 (1st Cir. 1995), *aff'd by an equally divided court*, 516 U.S. 233 (1996).

those Comments. As the government has explained to the Supreme Court, "[t]he precise rationale of *Lotus* is not clear."[13] And whatever the basis for its legal reasoning, that decision "cannot reasonably be read to treat Section 102(b) as applicable to computer code itself, a form of expression that the Copyright Act clearly protects and that the First Circuit took pains to distinguish."[14] Indeed, we are unaware of any decision "in the two decades since *Lotus*, in the First Circuit or elsewhere, in which Section 102(b) has been invoked to hold that original computer code is not copyrightable."[15]

Some have also read *Sega Enterprises Ltd. v. Accolade, Inc.*[16] to suggest that otherwise expressive computer code can be excluded from copyright protection via 102(b).[17] To reiterate, this view conflicts with the text of section 102(b). In any event, we believe that this reading of *Sega* is mistaken. To begin with, the *Sega* decision was based on fair use and, even there, was only limited to the question of whether the intermediate copying done for purposes of reverse engineering was an infringing act. *Sega* was not a case where the original code was copied wholesale into the competing product; the court stressed that Accolade's development manual "contained only functional descriptions of the interface requirements and did not include any of Sega's code."[18] Indeed, to the extent that the court addressed the functionality of Sega's code, it was focused on the "functional concepts *embodied* in the code," and the necessity of engaging in reverse engineering to understand those concepts.[19] That is an appropriate understanding of section 102(b), and one with which the Copyright Office agrees. Moreover, to the extent the court addressed copyrightability, it was because it regarded the relevant code as *insufficiently creative* to qualify for copyright protection under section 102(a). This is made clear by the manner in which the *Sega* court distinguished the Federal Circuit's arguably conflicting decision in *Atari Games Corp. v. Nintendo of America, Inc.*[20] which upheld the copyrightability of a lockout program for a different video game console. The court distinguished *Atari* on the ground that the Nintendo security system "consisted of an *original program* which generates an arbitrary data stream 'key'" and that "[c]reativity and originality went into the design of that program." [21] The *Sega* court concluded that the Sega code, by contrast, "appears to be functional" because "[i]t consists merely of 20 bytes of initialization code plus the letters S-E-G-A," and that there was not "a multitude of different ways to unlock the Genesis III console."[22] These points relate to the *creativity* of the code, and do not bear on the applicability of section 102(b). Indeed, to underscore this point, the *Sega* court also "note[d] that Sega's security code is of such de minimis length that it is probably unprotected under the words and short phrases doctrine."[23]

Separately, Reporters' Note to Comment *d* makes plain the Reporters' aversion to courts relying on fair use in the context of suits involving the creation of interoperable computer programs. It

---

[13] U.S. Google v. Oracle Br. at 20
[14] *Id.*
[15] *Id.* at 21.
[16] 977 F.2d 1510 (9th Cir. 1992).
[17] *See, e.g.*, Pamela Samuelson, *Functionality and Expression in Computer Programs: Refining the Tests for Software Copyright Infringement*, 31 Berkeley Tech. L.J. 1215, 1264 (2017).
[18] *See Sega Enters.*, 977 F.2d at 1515.
[19] *Id.* at 1517, 1519 (emphasis added).
[20] 975 F.2d 832 (Fed. Cir. 1992).
[21] *Sega Enters.*, 977 F.2d at 1524 n.7.
[22] *Id.*
[23] *Id.* (citing 37 C.F.R. § 202.1(a)).

states: "Allowing choices among methods to be protectable and relying instead on fair use presents its own set of problems," and quotes Judge Boudin's concurrence in *Lotus v. Borland*. This aversion is wholly unwarranted. The note ignores the significant body of case law—as well as the Copyright Office's own recommendations in section 1201 rulemakings—that have repeatedly relied on fair use to address situations where copying is needed to enable interoperability of software.[24]

In addition, Reporters' Note to Comment *e* evinces a fundamental misunderstanding of the facts of the *Oracle v. Google* case, and confuses and conflates distinct meanings of the term "API." In an effort to better explain our position, we have attached a more detailed factual and legal discussion of that case as an Addendum to this letter.

Section 2.05

Section 2.05, Comment *c*, lines 9-28, and Reporters' Note *c*, attempt to distinguish between two situations involving the application of merger and *scènes à faire* doctrines to computer programs. The passage treats "choices of an initial creator [that] immediately constrain the choices available to later authors" as subject to those doctrines, and "constraints on later authors [that] emerge subsequent to the creation of the first work" as not subject to them. This is done in an effort to explain and justify the result in the Sixth Circuit's 2004 decision in *Lexmark Int'l, Inc. v. Static Control Components, Inc.*, which the Preliminary Draft regards as involving the first situation rather than the second.

The Copyright Office disagrees with these passages. The Office has repeatedly taken the position that the merger and *scènes à faire* doctrines should look to the constraints that existed at the time the author created the work, and not later events.[25] Moreover, we believe this discussion is premised on a fundamental misunderstanding of the facts and reasoning of *Lexmark*. (This misunderstanding is also reflected on lines 1-8 on the same page.) To begin with, the Sixth Circuit did not itself resolve the merger/*scènes à faire* question, but instead left that question to the district court.[26] But even setting aside that point, as the Office has elsewhere explained,[27]

---

[24] *See generally* U.S. COPYRIGHT OFFICE, SOFTWARE-ENABLED CONSUMER PRODUCTS 54-59 (2016) ("SOFTWARE-ENABLED CONSUMER PRODUCTS"), *available at* https://www.copyright.gov/policy/software/software-full-report.pdf (discussing decisions); *see also Sega Enters.*, 977 F.2d at 1527-28 (holding that copying a video game console's computer program to decompile and reverse engineer the object code to make it interoperable with the defendant's video games was a fair use); *Atari*, 975 F.2d at 842-44 (holding that reverse engineering of a competitor's computer chips to "learn their unprotected ideas and processes" was a fair use); *Sony Comput. Entm't, Inc. v. Connectix Corp.*, 203 F.3d 596, 608 (9th Cir. 2000) (holding that that reverse engineering the operating system of a PlayStation gaming console to develop a computer program allowing users to play PlayStation video games on a desktop computer, as well as making copies in the course of such reverse engineering, was a fair use).

[25] *See, e.g.*, SOFTWARE-ENABLED CONSUMER PRODUCTS at 16 ("For both merger and *scènes à faire*, courts must focus on the options available to the author at the time a work is initially created, rather than the choices available to users after the fact.").

[26] *Lexmark Int'l, Inc. v. Static Control Components, Inc.*, 387 F.3d 522, 541 (6th Cir. 2004) ("At the permanent injunction stage of this dispute, we leave it to the district court in the first instance to decide whether the Toner Loading Program has sufficient originality to warrant copyright protection."). Note as well that *Lexmark* was a somewhat fractured opinion. Judge Merritt, in a separate concurring opinion, regarded the main opinion as having held that "the Toner Loading Program is not copyrightable because of the merger and scenes a faire doctrines," although the main opinion was explicit that it was not reaching that point. 387 F.3d at 551 (Merritt, J., concurring). And Judge Feikens dissented on the question of the copyrightability of the Toner Loading Program entirely. 387 F.3d at 553 (Feikens, J., dissenting).

Lexmark is not, in fact, a case where plaintiff's unconstrained choices limited the choices available to the defendant. It is, instead, an absolutely typical merger/*scènes à faire* case: the court's reasoning looked to the existence of *preexisting* constraints that limited the *plaintiff's* choices.

In *Lexmark*, the copyright infringement claim was based on defendant's replication of a program called the Toner Loading Program, copies of which were on Lexmark's toner cartridges.[28] The Toner Loading Program "measure[d] the amount of toner remaining in the cartridge based on the amount of torque (rotational force) sensed on the toner cartridge wheel."[29] It did so by "execut[ing] one of several mathematical equations that convert the torque reading into an approximation of toner level."[30]

In addressing the application of the merger and merger/*scènes à faire* doctrines to the Toner Loader Program, the court described "several external constraints" that "*limit[ed] the options available in designing the Toner Loading Program*."[31] These included the constraints imposed by separate Printer Engine Program that existed on Lexmark's printers, as well as "physical realities of the printer and toner cartridge."[32] The Court treated those as constraints as preexisting ones that limited the choices available to the plaintiff.[33] This is true, even though, as a theoretical matter, the plaintiff could have designed its Printer Engine Program or its printer hardware in a different manner. Additionally, the court relied heavily on testimony from defendant's expert that the Toner Loading Program was the "most efficient means of calculating toner levels," and was a "no-thought translation" of the relevant mathematical formula "to the language that the internal loading program must be written in."[34] Those (unprotectable) mathematical formula and the efficiency constraints were unquestionably preexisting constraints on the plaintiff.

Thus, *Lexmark* does not support the distinction this passage is attempting to draw, and we are unaware of any other legal support for the proposition. We urge the deletion of lines 1-28 on page 36, lines 29-39 on page 41, and lines 1-16 on page 42.

Section 2.06

Section 2.06, Comment *f* takes the position that a work, or elements of a work, that were originally sufficiently expressive and creative to warrant copyright protection under section 102(a), can be deprived of copyright protection over time by the *scènes à faire* doctrine, if such elements of that work "come to be standard or commonplace after the first author." The Copyright Office believes this view is contrary to section 302(a) the Copyright Act, which provides that "[c]opyright in a work . . . *subsists* from its creation and, except as provided by the following subsections, *endures* for a term consisting of the life of the author and 70 years after

---

[27] SOFTWARE-ENABLED CONSUMER PRODUCTS at 16.
[28] 387 F.3d at 529.
[29] *Id.*
[30] *Id.*
[31] *Id.* at 539 (emphasis added).
[32] *Id.* at 539-40.
[33] *See id.* at 539 ("To discern whether 'originality' exists in the work, the court should ask whether the ideas, methods of operation and facts of the program could have been expressed in any form other than that chosen by the programmer, taking into consideration the functionality, compatibility and efficiency demanded of the program.").
[34] *Id.* at 540

the author's death."[35] In any event, as the Comment acknowledges, no court has adopted that principle,[36] and a number of courts of appeals and the Copyright Office itself have adopted the opposite one. The Comment cites a case that relied heavily on the idea/expression dichotomy in finding that the works at issue were not substantially similar.[37] The Office believes that section 102(b) will generally be a sounder way of resolving cases like that one, providing copyright protection to expressive content, but withholding it from the underlying ideas (like a man with super strength who can lift a car, or secret marriages between Jewish and Irish children of domineering fathers[38]). The Copyright Office urges deletion of Comment *f*.

Section 2.07

We appreciate the Reporters' efforts to explain useful article design protection in Section 2.07 of the Preliminary Draft. The black letter law and comments—with a few specific exceptions discussed below—generally comport with the Office's own current thinking after the *Star Athletica v. Varsity Brands* decision.

