JAVA: AN INNOVATION IN SOFTWARE DEVELOPMENT AND A DILEMMA IN COPYRIGHT LAW

I. INTRODUCTION

The introduction of Java software development technology by Sun Microsystems, Inc. (Sun) raises new issues in an already unsettled and confusing area of law: the copyrightability of computer programming languages. In 1980, Congress adopted the recommendations of the National Commission on New Technological Uses of Copyrighted Works (CONTU) and passed legislation defining computer programs as copyrightable subject matter.[1] However, no mention was made about the copyrightability of the computer programming languages used to write computer software. No court has explicitly ruled on the issue of whether copyright protection subsists in a computer programming language.[2] Academics have argued both for and against extending copyright protection to computer programming languages.[3] The development of Java makes this unsettled issue more interesting, important, and complex.

Java was developed by Sun, who describes the technology as "a standardized application programming environment that affords software developers the opportunity to create and distribute a single version of programming code that is capable of operating on many different, otherwise incompatible systems platforms and browsers."[4]

This Note will examine how Java works, how it has changed the computer science industry, and how it affects the debate over the copyrightability of programming languages. We will first consider how traditional programming languages operate and how Java has changed the traditional framework. To facilitate a clear understanding of the copyright issues involved, we will consider

[1] Copyright protection subsists in "literary works." 17 U.S.C. § 102(a)(1) (1994). Computer programs are classified as literary works for copyright purposes. H.R. REP. NO. 94-1476, at 54 (1976).

[2] *But see* Lotus Dev. Corp. v. Paperback Software Int'l, Inc., 740 F. Supp. 37, 72, 15 U.S.P.Q.2d (BNA) 1577, 1602 (D. Mass. 1990) (criticizing, in dicta, the argument that computer programming languages are uncopyrightable subject matter).

[3] For arguments supporting copyrightability of computer programming languages, see generally Ronald Johnson & Allen Grogan, *Copyright Protection for Command Driven Interfaces*, 8:6 COMPUTER L. 1 (1991). For arguments against the copyrightability of computer programming languages, see generally Marci A. Hamilton & Ted Sabety, *Computer Science Concepts in Copyright Cases: The Path to a Coherent Law*, 10 HARV. J.L. & TECH. 239 (1997); Elizabeth G. Lowry, *Copyright Protection for Computer Languages: Creative Incentive or Technological Threat?*, 39 EMORY L.J. 1293 (1990); Steve Posner, *Can a Computer Language Be Copyrighted? The State of Confusion in Computer Copyright Law*, 11 COMPUTER/L.J. 97 (1991); Richard H. Stern, *Copyright in Computer Programming Languages*, 17 RUTGERS COMPUTER & TECH. L.J. 321 (1991).

[4] Sun Microsystems, Inc., *Memorandum of Points & Authority* (visited Oct. 11, 1998) <http://www.java.sun.com/lawsuit/counterclaimdoc2.html>.

a hypothetical involving a copier who has used the exact Java programming language specification without copying Sun's literal code. We will use this hypothetical to explore Sun's claim against the copier for infringement of the copyright in the actual language, if any such infringement has occurred. In doing so, we will examine the arguments for and against allowing copyright protection to subsist in a programming language by itself. We will then consider Sun's claim against the hypothetical copier for infringement of the copyright in the computer software programs that utilize the Java languages. Finally, we will examine the law surrounding the extension of copyright protection to non-literal elements of a computer program and whether copyright protection could be extended to give Sun copyright-like protection in the Java language.

II. BACKGROUND AND TECHNICAL OVERVIEW OF JAVA

A. THE COMPUTER SCIENCE WORLD BEFORE JAVA

The allure of Java is its ability to be *platform independent*. To understand what platform independent means, we will discuss what made other programming languages *platform dependent*.

First, what is a platform? The simple answer is that a platform is just a type of computer environment. For example, the most well-known platforms are currently the PC, or Windows platform, and the Macintosh platform. Unix and Linux are also platforms used today. What makes these platforms different are the different instruction sets they use. Computers operate on instructions in the form of binary strings, or strings of ones and zeroes called the *machine language*.[5] This is the most basic language in which the computer communicates.[6] Each instruction for the computer corresponds to a specific string of ones and zeroes. In each environment there are a vast number of different instructions, or binary strings, that the computer will recognize. This vast number of recognizable instructions constitutes that platform's *instruction set*. What makes platforms different from one another is the form of their instruction sets. Each platform's instruction set is comprised of a different number and type of instructions.

---

[5] HARREY M. DEITEL & PATRICK J. DEITEL, C++ HOW TO PROGRAM 8 (1994); JERI R. HANLY ET AL., PROBLEM SOLVING AND PROGRAM DESIGN IN C 17 (1993); ROBERT P. MERGES ET AL., INTELLECTUAL PROPERTY IN THE NEW TECHNOLOGICAL AGE 836 (1997).

[6] DEITEL & DEITEL, *supra* note 5, at 8; HANLY ET AL., *supra* note 5, at 17; MERGES ET AL., *supra* note 5, at 836.

Computer programmers formerly wrote computer programs directly in machine language,[7] but now they communicate with computers using computer programming languages. Examples of pre-Java, or traditional, computer programming languages are C, C++, Pascal, and FORTRAN. Programmers write a program in a given programming language and save it into a basic text file (just like a text file used to save a letter written to a friend). This text file is called the *source file*.[8] A program called a *compiler* then converts the text file into an *object file,*[9] or a file of binary machine language instructions. The object file contains the computer program in its machine language form; it is a file consisting of strings of ones and zeroes. Each string of ones and zeros corresponds to an instruction in the computer's machine language instruction set and instructs the computer to take some action. Thus, the compiler takes, as input, the textual source file and produces, as output, the machine language object file. This object file is platform-specific. It contains instructions out of a specific platform's instruction set. Therefore, an object file for a program written for the Windows platform will not run on a Macintosh computer and vice-versa.

A different compiler is therefore needed for each platform. A software developer wishing to create and develop a program that would run on all platforms would have to develop, compile, and create an object file for each platform. He would then have to transfer, or *port*, his source files over to the other platforms, and recompile them on those platforms to get object files specific to those platforms. This process was used in the past and was expensive and troublesome for software developers. Thus, many software developers opted to make their programs available on only one platform.

A way to write a program once and have it run on any platform was needed. As Sun stated:

> [t]he existence of multiple, platform-dependent systems, each implementing different incompatible programming environments, confronts software developers with a Hobson's choice: either they must develop and support different versions of each program for each different systems platform, or they must pick and choose among the many different possible systems platforms the one or more environments for which they will develop and

[7] Programmers would encode a list of instructions, or a list of ones and zeroes, onto punch cards, insert the punch cards into the punch card reader, and then start the computer working on the punch cards. The punch card and punch card readers simply allowed the programmer to communicate directly with the computer, albeit in the computer's native machine language.

[8] HANLY ET AL., *supra* note 5, at 17.

[9] *Id.* at 19.

> support a version of their programs. Obviously, a better choice would be to develop a single program version that could run on all platforms without modification.[10]

B. HOW JAVA CHANGED COMPUTING: WRITE ONCE, RUN ANYWHERE

Sun, in developing Java, added a new layer to the traditional sequence of events in the life of a computer program. This new layer was called the *Java bytecode interpreter*, otherwise known as the *Java Virtual Machine*.[11] We will start from the beginning of the programming life of a computer program written in Java.

