Disrupitve Effects of the Google v. Oracle Case on Copyright Protection of Software

Bogdanović, Marina

University of Zagreb

Faculty of Law

Department for Information Technology Law and Informatics

Marina Bogdanović

DISRUPTIVE EFFECTS OF THE GOOGLE V. ORACLE CASE ON COPYRIGHT PROTECTION OF SOFTWARE

Master's thesis

Mentor: Tihomir Katulić, Ph.D., Assoc. Prof.

Zagreb, May 2022

Sveučilište u Zagrebu

Pravni fakultet

Katedra za pravo informacijskih tehnologija i informatiku

Marina Bogdanović

UČINCI PREDMETA GOOGLE PROTIV ORACLEA NA AUTORSKOPRAVNU ZAŠTITU RAČUNALNIH PROGRAMA

Diplomski rad

Mentor: izv. prof. dr. sc. Tihomir Katulić

Zagreb, svibanj 2022.

Declaration of Authenticity

I, Marina Bogdanović, under full moral, material and criminal responsibility, herewith declare that I am the exclusive author of the master's thesis and that no unauthorised use of any part of other works (use without proper citation) was made and that I did not use any sources other than those listed herein.

Marina Bogdanović, m.p.

_____

Izjava o izvornosti

Ja, Marina Bogdanović, pod punom moralnom, materijalnom i kaznenom odgovornošću, izjavljujem da sam isključivi autor diplomskog rada te da u radu nisu na nedozvoljeni način (bez pravilnog citiranja) korišteni dijelovi tuđih radova te da se prilikom izrade rada nisam koristila drugim izvorima do onih navedenih u radu.

Marina Bogdanović, v.r.

_____

Table of Contents

1. Introduction

The development of computer programs[1] mirrors and surpasses in many ways the advances made in other fields of science and technology in the past decades. It would be hard to find a field in which the advance of technology has had such an impact on our daily lives. From parking meters to stock markets, today's world depends on the ones and zeroes hidden in computer code. At the same time, the speed of development of this field outstrips the speed by which established legal concepts can be adapted to facilitate its growth and protection.

Across the world, copyright is currently being applied as the optimal legal concept that regulates computer programs, affording them protection as literary works. *"The copyright framework has proven to be a secure and enforceable framework and has successfully fostered the growth and development of content industry, software included."*[2] Copyright facilitates the development of programs by forbidding the monopolisation of ideas while guaranteeing protection from the moment of a program's creation, that is, fixation in any tangible medium. Available worldwide[3] and requiring no additional steps to be established, copyright is deemed an elegant, cost-free solution that requires no or a minimal amount of paperwork and lawyer involvement.

However, when disputes arise that only courts can resolve, the rules of copyright do not always seem to give clear answers when it comes to the protection of programs. This can be seen from the following quotations:

"We are mindful that the application of copyright law in the computer context is often a difficult task. See Lotus Dev. Corp. v. Borland Int'l, Inc., 49 F.3d 807, 820 (1st Cir.1995) (Boudin, J., concurring) ('Applying copyright law to computer programs is like assembling a jigsaw puzzle whose pieces do not quite fit.')."[4]

"The fact that computer programs are primarily functional makes it difficult to apply traditional copyright concepts in that technological world."[5]

---

[1] For the purposes of this paper, the terms "computer program", "program" and "software" are used interchangeably, as defined in §101 of the Title 17 of the Code of Laws of the United States of America, see p. 32

[2] Katulić, T.: *Protection of Computer Programs in European and Croatian Law – Current Issues and Development Perspective*, Zbornik Pravnog fakulteta u Zagrebu, Vol. 65, No. 2, 2015, pp. 239 - 240

[3] The majority of the countries in the world are signatories to the World Trade Organisation ("WTO") Agreement on Trade-Related Aspects of Intellectual Property Rights ("TRIPS"). Under Article 10(1) of the TRIPS Agreement, programs enjoy copyright protection.

[4] United States Court of Appeals for the Federal Circuit, *Oracle America, Inc. v. Google, Inc*., May 9, 2014, p. 17

[5] Supreme Court of the United States, *Google LLC v. Oracle America, Inc*., April 5, 2021, p. 35

Due to a strong personal interest of the author of this paper in matters of the protection of intellectual property in general and computer programs in particular, the author chose to research this topic as it seemed fit to provide a deeper understanding of the nuances of computer programs protection. The essential issue, lying at the heart of both the protection of computer programs and this work, is the conflict of creativity and functionality inherent in programming. One can hardly argue that an efficient and highly functional piece of engineering lacks creativity, the same applies to well-written software, yet it is wholly functional and designed with the sole purpose of solving a problem. The creativity inherent in a well-designed computer program deserves protection, yet at the same time ideas, methods and processes are not copyrightable. The Google LLC v. Oracle America, Inc. case ("Google v. Oracle") reflects the current state of the copyright protection of computer programs and serves as a reminder of crucial gaps in that system that some authors seem to bridge more easily than others.

This work aims to review the Google v. Oracle case and discuss the implications thereof on the legal protection of computer programs in the United States of America ("U.S.") and in the European Union ("EU").

Before presenting the course of the *"copyright case of the century"*,[6] and to give much-needed context, it is necessary to become acquainted with the basics of the Java and Android technologies, the application programming interface technology, which is central to this case, and with the declaring code, a minuscule, misunderstood, yet surprisingly prevalent piece of code. Following this, the history preceding the case, and the relevant U.S. legal concepts and regulations will be presented in short. The case will be reviewed by presenting some of the most important copyright-related arguments of the parties and the diverging legal interpretation of both facts and law by the U.S. courts involved. To give a European perspective, the differences between the U.S. and the EU copyright systems shall be briefly presented to serve as the introduction to the analysis of how the EU courts might adjudicate such a case. In the conclusion, the current state of copyright protection of computer programs in the U.S. and the EU will be contemplated.

[6] Lemely, M.: "*This is the copyright case of the century. (…) However it is decided, it has the potential to reshape not only software copyright law but copyright doctrine more generally*." According to Stanford Law School press, available at: https://law.stanford.edu/press/supreme-court-finally-takes-up-google-v-oracle/, last accessed on December 15, 2021

2. Technology background

Before delving into the case *per se*, the complex technology, terms and concepts that are used throughout the case need to be elaborated in short to make the specificity of the object of dispute clear.

2.1. What is Java?

"Write once, run everywhere." This is the slogan under which Sun Microsystems, Inc., the company that developed Java back in 1995,[7] was marketing its software platform,[8] as well as encouraging the usage of its programming language,[9] both named – Java. *"From laptops to data centres, game consoles to scientific supercomputers, cell phones to the Internet, Java is everywhere!"*[10] The Java programming language was designed to ease the burden on computer programmers[11] which has, due to the rise of incompatible operating systems, been growing year in year out.[12]

2.1.1. Java as a platform

The Java platform is a type of computer software platform which serves as a base upon which programs, usually applications,[13] can be developed and subsequently run. It is intended primarily for use on desktop computers and laptops. There are four types of Oracle-issued Java

---

[7] According to Java T Point: https://www.javatpoint.com/history-of-java, last accessed December 15, 2021

[8] According to Techopedia, a platform is defined as "*a group of technologies that are used as a base upon which other applications, processes or technologies are developed*." Available at: https://www.techopedia.com/definition/3411/platform-computing, last accessed December 15, 2021

[9] According to Tech Terms: "*A programming language is a set of commands, instructions, and other syntax use to create a software program. Languages that programmers use to write code are called 'high-level languages.' This code can be compiled into a 'low-level language,' which is recognized directly by the computer hardware. High-level languages are designed to be easy to read and understand. This allows programmers to write source code in a natural fashion, using logical words and symbols*." Available at: https://techterms.com/definition/programming_language, last accessed December 15, 2021

[10] According to Java.com: https://web.archive.org/web/20111124090716/http:/www.java.com/en/about/, last accessed December 15, 2021

[11] The terms computer programmer ("programmer") and software developer ("developer"), as well as the terms programming and developing, are being used interchangeably for the purposes of this paper.

[12] According to Oracle: "*These past few years you've seen the growth of multiple incompatible hardware architectures, each supporting multiple incompatible operating systems, with each platform operating with one or more incompatible graphical user interfaces*." May 1996, available at: https://www.oracle.com/java/technologies/introduction-to-java.html, last accessed December 15, 2021

[13] Applications or application programs "*are programs written to solve specific problems, to produce specific reports, or to update specific files. A computer program that performs useful work on behalf of the user of the computer (for example a word processing or accounting program) as opposed to the [system software] which manages the running of the computer itself, or to the [development] software which is used by programmers to create other programs*." Thakur, D.: *What is Application Program and Application Software*?, available at: https://ecomputernotes.com/fundamental/terms/application-program, last accessed December 15, 2021

platform: Standard Edition (Java SE)[14], Enterprise Edition (Java EE)[15], Micro Edition (Java ME)[16] and Java FX[17]. Java SE is one of the most widespread Java platforms, around which the Google v. Oracle case revolves.

The Java platform consists of the following elements: the Java language, the Java Class Library ("JCL"), the Java Development Kit ("JDK"), the Java Runtime Environment ("JRE") and the Java Virtual Machine ("JVM"). The JVM and the JCL can be found in both the JDK and the JRE.[18]

The JDK is a type of software development kit[19] ("SDK") which enables programmers to write programs in the Java language by providing a set of components for developing and debugging the Java applications. The most important set of components in the JDK are the JVM, the JCL and development tools.[20] In contrast, the JRE enables end-users to run and use the readily available Java applications that run in the JVM, while taking up much less memory space than the JDK as it only contains the components of the Java platform necessary to perform the task, such as the JCL and files that the JVM uses at runtime.[21]

2.1.2. Java as a programming language

The Java programming language is a general-purpose, object-oriented[22], class-based[23] language. Java being an object-oriented programming language, objects are the focal point around which all Java syntax elements are built, and a common denominator for other object-

[14] For more information on Java SE, visit Java T Point, available at: https://www.javatpoint.com/java-se, last accessed December 15, 2021
[15] For more information on Java EE visit Oracle.com, available at: https://www.oracle.com/java/technologies/java-ee-glance.html, last accessed December 15, 2021
[16] For more information on Java ME visit Oracle.com, available at: https://www.oracle.com/java/technologies/javameoverview.html, last accessed December 15, 2021
[17] For more information on Java FX, visit Open JFX, available at: https://openjfx.io/, last accessed December 15, 2021
[18] For more information on the difference between JDK, JRE, and JVM, visit Java T Point, available at: https://www.javatpoint.com/difference-between-jdk-jre-and-jvm, last accessed December 15, 2021
[19] Valdellon, L.: *What is an SDK? Everything You Need to Know*, available at: https://clevertap.com/blog/what-is-an-sdk/, last accessed December 15, 2021
[20] For more information on tools and files in the JDK, for example the JDK 7, visit Oracle, available at: https://docs.oracle.com/javase/7/docs/technotes/tools/, last accessed December 15, 2021
[21] For more information on the difference between the JRE, the JDK and the Java SE platform, visit Java.com, available at: https://java.com/en/download/help/techinfo.html, last accessed December 15, 2021
[22] Fathima, S.: *Functional Programming VS Object Oriented Programming (OOP) Which is better….?*, available at: https://medium.com/@shaistha24/functional-programming-vs-object-oriented-programming-oop-which-is-better-82172e53a526, last accessed December 15, 2021
[23] For more information on class-based vs. prototype-based languages, visit Li60.zendesk.com, available at: https://li60.zendesk.com/hc/en-us/articles/115002448751-Class-based-vs-prototype-based-languages, last accessed December 15, 2021

oriented languages, like the C++ language[24]. Real-life objects (e.g., a television set) consist of their state (e.g., on, off, current channel, current volume) and related behaviour (e.g., turn on, turn off, change channel, change volume).[25] In object-oriented programming, software objects apply the same logic by consisting of their state and their behaviour. In the Java language, an object's state is stored in fields[26] (that contain data for that object e.g., values like the number π), and its related behaviour is executed ("implemented") through methods[27].

Both fields and methods are members of a class, the fundamental Java syntax element. A class serves as a layer of abstraction, a blueprint from which individual objects are created.[28] It consists of the class header and the class body. A class header introduces a class body by specifying ("declaring") the name of a class and other elements that define a class. A class body contains the already mentioned fields, methods, and other relevant parameters. For programs written in the Java language to be able to run, every program must have at least one class containing at least one method.

A class may have subclasses that inherit[29] the functionality of that class, creating a specialised type of object of the original class. In terms of real-life objects, a television set would be an object of a class, whereas a LED television set would be an object of a subclass which inherits the general functionality of its television set class. Therefore, instead of rewriting the code of an entire class anew, to make its subclass, the inheritance feature resolves that problem by creating an abstraction of an entire (super)class using the word extends plus the name of that (super)class. A subclass may inherit functionality only from one class. However, a subclass can inherit functionality from more than one interface[30] which is a Java

[24] The C++ language was formerly referred to as "*C with Classes*". See: Hossain, A.: *C++ vs Java: Basic Comparison, Key Differences, & Similarities*, available at: https://hackr.io/blog/cpp-vs-java, last accessed December 15, 2021

[25] For more information on objects in the Java language, visit Oracle.com, available at: https://docs.oracle.com/javase/tutorial/java/concepts/object.html, last accessed December 15, 2021

[26] *Ibid*., fields are also called variables in functional programming languages. For more information on fields and variables in the Java language visit Oracle.com, available at: https://docs.oracle.com/javase/tutorial/java/nutsandbolts/variables.html, last accessed December 15, 2021

[27] Lowe, D.: *Methods and Method Declaration in Java*, available at: https://web.archive.org/web/20180616174802/http://www.dummies.com/programming/java/methods-and-method-declaration-in-java/, last accessed December 15, 2021

[28] For more information on class in the Java language, visit Oracle.com, available at: https://docs.oracle.com/javase/tutorial/java/concepts/class.html, last accessed December 15, 2021

[29] For more information on inheritance in the Java language, visit Oracle.com, available at: https://docs.oracle.com/javase/tutorial/java/concepts/inheritance.html, last accessed December 15, 2021

[30] Not to be confused with the usual meaning of the word interface, or application programming interface. For more information on interface, visit Oracle.com, available at: https://docs.oracle.com/javase/tutorial/java/concepts/interface.html, last accessed December 15, 2021

syntax element of the same level as a class that also contains methods, thus being able to surpass the class single-inheritance problem.

A method is a lower Java syntax element, a block of command lines ("statements")[31] which consist of the method header and the method body. The method header introduces the method body by declaring the name and functionality of a method, the number, order, type and name of parameters used, as well as what type of value the method's result will hold. The method body implements a method by doing *"the heavy lifting, namely the actual work of taking the inputs, crunching them, and returning an answer"*.[32]

In the Google v. Oracle case, both method and class headers are referred to as the declaring code, while method and class bodies are referred to as the implementing code.[33] The implementing code statements are longer and more complex than the declaring code statements and can be changed for better performance, if need be, without changing the declaring code. It should be noted that, due to expediency, throughout the case, it was common for courts to almost exclusively discuss the relationship between the declaring code and the implementing code by comparing the method header and the method body, however, that does not exclude the class header and the class body from the scope of this case.

A higher Java syntax element is a package which is *"a namespace that organizes a set of related classes and interfaces"*.[34] In its function, it is very similar to an organisation folder. A group of packages constitute a Java library. The JCL consists of groups of Sun, now Oracle-developed packages.

A method can either be written from scratch in a program that is being developed and be invoked ("called") inside that program when needed for a certain operation or be called from the JCL or some other Java-compatible library.[35] The package–class–method format is

---

[31] For more information on statements, visit Oracle.com, available at: https://docs.oracle.com/javase/tutorial/java/nutsandbolts/expressions.html, last accessed December 15, 2021

[32] United States District Court For The Northern District Of California ("District court"), *Oracle America, Inc. v. Google Inc*., May 31, 2012, p. 8, available at: https://docs.justia.com/cases/federal/district-courts/california/candce/3:2010cv03561/231846/1202, last accessed December 15, 2021

[33] Instead of the term "declaring code", the District court used the term "declaration" line of code and "header" line of code interchangeably. See: The District court, *Oracle America, Inc. v. Google Inc*., May 31, 2012, p. 5

[34] For more information on packages, visit Oracle.com, available at: https://docs.oracle.com/javase/tutorial/java/concepts/package.html, last accessed December 15, 2021

[35] "*A method is like a subroutine*." District court, *Oracle America, Inc. v. Google Inc*., May 31, 2012, p. 6; Although similar in function, it should be noted that methods and subroutines operate differently. In functional programming languages, methods' counterparts are called functions or subroutines. In those languages, subroutines run their operations on the data given to them as an input, the data and the subroutines remaining separated. However, in object-oriented programming languages, objects encapsulate both the data and methods into instances (the concrete occurrence of objects) and in that way hide the fields from outside access ("data

important for making method calls. When making a method call from the JCL in the program that is being developed, a method would be called by using the command format "java.package.Class.method()". For example, "java.lang.Math.max()" is a real method call that would follow that command format. A method computes with the inputs (arguments) which are placed in the brackets of that method.