The Office is in the process of updating the relevant sections of *Compendium III* to account for that decision. One of the issues we are considering is whether to include a more comprehensive treatment of "works of artistic craftsmanship," because the Copyright Act by its express terms distinguishes between "works of artistic craftsmanship" and "useful articles," and treats the two categories of works differently.[39] Works of artistic craftsmanship are generally copyrightable "insofar as their form but not their mechanical or utilitarian aspects are concerned," and the Act does not impose a "separability" limitation on copyrightability.[40] Useful articles, in contrast, are generally *not* copyrightable as such, although copyright protects two- or three-dimensional artistic features incorporated into the design of a useful article, so long as those features are separable from the useful article.[41]

Although the Act itself provides no explicit means of distinguishing between "works of artistic craftsmanship" and "useful articles," the Copyright Office's pre-1976 regulations and practices, which were largely codified in the Copyright Act of 1976, are instructive. 37 C.F.R. § 202.10 (1960) (copyrightable works of art include "works of artistic craftsmanship insofar as their form but not their mechanical or utilitarian aspects are concerned such as artistic jewelry enamels, glassware and tapestries"); H.R. Rep. No. 94–1476, at 54, *as reprinted in* 1976 U.S.C.C.A.N. 5659, 5667. For decades prior to the 1976 Act, the Office's practices distinguished between, on the one hand, works of art that may incidentally serve useful functions, such as stained-glass windows, sculpted figures useable as bookends, sculpted figures with a slot for use as a bank,

---

[35] 17 U.S.C. § 302(a) (emphases added).
[36] Section 2.06, Reporters' Note *f* adopts a "dog that didn't bark" principle to justify the comment, considering it "odd that courts do not expressly address that timing in the vast majority of cases" outside the software context. But that may be because, in the vast majority of cases, the stock elements are so old, or are dictated by external constraints like actual events in history, that the timing question is irrelevant.
[37] *Warner Bros., Inc. v. Am. Broad. Cos., Inc.*, 654 F.2d 204, 209 (2nd Cir. 1981) (noting that "the works . . . share common ideas, themes, and general imagery" but that "the similarities are not sufficiently particular and concrete so as to represent an appropriation of the protected expression of the plaintiffs' works"); *see also Warner Bros., Inc. v. Am. Broad. Cos., Inc.*, 720 F.2d 231, 239-40 (2d Cir. 1983) (decision on appeal from summary judgment).
[38] *See Nichols v. Universal Pictures Corp*., 45 F.2d 119, 120 (1930).
[39] *See* 17 U.S.C. § 101 (definition of "pictorial, graphic, and sculptural works").
[40] *Id.*
[41] *See id.*

mortuary urns, and artistic vases and bowls, and, on the other hand, articles that principally had a utilitarian function, such as hand tools, medical instruments, automobiles, and home appliances. These categories were borrowed in modified form from the Supreme Court's decision in a tariff case, *United States v. Perry*, 146 U.S. 71 (1892), which had distinguished between "[o]bjects of art, which serve primarily an ornamental, and incidentally a useful, purpose" and "objects primarily designed for a useful purpose, but made ornamental to please the eye." *Id.* at 74-75. This distinction is generally consistent with the Preliminary Draft's approach, which distinguishes "'intrinsic' utilitarian functions . . . from what may be termed 'incidental' or 'marginal' utilitarian functions." The Office will ensure the Reporters are aware of any relevant Compendium updates when they are available.

The Copyright Office does have some concern about the Preliminary Draft's treatment of toys. The Office categorizes toys, dolls, and stuffed toy animals as *works of artistic craftsmanship*. *See Compendium III*, § 903.1. We are concerned, therefore, by the suggestion in Comment *a* (pp.63-64), and the related discussion in Reporters' Note *a*, that the stacking toy pictured has an "intrinsic utilitarian function" because it "is useful in improving young children's understanding of size and sequence." We believe that sort of functionality is not the type of "intrinsic utilitarian function" that Congress had in mind when defining the category of useful articles.[42] Indeed, we think that suggestion is contrary to the earlier acknowledgment that the definition of useful articles is meant to cover works that are "in the category of industrial design," which would exclude children's toys like the stacking game. On the logic of Comment *a*, even a stuffed toy bear would be considered a useful article, since stuffed animals provide developmental benefits to children.[43] In addition, the suggestion in the Reporters' Note to evaluate whether a toy's usefulness "is a significant part of its appeal" seems contrary to *Star Athletica*'s rejection of the "marketability" test.[44]

The Copyright Office also questions the statement in Comment *c* that "pictorial, graphic, or sculptural features that contribute to an article's utility should not be treated as separable from the article's utilitarian aspects." The discussion in Reporter's Note *a*, from page 76, line 26 to page 77, line 24, appears premised on the same understanding of the statute. We believe that understanding is inconsistent with *Star Athletica.* The petitioner in that case had argued that "a feature may exist independently only if it can stand alone as a copyrightable work *and* if the useful article from which it was extracted would remain equally useful."[45] Thus, according to the petitioner, "if a feature of a useful article 'advance[s] the utility of the article,' . . . then it is categorically beyond the scope of copyright."[46] The Court squarely rejected that argument, holding that "the statute does not require the imagined remainder to be a fully functioning useful

[42] *See Gay Toys, Inc. v. Buddy L. Corp.*, 703 F.2d 970 (6th Cir. 1983) ("The intention of Congress was to exclude from copyright protection industrial products such as automobiles, food processors, and television sets. The function of toys is much more similar to that of works of art than it is to the "intrinsic utilitarian function" of industrial products.") (internal citation omitted).
[43] *See, e.g.*, Kathleen M. Reilly, *First Friends: Toddlers and Stuffed Animals*, PARENTS (May 2009), *available at* http://www.parents.com/toddlers-preschoolers/development/social/stuffed-animals/. *See also Spinmaster, Ltd. v. Overbreak, LLC*, 404 F. Supp. 2d 1097, 1102-04 (N.D. Ill. 2005) ("A rule that considers any toy which helps to develop hand-eye coordination to be useful would cause the usefulness exception to swallow the general rule allowing copyrights for pictorial works.").
[44] *Star Athletica, LLC v. Varsity Brands*, 137 S.Ct. 1002, 1015 (2017).
[45] *Id.* at 1013.
[46] *Id.*

article at all, much less an equally useful one."[47] Indeed, the Court states that the removed feature can itself have *some* utilitarian function, relying on a dictionary definition of "applied art."[48] (Relatedly, the Court's decision clarifies that the statutory language that "an article that is normally a part of a useful article is treated as a useful article," is limited to a part that is *intrinsically* useful, not any part that simply adds increased functionality. For example, the zipper on a uniform is a useful article, but the stripes on a uniform are not.)

Reporters' Note *c*, on p.78, lines 28-38, attempts to preserve consumer perception as a possible relevant factor in assessing separability. Contrary to the suggestion in this passage, the Copyright Office believes that *Star Athletica* squarely forecloses this approach.[49] Moreover, it is inconsistent with the Office's longstanding approach in examination, which is based on an objective examination of the material deposited with the Copyright Office, rather than extrinsic factors.

Finally, we believe it would be helpful for the Restatement to emphasize that the statutory bar on copyright protection for utilitarian aspects of works is limited to *pictorial, graphic, and sculptural works.* The Copyright Act does not itself, however, bar copyright protection for other kinds of expressive works (including literary works) that might be described as "utilitarian" or "functional" (although certain *aspects* of those works may very well be outside the scope of copyright protection under section 102(b) or judicially created doctrines). There is often confusion about this point.

CHAPTER 3

Section 3.03

Section 3.03(b), and associated Comments and Notes, in the Copyright Office's view, correctly take the position that, to be considered a joint author, a person must contribute original authorship to the work.[50]

The statement of black letter law goes further, however, and states that a joint author's contribution must be "significant enough in the context of the entire work to qualify that author as one of the intellectual originators of the work as a unitary whole." The Office currently has no view about whether the Copyright Act imposes such a "significance" requirement, or other similar requirement, for contributions to joint works. On the one hand, we understand the policy rationales underlying this provision, and that, for certain industries, such a rule would be a convenient means of "keep[ing] the number of authors of a complex work to a manageable

---

[47] *Id.* at 1014; *see also id.* ("Indeed, such a requirement would deprive the *Mazer* statuette of protection had it been created first as a lamp base rather than as a statuette. Without the base, the 'lamp' would be just a shade, bulb, and wires. The statute does not require that we imagine a nonartistic replacement for the removed feature to determine whether that *feature* is capable of an independent existence.").

[48] *Id.* ("An artistic feature that would be eligible for copyright protection on its own cannot lose that protection simply because it was first created as a feature of the design of a useful article, *even if it makes that article more useful*." (emphasis added))

[49] *Id.* at 1015 (rejecting marketability test because "[n]othing in the statute suggests copyrightability depends on market surveys).

[50] *See Compendium III*, § 505.2. *See also* Section 3.02, Comment *d* ("[I]t will not affect an author's claim to sole authorship of a work if another person contributed only ideas, rather than expression, to that work, because copyright protection does not extend to ideas.").

number," as Comment *b* states. On the other hand, the statutory basis for such a limitation is not entirely clear. Section 201(a)'s reference to the "authors of a joint work" strikes us as an unsteady pillar on which to rest the entire body of law described in Comment *d*; that phrase adds nothing to the definition of "joint work" in section 101, which does not expressly limit the kinds of "authors" that are eligible to make contributions to a joint work. Nor does the statement of black letter law adopt a standard set forth in an established and uniform body of case law.[51] As a result, regardless of whether the Copyright Act is properly understood to impose a significance requirement, we do not believe that it is appropriate—at least under the current state of the law—to elevate this point to black letter law.

Section 3.04

In Section 3.04, the Office would recommend making the additional point that the United States' rules on joint authorship differ from those of many countries that require the consent of all co-owners for any license, not only an exclusive license.[52] Foreign law may, in some circumstances, govern the treatment of the joint works of a U.S. and foreign author.[53]

Section 3.04, Reporters' Note to Comment *e*, addresses the question of whether an agreement among co-authors altering the default allocation of ownership shares in a joint work must be in writing. As noted above, the Copyright Office agrees that agreements that alter initial ownership shares for a joint work should be in writing and signed. Unlike the Reporters' Note, however, we believe this is a necessary consequence of section 204 rather than merely the result of sound policy.[54] That is because, in our view, the best way to think about such an agreement is as a transfer of partial ownership shares in the whole work from one co-author to another. So, under the example in Illustration 12, Programmers B and C have effectively transferred their respective 8 and 1/3 percent ownership shares to Programmer A.

At the same time, it is not entirely clear whether an agreement that maintains the equal default shares, but limits each co-owner's rights to administration of only their respective share (*e.g.*, an agreement to override the default rule that would permit a co-owner to grant a nonexclusive

---

[51] *Compare Aalmuhammed v. Lee*, 202 F3d 1227, 1232 (2000) ("We hold that authorship is required under the statutory definition of a joint work, and that authorship is not the same thing as making a valuable and copyrightable contribution."); *id.* at 1231, 1235 (concluding that while "Aalmuhammed made substantial and valuable contributions to the movie," he was not an author because he "offered no evidence that he was the 'inventive or master mind' of the movie").