First, as with the traditional method, the programmer writes his program in the high-level Java language, which is very similar to a program written in C or C++ in terms of the words used and the structure of the actual lines of code.[12] The finished text file containing the high-level Java code is called a Java source file, similar to the traditional source file.

Next, the Java Compiler is run on the source files, but instead of producing machine language specific to one machine, as in the case of the traditional compiler, it generates Java bytecodes.[13] Bytecodes look similar to machine language, but are not specific to any one machine.[14] They are only specific to the Java Virtual Machine.[15] Now, the Java program, in its compiled bytecode form, may be run on any Java Virtual Machine.

The Java Virtual Machine is a computer program that takes the compiled Java bytecodes as input and carries out the bytecode instructions on a specific machine, with a different Java Virtual Machine program for each platform. For example, in the Java object code file there may be an instruction to add two numbers. The *add* instruction in the Java object file would be in Java bytecode format. When the Virtual Machine comes to the bytecode instruction for the *add* instruction, it simply tells the platform's operating system to carry out the *add* instruction. The Virtual Machine takes the generic Java bytecode instructions and interprets them into the specific machine language instructions for that particular platform. For this reason the Java Virtual Machine is sometimes referred to as an *interpreter*.[16] It interprets the Java bytecodes into corresponding machine language instructions. Thus, the same Java object file can be run on any platform that supports the Java

[10] Sun Microsystems, Inc., *supra* note 4.

[11] LAURA LEMAY ET AL., TEACH YOURSELF JAVA IN 21 DAYS 9 (1996).

[12] *Id.* at 11 (stating that "Java is modeled after C and C++, and much of the syntax and object-oriented structure is borrowed from the latter. . . If you are familiar with C++, learning Java will be particularly easy for you because you have most of the foundation already").

[13] *Id.*

[14] *Id.*

[15] *Id.*

[16] LEMAY ET AL., *supra* note 11, at 9.

Virtual Machine. Currently all major platforms support the Java Virtual Machine. A software developer can now write a program, compile it into a Java object file, place it on the Internet, and be confident that anyone can download it and run it on their specific platform.

Java Virtual Machines are also written for most Internet browsers themselves. An Internet browser that is capable of running Java object files is *Java enabled*. This means that a software developer can put his Java object files on the Internet and a user can run the program *inside their browser*, without downloading the program to their computer.

This technology has had a huge impact on the software industry. No longer must developers choose which platform to develop their software on; nor must they incur the expenses of porting their finished product to other platforms. With Java, developers simply write a program once and run it anywhere. As one commentator has stated:

> [a] Java executable [compiled program code] written for one computer can be run, without modification, on another computer supporting Java. The other computer does not need the corresponding source code to accomplish this feat; porting is automatic and virtually instantaneous. This means that users owning entirely different types of computers can download a Java executable from a server and run that executable on their systems and expect an identical result . . . *The capability of downloading a program and executing it in a variety of computers is expected to lead to entirely new kinds of application programs. This is the true magic of JAVA.*[17]

III. COPYRIGHT ISSUES

Java's ability to facilitate platform-independence makes the issue of copyrightability more complex. When considering copyright protection, one must examine the copyrightability of the traditional high-level Java programming language, just as one would in analyzing copyrightability of a traditional programming language. But with Java one must also consider the copyrightability of Java bytecodes, an element not present in traditional programming languages.

We will use a hypothetical to clarify the copyright issues related to programming languages and how these issues are affected by the innovations of

[17] Sun Microsystems, Inc., *supra* note 4 (quoting GILBERT & MCCARTY, OBJECT-ORIENTED PROGRAMMING IN JAVA 50 (1997)).

the Java technology. First, we will set forth the specifics of the proposed infringer. We will then examine two of Sun's possible claims against this infringer: first, a claim for infringement of Sun's copyright in the language itself, and second, a claim for infringement of Sun's copyright in the software that utilizes the language.

A. THE HYPOTHETICAL INFRINGER

To discuss copyright issues related to Java, we must visualize an infringer who has copied the Java language, without authorization, and developed a new Java compiler. The new compiler would be a computer program developed without copying any literal code from Sun's compiler. The compiler would take Java source code files as input and produce Java object files as output, just as Sun's Java Compiler would. Object files created by the new compiler would run on Sun's Virtual Machine. Thus, the infringer would have successfully utilized the high-level Java programming language and Java bytecodes without literally copying any of Sun's source code.

First, *how* would this be done? To construct his own Java compiler, the infringer would have to study simple Java source code fragments alongside the Java bytecode files produced by running Sun's Java Compiler on the simple source code fragment.[18]

Second, *why* would anyone want to do this? Why wouldn't the infringer just want to develop his own platform-independent programming language like Java, but which utilized a different source code language as input and produced a different bytecode object file as output? The answer is compatibility. The infringer's completely new system would lack compatibility with Java, thus making it unlikely that developers would switch from the established Java system to the infringer's new system without some incentive to do so. The computer industry would be wary of embracing a new platform-independent computer programming language for fear of reverting back to the pre-Java state of the industry: incompatible platforms running incompatible code. Thus, if one was to develop a Java-like system of software, it would likely be accepted only if it was completely compatible with Sun's Java language.

---

[18] For purposes of this Note we will not consider whether this process of reverse engineering constitutes infringement of Sun's copyrights in its Compiler and Virtual Machine software. In other words, we will not consider whether the infringer's compiler and virtual machine, being developed in this manner, infringe Sun's copyright in its own Compiler and Virtual Machine. We will simply consider whether Sun has a valid copyright in the language itself and whether the infringer, by using the language in his own software, has infringed that copyright. For more on the issue of reverse engineering of computer software, see Sega Enters. Ltd. v. Accolade, Inc., 977 F.2d 1510, 1527-28, 24 U.S.P.Q.2d (BNA) 1561, 1574 (9th Cir. 1992) (holding that "where disassembly is the only way to gain access to the ideas and functional elements embodied in a copyrighted computer program and where there is a legitimate reason for seeking such access, disassembly is a fair use of the copyrighted work, as a matter of law").

There are two distinct incentives for our proposed copier, which relate to the two levels of the Java language. First, the copier has an incentive to use the same specification as the high-level Java programming language so that programmers who have already learned Java could easily switch to his "new" language without having to learn a new language. This type of incentive is common among computer copyright cases.[19] Indeed, it has generally affected companies developing new compilers for traditional programming languages. If a new company wanted to develop its own C or C++ compiler, the programming language utilized by its compiler would be nearly identical to the C or C++ programming language utilized by another company's compiler. As a result, many companies would develop their own compiler for a single language, with only subtle differences between the languages. Thus, a programmer who learned C on Borland's compiler would not have to relearn the language if he switched to a Unix compiler.