In the first verdict of the Google v. Oracle case, judge William Haskell Alsup used the "java.lang.Math.max()" method call to show how a block of statements is organised in the package – class – method form and how a computer behaves when a method is called. Judge Alsup used the double slash sign to indicate his explanations for each line of the code:

```
"  package java.lang;                          // Declares package java.lang
   public class Math {                         // Declares class Math
        public static int max (int x, int y) { // Declares method max
             if (x > y) return x ;              // Implementation, returns x or
             else return y ;                    // Implementation, returns y
             }                                  // Closes method
   }                                           // Closes class
```

To invoke this method from another program (or class), the following call could be included in the program:

int a = java.lang.Math.max (2, 3);

Upon reaching this statement, the computer would go and find the max method under the Math class in the java.lang package, input '2' and '3' as arguments, and then return a '3,' which would then be set as the value of 'a.'

(…)

The word 'public' means that other programs can call on it [the Math class]. (If this instead says 'private,' then it can only be accessed by other methods inside the same class.) The word 'static' means that the method can be invoked without creating an instance of the class. (If this instead is an instance method, then it would always be invoked with respect to an object.) The word 'int' means that an integer is returned by the method. (Other alternatives are 'boolean,' 'char,' and 'String' which respectively mean 'true/false,' 'single character,' and 'character string.') The word 'max' is a [method's] name (…). The phrase '(int x, int y)'

encapsulation*")*, which reduces the cost of maintaining a large object-rich program, while increasing its security, quality, and reliability. See: Fathima, S., *op. cit.*, fn. 22.

identifies the arguments that must be passed into the method, stating that they will be in integer form."[36]

2.1.3. The Java Virtual Machine

Apart from the syntax, both the C++ and the Java language utilise language-specific libraries, and the so-called compilers which are translating programs that convert ("compile") the human-readable computer program ("source code") into a sequence of zeros and ones ("machine code") that is directly executable by a computer's physical processor.

However, unlike the C++ compiler, the Java compiler provides an intermediate step by translating source code into the Java bytecode.[37] Java bytecode is, in turn, converted into machine code by the Java interpreter,[38] which is a vital part of the JVM, the Java stack-based[39] virtual machine.[40] This additional translation step makes Java a portable,[41] platform-independent language[42] so that a Java code *"will run without modification on multiple operating systems and hardware architectures"*.[43] Hence, the source code does not need to be written anew by a programmer, saving the time otherwise needed to make a cross-platform program and cutting the cost of a program for the end-user. At the same time, this necessarily means that the JVM is platform dependent as it needs to be designed in a way that the same bytecode can run on different operating systems. Therefore, the Java platform's JVM is designed in a way that, once installed on the host operating system, enables running programs

---

[36] District court, *Oracle America, Inc. v. Google Inc*., May 31, 2012, p. 10

[37] "*Bytecode is the intermediate representation of a Java program, allowing a JVM to translate a program into machine-level assembly instructions. When a Java program is compiled, bytecode is generated in the form of a .class file. This .class file contains non-runnable instructions and relies on a JVM to be interpreted*." A. Bansal: *View Bytecode of a Class File in Java*, available at: https://www.baeldung.com/java-class-view-bytecode, last accessed December 15, 2021

[38] For more information on Java interpreter, visit Java T Point, available at: https://www.javatpoint.com/java-interpreter, last accessed December 15, 2021

[39] A stack-based virtual machine is "*an abstraction of a computer, that emulates a real machine. Generally, it is built as an interpreter of a special bytecode, that translates it in real time for execution on the CPU [computer's central processing unit]*." Bergia, A.: *Stack Based Virtual Machines*, available at: https://andreabergia.com/stack-based-virtual-machines/, last accessed December 15, 2021

[40] "*A virtual machine (VM) is a virtual environment that functions as a virtual computer system with its own CPU, memory, network interface, and storage, created on a physical hardware system (located off- or on-premises)*." For more information on virtual machines, visit Red Hat, available at: https://www.redhat.com/en/topics/virtualization/what-is-a-virtual-machine, last accessed December 15, 2021

[41] Roulo, M.: *Java's three types of portability*, available at: https://www.infoworld.com/article/2076944/java-s-three-types-of-portability.html, last accessed December 15, 2021

[42] The term "platform-independent" means that the Java language's compiled code (bytecode) can run on virtually any operating system, under the condition that a JVM version which supports that operating system exists. For more information, visit Geeks for Geeks, available at: https://www.geeksforgeeks.org/java-platform-independent/, last accessed December 15, 2021

[43] According to Oracle, see fn. 12

written in the Java language by executing the Java bytecode, or other programs compiled in the Java bytecode, on virtually any computer.

2.2. Libraries in software development

The conciseness and simplicity of design make source code written in any programming language easier to read and therefore, update, maintain, as well as find and correct errors. Like in human languages, copious repetition in the text makes it less comprehensible and harder to use for programmers. It also may prolong the time needed to run a program, making it less efficient.

In computer science, libraries[44] are collections of prewritten, non-volatile, reusable components, such as source code, documents and other files used by programmers for the development of computer programs in various programming languages. In layman's terms, libraries are collections of standardised, readily available blueprints that a software developer can use to accelerate the computer program creation process. It usually refers to collections of source code which are developed to execute frequently used functions in a computer program.

Although programmers may develop their own, tailor-made libraries in a certain programming language, it is a common practice to use prewritten blocks of statements from a standard library. Standard libraries[45] enable the large-scale reuse of standardised library components used by software developers worldwide. One of the most popular standard libraries for the Java language is the JCL, developed and maintained by Sun, now Oracle America.[46]

2.2.1. The Java Class Library

While developing a program, sometimes it might prove more beneficial to write it from scratch and build a custom, private library in the Java language, consisting exclusively of unique method calls, declaring and implementing code. Despite this option, the JCL is one of

---

[44] "*A software library is a suite of data and programming code that is used to develop software programs and applications. It is designed to assist both the programmer and the programming language compiler in building and executing software*." According to Techopedia, available at: https://www.techopedia.com/definition/3828/software-library, last accessed December 15, 2021

[45] "*A good standard library is essential to the long-term success of a programming language. However, there's no consensus on what or how much should be included in a language's standard library. Python and Java have large standard libraries while C takes a minimalistic approach*." According to Devopedia, available at: https://devopedia.org/standard-library, last accessed December 15, 2021

[46] There are other JCL-related standard libraries for the Java programming language such as the GNU Classpath *"Essential Libraries for Java"*, and the Apache Harmony library, see *infra* 3.2.2. and 3.3.2.1.

the key attributes of the Java programming language meaning that, once the JDK is installed on a computer, the Java programmer will usually start using prewritten methods contained in the JCL to expedite the writing process.

The JCL[47] is a Java language standard library and a part of the Java platform. As the name suggests, the JCL contains a vast collection of classes which are ingeniously organised in packages. Packages serve as folders or containers in which classes are grouped by their functional or other kinds of logical similarity. The package–class–method organisation within the JCL prevents naming conflicts and facilitates the reuse of methods and classes.

In 1996, the JCL was published with only 8 packages of prewritten programs. Its value was soon recognised, and it started to expand rapidly. It has been established that three JCL packages, java.lang, java.io and java.util, are the *"core"* Java packages, i.e., fundamental to using the Java language *"in order to make any worthwhile use of the language"*.[48] By 2008, the JCL contained 166 packages, with more than 600 classes and 6000 methods.[49] Ease of reuse and the variety of functionalities hidden in implementing code, as well as intuitive, easy-to-remember names selected as method calls are some of the reasons why the JCL gained popularity among programmers, who understood the value of elegant, quick-to-write code.

Both the JCL's method calls and the features of the Java language, e.g., inheritance, serve as abstractions, abbreviations, or means to achieving the conciseness of source code. There exists a degree of interdependence between the Java language and the JCL which, to an extent, limits programmers in their freedom of creation, tying together the public domain Java language and the copyrighted JCL. Therefore, to programmers, the line between the Java language and the JCL is, in a way, obfuscated.

2.2.2. Application programming interface

The way the JCL is organised, however ingenious, would mean little if it were not for the method calls. A method call written in the program connects to the related declaring code in the JCL, which in turn connects to the related implementing code. Once a programmer has placed a method call in a program, this process continues to be repeated every time a program

[47] According to Simply Coding: "*Java Class Library (JCL) are dynamically loadable libraries that Java applications can call at run time*." Available at: https://simplycoding.in/java-class-libraries-and-packages/, last accessed December 15, 2021
[48] The District Court for the Northern District of California, *Oracle America, Inc. v. Google Inc*., May 31, 2012, p. 12
[49] *Ibid*., p. 5

is run, creating a permanent link between that program and the JCL. Method calls are shorthand for declaring code, a way for a program to interact with the JCL in which the declaring code serves the function of a software interface.[50] It could be said that the declaring code contains an exact address, a location of the specific method that is being called upon. The JCL's declaring code is commonly referred to as the Java application programming interface ("Java API").[51]

2.2.2.1. What is an API?

The application programming interface ("API") is a type of software interface that originally served as an interface between end-user application programs and other computer programs, hence its name. Nowadays, API as a term also includes utility software and hardware interfaces.[52]

The API consist of two components:

a) a software interface, which enables one program to interact with another program, and
b) the API documentation,[53] also called the API specification or the API contract, in which it is described how to use the API and what functionality, or services, may be expected from that API.

Thanks to the API, a programmer knows what information needs to be provided to the library and what function execution is to be expected back. This means that not only does a programmer not need to write the desired function for his program, but he also does not need to know how that library function has been written, or how exactly it executes an operation.

[50] According to PC Mag: *"Software interfaces (programming interfaces) are the languages, codes and messages that programs use to communicate with each other and to the hardware."* Available at:https://www.pcmag.com/encyclopedia/term/interface, last accessed December 15, 2021

[51] Kaufman, J. R.: *What Google v. Oracle means for open source*: "*The declaring code is, in essence, a 'software interface' allowing access to a software module's various methods. Said another way, it allows one software module to interface, pass information to/from, and control another software module*." https://opensource.com/article/21/5/google-v-oracle, last accessed December 15, 2021

[52] Bloch, J.: *A Brief, Opinionated History of the API*, recorded at Q Con, August 8, 2018, available at: https://www.infoq.com/presentations/history-api/, last accessed December 15, 2021

[53] According to Altexsoft.com: "*No matter how many opportunities for creating or extending software products API gives, it would remain an unusable piece of code if developers didn't understand how to work with it. Well-written and structured API documentation that explains how to effectively use and integrate an API in an easy-to-comprehend manner will make a developer happy and eager to recommend the API to peers. (…) The API documentation is a reference manual with all needed information about the API*." Available at: https://www.altexsoft.com/blog/engineering/what-is-api-definition-types-specifications-documentation/, last accessed December 15, 2021

The API speeds up the writing of code and boosts programmers' productivity, enabling them to focus on the uniqueness of their program by streamlining the writing process.

The Java API defines a set of functionalities, describing what a certain code in a library does, serving as an abstraction layer independent of its implementing code. It allows Oracle engineers to make changes to the implementing code, e.g., to maintain or update it, without compromising the usage of the Java API, or rather, the declaring code, rendering it intact. Java API consistency is essential both for programmers who integrated that API into their program's code, as they do not need to rewrite that program every time an official update is made to the JCL, and for the end-users of that program, as the program always runs smoothly and without a hitch.[54]

2.2.2.2. Overview of API history

The API as an idea was developed before a term for the API was invented. In the 1940s, computer scientists Wilkes and Wheeler, while working on an early computer called EDSAC, developed a subroutine library that was stored on punched paper tape and organised in a filing cabinet, together with the library catalogue that contained the description of each subroutine and instructions on how to incorporate them into a computer program. In today's terms, a library catalogue is a functional equivalent of the API documentation.[55]

API as a term was mentioned for the first time in 1968, in the paper "Data structures and techniques for remote computer graphics"[56] as an application program interface, lacking the "-ing" suffix. The term was used in that paper to describe the interaction between an application, a graphics program, with the rest of the computer software on the same computer. The API was intended to make the library hardware independent, as computer hardware would eventually be replaced with other, more advanced hardware. Until this point in time, the API was considered an integral part of a library, as a library was necessarily an integral part of a computer. Not much thought was given to the portability of the library, and by the same token, to the API as means of achieving that portability.[57]

---

[54] Lane, K.: *Intro to APIs: What Is an API?*, available at: https://blog.postman.com/intro-to-apis-what-is-an-api/, last accessed December 15, 2021; see also Bloch, J., *op. cit.*, fn. 52

[55] The first published API documentation can be found in Wilkes, M.; Wheeler, D.; Gill, S.: *The Preparation of Programs for an Electronic Digital Computer*; according to Bloch, J., *op. cit.*, fn. 52

[56] Cotton, I.; Greatorex, F. S.: *Data structures and techniques for remote computer graphics*, AFIPS (Fall, part I), 1968, available at: https://www.semanticscholar.org/paper/Data-structures-and-techniques-for-remote-computer-Cotton-Greatorex/060f4a49e8984f7b82efff4c5ca13166e0ee4811, last accessed December 15, 2021

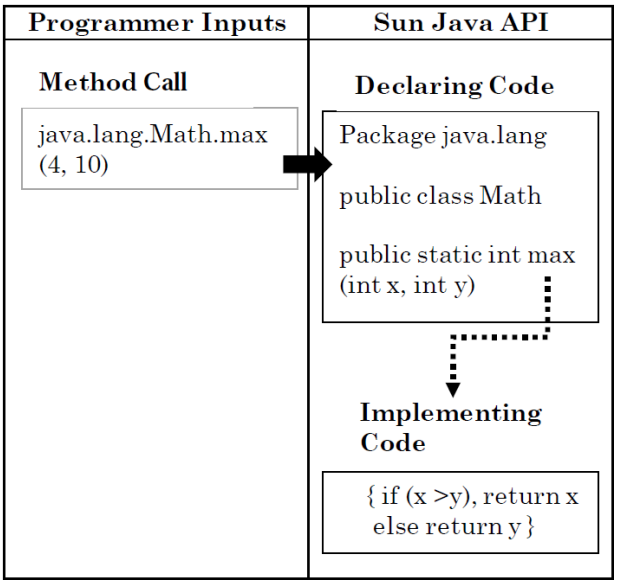
[57] Bloch, J., *op. cit.*, fn. 52

In the early 2000s, due to the spread of the Internet and the ever-growing need for web development and web services, the web API started to gain traction, as it served the purpose of an intermediary between a web page's server and a user's computer. The dot-com companies[58] like Amazon.com, Inc. ("Amazon") and eBay, Inc. used the web API to *"impact the commercial landscape"*.[59] The API evolved into a type of interface which allows any two separate computer programs to interact with each other, which is nowadays fuelling the development of Cloud computing, social networking services and the Internet of Things.

2.2.2.3. The API and the JCL

In the Google vs. Oracle case, the Java SE Class Library as a whole, its declaring code and implementing code, are referred to as the Java API, stressing the importance of API when it comes to using the JCL. However, courts only discuss JCL's declaring code as that is a point of contention. Therefore, the terms Java API and declaring code (as well as the method declaration as a synonym for the latter term) are often used interchangeably, effectively ruling only on the declaring code, as the computer scientists were originally defining the API, namely, as the declaring code.[60] The courts' view of the Java API remained consistent throughout the procedure. For this reason, this paper is also referring to the declaring code in one of the abovementioned three ways. The following graph[61] shows how the Supreme Court of the United States ("SCOTUS") understood[62] the Java API:

[58] See *infra* 3.1.2.
[59] Lane, K.: *Intro to APIs: History of APIs*, available at: https://blog.postman.com/intro-to-apis-history-of-apis/, last accessed December 15, 2021
[60] In the *Motion for Leave to File Brief of 78 Amici Curiae and Brief of 78 Amici Curiae Computer Scientists In Support Of Petitioner*, software interfaces are described as follows: "*Software interfaces, including those embodied in the Java Application Programming Interface (API) at issue here, (…) merely describe what functional tasks a computer program will perform without specifying how it does so. The Java API's functional interfaces, called declarations, are written using the Java programming language, which mandates each declaration's precise form. In contrast, implementations provide the actual step-by-step instructions to perform each task included in an interface*." See *infra* fn. 337
[61] SCOTUS, *Google LLC v. Oracle America, Inc*., *Appendix to opinion of the Court B Sun Java API Diagram*, April 5, 2021, p. 38
[62] It should be noted that, although not shown in this graph, the SCOTUS included in its analysis the structure, sequence and organisation in the Java API and ruled on the declaring code and SSO. The implementing code was not a point of contention. The SCOTUS analysed in detail only one element of the Java API, the declaring code. See *infra* 6.4.