[52] *See, e.g.*, 1 NIMMER ON COPYRIGHT § 6.10[D] (citing legal precedent from the United Kingdom, France, Italy, Japan, Mexico, and Canada to document that in many "foreign jurisdictions, a license will not be valid unless all joint owners are party to it . . . . [and thus], the licensee of a single joint owner will be unable to exploit his work [outside the United States]").

[53] *See* 2 SAM RICKETSON & JANE GINSBURG, INTERNATIONAL COPYRIGHT AND NEIGHBORING RIGHTS: THE BERNE CONVENTION AND BEYOND § 20.40, at 1318-20 (2005) (describing potential choice of law principles for determining joint ownership rights including "the law of the country in which a majority of the creators reside," or "the law of the country the creators designate by contract"); *Itar-Tass Russian News Agency v. Russian Kurier, Inc.*, 153 F.3d 82, 90 (2d Cir. 1998) (suggesting law of country with "'most significant relationship' to the property and the parties" would govern ownership rights (quoting RESTATEMENT (SECOND) OF CONFLICT OF LAWS § 222 (1971))).

[54] U.S. Copyright Office, *Views of the United States Copyright Office Concerning PRO Licensing of Jointly Owned Works* 10 n.54 ("In the case of an agreement concerning a joint work where the percentage shares deviate from the default rule of equally divided shares, the agreement represents a transfer of ownership interests and, under section 204 of the Act, must therefore be in writing and signed by the owner of the rights conveyed or the owner's duly authorized agent in order to be valid.").

license to use the whole work), would require a signed writing.[55] It seems difficult to describe that kind of an agreement as a "transfer of copyright ownership" within the meaning of the Copyright Act.

Section 3.05

Section 3.05, Comment *b*, lines 11-16, treat the decision in *Kirk v. Harter*[56] as holding that the provision of employee benefits and tax treatment are central to the determination of employee status. We do not read *Kirk* this way; the relevant passage from that decision makes clear that the court was considering all the factors identified in *CCNV*, though it specifically mentioned only the two:

> On balance, we conclude that the factors which might support a conclusion that an employer-employee relationship existed are insufficient to overcome the evidence that Harter was an independent contractor. Iowa Pedigree did not treat Harter as an employee for tax purposes and did not pay him traditional employee benefits. Furthermore, Harter was highly skilled, continued to consult with other companies, and on at least one occasion unilaterally hired a subcontractor. We find the Second Circuit's reasoning in *Aymes* persuasive, and we therefore conclude that Harter was an independent contractor.[57]

Section 3.05 Comment *e* and Reporters' Note *e* express concern that the categories of "compilation" and "collective works" could be read "very broadly." The problem identified is not unique to the work-for-hire context; it exists, at least as a theoretical matter, with compilations generally.[58] But if a work is, in fact, actually a "compilation" or "collective work," it is unclear what basis there is in the language of the Act for excluding it from the work-for-hire provision.

Section 3.06

Section 3.06, Comment *d* states that "the effect of a transfer that satisfies the applicable requirements described in this Section is that the transferee owns the exclusive rights (or subdivisions thereof) that are the subject of the transfer. . . . As the owner, the transferee has the exclusive rights to do and to authorize the activities that are covered by the transferred rights or subrights." That is far too broad a statement, given the Copyright Act's definition of "transfer of copyright ownership," which defines that term to include a "mortgage" or a "hypothecation"[59] of a copyright, neither of which would give the transferee the right to exercise exclusive rights. It

---

[55] *See id.*
[56] 188 F.3d 1005 (8th Cir. 1999).
[57] *Id.* at 1009 (referring to *Aymes v. Bonelli*, 980 F.2d 857, 862-64 (2d Cir. 1992) (finding that the skill, tax treatment, and employee benefit factors compelled a finding that a computer programmer was an independent contractor)).
[58] *See* Dennis S. Karjala, *Copyright and Creativity*, 15 UCLA ENT. L. REV. 169, 194-95 (2008) (noting that, if taken too far, "there is likely no work of authorship that is not a compilation under the statutory definition," since "[a] novel, for example, is a selection and arrangement of words (or letters), a musical work is a selection and arrangement of notes, and a painting is a selection and arrangement of forms and colors").
[59] *See* WEBSTER'S THIRD NEW INTERNATIONAL DICTIONARY 1117 (1981) (defining "*hypothecate*" as "to subject to a hypothec," and "*hypothec*" as "an obligation, right, or security given by contract or by operation of law to a creditor over property of the debtor without transfer of possession or title to the creditor").

would be more accurate to simply say that the terms of the transfer agreement that dictate the scope of the transferee's rights in the work.

Section 3.06, Comment *d* and the related Reporters' Note accurately describe the Copyright Office's current practices regarding registration—specifically, the fact that the Office permits only the author or the owner of all exclusive rights to be listed as the "claimant" in copyright registration. But contrary to Comment *d*, the Office does not regard the Copyright Act itself as imposing a limitation on who may be listed as a claimant; that limitation arises solely as a matter of regulation. The Office may decide in the future, in the interests of the national registration system, to reconsider the regulatory definition of a claimant.

Section 3.06, Comment *e*, deftly addresses the Ninth Circuit's decision in *Gardner v. Nike, Inc.*[60] We wonder whether it the Reporters should also address the Ninth Circuit's decision in *Sybersound Records, Inc. v. UAV Corp.*, which held that an exclusive transfer of the rights of a joint author did not give the transferee standing to sue.[61]

The Office welcomes public evaluation and discussion of our copyright law, and thanks the ALI and the Reporters for their attention to our comments.

Sincerely,

Sarang V. Damle
General Counsel and Associate Register of Copyrights

Robert J. Kasunic
Associate Register of Copyrights and Director of Registration Policy & Practice

[60] 279 F.3d 774 (9th Cir. 2002).
[61] 517 F.3d 1137, 1153 (9th Cir. 2008); *see also Corbello v. DeVito*, 777 F.3d 1058, 1065-66 (9th Cir. 2015) (arguably cabining *Sybersound* to its facts).

Addendum Regarding Preliminary Draft No. 3, § 2.03, comments (d) and (e)

The discussion of *Google Inc. v. Oracle America, Inc.* in the ALI's current preliminary draft of the Restatement suggests that the Reporters' understanding of the case differs from that of the U.S. Copyright Office ("Office"). As such, the Office hopes this lengthier discussion helps the Reporters reassess their views on the facts and the legal issues surrounding this case, and assists the ALI in its revision of the relevant portions of the draft Restatement.

As the Reporters are aware, the Federal Circuit held that the declaring code of the standard Java package library (sometimes called the "Java API") is copyrightable,[1] and in a CVSG brief, the United States agreed.[2] The Federal Circuit correctly applied the law in its holding. Moreover, the Preliminary Draft's suggestion that the Federal Circuit's decision was undermined by a later Ninth Circuit case called *Bikram's Yoga College of India, L.P. v. Evolation Yoga LLC* is incorrect; that case does not contradict the Federal Circuit's earlier holding on copyrightability.

BACKGROUND

A. Oracle's Library of Java Software

The *Oracle v. Google* case involves software written in the Java programming language. The Java language was developed by Sun Microsystems, Inc. ("Sun"). As the decisions below describe in greater detail, the major innovation of the Java platform was its ability to allow a programmer to create a single piece of software and have that software run in different types of computing environments. In doing so, Java "relieve[d] programmers from the burden of writing different versions of their computer programs for different operating systems or devices."[3] This feature of Java is facilitated through the use of the Java virtual machine—software developed by Sun that is installed on a computer that enables it to run Java programs. The Java virtual machine allows Java programs, which are written in a platform-neutral manner, to run on most computing platforms (Windows, Macintosh, Linux, etc.). As discussed below, however, Java programs cannot be run on Google's Android platform.

In addition to developing the Java programming language, Sun separately created a highly complex, reticulated library of prewritten Java software programs (the "Java package library" or "library") that can be used by programmers when writing their own software programs in Java.[4] The library is not self-executing software, however. Instead, it is essentially a collection of tens of thousands of specific computing tasks that programmers can use when creating their own Java

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

[2] Brief for the United States as Amicus Curiae, *Google Inc. v. Oracle Am., Inc.*, 135 S. Ct. 2887 (2015) (No. 14-410), *denying cert. to Oracle Am., Inc. v. Google Inc.*, 750 F.3d 1339 (Fed. Cir. 2014) [hereinafter "U.S. Br."].

[3] *Oracle*, 750 F.3d at 1348.

[4] *See* Barry Burd, JAVA FOR DUMMIES 38 (6th ed. 2014) (distinguishing between the "Java Language Specification," which defines the basic rules of syntax or "grammar" for the Java programming language—such as "[u]se an asterisk to multiply two numbers"—and the package library, which was added "after the language's grammar was defined" and includes prewritten code "rang[ing] from the commonplace to the exotic.").

programs. Technically speaking, programming in the Java programming language does not require the use of the Java package library. But, in practice, Java programmers regularly use elements of that library, albeit to varying degrees.[5]

At the heart of the Java package library are thousands of what are known as "classes," each of which contains a series of "methods" that can be used to issue instructions to the computer. Sun grouped these classes into a series of "packages." When it was first introduced in 1996, the Java package library included only "eight packages of pre-written programs."[6] Over time, Sun wrote more programs to carry out various functions "organized . . . into groups it called packages."[7] By 2008, the package library had grown from the original eight to 166 packages "with over six hundred classes, with over six thousand methods."[8] Google copied portions of 37 of those packages (including portions of the classes and methods within them).[9]

The Federal Circuit offered the following analogy: the "collection of API packages is like a library, each package is like a bookshelf in the library, each class is like a book on the shelf, and each method is like a how-to chapter in a book."[10] That analogy is slightly misleading, however, since it suggests that the packages and individual classes within them are independent of each other. In reality, the packages and classes are highly interrelated and interdependent—many classes rely upon the existence of other classes in the same package and in other packages in order to function.

In 2010, Oracle acquired Sun and as a result currently owns the copyrighted Java package library. Oracle does not claim copyright in the Java programming language itself—that is, the "words,

---

[5] The court of appeals noted that "three of [the] packages —java.lang, java.io, and java.util—[are] 'core' packages, meaning that programmers using the Java language had to use them 'in order to make any worthwhile use of the language.'" *Oracle*, 750 F.3d at 1349.

[6] *Id*.

[7] *Id*.

[8] *Id.* at 1349, 1351. Since 2008, Sun and Oracle have continued to add to the Java package library—there were two major releases of the Java Standard Edition library in 2011 and 2014, and each new release has added substantial new functionality to the Java package library. *See* Dustin Marx, *JDK 7: New Interfaces, Classes, Enums, and Methods*, JAVAWORLD (Mar. 31, 2011), http://www.javaworld.com/article/2074087/core-java/jdk-7--new-interfaces--classes--enums--and-methods.html; Alin Marian, *Java SE 8 and the New Date and Time Library*, DEVELOPER.COM (May 21, 2014), http://www.developer.com/java/java-se-8-and-the-new-date-and-time-library.html.