The second incentive is more important to the success of the copier's new language. The copier must make sure that his new compiler produces, as output, object files consisting of Java bytecodes that will run perfectly on Sun's Virtual Machine. In this way, any program developed with his new compiler would be as platform-independent as a program developed using Sun's own Java Compiler. He would thus be able to tap into the platform-independence of Java. The incentive for a copier to develop his new language system to be compatible with Sun's Java bytecodes did not exist with previous programming languages, because compilers written for these traditional languages compiled the source code files directly into machine language. Java, on the other hand, compiles source code into bytecodes to be utilized by another program, the Virtual Machine, which then turns the bytecodes into machine language. Because traditional languages did not have this extra layer of bytecode interpretation, there was no reason to worry about the various compiler outputs being compatible.

For purposes of discussion, we will assume our infringer has created a compiler that accepts, as input, a source code file written in the high-level Java programming language developed by Sun. This compiler produces, as output, a bytecode object file that will run on Sun's Virtual Machine. We will further assume that all of this has been done without literally copying any of Sun's source or object code.[20]

---

[19] *See, e.g.,* Lotus Dev. Corp. v. Borland Int'l, Inc., 49 F.3d 807, 34 U.S.P.Q.2d (BNA) 1014 (1st Cir. 1995), *aff'd by an equally divided Court*, 516 U.S. 233 (1996) (defendant copied entire command hierarchy from plaintiff's spreadsheet program so that users of plaintiff's program could more easily switch to defendant's program).

[20] The hypothetical situation considered in this Note is different from the issue to be resolved in Sun Microsystems, Inc. v. Microsoft Corp., No. C97-20884 PVT (N.D. Cal. filed Oct. 7, 1997). For a summary of the issues presented in that case and the possibility of the issue of copyrightability of programming languages arising, see *infra* Appendix I.

The focus of the rest of this Note will be on whether Sun could successfully claim that the hypothetical copier has infringed its copyright in the Java computer programming language. We will examine two arguments regarding this issue. First, we will consider whether copyright protection can be claimed for the language itself, without reference to the software that utilizes the language. Second, we will consider whether copyright protection extends to programs using the language, i.e., the Compiler and the Virtual Machine. Before considering arguments regarding copyrightability, we will look at why the copyrightability of Java as a programming language is particularly intriguing.

B. WHAT MAKES JAVA DIFFERENT WITH REGARD TO COPYRIGHT?

The copyrightability of the Java programming language is unique because it has two components, whereas traditional languages have only one. Both Java and the traditional languages have the high-level, pseudo-english component that programmers actually use to write code, but only Java has bytecodes, enabling it to be platform-independent. Thus, in the traditional language setting, one would consider only the copyrightability of the high-level language. With Java, one must consider the copyrightability of the high-level language as well as the copyrightability of the bytecode language. This prospect may lead to some interesting results. For example, if the high-level Java language is copyrightable and the bytecode language is not, then a copier's use of the high-level language in the compiler would be infringement. However, he could develop another high-level programming language, dissimilar to the Java language, that would compile into the same bytecodes. The copier's new language would, therefore, be platform-independent like Java. On the other hand, if bytecodes are copyrightable, then the copier would not be able to tap into the platform-independence of Java and his new language would not be as enticing to software developers.

C. COPYRIGHT IN A PROGRAMMING LANGUAGE GENERALLY

No court has directly ruled on the issue of copyrightability of a high-level computer programming language like Java.[21] However, courts have ruled on the copyrightability of items which share some characteristics with computer programming languages.[22] Further, a Massachusetts district court has considered the argument and has stated in dicta that there is no reason why a programming

[21] Hamilton & Sabety, *supra* note 3, at n.113.

[22] *See, e.g.,* Reiss v. National Quotations Bureau, 276 F. Supp. 717 (S.D.N.Y. 1921) (holding that a list of meaningless words is copyrightable subject matter).

language could never be copyrightable.[23] There is much disagreement in the academic world on this issue: commentators have argued both for and against allowing copyright protection in a computer programming language.[24]

This part will first examine the decision in *Lotus Development Corp. v. Paperback Software International, Inc.*[25] Consideration will then be given to each of the requirements of the Copyright Act (fixation, originality, and expression rather than idea) with regard to Java. Lastly, the First Amendment and public policy rationales used to argue against copyright protection for programming languages will be discussed.

1. Lotus Development Corp. v. Paperback Software International, Inc.[26] At issue in *Paperback Software International, Inc.* was the defendant's use of the plaintiff's command hierarchy in a spreadsheet program. The defendant, like the defendant in our hypothetical, did not copy the plaintiff's literal code. Thus, it was not a case of literal infringement. Instead, the defendant used the same input commands as were utilized by the plaintiff's program. For example, users of both programs could enter the command "/FR" to execute the file retrieval command.[27] The court explicitly stated that it did not consider these input commands to be a programming language and thus avoided the issue of whether programming languages are copyrightable subject matter.[28] Instead, the court held that the plaintiff's commands constituted a non-literal aspect of the program and that the plaintiff's copyright in the program itself extended to this non-literal element.[29] Therefore, the defendant had infringed the plaintiff's copyright in the software by copying this non-literal element.[30] But in a section at the end of the opinion titled "Strained Analogies and Word Games," the court considered, in dicta, the defendant's argument that the commands constituted an uncopyrightable "programming language" and that use of this programming language could therefore not be infringement.[31] The court summarized the argument as follows:

[23] Lotus Dev. Corp. v. Paperback Software Int'l, Inc., 740 F. Supp. 37, 72-73, 15 U.S.P.Q.2d (BNA) 1577, 1603 (D. Mass. 1990).

[24] For arguments for the copyrightability of programming languages, see generally Johnson & Grogan, *supra* note 3. For arguments against the copyrightability of computer programming languages, see generally Hamilton & Sabety, *supra* note 3; Lowry, *supra* note 3; Posner, *supra* note 3; Stern, *supra* note 3.

[25] 740 F. Supp. 37, 15 U.S.P.Q.2d (BNA) 1577 (D. Mass. 1990).

[26] *Id.*

[27] *See* Stern, *supra* note 3, at 326 (using this example).

[28] 740 F. Supp. at 68-70.

[29] *Id.* at 70.

[30] For discussion of copyright protection for non-literal elements of a program, see *infra* Part III.D. A finding that the copyright in the program itself extends to non-literal elements affords copyright-like protection to those elements but is doctrinally different from a finding that the non-literal elements are themselves copyrightable.

[31] Lotus Dev. Corp. v. Paperback Software Int'l, Inc., 740 F. Supp. 37, 70-72, 15 U.S.P.Q.2d (BNA) 1577, 1602-03 (D. Mass. 1990).

> (1) Although expression is copyrightable, the language in which the expression is written is not copyrightable. Thus, a book written in English or French may be copyrightable, but the English and French "languages" are not works in which copyright may subsist.
> (2) Like books, computer programs, written in computer programming "languages," may be copyrightable, but only the "sets of statements or instructions," and not the "language" in which they are written, are copyrightable. . . .
> (5) The macro, or "program," that the user writes may be copyrightable if original and nonobvious. The "language" in which the macro is written is never copyrightable.
> (6) Thus, when defendants copied the menu command hierarchy from 1-2-3, they did not copy a copyrightable element that embodied expression, but rather, copied only the "macro facility language" of 1-2-3, a non-copyrightable element in the public domain.[32]

Further, the court addressed three corollaries to the defendant's argument. First, the argument assumes that the word "language" has a "single, invariable meaning in all discourse about 'languages.' "[33] Second, the argument assumes not only that English and French are uncopyrightable as languages, but also that all languages are automatically ineligible for copyright.[34] The court noted that this argument was without authoritative support. Third, the argument assumes that "languages" and "sets of statements or instructions" are opposites and "never the twain shall meet."[35] The court stated that this argument was without merit and that it depended on "arbitrary definitions of words, adopted for undisclosed reasons."[36] The court also found this argument to be a hindrance to the court's function:

> [a]n argument that depends on the proponents' undisclosed definitions of words—and even different definitions as a word is used in different steps of the argument—becomes a word game that obscures the substantive meaning of the argument and is an

[32] *Id.* at 72.
[33] *Id.*
[34] *Id.*
[35] *Id.*
[36] Lotus Dev. Corp. v. Paperback Software Int'l, Inc., 740 F. Supp. 37, 72, 15 U.S.P.Q.2d (BNA) 1577, 1603 (D. Mass. 1990).