Sun Java API Diagram

The JCL necessarily contains the API, being a standard Java library. However, albeit an important part of well-designed libraries, the API should be distinguished from them.[63] When writing in the Java language, a program may be written without the JCL's API and without creating a program-specific API. Even when a programmer creates a custom-built declaring code for a non-standard library, it is possible that the declaring code in such a library is neither open nor intended for public use. This difference is worth highlighting because, as will be shown in this paper, in the Google v. Oracle case, Google had used the publicly available, standard JCL declaring code in the Android platform.

[63] According to the Rapid API Blog: "*An API can be thought of as the logical representation of what is in the library, or the consistent format that explains what a developer can do with the library. It is the part of the code that is accessible to programmers. The main difference is that the library refers to the code itself, while API refers to the interface*." Available at: https://rapidapi.com/blog/api-vs-library/, last accessed December 15, 2021

3. Historical background

Thanks to the platform independence made possible by the JVM and the Java API, programs could be written for the Java platform more quickly than in other object-oriented programming languages. Owing to those characteristics, it didn't take long for the Java platform to become, and stay, one of the most popular software development platforms in the world.[64]

To understand the evolution and importance of the Java platform, it is necessary to give context to the history of the company that developed Java, and the market conditions which colour the events that led to the case discussed in this paper.

3.1. The company that developed Java

Sun Microsystems, Inc. ("Sun"), founded in 1982, was a technology company known for developing and selling both hardware and software products, most notable being SPARC-based and x86-based computer systems, the Java platform, the Solaris operating system (based on AT&T's UNIX operating system, further developed by Sun), as well as software later acquired by Sun: StarOffice, VirtualBox and MySQL. It was described as *"the last standing, fully integrated computing company, adding its own value at the chip, operating system and systems level"*.[65]

3.1.1. The rising Sun

Up until the 1990s, Sun was first and foremost a hardware company, competing for its market share with companies like IBM Corporation ("IBM") and Hewlett-Packard, Inc., focusing on the B2B[66] segment by selling computers and servers, bundled together with the Solaris operating system. After the great Internet explosion in the mid-1990s,[67] the competition

---

[64] According to Greig, J.: *8 of the most popular programming languages*, available at: https://www.techrepublic.com/article/8-of-the-most-popular-programming-languages/, last accessed December 15, 2021

[65] Kirkpatrick, D.: *Meanwhile, Back at Headquarters... Scott McNealy Made Sun the Hottest Server Company. Now He's Got to Prove That Java's More Than Just a Good Buzz. His Toughest Job's The One Ahead*, Fortune Magazine, October 13, 1997, available at: https://archive.fortune.com/magazines/fortune/fortune_archive/1997/10/13/232511/index.htm, last accessed December 15, 2021

[66] According to Investopedia: "*Business-to-business (B2B) refers to business that is conducted between companies, rather than between a company and individual consumer. Business-to-business stands in contrast to business-to-consumer (B2C) and business-to-government (B2G) transactions*." Available at: https://www.investopedia.com/terms/b/btob.asp, last accessed December 15, 2021

[67] For more information on the Internet Explosion, visit Encyclopedia.com, available at: https://www.encyclopedia.com/science/encyclopedias-almanacs-transcripts-and-maps/internet-explosion, last accessed December 15, 2021

only grew when the Sun-developed Java platform caught traction in the worldwide software developer community.

Thus, the Microsoft Corporation ("Microsoft") was added to the list of Sun's competitors, because why devote one's time to develop an application just for Windows OS using Microsoft's tools, when Java could be utilised to develop it for Windows, as well as UNIX, and Apple OS, without the drudgery of rewriting it from scratch? Although at the time Sun was battling with other software companies in courts as well, the dispute with Microsoft stands out as it illustrates the fierceness of the competition between Information Technology ("IT") titans, as well as what happens when they clash due to one company encroaching the other company's market. This foreshadows what the future has in store for IT companies both big and small.

If everyone had switched to the Java platform for application development, it would mean that the other software companies' tools for development wouldn't be as used anymore, hence those companies were to lose a part of their market share and would potentially be pushed out of that market segment entirely.

Microsoft proceeded to license the Java platform for the development of MS Internet Explorer 4.0. However, by developing its own, Windows-only compatible version of Java, thus breaching Sun's Java interoperability policy, Microsoft engaged in the *"embrace, extend, extinguish"*[68] anti-competition strategy by trying to marginalise, or even extinguish, the Java platform as a whole.[69] This clash of the titans ended in Sun suing and eventually settling the case against Microsoft, terminating their licencing agreement, and permanently barring Microsoft from using the *"Java compatible"* trademark.[70]

[68] According to the Free Dictionary: "*A business strategy of implementing a public standard or developing software compatible with it, adding features not supported by the public standard, then marginalizing competition that does not (or cannot) support such proprietary additions. Popularized during the United States v. Microsoft Corp. antitrust trial as a phrase used internally by Microsoft regarding its own business philosophy*." Available at: https://idioms.thefreedictionary.com/Embrace%2c+extend%2c+and+extinguish, last accessed December 15, 2021

[69] Wong, W.: *Sun vs. Microsoft: Clash of the titans*, available at: https://www.zdnet.com/article/sun-vs-microsoft-clash-of-the-titans-5000121284/, last accessed December 15, 2021

[70] Shankland, S.: *Sun, Microsoft settle Java suit*, available at: https://www.cnet.com/news/sun-microsoft-settle-java-suit/, last accessed December 15, 2021

3.1.2. The dot-com bubble

Sun's market value peaked in the late nineties,[71] during the dot-com bubble,[72] thanks to the skyrocketing stock prices of the so-called dot-com companies. The dot-com companies were doing most of their business on the Internet, especially those with prefixes like "web-" and "e-", or the suffix "-.com" in their name. *"We are the dot in the dot-com"* was Sun's slogan at the time, as Sun was focused on the B2B segment selling servers and becoming the dominant server provider to every more serious dot-com start-up company. [73] These companies hoped to make easy money, anticipating high returns on their investments, which never came about.

Sun's decline began after the dot-com bubble burst, with the majority of the aforementioned start-up companies shutting down virtually overnight, emptying Sun's servers, the capacity of which was initially drastically expanded in order to support the incoming flood of all those dot-com companies. Sun's then-president and COO Ed Zander jokingly talked about changing their slogan into *"anybody wants to buy a server?"*.[74] Even before the dot-com bubble burst, in the late nineties, Sun was already losing its share in the chips marketplace, being pushed out by the Intel Corporation's low-end chips. In the software market, Sun's proprietary StarOffice kept losing its market share to Microsoft's MS Office, while Sun's Solaris as a proprietary operating system was losing its share to the open-source[75] Linux operating system.

3.1.3. Sunset policy changes

The company's new policy of giving to the open-source community came in the wake of a new course set for the old titan among technology companies by the new CEO Jonathan

[71] According to ARN, available at: https://www.arnnet.com.au/slideshow/334210/pictures-remember-rise-fall-sun-microsystems/, last accessed December 15, 2021
[72] According to Techopedia, available at: https://www.techopedia.com/definition/26175/dot-com-boom, last accessed December 15, 2021
[73] Gomes, L.: *Sun Microsystems' Rise and Fall*, available at: https://www.forbes.com/2009/03/18/sun-microsystems-internet-technology-enterprise-tech-sun-microsystems.html, last accessed December 15, 2021
[74] Vance, A.: *Sun ditches its dot in dot-com slogan*, available at: https://www.computerworld.com/article/2796937/sun-ditches-its-dot-in-dot-com-slogan.html, last accessed December 15, 2021
[75] According to Opensource.com: "*Open source software is software with source code that anyone can inspect, modify, and enhance.*" Available at: https://opensource.com/resources/what-open-source, last accessed December 15, 2021

Schwartz, who tried to make Sun predominantly a software company focused on Java.[76] [77] Sun became known for the contributions made to the open-source community, to the benefit of both proprietary, closed-source and open-source software development. In 2006 the UNU-MERIT reported[78] that Sun held the first place in the category of private companies that contributed the greatest amount of free (libre) and open-source code ("FLOSS")[79] to the open-source community. In other words, Sun had by 2006 made the most lines of publicly available source code. Sun's contribution to the open-source community can be seen in the following examples: the 2000 release of OpenOffice as the open-source version of StarOffice, the 2007 release of the Open Java Development Kit ("OpenJDK") as the open-source version of the JDK, and the 2008 release of OpenSolaris as the open-source version of Solaris.

Sun tried to make a profit by providing services and support (e.g., updates, security patches) for its open-source products[80], but it was too little too late. Even though Sun avoided its annihilation in 2003, in the following years it had never come close to its former profitability, and after 7 years of dedicated work and struggle, it was finally sold off. Sun was a victim of poor investments and the changing market environment at the beginning of the 21st century.[81] After failed negotiations with IBM, on April 20, 2009, the Oracle Corporation ("Oracle") finally struck a deal with Sun's board of directors, buying Sun for 7.4 billion dollars.[82] Sun became Oracle America, Inc., a subsidiary of Oracle.

---

[76] Rooney, P.: *Schwartz: Sun is world's largest open source company*, available at: https://www.zdnet.com/article/schwartz-sun-is-worlds-largest-open-source-company/, last accessed December 15, 2021

[77] Schwartz, J.: *The Rise of JAVA – The Retirement of SUNW*, available at: https://jonathanischwartz.wordpress.com/2007/08/23/the-rise-of-java-the-retirement-of-sunw/, last accessed December 15, 2021

[78] *Study on the: Economic impact of open source software on innovation and the competitiveness of the Information and Communication Technologies (ICT) sector in the EU*, UNU-MERIT, prepared by: Ghosh, R. A., p. 51, available at: https://szabadszoftver.kormany.hu/letoltesek/idegen-tanulmanyok/2006-11-20-flossimpact.pdf, last accessed December 15, 2021

[79] The author of this paper recognises the difference between the terms "free (libre)" and "open-source"; however, the discussion of these is beyond the scope of this paper. For the sake of brevity, the author uses only the term "open-source" when referring to FLOSS software as it is the most commonly used term when comparing FLOSS software to proprietary software. For more on the difference between free (libre) and open-source code see Stallman, R.: *Why Open Source Misses the Point of Free Software*, available at: https://www.gnu.org/philosophy/open-source-misses-the-point.en.html, last accessed December 15, 2021

[80] Anderson, T.: *Giving away software makes good sense for Sun*, available at: https://www.theguardian.com/technology/2008/feb/21/opensource.sunmicrosystems, last accessed December 15, 2021

[81] According to Information Week, available at: https://www.informationweek.com/sun-microsystems-falls-victim-to-internet-bust-and-strategy/d/d-id/1019496, last accessed December 15, 2021

[82] Dignan, L.: *Oracle buys Sun; Now owns Java; Becomes a hardware player*, available at: https://www.zdnet.com/article/oracle-buys-sun-now-owns-java-becomes-a-hardware-player, last accessed December 15, 2021

3.2. Communities that amplified Java's success

The two communities of programmers, the Java community and the open-source community, which overlap to an extent, have contributed additional value to the Java platform. The Java platform was already in and of itself a success, but these communities had greatly boosted its popularity, attracting other programmers and companies to use it. Over time, Java's ever-growing popularity would come back to haunt it.

3.2.1. Java Community Process

One of the reasons why the Java SE had been growing in its popularity ever since its 1995 release was its cross-platform interoperability and the Java community. The Java community[83] is a worldwide community of Java developers who share solutions for the problems they encounter while using the Java platform as well as ideas for its evolution and optimisation. Sun recognised this and encouraged programmers to give feedback and share their solutions and ideas. The Java Community Process[84] ("JCP") was, and still is, one of the ways programmers can participate in Java development. The JCP is a formalised mechanism founded by Sun and active since 1998, which accelerated the evolution of the Java platform. It gathers companies, non-profit organisations and individuals willing to contribute to the *"community development of Java technology specifications"*[85] by developing and revising new Java technology.

Official additions and changes to the Java platform are initiated by a JCP member submitting a Java Specification Request ("JSR").[86] The JSR is a formal proposal that contains *"the actual descriptions of proposed and final specifications*[87] *for the Java platform"*, that is proposed additions and changes to the Java platform. The JSR proposed source code needs to

[83] For more information on the Java community, visit Oracle, available at: https://www.oracle.com/java/technologies/javacommunity.html, last accessed December 15, 2021
[84] More information on the JCP available at: https://jcp.org/en/procedures/overview, last accessed December 15, 2021
[85] More on this topic available at: https://jcp.org/en/participation/membership, last accessed December 15, 2021
[86] More on the JCP available at: https://jcp.org/en/jsr/overview, last accessed December 15, 2021
[87] "*A precise statement of the effects that an individual program is required to achieve. It should clearly state what the program is to do without making any commitment as to how this is to be done*." Encyclopedia.com, available at: https://www.encyclopedia.com/computing/dictionaries-thesauruses-pictures-and-press-releases/program-specification, last accessed December 15, 2021; see The University of Iowa, *Prologue on Program Specification*, available at: https://homepage.cs.uiowa.edu/~fleck/spec.html, last accessed December 15, 2021; See also Java SE 6 API specification list as an example, Oracle, available at: https://docs.oracle.com/javase/6/docs/api/index.html, last accessed December 15, 2021

demonstrate its Java platform compatibility through the Technology Compatibility Kit[88] ("TCK"). The TCK was provided at Sun's discretion and included additional licensing terms and fees. The changes to the Java platform are made after the JCP Executive Committee,[89] a JCP body consisting of the biggest Java vendors and end-user companies led originally by Sun and now by Oracle, establishes the JSR's compatibility and finally approves its addition to the Java platform.

In this way, not only did Sun's programmers work on the development of Java software but also, by inviting all interested parties, Sun could respond more quickly to new emerging trends in Java development. At the same time, Java standardisation was ensured so that it would remain the *"write once, run everywhere"* compatible software platform, avoiding the fragmentation of the Java platform and maintaining its interoperability.[90]

3.2.2. Open-source community and the Java schism

Sun was developing and offering the Java platform under the Sun Community Source License,[91] a proprietary license[92] which incorporates some elements of the open-source licences.[93] This made the Java platform a proprietary software, while its JDK *per se* was being distributed at no cost, thus making it free for anyone wishing to program in Java. However, once a program was developed using the JDK, regardless of whether its author opted to put his newly created program under a proprietary license or an open-source software license, it would become JDK dependent, meaning that the program would not run on any other platform other than the Java platform. The program thus became permanently dependent on Sun's support, which came at a premium and was at the same time rendered useless outside of the Java platform. In 2004, Richard Stallman, a copyleft pioneer and the founder of the Free Software Foundation and the GNU Project, described this JDK dependency as *"the Java trap"*, a vendor

[88] For more information on the TCK visit Foojay.io, available at: https://foojay.io/pedia/tck/, last accessed December 15, 2021
[89] More on the JCP Executive Committee available at: https://jcp.org/en/participation/committee, last accessed December 15, 2021
[90] The JCP ensures "*Java technology's standard of stability and cross-platform compatibility*." More information available at: https://jcp.org/en/introduction/overview, last accessed December 15, 2021
[91] For more information on Sun Community Source License, visit Flylib.com, available at: https://flylib.com/books/en/2.603.1.35/1/, last accessed April 30, 2022
[92] For more information on proprietary licences, see *infra* 4.1.
[93] For more information on open-source licences, see *infra* 4.2.; see also *Open Source User Guide*, available at: https://www.open.ac.uk/library-research-support/sites/www.open.ac.uk.library-research-support/files/files/Open%20Source%20Software%20Licences%20User%20Guide%20V4%20-%20clean%20copy(1).pdf, last accessed December 15, 2021

lock-in.[94] Due to the growing discontent in the Java community, Sun's concerns about the fragmentation[95] of the Java platform came true, as new open-source, clean-room[96] developed versions of the Java platform's compilers, virtual machines and libraries had started to appear in the Java community.