[9] *Oracle*, 750 F.3d at 1347. These included the "core" packages (java.lang, java.io, and java.util) but also a variety of specialized packages. *Id.* at 1349 & n.2. For example, Google copied portions of the java.beans package developed by Sun. That package establishes a specialized programming framework called "JavaBeans" that is used to create "reusable software components" that "can be manipulated visually in a builder tool," *i.e.*, a graphical development environment. *See JavaBeans FAQ: General Questions*, ORACLE, http://www.oracle.com/technetwork/java/javase/faq-135947.html. If a programmer creates software components (called "Beans") using the JavaBeans framework, other developers can then "use a graphical tool to connect some Beans together and make an application without actually writing any Java code—in fact without doing any programming at all." Robert Englander, DEVELOPING JAVA BEANS xi (1997).

[10] *Oracle*, 750 F.3d at 1349.

symbols, and other units, together with syntax rules for using them to create instructions."[11] Instead, the copyright claim to the Java package library includes (i) the many millions of lines of source code written by Sun to create the software programs in the library, and (ii) the overall structure, sequence and organization ("SSO") of that code—that is, the complex taxonomy of interrelated packages, classes, and methods.

The precise legal issue in the case involved the copyrightability of what has been referred to as the "declaring code" contained within the Java package library. The term "API" (which stands for "application programming interface") is often used to refer to the Java package library (or aspects of it). The term "API," however, is a frequently invoked idiom that can mean different things in different contexts—and it appears to have been a source of significant confusion during the proceedings.[12] Google used the term in a manner that suggests that the "API" for the Java package library is something distinct from the underlying code in the library.[13] But—as demonstrated below—what Google called the "API" is actually an integral part of the source code for the library. For this reason, the Office believes that the term "declaring code" is a better way to refer to the portions of the software that were at issue in this case. In any event, regardless of the terminology, as the Federal Circuit indicated, what is most important is an accurate understanding of what was copied.[14]

The overall framework of the Java package library is established through the declaring code—the declaring code names the classes, methods, and packages; defines basic features of those classes and methods; defines the relevant inputs or variables that a programmer must provide to the methods; and describes the outputs those methods will generate. The declaring code also indicates the additional code that a programmer needs to add to her own program to invoke the functionality of a chosen class or method. In this sense, the declaring code is integral to the use of the Java package library, just as is the prewritten line-by-line source code that the declaring code references and calls upon to carry out the functions intended by the programmer. This other prewritten code is referred to by the parties (and herein) as "implementing code."

Furthermore, it is undisputed that, when creating the Java package library, Sun had countless options for creating and arranging the classes, methods, and packages. Indeed, the Federal Circuit found that "[t]he evidence showed that Oracle had *unlimited* options as to the selection and arrangement of the 7000 lines Google copied."[15] Indeed, the court found that "a quick

---

[11] *Id*. at 1348.

[12] *See Oracle America, Inc. v. Google Inc.*, 810 F.Supp.2d 1002, 1011 (2011) ("Google, however, does not use the term API consistently in the relevant portions of its briefs, so it is unclear precisely what Google is attempting to characterize as a method of operation."); Federal Circuit Appellant Br. at 9 (referring to the term as a "verbal chameleon").

[13] *See, e.g.*, Pet. 5 ("Programmers access the set of pre-written programs through the Java application programming interface ('API')—a highly structured system with its own nomenclature.").

[14] *Oracle*, 750 F.3d at 1347.

[15] *Id.* at 1361 (emphasis added); *see also id*. ("This was not a situation where Oracle was selecting among preordained names and phrases to create its packages.").

examination of other programming environments shows that creators of other development platforms provide the same functions with wholly different creative choices," with specific examples from Apple's iOS and Microsoft's Windows Phone platforms.[16]

The district court likewise acknowledged that, with respect to the SSO of the Java package library, "[t]here was nothing in the rules of the Java language that required that Google replicate the same groupings even if Google was free to replicate the same functionality."[17] Specifically, the court found that "the very same functionality could have been offered in Android without duplicating the exact command structure used in Java."[18] The Federal Circuit affirmed that factual finding in full. In doing so, it noted that the "evidence showed, moreover, that Google designed many of its own API packages from scratch, and, thus, could have designed its own corresponding 37 API packages if it wanted to do so."[19] Significantly, Google did not dispute the finding that there were essentially unlimited options for the creation and arrangement of the classes, methods, and packages in the Java package library. To the contrary, the question presented in its petition for certiorari to the Supreme Court was premised on the conclusion that the software here "could have [been] written in more than one way."[20]

It is critical to understand that the declaring code does not and cannot itself be used to "operate" a Java program on a computer. Instead, as explained, the Java package library is akin to a set of partially assembled materials. Creating a functional Java program using the Java package library requires Java programmers to write code of their own that invokes the appropriate classes and methods within the library to do particular tasks as needed. Indeed, for all but the most rudimentary of Java programs, a Java programmer creates her own library of custom classes and methods to perform the precise tasks required for the particular program she is creating.

To take one relatively simple example, the Java package library provides a prewritten program that Java programmers can use to, among other things, open an internet connection using a "uniform resource locator," or URL, and to retrieve information from that connection. That software is embodied in a "class" called "URL" that resides within the java.net package. The class contains over 30 "methods"—that is, over 30 different routines—that can be used to perform those and other related functions.

Below is a brief excerpt to demonstrate what Google copied from a single source code file from the URL class within the Java package library (the excerpt includes only a few of the methods in this class, with asterisks indicating portions of the file not included here).[21] The URL class is

---

[16] *Id.* at 1368 n.14.

[17] *Oracle*, 872 F. Supp. 2d at 999.

[18] *Id.* at 976.

[19] *Oracle*, 750 F.3d at 1368.

[20] Pet. i.

[21] The full source code can be viewed online. *See Source for java.net.URL*, DEVELOPER.CLASSPATH.ORG, http://developer.classpath.org/doc/java/net/URL-source.html (last visited Nov. 30, 2017); *see also* Joint Appendix, *Oracle Am., Inc. v. Google Inc.*, 750 F.3d 1339 (Fed. Cir. 2014), at 700-705 (comparison of source code for the PolicyNodeImpl class in Java and Android) [hereinafter "C.A."].

included in the java.net package, one of the source code packages at issue in this case. The portions of the file that were copied appear in bold. The bracketed explanations indicate the nature of the particular declaring code that was copied, as well as implementing code that was not copied:

```
package java.net;                                          [package statement]

import java.io.IOException;                                [import statement]
import java.io.Serializable;                               [import statement]

* * *

public final class URL implements Serializable            [class declaration]
{
    * * *
    private String host;                                   [field declaration]
    * * *
    private int port = -1;                                 [field declaration]

    * * *

    public URL(String spec) throws MalformedURLException   [method declaration]
    {
        [method body (i.e., implementing code) not copied]
    }

    public Object getContent() throws IOException          [method declaration]
    {
        [method body (i.e., implementing code) not copied]
    }

    private static synchronized URLStreamHandler           [method declaration]
    getURLStreamHandler(String protocol)
    {
        [method body (i.e., implementing code) not copied]
    }
}
```

As can be seen, the declaring code for this single URL class—that is, what Google copied—includes the Java package statement, import statements, the class declaration, field declarations, and method declarations. Google extracted approximately 7,000 lines of source code from 37

packages in the Java package library—that is, the skeletal code for hundreds of Java classes and thousands of Java methods as developed by Sun.[22]

Although it copied the declaring code for these 37 packages, Google stopped short of reproducing the packages in their entirety. Rather than copy the implementing code that Sun wrote to effectuate the individual methods that are included within these packages, Google created its own versions of this code.

Although the syntax rules of the Java language limit to some extent the manner in which the declaring code is written,[23] they do not constrain how functionality is defined in a method declaration. Indeed, for a given task, at an appropriate level of abstraction, there may be any number of ways to write the declaring code. For example, the URL class above provides just one possible way to perform the general task of "opening an internet connection using a URL and retrieving information from that connection." The URL class in the Java package library can only be associated with a single URL, and the class itself provides methods for opening the connection to the website and gathering the information from that connection. An alternative approach using the Java language—one actually found in a third-party software package developed by the Apache Software Foundation in response to shortcomings in the URL class[24]—is to create a class ("HttpClient") that can execute HTTP requests to any number of websites, and create separate classes to manage requests to a server ("HttpGet") and responses from a server ("HttpResponse").[25]

As discussed above, the declaring code specifies the inputs, name, and functionalities of the various methods (*i.e.*, the prewritten computer programs) within the Java package library. When writing programs in the Java language, programmers can access these methods by writing code to reference and call upon necessary classes and methods in the Java package library, to which the programmer must add the appropriate inputs or variables that the computer needs to execute the desired methods.

It might be helpful to see how a Java programmer could make use of the URL class and the specific methods discussed above in writing her own program in the Java language. Below is an

---

[22] C.A. 1059-65 (comparing classes and methods in the Java package library to those copied into the Android environment).

[23] For instance, under those syntax rules, the terms "public" and "private" as used in the method declarations have a defined meaning, and those terms must appear first in the method declaration; the type of object or data type returned by the method must immediately precede the name of the method, or the word "void" must appear if the method returns nothing; the name of the method must be followed immediately by a parenthetical listing of the parameters to be passed by the programmer; etc.

[24] *See HttpComponents*, APACHE SOFTWARE FOUNDATION, https://hc.apache.org/httpcomponents-client-ga/ ("Although the java.net package provides basic functionality for accessing resources via HTTP, it doesn't provide the full flexibility or functionality needed by many applications. HttpClient seeks to fill this void by providing an efficient, up-to-date, and feature-rich package implementing the client side of the most recent HTTP standards and recommendations.") (last visited Nov. 30, 2017).

[25] *See HttpClient Tutorial: Chapter 1. Fundamentals*, APACHE SOFTWARE FOUNDATION, https://hc.apache.org/httpcomponents-client-ga/tutorial/html/fundamentals.html

rudimentary console-based program that pulls the image of the Copyright Office seal from the Copyright Office website so it can be used by the program. In this case, the program (by virtue of the System.out.println command) displays a message to the console with the name of the type of content that is retrieved. The program also "catches" any errors that are generated and displays those error messages to the user:

```
import java.net.URL;

public class ExamineURLContent {

  public static void main(String[] args) {

    try {
      URL urlObj = new URL("http://copyright.gov/about/images/office-seal.png");
      Object objectObj = urlObj.getContent();

      System.out.println("Object type: " + objectObj.getClass().getName());


    } catch (Exception err) {

      System.err.println(err.getMessage());
    }
  }
}
```

The declaring code is invoked in the program, and instructs the computer to execute the implementing code for a particular method according to the programmer's inputs, and the implementing code provides the line-by-line instructions to execute that method for those inputs. Both types of code are necessary to operate the programmer's program on a computer.