> obstruction rather than an aid to the court's use of the adversary process to inform and thus improve decision-making.[37]

Thus, the court, while basing its finding of infringement on other grounds, took the opportunity to lash out at the argument that programming languages are per se uncopyrightable. While not offering much precedential support for the proposition that programming languages are copyrightable, the court clearly left the issue open.

2. Copyright Act Elements.

a. Fixation. In order for material to be copyrightable under the Copyright Act of 1976, it must be "fixed in any tangible medium of expression, now known or later developed, from which [it] can be perceived, reproduced, or otherwise communicated, either directly or with the aid of a machine or device."[38] Congress has elaborated on this requirement:

> [u]nder the bill it makes no difference what the form, manner, or medium of fixation may—be whether it is in words, numbers, notes, sounds, pictures, or any other graphic or symbolic indicia, whether embodied in a physical object in written, printed, photographic, sculptural, punched, magnetic, or any other stable form, and whether it is capable of perception directly or by means of any machine or device "now known or later developed."[39]

Programming languages are "fixed" in the computer program that utilizes them or embodies them; Java is fixed in the Java Compiler and Virtual Machine. The exact specification is embedded into the actual software that recognizes the language.

But is this fixation for section 102 purposes? The academics disagree. For example, one commentator concludes that this "fixation" is enough:

> [a]ll computer languages meet this requirement because they are embodied in computer programs. The program, stored on a diskette or other tangible medium, defines the computer language which it translates or requires for

[37] *Id.* at 72.
[38] 17 U.S.C. § 102(a) (1994).
[39] H.R. REP. NO. 94-1476, at 52 (1976).

> interaction with it. The user can perceive the computer language "with the aid of a machine," the computer. Thus, a computer language meets the copyright fixation requirement.[40]

On the other hand, Hamilton and Sabety argue that in order for a programming language to be "fixed in a tangible medium," the author would have to create one of two things.[41] First, the author could create "a list of all possible sentences in that language."[42] This would be nearly impossible, as languages may have an unbounded number of possible sentences.[43] Second, the author could give an expression of its specification.[44] However, the specification only tells us whether a given sentence is part of the language or not, and is not a fixation of any given sentence.[45]

In the Java context, Hamilton and Sabety's argument only applies to the high-level Java programming language because the Java bytecodes are finite. They could easily be listed in their entirety, and there is no separate grammar or set of rules to apply to build sentences out of them. They are simply a list of commands that the Java Virtual Machine will recognize. Of course, for the bytecodes to produce any meaningful output, they have to be in a meaningful order. But this order is not intrinsic to the language itself, as with a grammar. It is up to the programmer to organize the high-level program text in a manner that will allow the Compiler to produce bytecodes that will yield meaningful, useful output. A random assortment of bytecode instructions will produce an output, just not a meaningful output. Thus, although order is important for the usefulness of the language, there is not an intrinsic bytecode grammar as with the high-level Java language.

Further, Hamilton and Sabety's argument presupposes that in order for a programming language to be fixed, all of the possible sentences that could be formed using the language must also be fixed. This requirement finds no basis in the Copyright Act. Admittedly, the drafters of the Copyright Act likely did not contemplate fixation of a programming language, but the Act's most logical

[40] Lowry, *supra* note 3, at 1308-09.
[41] Hamilton & Sabety, *supra* note 3, at 269.
[42] *Id.*
[43] *Id.*
[44] A specification is a term referring to the mathematical way in which a programming language is defined. The specification is the quadruple (V, E, R, S) where V is the entire set of symbols used by the language, E is the set of all terminal symbols, R is the finite set of grammatical rules that transform non-terminal symbols into constituent terminal and non-terminal symbols, and S is the start symbol that tells the computer there is a sentence to parse. Hamilton & Sabety, *supra* note 3, at 266.
[45] *Id.*

application to programming languages would be to require that a copyright-seeker fix only that for which he would like to obtain copyright protection.

Hamilton and Sabety also assume that anyone seeking copyright protection would be seeking protection not only for the language, but also for any expressions made in that language. For example, Hamilton and Sabety state that "language copyright is doctrinally suspect because that would provide copyright protection for expressions not yet fixed."[46] If one wished to obtain copyright protection for every expression made in a programming language, then one would have to either list all possible expressions or list the specification of the language, as Hamilton and Sabety suggest. However, if one wished only to obtain copyright protection for the language itself, one should be able to fulfill the fixation requirement by simply fixing the language and the grammar rules for the language in a computer program utilizing the language.

A rule that would require a copyright-seeker to fixate all possible expressions in the language would be unduly harsh. The most logical application of the fixation requirement would be to require fixation only of the language if that is all the copyright-seeker wishes to protect. Thus, the fixation requirement of the Copyright Act should be deemed fulfilled where the language is embodied in the software that utilizes the language.

b. Originality. The 1976 Copyright Act provides that "[c]opyright protection subsists, in accordance with this title, in original works of authorship."[47] A programming language is sufficiently original to fulfill this requirement.[48] This standard requires little creativity because the author does not need to show "novelty, ingenuity, or aesthetic merit."[49]

However, some trouble may arise because computer languages generally build on predecessor languages. For example, the high-level Java programming language was modeled after the C and C++ programming languages.[50] A court would have to determine which elements of the new language were created by Sun and which elements were derived solely from C or C++. "Hence, while a computer language in all likelihood meets the originality standard, its developer may have difficulty establishing which elements of the language are original and which are copied from a preexisting language."[51] Note that this public domain problem will only arise regarding the copyrightability of the high-level language component of Java. The problem will not arise at the Java bytecode level, since the Java bytecodes were an innovation by Sun.