In computer science, a clean-room design is a type of reverse engineering[97] where a group of programmers, according to their client's general instruction on what the program is supposed to do, but not on how it is supposed to do it, develop a version of the copyrighted program in a so-called clean environment, that is, without seeing the inspiring program's source code. Afterwards, a third party checks for any infringing content by comparing the two source codes. In this way, the program is developed without infringing any copyrights of the original program's author. This technique, however, does not protect against patent infringement. The most notable open-source, clean-room implementations of the Java platform include the GNU compiler[98], the Kaffe[99] virtual machine, the GNU Classpath[100] libraries and the Apache Harmony[101] platform.

To the end of responding to the above-stated open-source trends, while at the same time wanting to attract as many new Java users as possible, in 2006 Sun released[102] most of its Java source code as the OpenJDK,[103] its open-source version of the JDK, which was under the GNU General Public License version 2 ("GPLv2").[104] The GPLv2 is a type of copyleft licence[105]

---

[94] Stallman, R.: *Free but Shackled – The Java Trap*, available at: https://www.gnu.org/philosophy/java-trap.html, last accessed December 15, 2021

[95] "*When used to describe software platforms, the term fragmentation generally refers to the proliferation of diverging variants—a situation in which many custom versions of the software platform emerge and coexist with the original. Platform fragmentation can weaken interoperability because applications that are built for one variant might not work on others.*" Paul, R.*: Android fragmentation: something to fear?*, available at: https://arstechnica.com/information-technology/2010/06/ars-explains-android-fragmentation/, last accessed December 15, 2021

[96] For more information on Clean Room, visit Technology and IP Law Glossary, available at: http://www.ipglossary.com/glossary/clean-room/#.YMzVbfkzY2w, last accessed December 15, 2021

[97] Schwartz, M.: *Reverse-Engineering*, available at: https://www.computerworld.com/article/2585652/reverse-engineering.html, last accessed December 15, 2021

[98] More information on GNU compiler available at: https://gcc.gnu.org/, last accessed December 15, 2021

[99] More information on Kaffe available at: http://www.kaffe.org/, last accessed December 15, 2021

[100] More information on GNU Classpath available at: https://www.gnu.org/software/classpath/, last accessed December 15, 2021

[101] More information on Apache Harmony library available at: https://cwiki.apache.org/confluence/display/harmony, last accessed December 15, 2021

[102] LeMonica, M.: *Sun picks GPL license for Java code*, available at: https://www.cnet.com/news/sun-picks-gpl-license-for-java-code/, last accessed on December 15, 2021

[103] Oracle America periodically releases an updated Java SE version for OpenJDK, usually after the initial Java SE release, which is intended for commercial use.

[104] More information on the GPLv2 available at: https://www.gnu.org/licenses/old-licenses/gpl-2.0.html, last accessed December 15, 2021

[105] For more information on copyleft licences, see *infra* 4.2.

that requires the source code, which is developed by using parts of the GPLv2 licensed source code, to be licensed under the same license, and published in its entirety. The GPLv2 licensed source code can be used under the condition that it satisfies all license obligations, despite any other conflicting legal obligations that might exist, such as software patents. This led to many for-profit companies who did not wish to open their proprietary source code to avoid using open-source code licensed under the GPLv2.

The Java platform's TCK became available for OpenJDK implementations licensed under the GPLv2 licence. This mostly halted the further fragmentation of the Java platform, with many open-source Java platform implementations being discontinued, no longer maintained, or seeing no new improved releases.[106]

Since Oracle's Java SE 9[107] release in 2017, the OpenJDK license has been changed from the GPLv2, a full copyleft licence, to the GNU General Public License version 2 with the linking (Classpath) exception ("GPLv2 with Classpath exception"), a partial copyleft licence.[108] In that way, when a developed program is linked to the OpenJDK's library so that it produces a larger executable program, it does not have to be licensed under the full terms of GPLv2, but rather the author may freely choose the licence under which he wants to put his program, be it proprietary or open-source, with no obligation to subject the developed program to the full terms of the GPLv2 and make the whole source code public.

3.3. Android on the horizon

The Android platform ("Android") project had been started in 2003 by Andy Rubin, Rich Miner, Nick Sears, and Chris White, the founders of the company Android, Inc. The aim of the project was originally envisioned as the development of an operating system for digital cameras, however, it soon grew into an operating system for smartphones. In 2005 Android,

---

[106] See *supra* fnn. 99, 100, 101 and 102

[107] According to Oracle, available at: https://www.oracle.com/java/technologies/javase/jdk-faqs.html, last accessed December 15, 2021

[108] For more information between a full and a partial copyleft licence, see *infra* 4.2., pp. 30-31.
More information on GPLv2 with Classpath exception available at: https://openjdk.java.net/legal/gplv2+ce.html, last accessed December 15, 2021
See also Sass, R.: *Top 9 GPL With the Classpath Exception Questions Answered*, available at: https://www.whitesourcesoftware.com/resources/blog/top-9-gpl-with-the-classpath-exception-questions-answered/, last accessed December 15, 2021

Inc. was bought by the search engine giant Google, Inc. ("Google"). The original Android founders came to work for Google and continued their work on the Android project.[109]

3.3.1. Open Handset Alliance

On November 5, 2007, Google announced the creation of the Open Handset Alliance ("OHA"), a consortium of numerous mobile device manufacturing companies, software development companies, mobile network operators and semiconductor manufacturing companies. The OHA was founded to further develop the Android operating system and *"significantly lower the cost of developing and distributing mobile devices and services"*.[110] The most notable companies in the OHA consortium are Google, Sony Corporation, T-Mobile (Deutsche Telekom AG), eBay, Inc., Qualcomm, Inc., China Mobile, Ltd., Intel Corporation, Samsung Group, Motorola, Inc., NVIDIA Corporation and Texas Instruments, Inc.[111]

3.3.2. First building blocks of Android

Android was envisaged as open source and its development would not have been possible without the open-source technology discussed below. These are the building blocks which have formed the foundation for Android development, enabling Android's rapid growth, but also lead to legal issues, including the Google v. Oracle case.

3.3.2.1. Apache Harmony library

Apache Harmony, the *"Open Source Java SE"* [112] is an open-source, clean-room Java implementation platform. It was announced in 2005 by the Apache Software Foundation,[113] a non-profit, open-source software development corporation. Apache Harmony was licensed

---

[109] Elgin, B.: *Google Buys Android for Its Mobile Arsenal*, available at: https://web.archive.org/web/20060427095759/http://www.businessweek.com/technology/content/aug2005/tc20050817_0949_tc024.htm, last accessed December 15, 2021

[110] According to T-Mobile, available at: https://www.t-mobile.com/news/press/industry-leaders-announce-open-platform-for-mobile-devices, last accessed December 15, 2021

[111] According to Open Handset Alliance, available at: http://www.openhandsetalliance.com/oha_members.html, last accessed December 15, 2021

[112] According to Apache Software Foundation, available at: http://harmony.apache.org/, last accessed December 15, 2021

[113] More information on the ASF available at: https://www.apache.org/foundation/, last accessed December 15, 2021

under the permissive[114] Apache License according to which it is not mandatory for Apache Harmony-derived software to be further distributed under the same license or required that its entire source code be made available to the public. As this type of license was deemed more favourable by for-profit corporations, many of them, including Google and IBM, both used and contributed source code to the Apache Harmony project.

The Apache Software Foundation ("ASF") was one of the JCP Executive Committee members who could not acquire a compatible license for the Java platform TCK from Sun. Therefore, the ASF could not test their Java implementation against the TCK,[115] and their version of the implementation never officially became compatible and interoperable with the Java platform. As a result, even though the Apache Harmony was *de facto* Java-compatible, it was never recognised by the Sun. In 2007 Sun released OpenJDK, granting the use of the TCK to OpenJDK derivatives or clean-room implementations that are licensed under the copyleft GPL v2 license. Apache Harmony did not qualify as it was licensed under the permissive Apache License, one of the main reasons why Sun never allowed ASF to acquire the TCK. Three years later, this led to ASF resigning in protest from its JCP Executive Committee seat.[116] The same year IBM, one of the biggest participants of the Apache Harmony project, decided to leave the project and join Sun's OpenJDK.[117] In 2011, one year later, the Apache Harmony project ended.[118] One of the reasons why the Apache Software Foundation left the JCP was the unwillingness of Sun to release the TCK for other GPL v2 incompatible licences, ultimately exposing the users of Apache Harmony's code to possible litigation, as can be seen in the Google v. Oracle case.

As the Apache Software foundation stopped maintaining Apache Harmony in 2011, Google, as a member of the OHA and the Android Open Source Project, took it upon itself to continue to both maintain and use the subset of the Apache Harmony library for Android development. Thus, since Android's inception, and even after the Apache Harmony project ended in 2011, a subset of the Apache Harmony project's library was still being used to develop

---

[114] For more information on permissive licences, see *infra* 4.2.; visit also FOSSA, available at: https://fossa.com/blog/all-about-permissive-licenses/, last accessed December 15, 2021

[115] For more information on this subject, visit the ASF, available at: https://www.apache.org/jcp/sunopenletter.html, last accessed December 15, 2021

[116] According to the ASF, available at: https://blogs.apache.org/foundation/entry/the_asf_resigns_from_the, last accessed December 15, 2021

[117] Paul, R.: *Java wars: IBM joins OpenJDK as Oracle shuns Apache Harmony*, available at: https://arstechnica.com/information-technology/2010/10/ibm-joins-openjdk-as-oracle-shuns-apache-harmony/, last accessed December 15, 2021

[118] According to InfoQ, available at: https://www.infoq.com/news/2011/11/apache-harmony-finale/, last accessed December 15, 2021

Android. Only in 2016, with the release of the newest Android version at the time, were the Apache libraries abandoned in favour of the OpenJDK,[119] because since that year Oracle had licensed it under the GPL v2 with the Classpath exception. Despite the Google v. Oracle case at that point being already ongoing for six years, the switch to OpenJDK had multiple advantages, as possible future legal issues for Google could be avoided with the GPL v2 Classpath exception version of the licence, while at the same time offering Google a more up-to-date code base supported by Oracle, to develop Android upon.

3.3.2.2. Linux kernel

The Linux kernel is an open-source, Unix-like[120] operating system kernel[121]. A kernel is the fundamental software at the core of an operating system. It was created in 1991 by Linus Torvalds as an open-source, clean-room version of AT&T's proprietary Unix kernel. It was used by Android developers as one of the foundational building blocks because it enabled them to adapt the kernel to Android's mobile phone software development needs,[122] making Android in fact a fork[123] of the Linux kernel.

The influence of Linux and Linus Torvalds in the development of Android was not only limited to the use of the Linux kernel but perhaps, it could be said that at least equally important was the adaptation of the concept of the "Bazaar", whose originator according to Eric Raymond was Linus Torvalds. Google managed to create a hybrid between the "Cathedral" and the "Bazaar" approaches to programming, having both a dedicated core of experts, *"individual*

---

[119] Amadeo, R.: *Android N switches to OpenJDK, Google tells Oracle it is protected by the GPL*, available at: https://arstechnica.com/tech-policy/2016/01/android-n-switches-to-openjdk-google-tells-oracle-it-is-protected-by-the-gpl/, last accessed December 15, 2021

[120] According to Known Host: "*When we call an OS 'Unix-like,' that generally will mean that the source code of the* [operating system] *(…) is directly traceable to, has similar properties to, and is explicitly based on Unix. (…) The umbrella-term Unix-like also refers to clones of Unix. A clone is software that performs in a similar way to other software but does not have the same source code*." Available at: https://www.knownhost.com/blog/unix-like-mean/, last accessed December 15, 2021

[121] "*The kernel is a core component of an operating system and serves as the main interface between the computer's physical hardware and the processes running on it. The kernel enables multiple applications to share hardware resources by providing access to CPU, memory, disk I/O, and networking*." Horcasitas, J.: *What Is a Kernel?*, available at: https://www.digitalocean.com/community/tutorials/what-is-a-kernel, last accessed December 15, 2021

[122] "*Linux gives the Android developers a pre-built, already maintained operating system kernel to start with so they don't have to write their own kernel*." Hoffman, C.: *Android is Based on Linux, But What Does It Mean*?, available at: https://www.howtogeek.com/189036/android-is-based-on-linux-but-what-does-that-mean/, last accessed December 15, 2021

[123] Definition of fork according to Webopedia: "*To split source code into different development directions. Forking leads to the development of different versions of a program. Forking often occurs when the development of a piece of open source code has reached an impasse. The project is forked so that the code can be developed independently in different ways with different results*." Available at: https://www.webopedia.com/definitions/fork/, last accessed December 15, 2021

wizards or small bands of mages working in splendid isolation, with no beta to be released before its time (…)" and the support of a wide and dedicated user base *"(…) a great babbling bazaar of differing agendas and approaches."*[124]

The Android SDK was released in 2009. Unlike the JCP, the Android Open Source Project[125] was not stifled by restrictive licensing and control over key development tools like the TCK. This approach netted Google a rapid yet controlled development cycle free from the rigidity of closed in-house development, all the while fostering popularity and a community of programmers that would guarantee the longevity and adaptability of their product.

An apparent legal issue arose from the union of the Linux kernel and the Apache Harmony library in Android. The Linux kernel was licensed under the GPLv2, while the Apache Harmony library was licensed under the Apache license. GPLv2 being a copyleft licence and the Apache licence being a permissive licence created a legal incompatibility between these two fundamental building blocks, as pointed out by Richard Stallman.[126] Thus there existed an additional legal issue, and a point in time in which Google could have been sued for a license violation, but these fears were laid to rest due to an understanding that *"the kernel system call*[127] *interfaces do not in any way result in a derived work as per the GPL, and the kernel details are exported through the kernel headers to all the normal [GNU C Library] interfaces (…)"*[128] because using *"kernel services by normal system calls (…) is merely considered normal use of the kernel"*.[129]

3.3.3. Development of Android

Although neither human nor programming languages can be copyrighted, the perks of the Java language, as shown above, are intertwined with the copyrightable Java platform. As the Java language seemed the most suitable for the rapid development of Android, Google

[124] Raymond, E. S.: *The Cathedral and the Bazaar*, O'Reilly, 1999, p. 21
[125] For more information on the Android Open Source Project, visit: https://source.android.com/, last accessed December 15, 2021
[126] Stallman, R.: *Android and Users' Freedom*, available at: https://www.gnu.org/philosophy/android-and-users-freedom.html, last accessed December 15, 2021
[127] According to GeeksforGeeks: "*In computing, a system call is the programmatic way in which a computer program requests a service from the kernel of the operating system it is executed on. A system call is a way for programs to interact with the operating system*." Available at: https://www.geeksforgeeks.org/introduction-of-system-call/, last accessed December 15, 2021
[128] Vaughan-Nichols, S. J.: *Linus Torvalds on Android, the Linux fork*, available at: https://www.zdnet.com/article/linus-torvalds-on-android-the-linux-fork/, last accessed December 15, 2021
[129] According to DejaCode, available at: https://enterprise.dejacode.com/licenses/public/linux-syscall-exception-gpl/, last accessed December 15, 2021

entered licencing negotiations with Sun regarding the rights to the Java platform. Google wanted to adapt the Java platform for use in mobile devices, so an Android co-development partnership deal with Sun was being negotiated over the span of several months.[130] Other documents show Google was considering the acquisition of a Java licence and was in periodical negotiations with Sun since 2005 and up until Oracle's acquisition of Sun in 2010.[131] However, according to Google's CEO Eric Schmidt's testimony, since Sun allegedly wanted a certain amount of control over Android along with royalties between $30 million to $50 million,[132] an agreement with Sun could not be reached. In the Google v. Oracle case, the United States Court of Appeals for the Federal Circuit found that Google did not make Android compatible with the JVM or interoperable with programs written for the Java platform, which has resulted in Sun not granting Google a Java licence.[133]

As an agreement with Sun could not be reached, Google decided to commence the development of Android by forking the Linux kernel and using the Apache Harmony libraries. Google chose code to integrate 37 JCL packages that consist of around 7000 lines[134] of declaring code into Android. *"Google believed Java application programmers would want to find the same 37 sets of functionalities in the new Android system callable by the same names as used in Java."*[135] To achieve this, it copied the declaring code of those 37 JCL packages, also present in the Apache Harmony library, and proceeded to write its own implementing code. As to those 37 packages, the JCL and Android library have the same package – class – method structure and the same name tree, enabling programmers to write the same method calls in Java and Android.

Instead of using the JVM, the Google-led team used Apache Harmony's library to develop the Dalvik virtual machine[136] ("Dalvik") and other remaining components of Android.