Programmers, however, use Oracle's declaring code only if they intend to incorporate methods from Oracle's Java package library into their programs. As the Federal Circuit explained, "[o]f course, once Sun/Oracle created 'java.lang.Math.max,' programmers who want to use that particular package have to call it by that name."[26] But there is no need for a programmer to use Oracle's declaring code; a software developer (like Google) can always create an entirely new library of methods.[27]

Another important point to understand is the limited extent to which Java programmers have actually memorized the packages, classes, and methods in the Java package library. According to one source, "a typical Java programmer" that lacked access to Oracle's explanatory materials

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

[27] *See id.* at 1361 ("[A]s the [district] court acknowledged, nothing prevented Google from writing its own declaring code, along with its own implementing code, to achieve the same result.").

for the package library "would be able to use less than 2 percent" of the methods in the library.[28] Java developers thus rely heavily on those explanatory materials to learn what classes and methods are available and what they do. In addition, most developers—including those using the Java library and the Android library—write their programs using what is known as an "integrated development environment" or "IDE."[29] Many popular Java IDEs will provide prompts to the programmer listing the various classes in a package, methods within a class, and parameters that the methods require, based on the particular package library being used.[30]

A more detailed explanation of the role and functioning of Java declaring code appears in the Appendix hereto.

B. Google's Library of Java Software

Google began developing Android software in 2005.[31] Android competes with Java in the mobile device market.[32] Google decided to replicate aspects of the Java platform in Android,[33] although that was not the only option available to Google. Both Apple and Microsoft, for example, built proprietary mobile platforms from scratch.[34] But Google did not.

Initially, Oracle and Google entered into negotiations to license the entire Java platform (including the Java package library) for use in Android. Negotiations for a paid license broke down, however, because of "Google's refusal to make the implementation of its programs compatible with the Java virtual machine or interoperable with other Java programs."[35] Oracle found that position to be "anathema to the 'write once, run anywhere' philosophy" underpinning the Java programming language.[36]

Instead of licensing Java or the Java package library, Google decided to use the Java programming language "to design its own virtual machine—the Dalvik virtual machine."[37] In addition, to attract the interest of Java programmers who were familiar with the Java package library, Google wanted to provide access to the same functionality for "the functions in the [Java package library] that were key to mobile devices."[38] Google identified 37 packages from the

---

[28] Burd, JAVA FOR DUMMIES at 39.

[29] *Id.* at 33-35.

[30] *See NetBeans IDE – The Smarter and Faster Way to Code*, *NETBEANS.ORG*, https://netbeans.org/features/index.html (last visited July 9, 2017).

[31] *Oracle*, 750 F.3d at 1350.

[32] *Id*.

[33] *Id*.

[34] *Id*. at 1360 n.5.

[35] *Id*. at 1350.

[36] *Id*.

[37] *Id*.

[38] *Id*.

Java package library that it "believed Java application programmers would want to find . . . in the new Android system callable by the same names as used in Java."[39]

To achieve that result, Google copied "the declaring source code from the 37 [Java package library] packages verbatim, inserting that code into parts of its Android software."[40] In other words, Google copied the package statements, import statements, class declarations, field declarations, and method declarations, from the vast majority of the classes, interfaces,[41] and other elements contained within those packages.[42] It appears that Google did not copy every class and interface in these 37 libraries, but it did copy 616 of the 677 classes and interfaces.[43] In copying that declaring code, Google also necessarily copied the complex and highly reticulated interrelationships between and among different classes and interfaces across the packages.[44]

Google stated that it copied the declaring code to give programmers access to the same "shorthand commands" and "basic vocabulary words" with which Java programmers were already familiar,[45] and that it "took pains to replicate only the elements necessary to allow programmers to use the shorthand commands."[46] But, notably, Google did not limit its copying to the declarations for *public* fields and methods—*i.e.*, those fields and methods within classes that can be invoked by a Java programmer in his own program. The record in the also appears to demonstrate instead that Google also copied the declarations for *private* fields and methods—*i.e.*, fields and methods that can only be invoked by implementing code within the pertinent class and that Java programmers can thus never directly invoke.[47]

Google then created its own implementing code for each of the methods within each of the 37 classes. To aid that effort, Google relied on the "specification" for the Java package library—the documentation Java provides to describe all the elements of the library and their relationships in plain English.[48] As explained by an Oracle witness at trial, once there is "a hierarchy of packages and classes and methods and fields, together with English descriptions of how

---

[39] *Id*.

[40] *Id*. at 1350-51.

[41] In the Java programming language, an "interface" is essentially a class that includes only method declarations but not implementing code. A class that "implements" such an interface must provide the implementing code for the method declarations specified in the interface. Interfaces are used to ensure that specific functionality is provided across different classes—in other words, a Java programmer will know that all classes that implement an interface will have certain functionality. For an extended discussion of these terms, see Appendix.

[42] *See, e.g.*, C.A.700-05 (side-by-side comparison of Java and Android versions of the PolicyNodeImpl class).

[43] C.A. 1065; Pet. App. 109.

[44] *Oracle*, 750 F.3d at 1350-51.

[45] Pet. 2.

[46] *Id*. at 30.

[47] *See* C.A. 700-05. For an extended discussion of public and private methods, see Appendix.

[48] *See Java 2 Platform Standard Edition 5.0 API Specification*, ORACLE, http://docs.oracle.com/javase/1.5.0/docs/api/ (last visited July 9, 2017).

everything is supposed to work together . . . to do an implementation from scratch is a relatively easier job."[49] Moreover, with respect to the Java library interfaces—which consist only of declaring code with no implementing code, but which require the programmer to add code in order to function—the Google source code was identical to the source code created by Sun.

While Android incorporates Java declaring code, it is not interoperable with the Java platform. As the court of appeals explained, "[a]lthough Android uses the Java programming language, it is undisputed that Android is not generally Java compatible" due to the fact that "Google ultimately designed Android to be *incompatible* with the Java platform, so that apps written for one will not work on the other."[50] Indeed, there was no evidence at trial that programs written in Java would be able to run in Android.[51]

DISCUSSION

I. The Legal Framework for Evaluating the Copyrightability of Computer Programs

To set the stage for the discussion of the issues presented in the *Oracle v. Google* litigation, it may be helpful to address the Copyright Office's understanding of some of the basic legal principles underlying those issues.

A. Computer Programs Are Eligible for Copyright Protection Under Section 102(a)

Today, the law is well-settled that computer programs generally are protected by copyright law, and are governed by the same doctrines as other types of works. In the Copyright Act of 1976, Congress acknowledged that copyright law covers computer programs, while simultaneously removing from 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 in such work."[52] Congress also created the National Commission on New Technological Uses of Copyrighted Works ("CONTU") to study issues raised by new technologies, including computers.[53] Congress eventually followed CONTU's recommendations to define "computer programs" in the Act and to amend section 117 to allow copies or adaptations of computer programs to be made either "as an essential step" of using the computer, or for archival purposes.[54]

---

[49] C.A. 22,405.

[50] *Oracle*, 750 F.3d at 1351 (internal quotation marks and citation omitted).

[51] *Id.* at 1371.

[52] 17 U.S.C. § 102(b).

[53] Pub. L. No. 93-573, § 201, 88 Stat. 1873, 1873-74 (1974); *see also* CONTU, FINAL REPORT 9 (1978) ("CONTU Report").

[54] *See* CONTU Report at 12-13; Act of Dec. 12, 1980, Pub. L. No. 96-517, § 10, 94 Stat. 3015, 3028-29.

Computer code can be copyrightable as a "literary work."[55] The Copyright Act defines a "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."[56] Copyright protects both the "source code," which consists of words, numbers, and symbols typed by a programmer, as well as the compiled "object code," which is used by the computer to carry out the instructions, but generally cannot be read by a human being.[57]

To qualify for copyright protection, a computer program—like any other work of authorship—must be original. The Supreme Court has explained that the originality requirement is "not particularly stringent."[58] The work must be independently created by the author (not be copied from another work) and must possess "at least some minimal degree of creativity."[59] The vast majority of works satisfy this requirement easily because most "possess some creative spark, 'no matter how crude, humble or obvious' it might be."[60] But the fact that a computer program is original and therefore *eligible* for copyright protection under section 102(a) does not necessarily mean that every aspect of the program is protected. The copyright in a computer program—or any other work of authorship—protects only the original expression that the author contributed to that work.

The Copyright Act provides the owner of copyright in an original work of authorship, the "exclusive rights to do and to authorize" specified things with that work.[61] As most relevant here, these include the rights to (1) reproduce the work in copies, (2) prepare derivative works based on the original work, (3) distribute copies of the work to the public "by sale or other transfer of ownership, or by rental, lease, or lending," and (4) display the work publicly.[62] Those exclusive rights are the same for a computer program as they are for a book or a song; they are not diminished simply because the software code has some functional purpose.

B. Section 102(b) and the "Idea/Expression" Dichotomy

Section 102(b) of the Copyright Act provides that "[i]n no case does copyright protection for an original work of authorship extend to 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 in such work."[63] This provision codifies a long-standing principle of

---

[55] 1-2A MELVILLE B. NIMMER AND DAVID NIMMER, NIMMER ON COPYRIGHT § 2A.10 (2015) ("NIMMER ON COPYRIGHT").

[56] 17 U.S.C. § 101.

[57] *See Apple Comput., Inc. v. Franklin Comput. Corp*., 714 F.2d 1240, 1248-49 (3d Cir. 1983).

[58] *Feist Publ'ns v. Rural Tel. Serv. Co.*, 499 U.S. 340, 358 (1991).

[59] *Id.* at 345.

[60] *Id.* (citation omitted).

[61] *See* 17 U.S.C. § 106.

[62] *Id.*

[63] 17 U.S.C. § 102(b).

copyright law known as the idea/expression dichotomy.[64] "Taken literally, 'idea' refers to the work's animating concept—the idea, for example, of a drama about two star-crossed lovers—while 'expression' refers to the precise words in which the playwright wrote the drama."[65] Thus, properly read, section 102(b) draws a line between *non-expressive* intellectual concepts or procedures—whether considered ideas, principles, processes, methods of operation, etc.—which are *not* subject to copyright protection, and the *expression* that embodies them, which is.[66]

Moreover, as the government explained to the Supreme Court, by its plain terms, "[s]ection 102(b) is not a limitation on what *kinds* of expressive works may be protected by a copyright."[67] Rather, "it is a limitation on how broadly the copyright [in an expressive work] *extends*."[68] Thus, section 102(b) codifies the important principle that "[c]opyright law protects the means of expressing ideas or concepts, but it does not give the copyright holder the right to exclude others from making use of the ideas or concepts themselves."[69] As discussed below, this is the crucial legal point in resolving the copyrightability dispute in the *Oracle v. Google* case.

This interpretation of the statutory text is buttressed by the legislative history of the Act, which illustrates that the purpose of section 102(b) was to codify the dichotomy between non-expressive idea and expression. The House Report on the 1976 Act, considered an authoritative source for the meaning of the Act,[70] stated expressly that "[s]ection 102(b) in no way enlarges or contracts the scope of copyright protection under the present law," but "[i]ts purpose is to restate, in the context of the new single Federal system of copyright, that the basic dichotomy between expression and idea remains unchanged."[71] Particularly notable is the House Report's discussion of the relevance of section 102(b) to the scope of protection for computer programs under the Act:

> Some concern has been expressed lest copyright in computer programs should extend protection to the methodology or processes adopted by the programmer, rather than merely to the "writing" expressing his ideas. Section 102(b) is intended, among other things, to make clear that the expression adopted by the programmer is the copyrightable element in a computer program, and *that the*

[64] *Golan v. Holder*, 565 U.S. 302, 328 (2012) ("The idea/expression dichotomy is codified at 17 U.S.C. § 102(b).").