[46] *Id.* at 269.
[47] 17 U.S.C. § 102(a) (1994).
[48] Lowry, *supra* note 3, at 1306-08.
[49] *See* 17 U.S.C.A. § 102 (1996) (for accompanying legislative history).
[50] LEMAY ET AL., *supra* note 11, at 11.
[51] Lowry, *supra* note 3, at 1308.

c. Idea/Expression Dichotomy. Copyright protection will not subsist in ideas but only in the expression of those ideas. This is made clear in section 102(b) of the Copyright Act:

> 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.[52]

The idea/expression dichotomy was dealt with in the early case of *Baker v. Selden*.[53] There, the Supreme Court's holding explained this proposition:

> [t]he 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. . . The very object of publishing a book on science or the useful arts is to communicate to the world the useful knowledge which it contains. But this object would be frustrated if the knowledge could not be used without incurring the guilt of piracy of the book. . . The description of the art in a book, though entitled to the benefit of copyright, lays no foundation for an exclusive claim *to the art itself*.[54]

The problem is that the high-level Java language, like the accounting system in *Baker*, is similar to a mere system. The language is a set of symbols, accompanied by grammar rules for building those symbols into sentences that will be recognizable by the Compiler. One commentator has described this problem as follows: "a computer language is a system of vocabulary and grammar rules. When the operator correctly applies the grammar rules to the vocabulary, the system works: he creates a program. If he does not follow the rules correctly, the program fails."[55]

Thus, under section 102(b) of the Copyright Act, the high-level Java language may not be copyrightable because it may too closely resemble a system. However, not all of this level of Java is a system because there are symbols used in the

[52] 17 U.S.C. § 102(b) (1994).

[53] 101 U.S. 99 (1879).

[54] *Id.* at 103-05 (emphasis added).

[55] Lowry, *supra* note 3, at 1311; *see also* Posner, *supra* note 3, at 106-07 (stating that a computer language is an uncopyrightable set of rules).

language. Nevertheless, a court may find that the copyrightable elements have merged into the idea, thus precluding copyright protection.[56]

Applying the idea/expression dichotomy to the Java bytecodes yields a different result. The Java bytecodes are not a combination of symbols and of rules to be applied to those symbols. They are simply a list of instruction codes that the Virtual Machine will recognize. Unlike the accounting system in *Baker*, the Java bytecodes do not constitute a system or means of doing something. Although the instructions need to be arranged in a meaningful order to produce meaningful output, the order is not intrinsic to the bytecodes. Unordered bytecodes are still valid and will still run on the Java Virtual Machine; they will just not produce meaningful output. Therefore, the bytecodes would seem to be copyrightable expression rather than an uncopyrightable system.

The bytecodes are akin to the compilation of coined words having no known meaning held to be copyrightable in *Reiss v. National Quotation Bureau, Inc.*[57] In an opinion by Judge Learned Hand, the court held that a group of words, having only an agreed meaning for purposes of cable correspondence, were copyrightable. Hand likened the plaintiff's group of words to "a set of words or symbols to form a new abstract speech, with inflection, but as yet with no meaning, a kind of blank Esperanto."[58] Hand later analogized the plaintiff's symbols to the language of math:

> [m]athematics has its symbols, indeed a language of its own, Peanese, understood by only a few people in the world. Suppose a mathematician were to devise a new set of compressed and more abstract symbols, and left them for some conventional meaning to be filled in. Still we should not be quite at the plaintiff's words, but again we should not be far away. The distinction is real, but for practical purposes seems to me irrelevant.[59]

Hand envisioned the copyrightability of an invented "language," similar to Sun's Java bytecodes. Thus, the bytecodes are expression deserving of copyright protection.

3. First Amendment and Public Policy Concerns. Some commentators have raised objections to the copyrightability of programming languages apart from the elements of the Copyright Act. We will first consider the argument that copyright

[56] *See* Lowry, *supra* note 3, at 1312 (arguing that the words which constitute part of a language are an integral part of the system, i.e. the programming language, and therefore cannot be separated from the system).
[57] 276 F. 717 (S.D.N.Y. 1921).
[58] *Id.* at 718.
[59] *Id.*

protection of a programming language would violate the First Amendment. Second, we will consider the argument that copyright protection in a programming language would violate public policy.

Commentators have argued that allowing copyright protection for programming languages would violate the First Amendment. For example, Hamilton and Sabety feel that such copyright protection would inhibit expression even before it has the chance to develop:

> [b]y authorizing protection for languages, the Act would be authorizing prior restraint of any expression in that language. . . . [Copyright law] exceeds First Amendment boundaries when it permits authors to own the rights to form any expression from the building blocks of language. Indeed, protection of linguistic building blocks impedes the progress of originality sanctioned by the Copyright Clause in addition to violating the First Amendment rule against suppressing speech before it has been expressed.[60]

Hamilton and Sabety's argument assumes that the copyright-seeker wishes to obtain copyright protection for all expressions written in the language, but this assumption has no basis in law or in reality. Sun does not wish to control the writing of programs in its language. Instead, Sun wants as many people as possible to be programming in Java. Sun only wants to prevent other companies from using its language in the other's compiler and virtual machine, as in our hypothetical.

When considering the hypothetical defendant's argument that granting copyright protection would violate the First Amendment, a court would likely find that Sun is not suing to stop expression in the language, but is instead suing because of the use of the language in the defendant's software. It would be difficult for a court to rule that the language should be deemed uncopyrightable because of First Amendment concerns when the defendant cannot claim that his First Amendment rights have been violated.

Policy issues regarding copyright protection of programming languages have also arisen.[61] It has been argued that "[p]rotection would inhibit technological development, create monopoly pricing, and prevent standardization of

[60] Hamilton & Sabety, *supra* note 3, at 270; *see also* Lowry, *supra* note 3, at 1315 (stating that "[a] programming language is also unprotectible because its commands are 'building blocks' for creative expression").

[61] Lowry, *supra* note 3, at 1338-46.

languages."[62] Likewise, it has been argued that protection is not needed to give developers the creative incentive to produce programming languages.

These concerns, although providing support for the non-protection of languages in general, lose their force when examined within the specifics of Java. Creating a programming language like Java that allows platform-independence will not inhibit technological development. On the contrary, this is the advancement for which the computer science world has been waiting. If anything, it will advance technology by enhancing the flow of information across multiple platforms.

Further, Java will enable the operating system market to become more competitive.[63] The current operating systems market is affected heavily by "network effects," meaning that "users will naturally gravitate towards a single standard platform and tend to stay there."[64] But with the introduction of Java, the need to use the same operating system disappears. One commentator has noted this benefit:

> [i]f each consumer could operate across platforms, . . . [c]onsumers could choose between IBM, Apple, Sun, Linux and Microsoft operating systems on their merits as programs, and not on the basis of what everyone else uses, or what compatible applications programs exist. In a world in which the platform from which one operates is irrelevant to one's ability to exchange data, the sheer number of existing platform users will be much less important to a consumer's purchasing decisions.[65]

Thus, it is clear that Java is an important advancement for the computer science industry. Allowing copyright protection in the language will not take away from its importance to the industry. If anything, copyright protection will encourage companies like Sun to invest the resources necessary to create such an advancement.[66]

4. How Our Hypothetical Copier is Affected. A court may find the high-level component of the Java language uncopyrightable for the reasons articulated above.

---

[62] *Id.*

[63] Mark A. Lemley & David McGowan, *Could Java Change Everything? The Competitive Propriety of a Proprietary Standard*, 520 PLI/Pat 453, 456-57 (1998).

[64] *Id.* at 456.

[65] *Id.*

[66] *But see id.* at 466 (raising concerns about a single firm being able to control the standardization process for the computer science industry).

However, the Java bytecodes seem to be copyrightable subject matter under section 102 of the Copyright Act, section 102(b) and First Amendment objections notwithstanding.