---

[130] District court, *Oracle America, Inc. v. Google Inc*., May 31, 2012, p. 6, line 1-5

[131] According to Technical Guidance, available at: https://technical-guidance.com/article/google-wanted-to-license-java-from-sun1619020366, last accessed December 15, 2021; See also: J. Niccolai: *Google: Sun offered to licence Java for $100M*, available at: https://www.computerworld.com/article/2509401/google--sun-offered-to-license-java-for--100m.html, last accessed December 15, 2021

[132] Brodkin, J.: *Sun wanted up to $50 million from Google for Java license, Schmidt says*, available at: https://arstechnica.com/tech-policy/2012/04/sun-wanted-up-to-50-million-from-google-for-java-license-schmidt-says/, last accessed December 15, 2021

[133] "*The point of contention between the parties was Google's refusal to make the implementation of its programs compatible with the Java virtual machine or interoperable with other Java programs. Because Sun/Oracle found that position to be anathema to the 'write once, run anywhere' philosophy, it did not grant Google a license to use the Java API packages*." CAFC, *Oracle America, Inc. v. Google Inc*., May 9, 2014, p. 6

[134] *Ibid*., p. 30

[135] District court, *Oracle America, Inc. v. Google Inc*., May 31, 2012, p. 6, line 15-17

[136] For more information on Dalvik, visit GeeksforGeeks, available at: https://www.geeksforgeeks.org/what-is-dvmdalvik-virtual-machine/, last accessed December 15, 2021

Dalvik is an open-source, clean room, register-based[137] virtual machine optimised for mobile devices which is licensed under the Apache License.[138] As Android programs are predominantly written in the Java language, their source code was compiled into Java bytecode by a Java language just-in-time compiler, and then additionally to Dalvik bytecode so that they could run on Dalvik. As their function is very similar, Dalvik is JVM analogous. However, Dalvik was designed with low memory mobile devices in mind, as a result being more efficient when running multiple applications at the same time as opposed to the JVM.[139]

3.3.4. The release of Android

Riding the ever-growing wave of the open-source community, on November 5, 2007, the development team led by Andy Rubin, in cooperation with members of the Open Handset Alliance, proceeded to announce Android, an open-source mobile operating system, and *"the first truly open and comprehensive platform for mobile devices"*.[140] In his blog, Sun's CEO offered his heartfelt congratulations by saying that *"(...) Google and the Open Handset Alliance just strapped another set of rockets to the community's momentum (...)."*[141]

At the start of the Google v. Oracle case, Android contained 168 packages in its library, of which 37 corresponded in functionality to the previously mentioned 37 JLC packages. This creates limited interoperability[142] between the Android and Java platforms, under the condition that a certain program would use only those 37 packages.[143] Otherwise, code written for one cannot be executed on the other. This effectively means that the similarity between these two

---

[137] "*Closely related to stack based VM are register based virtual machines. They are also interpreters of bytecode, but their design is quite different, since they don't use the stack for the operands but rather a set of registers. While they tend to be more complex, they are also generally faster at runtime*." Bergia, A.: *op. cit.*, fn. 39, p. 8

[138] Both Dalvik and the majority of Android is licenced under the Apache licence. For more information visit GoogleGit, available at: https://android.googlesource.com/platform/dalvik/+/2ad60cfc28e14ee8f0bb038720836a4696c478ad/README.txt, last accessed December 15, 2021

[139] According to Baeldung, available at: https://www.baeldung.com/java-jvm-vs-dvm, last accessed December 15, 2021

[140] According to the OHA, available at: http://www.openhandsetalliance.com/press_110507.html, last accessed December 15, 2021

[141] Schwartz, J.: *Congratulations Google, Red Hat and the Java Community!*, available at: https://web.archive.org/web/20101023072550/http:/blogs.sun.com/jonathan/entry/congratulations_google, last accessed December 15, 2021

[142] "*Although Android uses the Java programming language, it is undisputed that Android is not generally Java compatible. As Oracle explains, 'Google ultimately designed Android to be incompatible with the Java platform, so that apps written for one will not work on the other.'*" CAFC, *Oracle America, Inc. v. Google Inc*., May 9, 2014, p. 12

[143] *"During oral argument, Google's counsel stated that 'a program written in the Java language can run on Android if it's only using packages within the 37. (...)' Counsel did not identify any programs that use only the 37 API packages at issue, however, and did not attest that any such program would be useful. Nor did Google cite to any record evidence to support this claim*." *Ibid*., p. 51

platforms mainly extends as far as the familiarity of programmers with the method calls and the declaring code of those 37 JCL packages.

Open source does not mean not-for-profit; Google generates revenue primarily *"through advertisement whenever a consumer uses particular functions on an Android smartphone"*.[144] For this reason, Google's market strategy is focused on reaching as many end-users as possible. This is achieved on the one hand by inviting programmers to develop Android applications for the end-users and on the other hand, by having as many devices as possible run Android. To this end, the OHA benefited from the Apache License by making use of Android as an open-source platform, while not being restricted to making their own products open source, enabling them to keep the source code of their products closed and patents protected.

[144] District court, *Oracle America, Inc. v. Google Inc*., May 31, 2012, line 22-23

4. Software licencing

The WTO TRIPS Agreement[145] and the World Intellectual Property Organisation ("WIPO") Copyright Treaty[146] prescribe that computer programs are protected as literary works, as per the Berne Convention.[147] This *ex lege* type of protection, which is enshrined in the copyright law of the WTO and WIPO member states, is commonly referred to as the "all rights reserved" licence model. The author of a computer program may disclaim some or all of his rights to the computer program. Depending on which rights are to be disclaimed, the author may opt for placing the program under the "copyleft licence" model, the "permissive licence" model or donate it to the "public domain".

4.1. The all rights reserved licence model and the public domain

In the "all rights reserved" licence model or the "proprietary licence" model, the licensor is reserving all rights attached thereto, be they moral or economic. To use, copy, modify and/or distribute the original computer program or its modified version, a contract needs to be signed with the licensor determining the price, duration, possible restrictions and other terms of use. Any deviation from the contract constitutes a copyright infringement. This type of contract usually restricts the licensee's access to the source code of the licensed program, making it closed-source software.[148] It may come in the form of an adhesive contract requiring the licensee to agree to all of its terms and conditions, in order to proceed with the intended use of that program.

In contrast to this type of computer program copyright protection, which is considered to be the most restrictive, the licensor may disclaim all of his rights to the software and donate it to the "public domain". Once in the public domain, a computer program is free for all to use, copy, modify and distribute, be it for commercial or non-commercial purposes, without any restrictions, including no obligation to give credit to the author of the program.

---

[145] Article 10(1) of the TRIPS Agreement

[146] Article 4 of the WIPO Copyright Treaty, available at: https://wipolex.wipo.int/en/text/295166, last accessed April 30, 2022

[147] Berne Convention for the Protection of Literary and Artistic Works, Paris Act, 1971, amended in 1979, available at: https://wipolex.wipo.int/en/text/283698, last accessed April 30, 2022

[148] For more information on closed-source software visit Kaspersky IT Encyclopedia, available at: https://encyclopedia.kaspersky.com/glossary/closed-source/, last accessed April 30, 2022

4.2. The copyleft licence model and the permissive licence model

Apart from reserving or disclaiming all rights to a computer program, a licensor may also opt for one of the two open-source licence models that lie in between: the "copyleft licence" model and the "permissive licence" model. These licence models allow for more flexibility when it comes to the licensor–licensee relationship, which is especially common when it comes to collaboration between multiple parties.

The "copyleft licence" model, also known as the "reciprocal", "restrictive" or "protective open-source licence" model,[149] allows the licensee to use, copy, modify and/or distribute the original computer program under the condition that the derivative computer program has to either be licensed under the same licence as the original program or under another compatible copyleft licence. The copyleft licence model requires that the original program is attributed to its licensor and *"that anyone who redistributes the software, with or without changes, must pass along the freedom to further copy and change it."*[150] In terms of computer programs, this means that the original and its derivatives have to be distributed in the source code form, as distributing it in the object code would require the program to be reverse-engineered first, which would hinder the freedom of others to copy and modify the original. The derivative work may be commercialised, but its source code must remain open. This means the author of a derivative work may not be able to sell his derivative work, but profit may be gained through other means, e.g., through selling services related to that derivative work. The most notable copyleft licence is the GNU General Public License.

The copyleft licence model is usually applied as the so-called "full copyleft" licence which means that all parts of a program are put under the abovementioned terms. However, it may also be applied as the "partial copyleft" licence which means that it *"exempts some parts of the work from the copyleft provisions, thus permitting distribution of some modifications under terms other than the copyleft license, or in some other way does not impose all the principles of copylefting on the work."*[151] The most notable example of the "partial copyleft" licence is the GNU General Public License with the linking (Classpath) exception.

[149] The described copyleft license model is also referred to as the "strong copyleft" license model. There is another type of the copyleft license, the so-called "weak copyleft" license model which is not as relevant to this paper and is therefore not discussed. For more information on the "weak copyleft" license model visit Fossa.com, available at: https://fossa.com/blog/all-about-copyleft-licenses/, last accessed on April 30, 2022

[150] According to GNU.org, available at: https://www.gnu.org/copyleft/, last accessed April 30, 2022

[151] According to Sources Select Resources, available at: https://www.sources.com/SSR/Docs/SSRW-Copyleft.htm, last accessed on April 30, 2022

The “permissive licence” model or “non-protective licence” model allows the licensee to use, copy, modify and/or distribute the original computer program under the condition that the original program is attributed to its licensor. The original program licensed under a permissive licence must remain publicly available in its source code form, yet no such obligation exists for its derivatives. The original program’s derivatives may be licensed under any other software license, including under a proprietary licence, and its derivatives may be closed-source programs. The most notable permissive licences are the Apache License, MIT License and BSD License.

5. Google, LLC v. Oracle America, Inc: Relevant legislation and legal doctrines

After having elaborated on the technological aspects of the claims and the history preceding the case, a short review of the most relevant legislation and legal doctrines derived from case law must be presented to contextualise the case in the U.S. legal system. Considering that the Google v. Oracle case predominantly revolves around the question of copyrightability and fair use, this paper is concerned with the relevant substantive legislation of the copyright law. The substantive legislation of patent law shall not be presented in this paper as all the patent infringement claims were dismissed early on during the Google v. Oracle litigation, making patent law not central to the case at hand. The procedural legislation and thereto related questions shall also not be presented as they are not relevant for the purposes of this paper.

5.1. Title 17 of the Code of Laws of the United States of America

The Code of Laws of the United States of America ("U.S. Code") is the official codification of the general and permanent United States Congress enacted federal statutes. It is divided into 53 sections, or titles, regulating different fields of law. For the purposes of this paper, the most important title is Title 17 of the U.S. Code ("17 U.S. Code")[152] which codifies the U.S. copyright law, specifically Chapter 1 – Subject Matter and Scope of Copyright.[153]

17 U.S. Code §101, §102 and §107 should be noted as the most relevant, especially §102(b) due to its interpretation being strongly disputed by both parties. They are therefore cited for future reference:

17 U.S. Code §101 – Definitions

"A 'computer program' is a set of statements or instructions to be used directly or indirectly in a computer in order to bring about a certain result."

"'Literary works' are works, other than audiovisual works, expressed in words, numbers, or other verbal or numerical symbols or indicia, regardless of the nature of the material objects, such as books, periodicals, manuscripts, phonorecords, film, tapes, disks, or cards, in which they are embodied."

[152] The term "17 U.S. Code" is being used interchangeably with the term "Copyright Act" for the purposes of this paper.

[153] 17 U.S. Code Chapter 1 - Subject Matter and Scope of Copyright, available at: http://uscode.house.gov/browse/prelim@title17/chapter1&edition=prelim, last accessed December 15, 2021

17 U.S. Code §102 – Subject matter of copyright: In general

"(a) Copyright protection subsists, in accordance with this title, 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. Works of authorship include the following categories:

(1) literary works; (…).

(b) 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."

17 U.S. Code §107 – Limitations on exclusive rights: Fair use

"(…) the fair use of a copyrighted work (…) for purposes such as criticism, comment, news reporting, teaching (including multiple copies for classroom use), scholarship, or research, is not an infringement of copyright. In determining whether the use made of a work in any particular case is a fair use the factors to be considered shall include—

(1) the purpose and character of the use, including whether such use is of a commercial nature or is for nonprofit educational purposes;

(2) the nature of the copyrighted work;

(3) the amount and substantiality of the portion used in relation to the copyrighted work as a whole; and

(4) the effect of the use upon the potential market for or value of the copyrighted work."

Not all of the four factors must be met, but instead *"fair use is appropriate where a 'reasonable copyright owner' would have consented to the use, i.e., where the 'custom or public policy' at the time would have defined the use as reasonable."*[154] It should also be mentioned that *"[t]he Supreme Court has cautioned against adopting bright-line rules and has emphasized that all of the statutory factors 'are to be explored, and the results weighed together, in light of the purposes of copyright.'"*[155]

[154] CAFC, *Oracle America, Inc. v. Google LLC*, March 27, 2018, p. 17, quoting U.S. Court of Appeals for the Ninth Circuit, *Wall Data Inc. v. L.A. Cty. Sheriff's Dep't*, 2006

[155] *Ibid*., p. 16, quoting SCOTUS, *Campbell, aka Skyywalker, et al. v. Acuff-Rose Music, Inc.*, March 1994. Available at: https://www.law.cornell.edu/supremecourt/text/510/569, last accessed December 15, 2021

5.2. Copyright-related legal doctrines

Legal doctrines which were applied in this case are derived from the case law developed by the U.S. courts. Some of the case law doctrines eventually become a part of the U.S. statutory law, meaning they have been adopted and passed by a legislative body and are enshrined in the code of law.

5.2.1. Idea – expression dichotomy

The idea-expression dichotomy is a legal doctrine first developed in the Baker v. Selden[156] case by the Supreme Court of the United States ("SCOTUS"). It grants copyright protection to the expression of an idea, but not to the idea itself. It is enshrined in 17 U.S. Code §102(b).

5.2.2. Merger doctrine

The merger doctrine was also developed in the Baker v. Selden case.[157] Under the merger doctrine, when the idea and the expression merge together, meaning that the idea and the expression are so tied together that there is only one or a limited number of ways to express that idea, anyone is allowed to express that idea or a function in the exact same way.

5.2.3. Structure, sequence and organisation

In the Whelan v. Jaslow case[158], the U.S. Court of Appeals for the Third Circuit decided that not only the program's literal elements[159] could be infringed by copying the source code verbatim, but also the program's more abstract, non-literal elements by copying the overall structure and organisation i.e., the structure, sequence and organisation ("SSO") of the program. The court corroborated its decision by stating that *"the 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."*[160]

---

[156] SCOTUS, *Baker v. Selden*, October 1879, available at: https://www.law.cornell.edu/supremecourt/text/101/99, last accessed December 15, 2021
[157] *Ibid.*
[158] United States Court of Appeals for the Third Circuit, *Whelan Assocs., Inc. v. Jaslow Dental Laboratory, Inc.*, 1986, available at: https://casetext.com/case/whelan-associates-v-jaslow-dental-laboratory, last accessed on December 17, 2021
[159] See explanation of terms "literal" and "non-literal" elements *infra* 6.2.2.
[160] United States Court of Appeals for the Third Circuit, *Whelan Assocs., Inc. v. Jaslow Dental Laboratory, Inc.*, 1986

5.2.4. Abstraction – filtration – comparison test

In Computer Associates International, Inc. v. Altai case,[161] the U.S. Court of Appeals for the Second Circuit developed the "Substantial Similarity Test for Computer Program Structure", better known as the abstraction-filtration-comparison test. This test was created by the court to determine whether the two programs are substantially similar, which is achieved after finding and comparing their copyrightable parts.

In the abstraction step, *"a court would first break down the allegedly infringed program into its constituent structural parts."*[162]

In the filtration step, *"a court would (...) be able to sift out all non-protectable material"* by analysing those constituent parts for any *"incorporated ideas, expression that is necessarily incidental to those ideas, and elements that are taken from the public domain."*[163] The court outlined three elements in the program that are precluded from copyright protection:

a) Elements dictated by efficiency

If the programmer necessarily had to write an element of the program in a certain way to *"efficiently (...) implement that part of the program's process"*,[164] it is considered that the programmer's expression of that element has merged with the idea, hence the merger doctrine applies to that element. The court elaborated that *"(...) the concept of efficiency is akin to deriving the most concise logical proof or formulating the most succinct mathematical computation. Thus, the more efficient a set of modules are, the more closely they approximate the idea or process embodied in that particular aspect of the program's structure."*[165]

b) Elements dictated by external factors

The court gave some examples of those factors, such as: *"(1) the mechanical specifications of the computer (...); (2) compatibility requirements of other programs (...); (3) computer manufacturers' design standards; (4) demands of the industry being serviced; and (5) widely accepted programming practices (...)."*[166]

[161] United States Court of Appeals for the Second Circuit, *Computer Associates International, Inc. v. Altai*, 1992, available at: https://caselaw.findlaw.com/us-2nd-circuit/1233733.html, last accessed December 15, 2021
[162] *Ibid.*, p. 12
[163] *Ibid.*
[164] *Ibid.*, p. 15
[165] *Ibid.*, p. 14
[166] *Ibid.*, p. 16

c) Elements taken from the public domain

If certain elements of a program are taken from the public domain, they are not protected by an exclusive intellectual property right and therefore cannot be considered a part of the programmer's original work of authorship.