[65] PAUL GOLDSTEIN, GOLDSTEIN ON COPYRIGHT § 2.3.1 (2015).

[66] 1-2 NIMMER ON COPYRIGHT § 2.03[D][1] ("[A]lthough Section 102(b) denies that copyright may 'extend to' an 'idea, procedure, process,' as contained in a given work, it does not deny copyright to the work itself, merely because it consists of an 'idea, procedure, process,' etc.").

[67] U.S. Br. at 12 (emphasis added).

[68] *Id.* (emphasis added).

[69] *Id.*

[70] *See, e.g., Feist Publ'ns*, 499 U.S. at 355.

[71] H.R. REP. NO. 94-1476, at 57 *reprinted in* 1976 U.S.C.C.A.N. 5659, 5670.

> *actual processes or methods embodied in the program are not within the scope of the copyright law*.[72]

Thus, the idea/expression dichotomy, as applied to software, excludes from copyright protection the abstract "methodology or processes adopted by the programmer" in creating the code.[73] It makes clear that the copyright law does not prevent anyone from studying the code for an existing program for the purpose of identifying the underlying ideas or processes embodied in that program. Nor does it prevent them from writing new routines or entirely new programs performing those same functions. Section 102(b) allows use of any of the ideas, methods, or other insights that make the program work—so long as the specific lines of code from the existing program are not copied.

II. The Declaring Code at Issue in *Oracle v. Google* was Eligible for Copyright Protection

A. As the Government Explained to the Supreme Court, Section 102(b) Does Not Exclude Declaring Code from Copyright Protection

As explained in the Government's brief submitted to the Supreme Court, the original source code copied quite clearly falls on the "expression" side of the idea/expression dichotomy, and is not an unprotectable "system" or "method of operation."

To be sure, as the government observed in its Supreme Court brief, computer code differs in a fundamental way from many traditional means of literary expression, such that, if the Copyright Act did not explicitly reference computer software, one might not think of it as protectable authorship.[74] Code, unlike other categories of protectable literary expression, constitutes both expression and the actual means by which a computer is induced to perform the desired function.[75] But the Copyright Act specifies that computer code is protectable expression, and therefore its functional nature does not preclude its protection. The uncopyrightable "method of operation" or "system" or "process" is the underlying computer function triggered by the written code—for example, an algorithm that the computer executes to sort a data set. The code itself is eligible for copyright protection, even though it causes a computer to function.

Nothing about the declaring code at issue in the *Oracle v. Google* case materially distinguishes it from other kinds of computer code. Indeed, as the discussion above and the example provided in the Appendix make clear, that declaring code is unquestionably expressive computer code that is eligible for copyright protection under section 102(a). The only real question in the *Oracle v. Google* case was whether section 102(b) excluded that computer code from copyright protection. But, as explained, section 102(b) is simply not a limitation on what *kinds* of expressive works

---

[72] *Id.*; *see also* CONTU Report at 22 ("[C]opyright leads to the result that anyone is free to make a computer carry out any unpatented process, but not to misappropriate another's writing to do so.").

[73] H.R. REP. NO. 94-1476, at 57 *reprinted in* 1976 U.S.C.C.A.N. 5659, 5670.

[74] U.S. Br. at 13.

[75] *Id.*

may be protected by a copyright. Rather, it is a limitation on how broadly the copyright in expressive works "extends." Thus, as the government observed in its Supreme Court brief, it is incorrect to argue that a work can be a "protectable work of authorship" under section 102(a) as well as a "system" or "method of operation" under section 102(b).[76] If a work constitutes expression (and if it is original), it is copyrightable under section 102(a). Section 102(b) merely excludes from copyright protection the subject matter explained or described in the expressive work.

Any argument that the declaring code may not be protected because it is "functional"—as the ALI Preliminary Draft suggests—is thus at odds with Congress' desire to protect computer programs. As noted, under the statutory definition, computer programs consist of written expression designed to "bring about a certain result" in a computer.[77] Congress could not have intended to bar copyright protection for a computer program merely because it serves a functional or utilitarian purpose.[78] Section 102(b) must be read in light of Congress's decision to extend copyright protection to computer programs; it cannot be read to exclude software merely because it can at some level be described as a "method of operation" or "system." If this were the actual meaning of section 102(b), no computer program would be protected.[79]

Nor does it matter that the declaring code defines what programmers must learn in order to use the Java package library. To begin with, this view is in tension with the statute, which defines a 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."[80] This broad terminology indicates that Congress intended to protect all kinds of instructions, without distinguishing between user-based or non-user-based instructions.

---

[76] U.S. Br. at 13.

[77] 17 U.S.C. § 101 (defining computer program as a set of statements or instructions used "in a computer in order to bring about a certain result").

[78] Like computer programs, Congress recognized that architectural works are utilitarian, yet still chose to protect them because architecture is "an art form that performs a very public, social purpose." H.R. REP. NO. 101-735 (1990), *reprinted in* 1990 U.S.C.C.A.N. 6935, 6943. Although Congress specified that individual standard features, such as common windows or doors, would not be protected, it emphasized that section 102(b) would not exclude protection for individual features reflecting the architect's creativity. *Id.* at 6949. Consequently, not only did Congress intend for the artistic expression embodied in an architectural work to be protected, but more importantly, it expressly provided for architectural works to be protected to the same degree as other forms of protected subject matter. *Id.* at 6952.

[79] *See, e.g.*, Jane C. Ginsburg, *Four Reasons and a Paradox: The Manifest Superiority of Copyright over Sui Generis Protection of Computer Software*, 94 COLUM. L. REV. 2559, 2569-70 (1994).

[80] 17 U.S.C. § 101 (emphasis added). Congress repeatedly used similar language elsewhere in the statute. *See, e.g., id.* ("'Copies' are material objects . . . in which a work is fixed by any method now known or later developed, and from which the work can be perceived, reproduced, or otherwise communicated, *either directly or with the aid of a machine or device*.") (emphasis added); *id.* § 102(a) ("Copyright protection subsists . . . in original works of authorship fixed in any tangible medium of expression, now known or later developed, from which they can be perceived, reproduced, or otherwise communicated, *either directly or with the aid of a machine or device*.") (emphasis added).

That reasoning would also seemingly eliminate copyright protection for all user interfaces and screen displays, contrary to established case law and more than two decades of Copyright Office practice.[81] Since the 1980s, the Office has recognized that user interfaces may be copyrightable, and its registration practices expressly recognize that the copyright in a computer program extends to any copyrightable screen displays generated by the copyrighted code.[82] Professor Nimmer has opined that extending copyright protection in the computer program to its screen displays is the "correct approach,"[83] and a number of courts have held that user interfaces and screen displays are copyrightable.[84]

That reasoning is also in tension with the CONTU report. Notably, Commissioner Hersey had argued in his dissent that copyright should protect a computer program during the development phase when it is capable of being seen or heard by a human being, but should "not extend to a computer program in the form in which it is capable of being used to control computer operations."[85] But the CONTU Commission considered and rejected that proposal, concluding

---

[81] *See Engineering Dynamics, Inc. v. Structural Software, Inc.*, 26 F.3d 1335, 1343-44 (5th Cir. 1994); *Autoskill Inc. v. Nat'l Educ. Support Sys., Inc.*, 994 F.2d 1476, 1492-96 (10th Cir. 1993).

[82] *See* COMPENDIUM OF U.S. COPYRIGHT OFFICE PRACTICES §§ 721.10(A); 1509.1(C)(7) (3d ed. 2014); Registration of Claims to Copyright Deposit Requirements for Computer Programs Containing Trade Secrets and for Computer Screen Displays, 54 Fed. Reg. 13,177 (Mar. 31, 1989); Registration Decision; Registration and Deposit of Computer Screen Displays, 53 Fed. Reg. 21,817, 21,819-20 (June 10, 1988). For purposes of registration, user interfaces are classified as literary works because they are usually generated by source code, and as such, are considered part of the computer program. Although the interface may contain pictorial or audiovisual elements, the computer program usually constitutes the predominant type of authorship in the work. When a work contains more than one type of authorship, the Office advises applicants to select the class of authorship that corresponds to the predominant form of authorship in the work. For example, if a website contains a substantial amount of text combined with a few photographs, the Office would classify that work as a literary work. If a website mostly contains photographs with a small amount of text, the Office would classify that work as a pictorial work. If the types of authorship are roughly equal (*e.g.*, 50% text and 50% photographs), the work could be classified in either category that would be appropriate for those types of authorship. *See* COMPENDIUM § 609.2(C).

[83] 2 MELVILLE & DAVID NIMMER, NIMMER ON COPYRIGHT § 7.18[C][3] (2014).

[84] *See, e.g., BUC Int'l Corp. v. Int'l Yacht Council Ltd.*, 489 F.3d 1129, 1149 (11th Cir. 2007) (noting that *Mitek* "recognized that 'a user interface, [such as] a screen display (itself an audiovisual work), may be entitled to copyright protection as a compilation.'"); *Atari Games Corp. v. Oman*, 888 F.2d 878, 882-86 (D.C. Cir. 1989) (screen display, though perhaps not sufficiently protected by copyright in the computer program, deemed copyrightable as an audiovisual work); *Broderbund Software, Inc. v. Unison World, Inc.*, 648 F. Supp. 1127, 1131-34 (N.D. Cal. 1986) (copyright protection in computer program found to extend to the user interface and screen displays); *Digital Commc'ns Assocs., Inc. v. Softklone Distrib. Corp.*, 659 F. Supp. 449, 453, 462-63 (N.D. Ga. 1987) (screen display copyrightable as a compilation of program terms through its expressive arrangement and design); *Mfrs. Technologies, Inc. v. Cams, Inc.*, 706 F. Supp. 984, 993 (D. Conn. 1989) (registration for computer program extended to user interface); *O.P. Solutions, Inc. v. Intellectual Prop. Network, Ltd.*, 1999 U.S. Dist. LEXIS 979, at *29-32 (S.D.N.Y. Feb. 1, 1999) (noting user interfaces are routinely deemed copyrightable as nonliteral elements of computer programs, while the constituent elements of user interfaces have been protectable as compilations).