Where does this leave our hypothetical copier? There would be little incentive to pursue the use of the high-level language without the bytecodes. The copier would not be able to take advantage of the platform-independence of the language. If the copier developed a new bytecode language, users would have little incentive to switch to the new language, which would be incompatible with the Java code already in use. Thus, Sun, although losing on the issue of copyrightability of their high-level language, would win overall.

D. EXTENSION OF COPYRIGHT PROTECTION TO NON-LITERAL ASPECTS OF A COMPUTER PROGRAM

In this part we will consider whether Sun could argue that its copyright in the programs that utilize Java has been violated.

It is well-settled that copyright protection can be obtained for the literal elements of a computer program.[67] Congress has provided that copyright protection exists in "literary works,"[68] and the legislative history of the Copyright Act makes it clear that computer programs fall within this category.[69] Thus, the literal elements of a computer program, i.e., the source and object codes, are protected by copyright, just as the actual text of a book is protected.

This part explores whether Sun's copyright in the software that utilizes Java (the Java Compiler and Virtual Machine) extends to other non-literal elements of the computer program, namely to the high-level Java language and to the Java bytecodes. The Court of Appeals for the Third Circuit explained the problem by noting that "copyrights of other literary works can be infringed even when there is no substantial similarity between the works' literal elements. One can violate the copyright of a play or book by copying its plot or plot devices."[70] There has been disagreement among the courts of the Federal Circuit as to what test should be used to determine whether copyright protection extends to non-literal elements of a computer program.[71] There is also some issue as to whether a particular

[67] Apple Computer v. Franklin Computer, 714 F.2d 1240, 219 U.S.P.Q. (BNA) 113 (3d Cir. 1983) (holding that "a computer program, whether in object code or source code, is a 'literary work' and is protected from unauthorized copying, whether from its object or source code version").

[68] 17 U.S.C. § 102(a) (1994).

[69] H.R. REP. NO. 94-1476, at 54 (1976).

[70] Whelan Assocs., Inc. v. Jaslow Dental Lab., Inc., 797 F.2d 1222, 1234, 230 U.S.P.Q. (BNA) 481, 490 (3d Cir. 1986).

[71] *Compare Whelan Assocs., Inc.*, 797 F.2d 1222, 230 U.S.P.Q. (BNA) 481 (applying the idea/expression dichotomy standard to determine infringement of a non-literal element) *with* Computer Assocs. Int'l, Inc. v. Altai, Inc. 982 F.2d 693, 23 U.S.P.Q.2d (BNA) 1241 (2d Cir. 1992) (utilizing a three-step approach to determine

element is a literal or non-literal element of a computer program.[72] Further, Congress has precluded copyright protection for any "idea, procedure, process, system, [or] method of operation."[73] The legislative history states that "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 *actual processes or methods embodied in the program are not within the scope of the copyright law*."[74]

Thus, copyright protection exists in the actual expression embodied in the source code of a computer program but does not extend to "processes or methods" used by the program. In determining whether copyright protection extends to a non-literal element of a computer program, one must keep in mind that processes and methods intrinsic to the program are not copyrightable. At the heart of this issue is the idea/expression dichotomy: copyright protection should only extend to those non-literal elements that constitute part of the expression of the computer program, not to the idea of the computer program.

This problem was considered by the Court of Appeals for the Third Circuit in *Whelan Associates, Inc. v. Jaslow Dental Laboratory, Inc.*[75] The specific issue there was "whether the structure (or sequence and organization) of a computer program is protectible by copyright, or whether the protection of the copyright law extends only to the literal computer code."[76] The plaintiff in *Whelan Associates, Inc.* did not allege that the defendant had copied any of the literal source or object code of its program. Rather, the plaintiff alleged that the defendant copied the "structure, sequence, and organization" of the plaintiff's program.[77] The court, considering whether a non-literal element is part of the expression of the program, stated that "the purpose or function of a utilitarian work would be the work's idea, and everything that is not necessary to that purpose or function would be part of the expression of the idea."[78] In applying the rule to the facts of that case, the court held:

> it is clear that the purpose of the utilitarian Dentalab program was to aid in the business operations of a dental laboratory. It is equally clear that the structure of the program was not

infringement of a non-literal element).

[72] *See* Lotus Dev. Corp. v. Borland Int'l, Inc., 49 F.3d 807, 34 U.S.P.Q.2d (BNA) 1014 (1st Cir. 1995) (holding that menu-command hierarchy is uncopyrightable subject matter), *aff'd by an equally divided Court*, 516 U.S. 233 (1996).

[73] 17 U.S.C. § 102(b) (1994).

[74] H.R. REP. NO. 94-1476, at 57 (1976) (emphasis added).

[75] Whelan Assocs., Inc. v. Jaslow Dental Lab., Inc., 797 F.2d 1222, 230 U.S.P.Q. (BNA) 481 (3d Cir. 1986).

[76] *Id.* at 1224.

[77] *Id.*

[78] *Id.* at 1236 (emphasis omitted).

> essential to that task: there are other programs on the market, competitors of Dentalab and Dentcom, that perform the same functions but have different structures and designs. . . . The conclusion is thus inescapable that the detailed structure of the Dentalab program is part of the expression, not the idea, of that program. . . . Because there are a variety of program structures through which that idea can be expressed, the structure is not a necessary incident to that idea.[79]

The *Whelan* test gives a very broad definition of which non-literal elements fall within the protection of copyright.

Applying the *Whelan* test to our hypothetical would likely yield a finding that the high-level Java language and bytecodes are protected by the copyright in the Java Compiler and the Virtual Machine software. In this case the "idea" would be the creation of a program which facilitates platform-independent software development. The choices of words and symbols that make up the high-level Java language and Java bytecodes are not essential to that idea. Any number of other words and bytecodes could be used to facilitate the idea. Because there are a variety of ways that Sun could have constructed its high-level language, and a variety of ways that it could have specified its bytecodes, the high-level Java language and the Java bytecodes would be protected under the *Whelan* test.

However, the Court of Appeals for the Second Circuit in *Computer Associates International, Inc. v. Altai, Inc.*,[80] chose not to adopt the *Whelan* test.[81] *Altai, Inc.* involved a scheduling program for IBM mainframe computers. The defendant unknowingly hired a programmer who had worked on the plaintiff's program (CA-Scheduler) and who had in his possession copies of the literal code from the plaintiff's program. This employee used his copies of the plaintiff's literal code to write the defendant's OSCAR program. After learning of the infringement, the defendant decided to rewrite OSCAR from scratch, using none of the code from OSCAR or from CA-Scheduler and none of the programmers who had worked on the tainted OSCAR. In this way, the defendant ensured that the new version of OSCAR in no way encompassed any of the literal elements of the plaintiff's CA-Scheduler. However, the plaintiff argued that OSCAR still infringed its CA-Scheduler because the structures of the two programs were substantially similar. The court noted that the *Whelan* rule had "received a mixed reception" from other courts and an even worse reception from the academic community.[82] The court

[79] *Id.* at 1238-40.

[80] 982 F.2d 693, 23 U.S.P.Q.2d (BNA) 1241 (2d Cir. 1992).

[81] Every court that has dealt with this issue since 1992 has declined to follow the *Whelan* test. MERGES ET AL., *supra* note 5, at 889.