Lastly, in the comparison step, a court would proceed to *"compare this material with the structure of an allegedly infringing program"*[167] and decide whether the copyright is infringed.

[167] *Ibid.*, p. 12

6. The Google, LLC v. Oracle America, Inc.: Litigation

The Google v. Oracle case could be divided into three phases; in the first phase, the copyrightability question was the focus of discussion, while in the second phase, the fair use question became central to resolving the legal issue at hand, while in the third phase the SCOTUS reviewed both the issue of copyrightability as well as fair use.

As is to be expected from a case which reaches all the way back to 2010, only finding its conclusion in the April 2021 decision of the SCOTUS, the development of the case is complex and somewhat challenging to follow. The sheer volume of filing makes a thorough analysis impractical. The essence of the arguments contained in the claim, response and the amendments thereto evolve as the case progresses. The development of the parties' main arguments will be presented through the verdicts of each court, as they represent a distillation of the parties' claims made up to that point. Nevertheless, it is necessary to review the parties' initial claims and arguments so as to give proper context to the rulings.

6.1. Initial arguments of the parties

On August 12, 2010, Oracle filed the complaint[168] and the demand to the United States District Court, Northern District of California against Google. By filing an action for patent and copyright infringement, Oracle alleged that Google actively and knowingly had infringed and was infringing the following Oracle-owned patents: the '477 patent[169], the '476 patent[170],

[168] *COMPLAINT (with jury demand) For Patent and Copyright Infringement against Google Inc*., filed by Oracle America, Inc. on December 8, 2010, available at: https://docs.justia.com/cases/federal/district-courts/california/candce/3:2010cv03561/231846/1, last accessed December 15, 2021

[169] Patent No. 6,125,447, *Protection Domains To Provide Security In A Computer System* ("the '477 patent"), available at: https://patft.uspto.gov/netacgi/nph-Parser?Sect1=PTO1&Sect2=HITOFF&d=PALL&p=1&u=%2Fnetahtml%2FPTO%2Fsrchnum.htm&r=1&f=G&l=50&s1=6,125,447.PN.&OS=PN/6,125,447&RS=PN/6,125,447, last accessed December 15, 2021

[170] Patent No. 6,192,476, *Controlling Access To A Resource* ("the '476 patent"), available at: https://patft.uspto.gov/netacgi/nph-Parser?Sect1=PTO1&Sect2=HITOFF&d=PALL&p=1&u=%2Fnetahtml%2FPTO%2Fsrchnum.htm&r=1&f=G&l=50&s1=6,192,476.PN.&OS=PN/6,192,476&RS=PN/6,192,476, last accessed December 15, 2021

the '702 patent[171], the '720 patent[172], the '104 patent[173], the '205 patent[174] and the '520 patent[175]. The patents were all issued to Sun by the United States Patent and Trademark Office, which would have made Google liable to Oracle for patent infringement under United States patent law, 35 U.S. Code §271.

Oracle also alleged Google's copyright infringement of the Java platform, claiming that Google knowingly copied, prepared, published, and distributed portions of the Java platform and works derived from the Java platform (forks, distributions etc.) and at the same time enabled third parties to use, copy and distribute content which was copyrightable under the United States copyright law, 17 U.S. Code §101 *et seq.*, without such use having been licensed by Oracle, thus having violated Oracle's exclusive rights under 17 U.S. Code §106. Oracle's Java copyrights were registered[176] with the United States Copyright Office, including those registered under registration numbers TX 6-196-514[177] and TX 6-143-306[178].

Oracle claimed that Google had realised and was continuing to realise unmerited profits and advantages because of the infringement. Therefore, Oracle demanded a jury trial and prayed for a judgement that would have held Google liable on all the above-stated grounds, permanently enjoined Google and third affiliated parties from continued acts of infringement and ordered the destruction or disposal of all infringing copies of the copyrighted work. Oracle

---

[171] Patent No. 5,966,702, *Method And Apparatus For Preprocessing And Packaging Class Files* ("the '702 patent") , available at: https://patft.uspto.gov/netacgi/nph-Parser?Sect1=PTO1&Sect2=HITOFF&d=PALL&p=1&u=%2Fnetahtml%2FPTO%2Fsrchnum.htm&r=1&f=G&l=50&s1=5,966,702.PN.&OS=PN/5,966,702&RS=PN/5,966,702, last accessed December 15, 2021

[172] Patent No. 7,426,720, *System And Method For Dynamic Preloading Of Classes Through Memory Space Cloning Of A Master Runtime System Process* ("the '720 patent"), available at: https://patents.google.com/patent/US7426720, last accessed December 15, 2021

[173] Patent No. RE38,104, *Method And Apparatus For Resolving Data References In Generate Code* ("the '104 patent"), available at: https://patft.uspto.gov/netacgi/nph-Parser?Sect1=PTO1&Sect2=HITOFF&d=PALL&p=1&u=%2Fnetahtml%2FPTO%2Fsrchnum.htm&r=1&f=G&l=50&s1=RE38,104.PN.&OS=PN/RE38,104&RS=PN/RE38,104, last accessed December 15, 2021

[174] Patent No. 6,910,205, *Interpreting Functions Utilizing A Hybrid Of Virtual And Native Machine Instructions* ("the '205 patent"), available at: https://patft.uspto.gov/netacgi/nph-Parser?Sect1=PTO1&Sect2=HITOFF&d=PALL&p=1&u=%2Fnetahtml%2FPTO%2Fsrchnum.htm&r=1&f=G&l=50&s1=6,910,205.PN.&OS=PN/6,910,205&RS=PN/6,910,205, last accessed December 15, 2021

[175] Patent No. 6,061,520, *Method And System for Performing Static Initialization* ("the '520 patent"), available at: https://patents.google.com/patent/US6061520A/en, last accessed December 15, 2021

[176] More information on applying for copyright registration in the U.S. available at: https://www.copyright.gov/circs/circ01.pdf, p. 4 and 5, last accessed December 15, 2021

[177] Java 2 standard edition 1.4., copyright registration in 2002, Copyright Catalog, the U.S. Copyright Office, available at: https://cocatalog.loc.gov/cgi-bin/Pwebrecon.cgi?Search_Arg=TX0006196514&Search_Code=REGS&PID=ssHB3uJGf4NsmyKlPRytdQcTqSm-NS&SEQ=20210627073715&CNT=25&HIST=1, last accessed December 15, 2021

[178] Java 2 standard edition, version 5.0. By Sun Microsystems, Inc., Comsys, the Carl Group, PrO Unlimited, ZAO Elbrus MCST, TelTech International Corporation., copyright registration in 2004, Copyright Catalog, the U.S. Copyright Office, available at: https://cocatalog.loc.gov/cgi-bin/Pwebrecon.cgi?Search_Arg=TX0006143306&Search_Code=REGS&PID=55I_hpsI8vTGym6afeAw6FQ9j6REAg_&SEQ=20210627074330&CNT=25&HIST=1, last accessed December 15, 2021

also prayed for an award in statutory damages, damages according to proof resulting from Oracle's infringement, together with prejudgment and post-judgment interest, treble damages[179] under 35 U.S. Code §284 for deliberate infringement of Oracle's patents, and attorney's fee, litigation, and other related costs.

On October 4, 2010, Google submitted its answer[180] to Oracle's complaint. Google denied both the patent and copyright infringement and the related liability that would result in the realisation of the requests enlisted in Oracle's prayer for relief. More specifically, in the defence related to patent infringements, Google stated that the patents were either invalid, unenforceable, or used in a substantially non-infringing way, that the licence for them was implied, or that they fell under the disclosure-dedication doctrine,[181] making them dedicated to the public domain by default. Furthermore, in the defence, Google claimed protection by the doctrine of misuse of patent[182] and the doctrine of unclean hands.[183] Google also claimed there were no grounds for an injunction.

In its amended complaint, [184] Oracle put emphasis on the amendments to its copyright infringement claims by stating that Google had infringed on the Java platform copyright and enabled third parties to do so, to the end of using or developing Android specifically by deriving one third of Android's API packages from Java API packages. Oracle also introduced its structure, sequence, organisation argument by stating that *"the infringed elements of Oracle*

---

[179] "*Treble damages is a term that indicates a statute exists to award a prevailing plaintiff up to three times actual or compensatory damages. (…) Statutes exist to award treble damages in cases involving patent infringement, willful trademark counterfeiting, and antitrust violations*." According to Investopedia, available at: https://www.investopedia.com/terms/t/trebledamages.asp, last accessed December 15, 2021

[180] *GOOGLE INC.'S ANSWER to Complaint with Jury Demand, COUNTERCLAIM against Oracle America, Inc. by Google Inc..*, filed by Google, Inc. on April 10,2010, available at: https://docs.justia.com/cases/federal/district-courts/california/candce/3:2010cv03561/231846/32, last accessed December 15, 2021

[181] "*Under the disclosure-dedication doctrine, when a patentee discloses subject matter but does not claim it, the patentee dedicates the unclaimed subject matter to the public and cannot recapture it through the doctrine of equivalents. The public can then practice the unclaimed subject matter without fear of infringement*." Mahanta, S.: *Disclosure-Dedication Rule: An Effective Tool Against Infringement Claims Under the Doctrine of Equivalents*, available at: https://www.ipwatchdog.com/2020/07/13/disclosure-dedication-rule-effective-tool-infringement-claims-doctrine-equivalents/id=123275/, last accessed December 15, 2021

[182] "*Patent misuse is an affirmative defense that recognizes that it is possible for a patent owner to abuse the exclusive right enjoyed as a result of the issuance of a patent*." Quinn, G.: *Patent Misuse, Exploring the Basics* available at: https://www.ipwatchdog.com/2011/11/18/patent-misuse-exploring-the-basics/, last accessed December 15, 2021

[183] "*The unclean hands doctrine applies to cases where the plaintiff has acted unethically in connection to the circumstances that have led to the suit. The unethical behavior does not have to be criminal nor enough to justify independent judicial proceedings, but must be a willful act directly related to the issue in court*." Vredenburg, A. C.; Poindexter, A. A.: *No Justice for Unclean Hands*, available at: https://www.fosterswift.com/communications-no-justice-clean-hands-doctrine.html, last accessed December 15, 2021

[184] *Amended Complaint for Patent And Copyright Infringement,* filed by Oracle on October 27, 2010, available at: https://cases.justia.com/federal/district-courts/california/candce/3:2010cv03561/231846/36/0.pdf?ts=1288282135, last accessed December 15, 2021

America's copyrighted work include Java method and class names, definitions, organization, and parameters; the structure, organization and content of Java class libraries; and the content and organization of Java's [API] documentation."

In its amended response to the complaint,[185] Google walked back on its dedication to the public defence and instead claimed to have committed no copyright infringement, having only used elements not protected by copyright, subject to fair use, that the copying was *de minimis*, and that the infringing material was, in fact, an independent creation.

6.2. The first phase: Copyrightability of the Java API

The first phase includes the proceedings before the United States District Court for the Northern District of California and the United States Court Appeals for the Federal Circuit. The former proceedings lasted from August 2010, when the Oracle's complaint was filed, until the first verdict was reached in favour of Google, in May 2012. The latter proceedings lasted from October 2012, when Oracle's appeal was filed, until the second verdict was reached in favour of Oracle, in May 2014.

6.2.1. The first District Court for the Northern District of California ruling

On the 31st of May 2012, the United States District Court for the Northern District of California ("District court") ruled that the declaring code and the structure, sequence, and organisation ("SSO") of the Oracle's 37 Java API packages are not entitled to copyright protection, whereas the rangeCheck function is entitled to copyright protection, even though copying was *de minims*. The question of Google's fair use of the 37 Java API packages remained undecided.

Judge Alsup explained at the beginning of this verdict how, due to the complexity of the subject, he decided to conduct this trial in three phases as it would be easier to understand the issues concerned in that way. The first phase concerned copyright, in which the judge decided on the issue of copyrightability and equitable defences[186], while the jury decided on

[185] *Google Inc.'S Answer to Plaintiff's Amended Complaint for Patent and Copyright Infringement and Amended Counterclaim*, filed by Google on November 10,2010, available at: https://cases.justia.com/federal/district-courts/california/candce/3:2010cv03561/231846/51/0.pdf?ts=1376938370, last accessed December 15, 2021

[186] Equitable defence is a type of an affirmative defence in which the defendant prays for the court to dismiss the plaintiffs' claims as the plaintiff had acted in an inequitable way (e.g., fraud, unclean hands). For more on equitable defence visit Upcounsel, available at: https://www.upcounsel.com/equitable-defenses, last accessed December 15, 2021

the questions of copyright infringement, fair use, and whether the copying was *de minimis*. The second phase concerned the patent infringement, and the third phase concerned the damages.

6.2.1.1. Decisions made in the copyrightability trial phase

The judge decided as follows: Under the merger doctrine, nobody can monopolise an expression if there is only one way to express it. Therefore, anyone is allowed to write source code which carries out *"exactly the same function or specification of any and all methods used in the Java API"*.[187] In the "java.lang.Math.max()" example[188] the judge elaborated how the rules of the Java language dictate that the method declaration must be identical to declare a method with the same functionality both in Java and Android, although the implementation code of that method *per se* may be different. The method declaration represents the idea, while the method implementation represents the expression. The court concluded that *"there is only one way to write"* the declaring code, and thus the *"merger doctrine bars anyone from claiming exclusive copyright ownership of that expression."* [189] Therefore, there is no copyright infringement in using identical method declarations.

Names or short phrases are not protected under the Copyright Act, but only under the Trademark Act if trademarked. Therefore, Google had the right to use the same names of the Java classes and methods in Android, although judge Alsup recognised the fact that Google could have named Android's classes and methods differently and would still have retained the same functionality of the original, Oracle-given class and method names.

Regarding Oracle's structure, sequence and organisation argument that Google replicated the names, organisation of those names, and functionality of 37 out of 166 packages in the Java API, the judge finds that the command structure of Oracle's Java API is not protected, for any system or method of operation does not enjoy protection under the Copyright Act §102(b). However, judge Alsup recognises that the same functionality could have been achieved in Android without the identical groupings in the command structure, even though Google argued without success that *"the groupings would be so expected and customary as to be permissible under the scène à faire doctrine."*[190] As an example of this, Google pointed out

---

[187] District court, *Oracle America, Inc. v. Google Inc*., May 31, 2012, p. 34, line 15-17 available at: https://docs.justia.com/cases/federal/district-courts/california/candce/3:2010cv03561/231846/1202, last accessed December 15, 2021

[188] See *supra* 2.1.2., p. 7

[189] District court, *Oracle America, Inc. v. Google Inc*., May 31, 2012, p. 35, line 23-24

[190] *Scène à faire doctrine* (French for "scenes that must be made") is a legal principle in copyright law under which certain elements of a creative work are deemed uncopyrightable as they are customary and recurring in a book or film genre. For example, a *femme fatale* is an uncopyrightable element of a *film noire* movie.

that *"the methods included under the Math class are typical of what one would expect to see in a group of math methods"*.[191] Due to the lack of evidence for all such class groupings, the judge rejected Google's *scène à faire* argument.