[85] *See* CONTU report at 27-37 ("The line should be drawn at the moment of the program's transformation, by whatever present or future technique, to a mechanical capability. This is the moment at which the program ceases to communicate with human beings and is made capable of communicating with machines.").

that copyright should protect elements of a computer program that are comprehensible to users and programmers, as well as elements that can be interpreted only by the computer itself.[86]

In addition, though it may be convenient for programmers familiar with the Java package library that Sun developed to use the same class and method calls when writing programs for Android or some other platform, this convenience, too, does not render the code unprotectable. Both declaring code and implementing code ultimately perform the same practical function: They instruct a computer to work, and both are necessary components of a Java or Android method. Though declaring code is one step further removed from actually causing a computer to perform the requested function, there is no basis to conclude that section 102(b)'s application depends on the directness of the link between code and its function. Further, as explained, the purpose of section 102(b) is not to distinguish copyrightable from uncopyrightable parts of an original work of authorship; it is to distinguish between the work itself and the underlying idea, abstract concept, or system.

B. Subsequent Case Law Does Not Cast Doubt on The Federal Circuit's Decision

No new cases have been decided that cast doubt on the government's interpretation of section 102(b) or its position in the *Oracle v. Google* litigation. The Preliminary Draft No. 3, § 2.03, comment *d*, appears to suggest that the Ninth Circuit's decision in *Bikram's Yoga College of India v. Evolation Yoga*[87] undermined the Federal Circuit's decision. In the *Bikram* case, the court explained that, although the plaintiff Bikram Choudhury possessed a copyright registration for his 1979 book containing a description of a yoga sequence, the copyright in that book did not extend to the sequence itself. Rather, "[a]s Choudhury describes it, the Sequence is a 'system' or a 'method' designed to 'systematically work every part of the body, to give all internal organs, all the veins, all the ligaments, and all the muscles everything they need to maintain optimum health and maximum function.'"[88] The court explained that Choudhury could not "secure copyright protection for a healing art: a system designed to yield physical benefits and a sense of well-being," and "if [the sequence] is entitled to protection at all, that protection is more properly sought through the patent process."[89] The court also rejected the sequence's protectability as a compilation, and in doing so, noted that "[i]t makes no difference that similar results could be achieved through a different organization of yoga poses and breathing exercises. . . . [T]he possibility of attaining a particular end through multiple different methods does not render the uncopyrightable a proper subject of copyright."[90] Rather, the sequence remains a process that is ineligible for protection.

---

[86] *See id.* at 25 ("[C]ourts have assiduously avoided adopting the critic's role in evaluating the aesthetic merits of works of authorship. To attempt to deny copyrightability to a writing because it is capable of use in conjunction with a computer would contravene this sound policy.").

[87] 803 F.3d 1032 (9th Cir. 2015).

[88] *Id.* at 1038.

[89] *Id.* at 1039.

[90] *Id.* at 1042.

This outcome accords with the Office's opinions on the matter. In June 2012, the Office published a statement of policy in the Federal Register explaining that a compilation of yoga poses would not be copyrightable under section 102(a).[91] The Office made clear in this policy statement that a compilation of yoga poses did not constitute copyrightable subject matter under section 102(a) because a compilation of yoga poses would not result in a work of authorship that could fit into any of the eight categories of copyrightable subject matter enumerated in the statute.[92] The closest possible analogue was "choreography," but the Office concluded that "[c]horeographic authorship is considered, for copyright purposes, to be the composition and arrangement of a related series of dance movements and patterns organized into a coherent, and expressive whole," and that "functional physical movements such as sports movements, exercises, and other ordinary motor activities alone do not represent the type of authorship intended to be protected under the copyright law as a choreographic work."[93] Additionally, the Office noted that a compilation of yoga poses could also be denied protection under section 102(b); the Office explained that while "[a] film or description of an exercise routine or simple dance routine may be copyrightable, as may a compilation of photographs of such movements . . . such a copyright will not extend to the movements themselves, either individually or in combination."[94]

Though some have argued otherwise, the result in *Bikram* is entirely consistent with the views the government has previously expressed with respect to the copyrightability of the declaring code in *Oracle v. Google*. At bottom, the Ninth Circuit in *Bikram* held that while a book describing yoga poses was indisputably copyrightable, that copyright did not extend to prevent others from practicing those poses.[95] As the court explained, the question was not about "the validity of a copyright but rather its scope."[96] *Bikram*'s holding thus reflects precisely the interpretation of section 102(b) that the government advocated in its brief in *Oracle v. Google*. As the government explained, "[s]ection 102(b) is not a limitation on what kinds of expressive works may be protected by a copyright" but "a limitation on how broadly copyright extends."[97] Indeed, the government provided an example of section 102(b)'s operation that is closely analogous to the facts of the *Bikram* case: "Although a book on how to build a bicycle may be eligible for copyright protection, that copyright does not include any exclusive right to practice the bicycle-building method that the book explains[.]"[98]

---

[91] Registration of Claims to Copyright, 77 Fed. Reg. 37,605 (June 22, 2012).

[92] Registration of Claims to Copyright, 77 Fed. Reg. 37,605, 37,607 (June 22, 2012).

[93] Registration of Claims to Copyright, 77 Fed. Reg. 37,605, 37,607 (June 22, 2012).

[94] Registration of Claims to Copyright, 77 Fed. Reg. 37,605, 37,607 (June 22, 2012).

[95] 803 F.3d at 1038 ("Does Choudhury's copyright protection for his 1979 book extend to the Sequence itself? Under the fundamental tenets of copyright law and consistent with the precedents discussed above, the answer is no.").

[96] *Id.*

[97] Brief at 12.

[98] Brief at 12.

That having been said, some of the reasoning of the *Bikram* case might be read to suggest a competing view of section 102(b)—that something that constitutes an expressive work of authorship can nonetheless be uncopyrightable because it is a "system" or a "method" under section 102(b). Specifically, in addressing the claim that the sequence of yoga poses constitutes a copyrightable choreographic work, the *Bikram* court concluded that it "need not decide whether to adopt the Copyright Office's definition of 'choreographic work' or fashion another on [its] own because all categories of works eligible for copyright protection, including choreographic works, are subject to the critical requirements and limitations of Section 102."[99] The court then says that the sequence of yoga poses "is not copyrightable as a choreographic work" because "it is an idea, process, or system to which copyright protection may '[i]n no case' extend."[100] One could argue, then, that the Ninth Circuit's decision effectively held that even though a sequence of yoga poses might be considered an expressive work of choreography, it is nonetheless excluded from copyright protection by section 102(b).

The Copyright Office thinks that such an argument would represent a significant over-reading of *Bikram.* Nowhere in *Bikram* does the court expressly conclude—or even assume *arguendo*—that the sequence of yoga poses could constitute an expressive work under section 102(a). At most, the court notes that the sequence might fit within "some colloquial definitions of dance or choreography," or that it "could be . . . characterized as a form of dance."[101] Moreover, unlike in the *Oracle v. Google* litigation, where Google conceded that the software code is expressive, the defendant in *Bikram* disputed the claim that the sequence constituted an expressive work of authorship under section 102(a).[102]

Accordingly, the Office believes that *Bikram* should be understood as essentially holding that a sequence of yoga poses is not copyrightable because it does not constitute an expressive work of authorship. The court in *Bikram* repeatedly emphasized that the sequence of yoga poses "serve[s] basic functional purposes," is made up of "routinized physical movements," and is a "functional physical sequence[]."[103] These facts are what remove the sequence from the category of "choreography" under section 102(a). The Office's 2012 policy statement addressed in detail the question of whether yoga poses constitute copyrightable subject matter under section 102(a).[104]

---

[99] *Bikram*, 803 F.3d at 1043.

[100] *Id.* at 1044.

[101] *Id.*

[102] Appellees' Answering Brief at 17-24, *Bikram's Yoga College of India v. Evolation Yoga*, No. 13-55763 (9th Cir. Jan. 13, 2014).

[103] *Bikram*, 803 F.3d at 1044.

[104] Registration of Claims to Copyright, 77 Fed. Reg. 37,605, 37,605-07 (June 22, 2012). The policy statement also noted that attempts to claim copyright in yoga poses, as such, could also be rejected under section 102(b), but again, that should be understood as addressing the situation where a copyrightable work is said to extend to the poses themselves. *See* 77 Fed. Reg. at 37,607 ("A film or description of . . . an exercise routine may be copyrightable, as may a compilation of photographs of such movements. However, such a copyright will not extend to the movements themselves, either individually or in combination. But only to the expressive description, depiction or illustration of the routine that falls within a section 102(a) category of authorship.").

It concluded that where such poses were not expressive choreography but were simply "said to result in improvements in one's health or physical or mental condition," they could not be protected by copyright.[105]

In contrast, if George Balanchine had written a book outlining the choreography of the New York City Ballet's *Swan Lake*, the result would be different. In that case, Balanchine would be entitled to a copyright claim in the book as a literary work, and also a copyright claim in the indisputably expressive choreography described in the book. And, significantly, even if Balanchine claimed that performance of the *Swan Lake* choreography was beneficial to human health, that functionality would not deprive the entire work of copyright protection. The Office does not believe that the *Bikram* court intended to hold otherwise. Accordingly, the *Bikram* decision does not preclude the Ninth Circuit, in a future case, from holding that a work that is truly expressive (*e.g.*, choreography, photography, software code) is nevertheless protectable even if it serves some function (*e.g.*, conveys health benefits, instructs a computer).

Furthermore, the court of appeals in *Bikram* did not mention, much less disagree with, a key piece of reasoning in the Federal Circuit's copyrightability decision: the fact that software code is protectable by copyright means that section 102(b) cannot be read to exclude "functional" works from copyright protection. Given that the *Bikram* court nowhere grappled with this point, it would be difficult to posit that the court nevertheless intended to contradict the Federal Circuit's copyrightability decision in this case. Indeed, in a later district court decision, *Cisco Sys., Inc. v. Arista Networks, Inc.*, regarding "command line interfaces" ("CLI") used by customers to configure or check settings of ethernet switches, distinguished *Bikram* on precisely these grounds.[106]

[105] Registration of Claims to Copyright, 77 Fed. Reg. 37,605, 37,607 (June 22, 2012).

[106] *Cisco Sys., Inc. v. Arista Networks, Inc.*, Dkt. 482, No. 14 Civ. 05344, 14 (N.D. Cal. Aug. 23, 2016).