[82] *Altai, Inc.*, 982 F.2d at 705, 23 U.S.P.Q.2d (BNA) at 1252.

stated that "the crucial flaw in [*Whelan's*] reasoning is that it assumes that only one 'idea,' in copyright law terms, underlies any computer program, and that once a separable idea can be identified, everything else must be expression."[83] The court went on to state that "[*Whelan's*] approach to separating idea from expression in computer programs relies too heavily on metaphysical distinctions and does not place enough emphasis on practical considerations."[84]

The court, in place of the *Whelan* test, applied an abstraction-filtration-comparison test. The first step, abstraction, was described by the court as follows: "[i]nitially, in a manner that resembles reverse engineering on a theoretical plane, a court should dissect the allegedly copied program's structure and isolate each level of abstraction contained within it. This process begins with the code and ends with an articulation of the program's ultimate function."[85]

The next step, filtration, was described by the court as:

> [the examination of] the structural components at each level of abstraction to determine whether their particular inclusion at that level was "idea" or was dictated by considerations of efficiency, so as to be necessarily incidental to that idea; required by factors external to the program itself; or taken from the public domain and hence is nonprotectable expression.[86]

The third and final step is comparison. Once a court has completed the first two steps, it will be left with a "core of protectable expression," or a "golden nugget."[87] The court should then determine whether the "golden nugget" of the defendant's work is substantially similar to the plaintiff's "golden nugget."

The *Altai, Inc.* court applied this test to the facts, accepting most of the district court's findings. The district court had determined that there were five levels of abstraction: object code, source code, parameter lists, services required, and the general outline.[88] The court first observed that there was no similarity in the source or object codes.[89] In considering the parameter lists and macros, the court held that many of the lists and macros were in the public domain or dictated by the functional demands of the program, which are "filtered" out and not given

[83] *Id.* at 705 (citation omitted).
[84] *Id.* at 706.
[85] Computer Assocs. Int'l, Inc. v. Altai, Inc., 982 F.2d 693, 707, 23 U.S.P.Q.2d (BNA) 1241, 1253 (2d Cir. 1992).
[86] *Id.* at 707.
[87] *Id.* at 710.
[88] *Id.*
[89] *Id.*

copyright protection.[90] The court found that the remaining lists and macros were not infringed.[91] Finally, the district court concluded that the organizational charts were simple and would be obvious to anyone exposed to the operation of the program.[92] Thus, the court found no infringement.[93]

To apply this test to our hypothetical case, we must first apply the abstraction step to two programs: the Java Compiler and the Java Virtual Machine. The Java high-level language and bytecodes are at a greater level of abstraction than source or object code, but they are at a lower level of abstraction than a "general outline." They may be somewhat akin to the "parameter lists" used in the *Altai, Inc.* court's analysis. The text files written in the high-level Java language are used as input to the Java Compiler, which produces an object file consisting of Java bytecodes as output. The object file is then used as input to the Java Virtual Machine.

At the second step, filtration, we must determine whether the inclusion at that level was "idea" or (1) dictated by considerations of efficiency, (2) required by factors external to the program itself, or (3) taken from the public domain. It is difficult to imagine how these factors apply to something like a language. However, the high-level Java language was in large part based on an item in the public domain, the prior programming languages C and C++.[94] For this reason much of the high-level Java language may be filtered out in step two.

The Java bytecodes will not be filtered out for this same reason. The bytecodes were not taken from anything in the public domain, unlike the parameters and macros in *Altai, Inc.* They were wholly created by Sun. Further, unlike the non-literal elements in *Altai, Inc.*, the bytecodes were not required by factors external to the program itself or by considerations of efficiency, but rather were at the heart of the very purpose of the program itself. The bytecodes directly facilitate platform-independence. Also, the bytecodes are utilized by both the Compiler and the Virtual Machine. They are possibly protected by copyrights in each program. Thus, the Java bytecodes are likely "golden nuggets" of material fit for copyright protection.

The third step, comparison, is the easiest of the steps in our hypothetical. As specified earlier, the language and bytecodes used by our hypothetical infringer are identical to the high-level Java language and Java bytecodes. If the language is not filtered out in step two, it will prove to be "substantially similar" in step three. Even if the high-level Java language is filtered out, the bytecodes will remain

[90] Computer Assocs. Int'l, Inc. v. Altai, Inc., 982 F.2d 693, 714-15, 23 U.S.P.Q.2d (BNA) 1241, 1259 (2d Cir. 1992).

[91] *Id.*

[92] *Id.* at 715.

[93] *Id.*

[94] LEMAY ET AL., *supra* note 11, at 11.

protected. Our hypothetical copier will have violated Sun's copyright in the Virtual Machine and in the Compiler through the use of bytecodes identical to Sun's Java bytecodes.

It is notable that the filtration test in step two really just asks whether the non-literal element is copyrightable. The test considers factors such as whether the element came from the public domain and whether the element was dictated by considerations of efficiency so as to be necessarily incidental to the idea. Thus, the filtration step of the test seems to reduce the question of whether the copyright extends to a non-literal element of the program to the question of whether that non-literal element, in and of itself, is copyrightable.[95]

This same concern caused the Court of Appeals for the First Circuit in *Lotus Development Corp. v. Borland International, Inc.*[96] to construe *Altai, Inc.* not to apply to cases involving the literal copying of some element of the computer program. In *Borland International, Inc.*, the plaintiff alleged that the defendant had copied the entire command hierarchy from its spreadsheet program Lotus 1-2-3 into the defendant's own spreadsheet program, Quattro. The defendant did not copy any of plaintiff's literal code, but instead used the Lotus command hierarchy so that Lotus users would not have to relearn a new command hierarchy or rewrite their Lotus macros if they chose to switch to Quattro. The court declined to follow the *Altai, Inc.* test:[97]

> [t]he Second Circuit designed its [*Altai*] test to deal with the fact that computer programs, copyrighted as "literary works," can be infringed by what is known as "nonliteral" copying, which is copying that is paraphrased or loosely paraphrased rather than word for word . . . The Second Circuit designed its *Altai* test to deal with this situation in the computer context, specifically with whether one computer program copied nonliteral expression from another program's code . . . In the instant appeal, we are not confronted with alleged nonliteral copying of computer code. Rather, we are faced with Borland's deliberate, literal copying of the Lotus menu command hierarchy. Thus, we must determine not whether nonliteral copying occurred in some amorphous sense,

[95] This same issue was raised by Lowry, *supra* note 3, at 1306. Although not specifically referring to the abstraction-filtration-comparison test, she notes that "the question, '[d]oes the copyright on the underlying computer program extend to the language defined by the program?' cannot be answered before determining whether a computer language is copyrightable." *Id.* at 1306.

[96] 49 F.3d 807, 34 U.S.P.Q.2d (BNA) 1014 (1st Cir. 1995), *aff'd by an equally divided Court*, 516 U.S. 233 (1996).

[97] *Id.* at 815.

> but rather whether the literal copying of the Lotus menu command hierarchy constitutes copyright infringement.[98]

This decision seems to discard the *Altai, Inc.* test altogether. Any claim for non-literal infringement of a copyright in a computer program could be recast as a claim for literal infringement of whatever the non-literal element was. For example, *Altai, Inc.* itself could be recast as a claim for infringement of the plaintiff's copyright in the structure of the computer program. Then the case would have turned on the issue of whether this type of structure, or outline, is copyrightable in and of itself.