Despite resembling a taxonomy,[192] which would enjoy copyright protection and has creative elements, the names in Java's command structure (the so-called "name tree") are not protected under the Copyright Act as they are symbols in the command structure in which the commands take the "java.package.Class.method()" form, and therefore have a functional purpose. Each of those words is a command which performs a specific, pre-assigned function. The judge also emphasises that copying Java's command structure is necessary for interoperability. *"Google replicated what was necessary to achieve a degree of interoperability — but no more, taking care (…) to provide its own implementations."*[193]

"Google believed Java application programmers would want to find the same 37 sets of functionalities in the new Android system callable by the same names as used in Java. Code already written in the Java language [owned by the programmers themselves and not by Oracle] would, to this extent, run on Android and thus achieve a degree of interoperability."[194]

Both parties agreed that Google's Dalvik is not infringing on Oracle's copyright. Both parties agreed that Google's remade method implementations are not infringing on Oracle's copyright. Google's Android has its own, JCL-independent API which has 168 packages. The crux of the first phase is whether Google has had the right to remake the declaring code as well as the SSO of 37 of 166 Java API packages. The judge established that after comparing the relevant 37 Java API and Android API packages, only 3% of the source code is identical. Those identical lines of source code are method declarations and class declarations. That 3% of the source code *"were freely replicable under the merger doctrine and names doctrines"*.[195] Apart from the implementing code, Google also remade the definitions and parameters of those 37 packages.

"*An intellectual property term used in reference to copyright to exclude from copyright and prevent alleged holders of copyright from attempting to assert exclusive ownership of elements of a work that are standard, stock or common either in general or in relation to a particular topic*." According to Duhaime.org, available at: https://www.duhaime.org/Legal-Dictionary/Term/ScenesAFaire, last accessed December 15, 2021

[191] District court, *Oracle America, Inc. v. Google Inc*., May 31, 2012, p. 36

[192] According to the Cambridge Dictionary, taxonomy is "*[a] system for naming and organizing things, especially plants and animals, into groups that share similar qualities*." Available at: https://dictionary.cambridge.org/dictionary/english/taxonomy, last accessed December 15, 2021
Taxonomy is copyrightable under United States Court of Appeals for the Seventh Circuit, *American Dental Association v. Delta Dental Plans Association*, 1997.

[193] District court, *Oracle America, Inc. v. Google Inc*., May 31, 2012, p. 38, line 12-13

[194] *Ibid.*, p. 6, line 15; see also p. 38 line 6 - 13

[195] *Ibid.*, p.40, line 25-26

The judge emphasised that *"the rules of Java dictate the precise form of certain necessary lines of code called declarations, whose precise and necessary form explains why Android and Java must be identical when it comes to those particular lines of code."*[196] More importantly, *"code written for one API would not run on an API organized differently, for the name structure itself dictates the precise form of command to call up any given method."*[197] The judge established that there is no specific law which regulates the legal question at hand, thus the verdict relied on general principles of copyright law that are found in the relevant decisions of the Congress of the United States ("Congress"), the SCOTUS and the United States Court of Appeals for the Ninth Circuit's ("the Ninth Circuit"). Judge Alsup deduced that there is only one way to write the declaring code in those 37 Java API packages and therefore the *"merger doctrine bars anyone from claiming exclusive copyright ownership of that expression."*[198] Judge Alsup concluded: *"To accept Oracle's claim would be to allow anyone to copyright one version of code to carry out a system of commands and thereby bar all others from writing their own different versions to carry out all or part of the same commands."*[199]

The jury decided as follows: The jury found that Google infringed Oracle's copyright regarding Google's implementation of Oracle's 37 packages of the Java SE Class Library, both the source code and SSO. Nevertheless, judge Alsup had overruled the jury's decision, establishing that a jury could not have ruled on that question as the API at issue is not copyrightable, and therefore, not protected under the Copyright Act. As a result, the judge concluded that Oracle's copyright could not have been infringed.

Regarding the documentation of 37 Java API packages, the jury found no copyright infringement.

The related question of whether Google's use of the 37 Java API packages is protected by the doctrine of fair use remained undecided by the jury.

Regarding Oracle's rangeCheck function, the jury found, and the judge concurred with the jury, that Google infringed on Oracle's copyright regarding the nine lines of rangeCheck source code by copying both their declaring and implementing code verbatim.

[196] *Ibid.*, p. 7, line 16-19
[197] *Ibid.*, p. 12, line 19 - 24
[198] *Ibid.*, p. 35, line 23-25
[199] *Ibid.*, p. 41, line 1-4

Regarding the eight decompiled Java security files Google had also copied verbatim, the jury found no copyright infringement as they were used as test files but never incorporated into the Android source code.

6.2.1.2. Decisions made in the patent and damages trial phase

The second phase revolved around the patents and their infringement. By the time jury made its decision, Oracle continued to pursue only 2 out of the original 7 patent infringement claims, the '520 patent and the '104 patent.[200] The jury found no infringement on all patent claims.

In the third phase concerning related damages for the rangeCheck and security files copyright issue, the judge awarded Oracle the statutory damages relief of US$150,000. The parties agreed on $0 in statutory damages.

6.2.2. The first Court of Appeals for the Federal Circuit ruling

On the 9th of May 2014, the United States Court of Appeals for the Federal Circuit ("CAFC") ruled that both the declaring code and SSO of the 37 Java API packages do enjoy protection under the Copyright Act and are thus copyrightable.[201] The CAFC reversed the District court's copyrightability decision with instructions to reinstate the jury's copyright infringement finding regarding the 37 Java API packages. As the fair use question remained undecided by the jury, the CAFC could not find sufficient facts in the District court's ruling to decide on that point. It therefore remanded the fair use question for a second trial to the District court. The CAFC also ruled in favour of Oracle regarding Google's literal copying of the Java declaring code and the rangeCheck code. Patent claims were not at issue in the appeal.

On the issue of copyrightability, the CAFC emphasised that under the Copyright Act computer programs are subject to copyright protection as literary works. They must be original to qualify for copyright protection. The CAFC used the originality threshold as defined in the Feist case: *"Original, as the term is used in copyright, means only that the work was independently created by the author (as opposed to copied from other works), and that it*

[200] "*Five of the seven patents were dropped from the case, as one by one they were rejected upon reexamination by the U.S. Patent and Trademark Office (USPTO).*" Brodkin, J.: *Newly revived patent gives Oracle extra ammunition in Google trial*, available at: https://arstechnica.com/tech-policy/2012/04/newly-revived-patent-gives-oracle-extra-ammunition-in-google-trial/, last accessed December 15, 2021

[201] CAFC, *Oracle America, Inc. v. Google, Inc.*, May 9, 2014, available at: https://law.justia.com/cases/federal/appellate-courts/cafc/13-1021/13-1021-2014-05-09.html, last accessed December 15, 2021

possesses at least some minimal degree of creativity."[202] Therefore, Oracle's 37 API packages have met the originality requirement under §102(a) of the Copyright Act.

The CAFC continued to explain that a computer program consists of its literal and non-literal elements. The literal elements are its source code and object code, whereas the non-literal elements include its SSO and user interface. Oracle claims copyright protection to the literal elements of its API packages, the 7000 lines of declaring source code, and to the non-literal elements, the SSO of each of the 37 Java API packages. The literal elements always enjoy copyright protection, whereas the non-literal elements can enjoy copyright protection, depending on *"whether, on the particular facts of each case, the component in question qualifies as an expression of an idea, or an idea itself"*.[203] It was established that the originality requirements are met for both the declaring code and the SSO.

Literal copying is word-for-word copying, whereas non-literal copying is paraphrasing or loosely paraphrasing rather than copying verbatim. It was established that Google copied the declaring code literally. Oracle claimed that, rather than copying the entire SSO literally, Google proceeded to paraphrase the rest of the SSO by writing its own implementing code.

The District court agreed with Google's *"two-step copyrightability analysis"* argument, meaning that even though under §102(a) Oracle's declaring code and SSO are original and therefore would enjoy copyright protection, under §102(b) they are not as they have a *"functional component"*.[204] The CAFC states that *"the problem with the District Court's approach is that computer programs are by definition functional—they are all designed to accomplish some task"*[205] and that the District court failed to apply the abstraction-filtration-comparison test to determine the copyrightability, yet had established that the merger doctrine is applicable to the Java declaring code and SSO, rendering them uncopyrightable. The CAFC emphasises that this test *"rejects the notion that anything that performs a function is necessarily uncopyrightable."*[206]

The CAFC established that the merger doctrine is a tool in determining *"whether a copyright infringement has occurred, rather than whether a copyright is valid"*[207] and that the

---

[202] *Ibid*, p. 18, quoting SCOTUS, *Feist Publications, Inc., Petitioner v. Rural Telephone Service Company, Inc*., March 27, 1991, available at: https://www.law.cornell.edu/supremecourt/text/499/340, last accessed December 15, 2021
[203] *Ibid*., p. 20, quoting the U.S. Court of Appeals, Ninth Circuit, *Johnson Controls v. Phoenix Control Systems*, October 3, 1989, available at: https://casetext.com/case/johnson-controls-v-phoenix-control-systems, last accessed December 15, 2021
[204] *Ibid*., p. 22
[205] *Ibid*., p. 42
[206] *Ibid*., p. 23
[207] *Ibid*., p. 25, quoting the U.S. Court of Appeals, Second Circuit, *Kregos v. Associated Press*, 1993

District court misapplied it. It recognised the Atari case criteria for the merger doctrine: *"unique arrangement of computer program expression (...) does not merge with the process so long as alternate expressions are available"*.[208] The CAFC stated that the evidence showed that Oracle had an infinite number of ways to organise and name the 7000 lines of declaring code and that therefore there is no merger. The District court also misapplied the merger doctrine when analysing the options available to Google at the time of copying, instead of those available to Oracle at the time of the code's fixation.

While discussing Oracle's appeal against the District court's ruling, the CAFC reiterated that *"except to the limited extent (...) regarding three of the API packages, it is (...) undisputed that Google could have written its own API packages using the Java language"*.[209] Because the authors of Java, while complying with the rules of the Java language, did not have only a limited number of ways to express the declaring code of the 37 Java API packages, the merger doctrine does not apply. This logic extends to the three core packages, as Google did not argue or prove that Sun had only a limited number of ways to express the declaring code of these "core" packages. Therefore, the District court was wrong to *a priori* exclude the three "core" API packages from the scope of copyright protection.

The CAFC agreed that *"short phrases such as names, titles and slogans"* are not subject to copyright protection, however, in its analysis the District court failed to recognise the context in which those short phrases appear, which does make them copyrightable. Namely, literary works often contain short phrases, but *"not all short phrases will automatically be deemed uncopyrightable"*.[210] It is wrong to analyse those short phrases individually, out of context; it is essential to recognise that *"an original combination of elements can be copyrightable"*.[211] The CAFC used the opening of Charles Dickens' *A Tale of Two Cities* as an analogy, as it is also a *"string of short phrases"*, yet nobody could claim that it is uncopyrightable because it could be *"broken into those shorter constituent components"*.[212]

---

[208] *Ibid.*, p. 28, quoting the U.S. Court of Appeals, Ninth Circuit, *Atari Games Corp. v. Nintendo of America*, March1990, available at: https://casetext.com/case/atari-games-corp-v-nintendo-of-america, last accessed December 15, 2021

[209] *Ibid.*, p. 15

[210] *Ibid.*, p. 34, quoting the U.S. Court of Appeals for the First Circuit, *Soc'y of Holy Transfiguration Monastery, Inc. v. Gregory*, 2012

[211] *Ibid.*, here CAFC additionally quoted SCOTUS, *Feist Publications, Inc., v. Rural Telephone Service*, 1991: "*The Court made quite clear that a compilation of non-protectible elements can enjoy copyright protection even though its constituent elements do not*."

[212] *Ibid*, p. 34

Regarding Google's cross-appeal, the CAFC rejects the defence that the literal copying of the rangeCheck code and the decompiled security files was *de minimis*, for it was quantitatively significant and constituted a *"significant use"*.[213]

6.3. The second phase: Fair use of the Java API

The second phase includes proceedings before two courts: the District court and the CAFC. Following the remand of the CAFC, the repeated first instance court proceedings lasted throughout May 2016 until the jury verdict was reached in favour of Google. The second instance court proceedings lasted from October 2016, when Oracle's appeal was filed, until a verdict was reached in favour of Oracle in March 2018.

6.3.1. The second District Court for the Northern District of California ruling

Per the CAFC's order, the fair use question was remanded for a second jury trial to the District court because the fair use question in the first District court trial had remained undecided by the jury, and also due to the fact that the CAFC had ruled the Java API declaring code and SSO to be copyrightable and enjoy copyright protection.

After hearing the arguments of both Google and Oracle, it was entirely on the jury to unanimously answer the following question: *"Has Google shown by a preponderance of the evidence that its use in Android of the declaring lines of code and their structure, sequence and organisation from Java 2 Standard Edition Version 1.4 and Java 2 Standard Edition Version 5.0 constitutes a 'fair use' under the Copyright Act?"*[214]

On the 26th of May 2016, the jury unanimously answered *"Yes (finding for Google)"*.

Oracle filed a motion to the District court to disregard the jury's verdict and to enter a judgement as a matter of law ("JMOL").[215] Therein, Oracle claimed that no reasonable jury could have found in favour of Google, as there was not enough supporting evidence based on

---

[213] *Ibid*, p. 66

[214] Jury Verdict - Document #1982, District court, *Oracle America, Inc. v. Google Inc., Special verdict form [jury verdict]*, May 26, 2016, available at: https://www.courtlistener.com/docket/4177532/1982/oracle-america-inc-v-google-inc/, last accessed December 15, 2021

[215] Legal Information Institute defines JMOL as "[a] *motion asking the court to enter judgment as a matter of law. This motion is made before a case is submitted to the jury and argues that no reasonable jury could find for the opposing party (i.e., whatever evidence exists for such ruling is legally insufficient). In federal court, this motion may be renewed after an adverse jury finding*." For more information on JMOL, visit Legal Information Institute, available at: https://www.law.cornell.edu/wex/motion_for_judgment_as_a_matter_of_law, last accessed December 15, 2021

which such a finding could have reasonably been made. The District court denied Oracle's motion for JMOL and entered a final judgement in favour of Google.[216]

The District court explained it *"supports Google's contention that the Java API library is simply an extension of the programming language itself."*[217] It continued to explain that both Android and Java *"presupposed the Java programming language in the first place"*[218] and that therefore, a jury reasonably could have found that it *"was better for both to share the same SSO insofar as they offered the same functionalities, thus maintaining usage consistency across systems and avoiding cross-system confusion, just as all typewriter keyboards should use the QWERTY layout."*[219]

While discussing the four statutory fair use factors, the District court found that a reasonable jury could have concluded that:

a) as to the purpose and character of the use, i.e., the first factor, Google's use was sufficiently transformative because, although commercial, Google had selected and used only certain elements of the Java API packages, added its own implementing code *"adapted to the constrained operating environment of mobile smartphone devices with small batteries"*,[220] wrote new API packages, classes and methods specifically for its mobile smartphone platform and by doing so, gave new expression or meaning to the copyrighted Java API code, which was originally designed by Sun for desktop and laptop computers;

b) as to the nature of the copyrighted work, i.e., the second factor, even though the Java declaring code and SSO were creative, and thus may enjoy copyright protection, they were *"not highly creative"* and *"functional considerations predominated in their design"*[221];

c) as to the amount and substantiality of the portion used, i.e., the third factor, Google copied *"only so much as was reasonably necessary for a transformative use"*[222] and the copying was minimal; and

d) as to the market harm, i.e., the fourth factor, *"use of the declaring lines of code (including their SSO) in Android caused no harm to the market for the copyrighted works, which were for desktop and laptop computers"*.[223]

---

[216] District court, *Oracle America, Inc. v. Google Inc., Order Denying Rule 50 Motions*, June 8, 2016, available at: https://www.leagle.com/decision/infdco20160609998, last accessed December 15, 2021
[217] *Ibid.*
[218] *Ibid.*
[219] *Ibid.*
[220] *Ibid.*
[221] *Ibid.*
[222] *Ibid.*
[223] *Ibid.*

6.3.2. The second Court of Appeals for the Federal Circuit ruling

Oracle appealed to the CAFC after filing unsuccessful motions for JMOL to the District court. The CAFC considered the appeal and held a second trial on the fair use question.

"Specifically, [Oracle] submits that:

(1) the purpose and character of Google's use was purely for commercial purposes;

(2) the nature of Oracle's work is highly creative;

(3) Google copied 11,330 more lines of code than necessary to write in a Java language-based program; and

(4) Oracle's customers stopped licensing Java SE and switched to Android because Google provided free access to it."[224]

On the 27th of March 2018, the CAFC ruled that Google's verbatim copying and use of both the Java declaring code and SSO is not fair use as a matter of law. The CAFC reversed the District court's ruling, which denies the Oracle's JMOL, and remanded for a trial on damages.