Appendix to
Addendum Regarding Preliminary Draft No. 3, § 2.03

Overview of the Specific Elements within Oracle's Declaring Code

To illustrate what was copied from the Java package library, this appendix explains each line of the following source code from the library's URL class, discussed in section I.A above. As noted above, ellipses indicate portions of the code omitted from the excerpt; copied code appears in bold; and the bracketed explanations indicate the type of declaring code copied, as well as implementing code that was not copied:

```
package java.net;                                          [package statement]

import java.io.IOException;                                [import statement]
import java.io.Serializable;                               [import statement]

* * *

public final class URL implements Serializable            [class declaration]
{
    * * *
    private String host;                                   [field declaration]
    * * *
    private int port = -1;                                 [field declaration]

    * * *

    public URL(String spec) throws MalformedURLException   [method declaration]
    {
        [method body (i.e., implementing code) not copied]
    }

    public Object getContent() throws IOException          [method declaration]
    {
        [method body (i.e., implementing code) not copied]
    }

    private static synchronized URLStreamHandler           [method declaration]
    getURLStreamHandler(String protocol)
    {
        [method body (i.e., implementing code) not copied]
    }

}
```

The first line of the source code for the URL class is called the "package statement":

package java.net;

This statement assigns the class to a "package," or a particular grouping of classes—here, the URL class is assigned to the "java.net" package.[107] Sun organized its full library of prewritten software programs into 166 such packages, including java.net, java.io, java.lang, java.math, etc.[108] Google copied declaring code from 37 of these packages, representing 616 classes. These 616 classes, in turn, encompass over 6,000 individual methods, or program routines.[109]

The next lines of source code are the "import statements":

import java.io.IOException;
import java.io.Serializable;

These statements are used so that classes (or related elements like exceptions or interfaces) that are contained in packages other than java.net can be used in the URL class more easily. Without these lines, each time a class from a different package is referenced in the remainder of the source code, one would have to type out the full package path, *e.g.*, "java.io.IOException" or "java.io.Serializable."[110]

The next line of code cited in the example above is the "class declaration":

public final class URL implements Serializable

This declaration defines the basic features of the URL class:

- The term "public" means that the class can be called by Java programmers that are using the Java package library. Without that signifier, the class could be "package-private," meaning that it could only be called by implementing code within other classes in the same package.[111]

- The term "final" means that the class cannot be "extended"—it prohibits a programmer from creating a subclass that "inherits" the functionality (*i.e.*, the fields and methods) of

---

[107] *See The Java Tutorials, Creating and Using Packages*, ORACLE JAVA DOCUMENTATION, http://docs.oracle.com/javase/tutorial/java/package/packages.html (last visited Mar. 11, 2015).

[108] *Oracle*, 750 F.3d at 1349. The Java packages from the pertinent version of Java are available to browse online. *Java 2 Platform Standard Edition 5.0 API Specification*, ORACLE, http://docs.oracle.com/javase/1.5.0/docs/api/ (last visited Mar. 11, 2015).

[109] C.A. 1065; Pet. App. 109.

[110] The one exception to this rule is that the "java.lang" package is imported by default into all source code.

[111] *See The Java Tutorials, Controlling Access to Members of a Class*, ORACLE JAVA DOCUMENTATION, http://docs.oracle.com/javase/tutorial/java/javaOO/accesscontrol.html (last visited Mar. 11, 2015).

the URL class.[112] In contrast, a related class in the same java.net package, URLConnection, does not specify that it is "final," which means that a programmer may create a subclass that extends the class and adds additional functionality.[113]

- The words "class URL" indicate that the source code establishes a "class" (rather than some other Java element like an "interface" or an "exception") and names the class "URL."

- The words "implements Serializable" mean that the class is an implementation of what is known in Java as an "interface"—in this case, an interface called "Serializable" (the same "Serializable" referenced in the import statement "java.io package" above). Interfaces are an important concept in Java and are discussed below.

The next lines of code quoted in the example above are the "field declarations":

private String host;
private int port = -1;

Classes contain "fields"[114] and "methods."[115] Fields are simply variables that can be assigned to objects of a particular class. Fields can hold one of eight "primitive data types," including an integer, decimal number, or character; fields can alternatively be an object of any particular class. For example, the excerpt from the URL class above shows two fields. The field called "port" is an integer, and is used to hold the port of a remote server to which a URL object will connect, and is set to the value of -1 by default.[116] The field called "host" is an object of the class "String." The String class is found in the java.lang package, and String objects are used to represent

---

[112] *See Oracle*, 872 F. Supp. 2d at 980; *see also The Java Tutorials, Inheritance*, ORACLE JAVA DOCUMENTATION, http://docs.oracle.com/javase/tutorial/java/IandI/subclasses.html (last visited Mar. 11, 2015); *Writing Final Classes and Methods*, ORACLE JAVA DOCUMENTATION, http://docs.oracle.com/javase/tutorial/java/IandI/final.html (last visited Mar. 11, 2015).

[113] For example, the java.net package contains another class, HttpURLConnection, that inherits the functionality of the URLConnection class and adds its own functionality specific to the hypertext transfer protocol. The HTTPConnection class is defined using the following code: "`public abstract class HttpURLConnection extends URLConnection`." *See Source for java.net.HttpURLConnection*, DEVELOPER.CLASSPATH.ORG, http://developer.classpath.org/doc/java/net/HttpURLConnection-source.html (last visited Mar. 11, 2015). Still another class, the HttpsURLConnection class contained in the javax.net.ssl package, extends the functionality of HttpURLConnection, and adds functionality specific to the hypertext transfer protocol secure. *See Class HttpsURLConnection*, ORACLE, http://docs.oracle.com/javase/7/docs/api/javax/net/ssl/HttpsURLConnection.html (last visited Mar. 11, 2015).

[114] *The Java Tutorials, Declaring Member Variables*, ORACLE JAVA DOCUMENTATION, http://docs.oracle.com/javase/tutorial/java/javaOO/variables.html (last visited Mar. 11, 2015) (noting that "member variables in a class . . . are called *fields*).

[115] *See The Java Tutorials, Defining Methods*, ORACLE JAVA DOCUMENTATION, http://docs.oracle.com/javase/tutorial/java/javaOO/methods.html (last visited Mar. 11, 2015).

[116] *The Java Tutorials, Primitive Data Types*, ORACLE JAVA DOCUMENTATION, http://docs.oracle.com/javase/tutorial/java/nutsandbolts/datatypes.html (last visited Mar. 11, 2015).

character strings (*e.g.*, "abcde").[117] In the URL class, the "host" field is used to hold the name of the remote server to which the URL connects (*e.g.*, www.copyright.gov).

The following are the "method declarations" for three of the methods in the URL class:

> **public URL(String spec) throws MalformedURLException**
>
> **public Object getContent() throws IOException**
>
> **private static synchronized URLStreamHandler getURLStreamHandler(String protocol)**

Methods are what provide specific functionality in a program. A method consists of the method declaration and the method body.[118] A method declaration (also referred to as the "method header") contains several elements that define and describe the operation of the method, including "the name of the method; the number, order, type and name of the parameters used by the method; the type of value returned by the method; the checked exceptions that the method can throw; and various method modifiers that provide additional information about the method."[119] The method body is the "block of code that . . . implements the method" [120]; those blocks of code have been referred to in the course of the litigation as "implementing code."

Java allows developers to control access to fields and methods within a class. "Public" fields and methods are those that can be used or invoked by implementing code from within any other class, including code written by third-party Java programmers who write their own programs using the Java package library. In contrast, fields and methods designated as "private" may only be used or invoked by implementing code that is included within the class itself. For instance, in the case of the URL class, the fields "port" and "host" are private, and cannot be directly used by third-party programmers. Instead, those fields can only be set and accessed by the implementing code for the methods within the URL class. Similarly, the "getURLStreamHandler" method is private, and can only be called from within the URL class. Private methods are often used when multiple methods within a class need to run a common subroutine; rather than repeating the same lines of code in each such method, the developer can create a private method that the other methods can each call as needed.[121] By making the method private, the developer ensures that the method cannot be invoked within other classes or by programmers writing their own code.

---

[117] *Class String*, ORACLE, https://docs.oracle.com/javase/8/docs/api/java/lang/String (last visited Mar. 11, 2015).

[118] *Oracle*, 872 F. Supp. 2d at 979.

[119] *Id.*

[120] *Id*. at 980.

[121] "Package-private" fields and methods may be called from within all classes in the same package. "Protected" fields and methods can be called from within that class and any subclasses that extend that class. *See generally The Java Tutorials, Controlling Access to Members of a Class*, ORACLE JAVA DOCUMENTATION, http://docs.oracle.com/javase/tutorial/java/javaOO/accesscontrol.html (last visited Mar. 11, 2015).

The method declaration "public URL(String spec) throws MalformedURLException" describes what is known as a "constructor" method. A constructor method is used to create a new object of the specific class. For example, a Java programmer could call the constructor method in the following manner: URL u = new URL("http://wwww.copyright.gov/"). That code creates a new URL object connected to the Copyright Office main page. The words "throws MalformedURLException" indicate that the method will kick out a specific kind of error. In Java, such errors are communicated using objects of a special type called "exceptions." The exception in this case is of the type "MalformedURLException," signifying that the error has to do with the form of the URL.

The method declaration "public Object getContent() throws IOException" references a method that Java programmers use to pull content from a URL and save it as an object in their own programs. This method declaration indicates that the method will kick out a different type of error—IOException—if there is an "input/output" error.

The method declaration "private static synchronized URLStreamHandler getURLStreamHandler(String protocol)" defines a method that creates an object of the class type URLStreamHandler for a given communication protocol (http, https, ftp, etc.). The URLStreamHandler class was developed to handle the mechanics of connecting to a remote server using a specific protocol. The key point to understand with respect to this method is that it is designated as "private"—it provides functionality that is and can only be invoked by implementing code *within* the URL class.

Returning to the "interface" concept, an interface is a Java element that specifies methods that implementing classes *must* provide. It is created in a similar manner to a class, but the interface source code does not include any implementing code—the interface contains only the method declaration. For example, the full source code for the interface "Comparable" in the java.lang package (used for objects that can be ordered among other objects) reads as follows[122]:

```
package java.lang;
public interface Comparable<T>
{
    int compareTo(T o);
}
```

All classes "implementing" the Comparable interface must then define how objects of that class can be compared with other objects by providing code to implement the "compareTo" method. Classes can implement multiple interfaces—a class doing so must provide the methods required by each of those interfaces.[123]

---

[122] *Source for java.lang.Comparable*, *DEVELOPER.CLASSPATH.ORG*, http://developer.classpath.org/doc/java/lang/Comparable-source.html#line.../../java/lang/Comparable.java:71 (last visited Mar. 11, 2015).

[123] *Oracle*, 872 F. Supp. 2d at 980 (note that "interfaces" here "refers to something different from the word 'interface' in the API acronym").

* * * * *

From the above discussion of an extremely brief excerpt from a relatively simple class, it is apparent that the declaring code used in the Java package library does several things. Viewed somewhat simplistically from a user's perspective, the declaring code defines the functionality provided by the underlying implementing code and identifies the words that the programmer should use (together with the relevant inputs or variables the programmer wants to provide to the computer) to invoke that functionality in the programming process. It also, however, expresses the thousands of elements that Sun chose to include in the Java package library (each class, interface, exception, etc.), as well as the key features of each of those elements (whether they are public, final, etc.). In addition, the declaring code embodies the complex organization—that is, the overall taxonomy—that Sun created for the various packages, classes and methods within the library. It defines the complex interrelationships among those elements—what classes are subclasses of other classes, what classes implement what interfaces, what types of objects must be passed to what methods, and what objects are returned by those methods, etc.

Properly understood, the declaring code within the Java package library would seem to be even more elaborate than what may be apparent in the examples cited in the Federal Circuit's decision. The interrelationships between and among the various packages, classes, and methods are so complex it is difficult to present them in chart form; such a representation cannot fully capture the interconnections between the various classes, packages, and methods, which are not merely linear or hierarchical. Simply put, the declaring code is far more than a mere user interface; it provides the central organizing framework of the Java package library and articulates the highly reticulated functionalities within that library.