The *Borland International, Inc.* decision has received some criticism for its characterization of the *Altai, Inc.* test. As one commentator points out, the court's reasoning for not applying the *Altai, Inc.* test was incorrect in that the test "was designed to determine if the non-literal elements of two computer programs were substantially similar, and not, as the First Circuit maintained, to determine if a copyright has been infringed by 'nonliteral' copying."[99] However, another commentator has attempted to find compatibility between the two decisions:

> [o]ne possible way to reconcile *Lotus* and *Altai* is to treat *Lotus* not as a case involving "literal" copying of a menu structure, but as a case involving nonliteral copying of the program itself. Lotus distributed object code which, in certain combinations, produces on a screen physical images that represent certain words and have certain effects. Borland did not copy that code; rather, it wrote its own code, which produced similar images and had the same effects as the Lotus code. The court could then have applied *Altai's* abstraction-filtration-comparison analysis to this "non-literal" copying. This is in effect precisely what the *Lotus* court did—although when it reached the "filtration" step, it determined that the entire program was unprotectable at the menu command hierarchy level.[100]

It is unclear where the *Borland International, Inc.* decision leaves us. That court would most likely opt not to use the *Altai, Inc.* test in a case like our hypothetical.

[98] *Id.* at 814-15.

[99] Jeffrey M. Gott, *Lotus Development Corporation v. Borland International: The United States Court of Appeals for the First Circuit Takes a Step Backward for the Copyright Protection of Computer Programs*, 30 CREIGHTON L. REV. 1349, 1399 (1997).

[100] MERGES ET AL., *supra* note 5, at 906-07.

It would instead focus on whether the high-level Java language and the Java bytecodes are copyrightable in and of themselves, as considered above.

IV. CONCLUSION

Sun would likely have a valid copyright infringement claim against a copier for the use of Sun's Java bytecodes in his compiler and virtual machine. Its claim could either be based on infringement of the copyright in the bytecodes or on infringement of a non-literal element of its Java Compiler and Virtual Machine software programs.

On the other hand, Sun may have some difficulty in claiming copyright infringement for the copier's use of its high-level Java programming language because much of the programming language is based on other languages already in the public domain: C and C++.

Having a valid copyright in the Java bytecodes helps Sun protect its technology because a prospective copier has little incentive to copy and to use the high-level component of the Java programming language without using the Java bytecodes. Without the bytecodes, the copier would be unable to create a language compatible with Java. Thus, if the copier wished to create a platform-independent language, he would have to develop his own bytecode instruction set. The object files created in Java could not be executed on the copier's virtual machine. Likewise, object files created with the copier's new compiler could not be executed on the Java Virtual Machine.

Developers would have to choose the platform-independent language on which to develop. It is unlikely that the computer industry would embrace a new platform-independent language. With two incompatible platform-independent languages available, the industry would be back to where it was before Java. Developers would now choose the platform-independent language on which to develop, just as they previously had to choose. A copier would, however, be able to make his high-level component extremely similar to Java. Thus, developers would not have to learn anything new to switch between the two incompatible platform-independent languages.

For doctrinal consistency, Congress, having already determined that computer programs are copyrightable subject matter, needs to make a similar determination as to the copyrightability of the languages that are used to write the computer programs. However, as a practical matter, the issue does not frequently arise. Computer programming languages are not invented everyday. It is also unclear whether the copier in our hypothetical would actually find it worth the time and effort to develop a new language, whether compatible with Java or not. However, the copier may be driven by the incentives of the current market state. Technology is one of the fastest moving industries. The market conditions and technological demands of tomorrow may turn the hypothetical considered here into reality, making the issue practically, as well as doctrinally, crucial.

MICHAEL P. DOERR

APPENDIX I

The hypothetical situation considered in this Note is distinguishable from the situation in *Sun Microsystems, Inc. v. Microsoft Corp.*[101] In that case Sun is claiming that Microsoft breached the Java licensing agreement entered into between the two companies by adding keywords to the Java programming language.[102] Sun claims that where the licensing agreement has been breached, any use of the language by Microsoft constitutes copyright infringement.[103]

Although the suit will likely be decided on the terms of the licensing agreement, the issue of the copyrightability of programming languages may be addressed. For example, the following is an excerpt from oral arguments regarding preliminary motions:

> Mr. Quackenbush [attorney for Microsoft]: . . . What's really going on here is Sun just disagrees with these enhancements. Sun doesn't think they're good. Sun doesn't think Microsoft should be able to go forward and do those. Sun thinks that everybody, all the tools vendors, all the competitors, ought to get together and say, "Well, let's decide and go one slow step at a time."
>
> Where does the contract say that? It doesn't say that. Microsoft doesn't need a license to add a keyword. Sun doesn't have a copyright in the language. Their copyright — there is no copyright registration in their papers for the language. There's a copyright for the JDK. [Java Developer's Kit]
>
> A language is just that. It's a language. Maybe Sun wishes Microsoft would not have gone forward and made these enhancements. Maybe Sun is afraid they'll be popular. Maybe Sun is afraid they'll have to implement them themselves because people will want them. That doesn't mean it's a breach of contract. . .
>
> If Sun claims that there is a copyright in the language, which I know, I can't believe that they would, and that it's covered by the JDK registration, then its license for the

[101] Civil Action No. C97-20884 PVT (N.D. Cal. filed Oct. 7, 1997).

[102] Sun Microsystems, Inc., Second Amended and Supplemental Complaint¶ 83 (visited Oct. 5, 1999) <http://java.sun.com/lawsuit/amend_complaint.html>.

[103] *Id.* at ¶ 118.

> JDK is licensed to Microsoft with right to modify, adapt and create derivative works.
>
> If Sun claims that the JDK registration doesn't cover it, which I think is the correct result, there's no need for a license.
>
> If I want to add a new word to the English language, am I creating a dialect? Do I need to ask somebody's permission? I can't copy somebody's dictionary.
>
> Say I want to invent a word and I want people to adopt it. 30 years ago, there was no word PC or personal computer. Somebody started calling it a PC. Now it's part of the English language. Does that mean we have a new English, a new dialect of English? It doesn't mean that. It's just a new word. It doesn't harm anybody. It's a useful tool.
>
> Mr. Day [attorney for Sun]: . . . Mr. Quackenbush said that Sun does not have a copyright in the language. I suggest you look at the front page of the Java language specification. You'll see the copyright notice there. Java does claim, or Sun does claim a copyright in the Java language.[104]

One commentator has addressed the possibility of the issue of copyrightability of programming languages arising in the lawsuit, stating:

> A dispute between Sun and one of their [sic] licensees, Microsoft, is emerging because Microsoft has modified its version of Java to increase compatibility with their [sic] Windows operating system without Sun's permission. This dispute may erupt into litigation centering on the question of whether computer languages are copyrightable subject matter — and it may turn out that Sun's rigorous licensing requirements are unenforceable.[105]

[104] Sun Microsystems, Inc., Transcript of Proceedings in Sun Microsystems, Inc. v. Microsoft Corp., 9/10/98 (visited Oct. 5, 1999) <http://java.sun.com/lawsuit/hearing.091098.html>.

[105] Hamilton and Sabety, *supra* note 3, at 241-42.