In brief, the CAFC explained its standards of review by citing relevant case law and pointing out the complexity of the procedural aspects of adjudicating mixed questions of fact and law, as it is in this case. *"The doctrine of fair use has long been considered 'the most troublesome in the whole law of copyright.'"*[225] While it is a stance of the Ninth Circuit that a jury may decide on copyrightability questions of fair use, this stance has not been elaborated in depth. It has, however, been clarified that the jury's role is limited to determining disputed historical facts, and not to drawing conclusions from them. As fair use is equitable in nature, and the question of fair use was in this case primarily a question of law, it would have seemed natural for a judge and not a jury to decide, despite there being disputes about the historical facts regarding its application.

Nonetheless, all aspects of Google's fair use defence went to the jury without any party arguing against it. Given the brevity of the content of the jury verdict, the CAFC had to assume that the jury had resolved all factual issues relating to the historical facts in favour of the verdict. Issues of fair use may usually, with regards to standards of review, be broken down into three parts:

[224] CAFC, *Oracle America, Inc. v. Google, Inc.*, March 27, 2018, p. 17, available at: https://law.justia.com/cases/federal/appellate-courts/cafc/17-1118/17-1118-2018-03-27.html, last accessed December 17, 2021

[225] *Ibid.*, p. 17, quoting the U.S. Court of Appeals, Ninth Circuit, *Monge v. Maya Magazines, Inc.*, 2012

a) whether the correct legal standards were applied, which is reviewed *de novo*,[226]

b) whether the findings of relevant historical facts were correct, which is reviewed with deference, and

c) whether the use was ultimately fair, which is reviewed *de novo*.

Taking the circumstances of this particular case into account, and the manner in which the jury's decision has come about, the CAFC opted to work on the assumption that it was not an error to send the question to the jury, but given the nature of the decision, i.e. deciding whether the use was fair, and not merely on historical facts, all legal implications which were to be drawn from the historical facts, as determined by the jury in their fair use finding, had to be reviewed *de novo*. So, in this case, given that the jury had been tasked with deciding on issues beyond its established competency, the CAFC was effectively forced to apply a stricter than usual standard of review to the jury decision, granting the District court less deference.

The CAFC has found that only four historical facts remained at issue, *"whether the use was commercial in nature"*, *"whether Google acted in bad faith in copying the work"*, *"whether there are functional aspects to the copyrighted work that make it less deserving of protection"* and *"whether there was harm to the actual or potential markets for the copyrighted work"*.[227]

As the first factor, the CAFC considered the purpose and character of the use. Commercial use is generally taken not to fall into the category of fair use and the court has found that Google's decision to make Android available as an open-source platform does not constitute a non-commercial use. This is because the commerciality of a product does not depend on how exactly Google earns money through Android, in this case through advertisements and not Android itself, achieving this goal by providing for free an otherwise paid-for platform. The use was therefore found to be overwhelmingly commercial and therefore weighs against a finding of fair use.

The CAFC then considered whether the use was transformative and found that it:

a) neither served to criticise, comment nor report on news and the like, as it did not fit the Copyright Act;

---

[226] According to Legal Information Institute: "*When a court hears a case de novo, it is deciding the issues without reference to any legal conclusion or assumption made by the previous court to hear the case. An appellate court hearing a case de novo may refer to the lower court's record to determine the facts, but will rule on the evidence and matters of law without deferring to that court's findings. (…) De novo review occurs when a court decides an issue without deference to a previous court's decision*." For more information visit Legal Information Institute, available at: https://www.law.cornell.edu/wex/de_novo, last accessed December 15, 2021

[227] CAFC, *Oracle America, Inc. v. Google, Inc.*, March 27, 2018, p. 27

b) had no different purpose when comparing the purpose of the API packages in Java and Android, as it clearly served the same functions in both works. The volume of copied content is to be considered together with the quality and the importance of the copied content. *"To hold otherwise would mean that verbatim copying could qualify as fair use as long as the plagiarist stops short of taking the entire work"*[228];

c) contained no alteration to the expressive content or message of the copyrighted material, as the new implementing code added by Google did not change the expression or message of the declaring code, adding that *"no plagiarist can excuse the wrong by showing how much of his work he did not pirate."*[229] Furthermore, the CAFC gave the example that Google's use could have been transformative if it had copied the API for an educational purpose; and

d) finally, the CAFC found that smartphones were a not new context, as the case file clearly demonstrated the presence of the Java API in early smartphones (e.g., Danger and Nokia), even before Android entered the smartphone market and elaborated that *"moving material to a new context is not transformative in and of itself"*.[230]

After concluding that the use of Java API was not transformative, the CAFC examined whether Google had acted in bad faith, as *"one who acts in bad faith should be barred from invoking the equitable defence of fair use"*.[231] The CAFC pointed out that *"a copyist's good faith cannot weigh in favour of fair use"*,[232] but his bad faith is relevant in finding against the existence of fair use. The CAFC concludes that whether or not Google acted in bad faith when using Oracle's code, even after having unsuccessfully attempted to negotiate a licence, had little weight when considering the *"highly commercial and non-transformative nature of the use"*[233] which already strongly weighed against finding the use fair.

As the second factor, the CAFC considered the nature of the copyrighted work, explaining that a creative work enjoys more copyright protection when compared to a functional or informational work. Considering the issue of whether the API packages were creative or functional in their nature, the CAFC found that this particular code was more informational than creative and thus found the use fair. The CAFC quotes Dr. Seuss v. Penguin Books, stating that *"this second factor 'typically has not been terribly significant in the overall*

[228] *Ibid*., p. 35
[229] *Ibid*., p. 36, quoting SCOTUS, *Harper & Row v. Nation Enterprises*, 1985, available at: https://supreme.justia.com/cases/federal/us/471/539/, last accessed December 17, 2021
[230] *Ibid*., p. 37
[231] *Ibid*., p. 39, referring to the U.S. Court of Appeals, Ninth Circuit, *Fisher v. Dees*, 1986
[232] *Ibid*., p. 41
[233] *Ibid*., p. 42

fair use balancing'",[234] and that even though the use was considered fair by this factor, it had less significance to the overall analysis.

As the third factor the CAFC reviewed the amount and substantiality of the portion used, explaining that while the volume of the copied work was less important than its quality, the *"extent of permissible copying varies with the purpose and character of the use."*[235] The CAFC found that Google's use was aimed at making Android attractive to programmers, by reusing the familiar features of Java. Furthermore, the CAFC pointed out that while the copying of merely 37 API packages would appear to be minimal, Google did in fact not only copy the necessary 170 lines of code but copied 11,330 additional lines which were not necessary to write in the Java language.

The CAFC concluded that *"copying the most famous and well-recognized aspects of a work 'to get attention' or 'to avoid the drudgery in working up something fresh' (…) is at best, neutral in the fair use inquiry, and arguably weighs against such a finding"*,[236] as Google did not *"seek to foster any 'inter-system consistency' between its platform and Oracle's Java platform"* and instead sought *"to capitalize on the fact that software developers were already trained and experienced in using the Java API packages at issue."*[237]

As the fourth factor, the CAFC considered the effect on the potential market. The main idea behind fair use in this context is whether the work materially impairs the marketability of the copied work. To determine this, a court must first determine the extent of market harm caused by the infringer. While once a central question to fair use, in recent times the Ninth Circuit has indicated that market harm may be presumed if the use *"was commercial and not transformative"*.[238] On the other hand, market harm must be proven if the use was non-commercial.[239] The CAFC points out that not only the harm to actual markets is important, but also the harm to the market for potential derivative uses of the copyrighted work, regardless of whether the work is being developed further by the author or is licensed to another person. Furthermore, the CAFC stresses the right of the copyright holder to determine *"'when, whether*

---

[234] *Ibid.*, p. 44, quoting the U.S. Court of Appeals, Ninth Circuit, *Dr. Seuss Enters., L.P. v. Penguin Books USA, Inc.*, 1997, available at: https://casetext.com/case/dr-seuss-enterprises-lp-v-penguin-books-usa-inc, last accessed December 15, 2021

[235] *Ibid.*, p. 45, quoting SCOTUS, *Campbell, aka Skyywalker, et al. v. Acuff-Rose Music, Inc.*, March 1994. Available at: https://www.law.cornell.edu/supremecourt/text/510/569, last accessed December 15, 2021

[236] *Ibid.*, p. 47, quoting SCOTUS, *Campbell, aka Skyywalker, et al. v. Acuff-Rose Music, Inc.*, March 1994

[237] *Ibid.*

[238] *Ibid.*, p. 49, quoting the U.S. Court of Appeals, Ninth Circuit, *Disney Enters., Inc. v. VidAngel, Inc.*, 2017

[239] *Ibid.*, quoting SCOTUS, *Sony Corp. of America v. University City Studios*, 1982: "*[T]he Supreme Court stated that, '[i]f the intended use is for commercial gain, that likelihood [of future harm] may be presumed. But if it is for a noncommercial purpose, the likelihood must be demonstrated.*'"

and in what form to release' the copyrighted work into new markets",[240] to not release it at all and to change his mind.

Regarding the harm to current, actual market potential, the CAFC found that Java SE had been used for years in mobile devices prior to Android's release and that Android directly competed with Java SE in the market for mobile devices. Furthermore, the CAFC found that the effect of Android on *"the market for the copyrighted works paralleled what Sun already expected via its OpenJDK"*[241] and that there is evidence that the release of Android caused Oracle substantial losses when negotiating licences, due to Java having to compete against a free product. For example, Oracle had to give a 97% discount for its Java licence to enter Amazon's Kindle platform.

The fact that Google and Oracle had entered licensing negotiations further provided evidence of market harm as both parties were aware of the developing mobile segment of the market. The CAFC dismissed Google's arguments that Oracle was not a device maker and had not yet built its own smartphone platform, as when considering potential markets, the option to issue licences to others to develop derivative works must be taken into account. Therefore, the CAFC weighed the fourth factor in favour of Oracle and concluded that *"'unrestricted and widespread conduct of the sort engaged in by' Google would result in 'a substantially adverse impact on the potential market for the original' and its derivatives."*[242]

Finally, the CAFC balanced the four factors together *"in light of the purposes of copyright"*[243] and concluded that Google's commercial exploitation of Oracle's work would not advance the purposes of copyright. Even though Google had the option to license Oracle's work, Google chose to copy it while presenting this conduct as promoting creative expression and innovation. The CAFC found that Google had thereby prevented Oracle from participating in developing markets, and acting inherently unfairly when using copyrighted work verbatim, for the same purpose and function as the original, in a competing platform. The CAFC found that factors one and four weighed against fair use, factor three was neutral at best, and that factor two weighed in favour of fair use, concluding that Google's *"copying and use of this particular code was not fair as a matter of law."*[244]

[240] *Ibid.*, p. 50, quoting the U.S. Court of Appeals, Ninth Circuit, *Monge v. Maya Magazines, Inc*., 2012
[241] *Ibid*., p. 51, referring to District court, *Oracle America, Inc. v. Google Inc., Order Denying Rule 50 Motions*, June 8, 2016
[242] *Ibid.*, p. 53, quoting SCOTUS, *Campbell, aka Skyywalker, et al. v. Acuff-Rose Music, Inc*., March 1994
[243] *Ibid.*, quoting SCOTUS, *Campbell, aka Skyywalker, et al. v. Acuff-Rose Music, Inc*., March 1994
[244] *Ibid.*, p. 54

6.4. The third phase: The Supreme Court of the United States ruling

The final instance court proceedings lasted from January 2019, when Google's petition for a writ of certiorari was filed to the SCOTUS, until the final verdict was reached in favour of Google, in April 2021.

6.4.1. The Supreme Court of the United States: Majority opinion

On the 5th of April 2021, the SCOTUS ruled[245] in a 6 – 2 majority that Google's use of Oracle's Java declaring code and its SSO of the 37 packages does constitute a fair use as a matter of law, and that all four fair use factors weigh in favour of Google's use as fair. It therefore reversed the CAFC's ruling on fair use and remanded the case for further proceedings in conformity with this ruling.

It addressed Google's allegations regarding the CAFC's violation of the Seventh Amendment's *"right of trial by jury"* which prohibits the re-examination of facts tried and established by a jury. The CAFC was right to decide *de novo* whether the facts established by a jury amount to a fair use, for it is a question of law and not a question of fact.

The SCOTUS decided not to analyse whether the Java API is copyrightable due to *"the rapidly changing technological, economic and business-related circumstances"* and that only the answers necessary to resolve the dispute at hand should be given.[246] Rather, it assumed *"for argument's sake"*[247] that the copied lines of Java API can be copyrighted and focused on the fair use question.

The SCOTUS first delved into the second fair use factor of the fair use analysis, the nature of the copyrighted work. It identified the copied lines of the Java API as a part of a *"user interface"*, the users being programmers in this case, that allows users to access the prewritten source code by using *"simple commands"*, the method calls. [248] In its nature, the declaring code differs from the implementing code i.e., the type of code that gives instructions to a computer which then executes a task. The value of the Java API is tied to the investment made by programmers to learn and master the Java API system. In other words, *"its value lies in its efforts to encourage programmers to learn and to use that system so that they will use (and continue to use) Sun-related implementing programs."*[249] The creativity to write the declaring

[245] SCOTUS, *Google LLC v. Oracle America, Inc*., April 5, 2021, available at: https://www.supremecourt.gov/opinions/20pdf/18-956_d18f.pdf, last accessed December 15, 2021
[246] *Ibid.*, p. 15
[247] *Ibid.*
[248] SCOTUS, *Google LLC v. Oracle America, Inc*., Syllabus, April 5, 2021, p. 2
[249] SCOTUS, *Google LLC v. Oracle America, Inc.*, Majority opinion, April 5, 2021, p. 24

code lies in finding names *"that would prove intuitively easy to remember"* so as to *"attract programmers who would learn the system, help to develop it further, and prove reluctant to use another."*[250] On the other hand, the creativity needed to write the implementing code, specifically for the use in smartphones versus desktop computers, requires a different kind of creativity and resourcefulness in adapting code to *"how quickly a computer can execute a task or the likely size of the computer's memory"*, for example. [251]

The copied lines of the declaring code are as functional in nature as any other computer program, but dissimilarly, the declaring code is not considered as creative because it is bound together with uncopyrightable ideas of *"general task division and organisation"*.[252] The declaring code is also intertwined with the implementing code which, in contrast, represents a more creative, albeit functional expression. In the view of SCOTUS, *"the declaring code is, if copyrightable at all, further than most computer programs (such as the implementing code) from the core of copyright."*[253]

The analysis continued with the first fair use factor, the purpose and character of the use. The SCOTUS considered whether Google's use *"adds something new, with a further purpose or different character"* by altering the copyrighted work at issue *"with new expression, meaning or message"*. [254] The SCOTUS concluded Google's copying constituted a transformative use, for Google only copied the necessary parts of the Java API so as to allow programmers to *"work in a different computing environment without discarding a portion of a familiar programming language"*.[255] Google used parts of the Java API *"in part for the same reason that Sun created those portions, namely, to enable programmers to call up implementing programs that would accomplish particular tasks"*.[256] By doing so, Google sought to create a new product, a product that *"offers programmers a highly creative and innovative tool for a smartphone environment."*[257] The SCOTUS established that Google's use had promoted the *"Progress of Science and useful Arts"*, as written in the U.S. Constitution.[258]

By reimplementing a user interface, i.e., repurposing the same words and syntaxes, the evolution of computer programs can be achieved. Reimplementation is essential for the programmers' ability to use their acquired skills, and the SCOTUS reiterated that Sun was

[250] *Ibid.*, p. 23
[251] *Ibid.*
[252] *Ibid.*, p. 24
[253] *Ibid.*
[254] *Ibid.*
[255] SCOTUS, *Google LLC v. Oracle America, Inc*., Syllabus, April 5, 2021, p. 2
[256] SCOTUS, *Google LLC v. Oracle America, Inc*., Majority opinion, April 5, 2021, p. 25
[257] *Ibid.*
[258] The Constitution of the United States of America, Art. I, §8, Cl. 8

aware of this when it made use of already existing user interfaces while creating the Java platform.[259] In this context, Google's goal was to create a different task-related system for smartphones, which is a different computing environment, and ultimately proceeded to create the Android platform to further that goal and popularise it among programmers.

Regarding the commerciality factor, although Google's use does qualify as a commercial one, the SCOTUS decided it does not weigh against Google due to its transformative use. News reporting is also usually done for commercial profit, but nonetheless qualifies as fair use according to the 17 U.S. Code §107.

As for Google's potential bad faith, the SCOTUS quoted Campbell by saying that *"copyright is not a privilege reserved for the well-behaved"*[260] and concluded that bad faith "*is not determinative in this context*" due to the other factors weighing for the fair use.[261]

Regarding the third fair use factor, the amount and the substantiality of the portion used, the SCOTUS viewed Google's copying of the declaring code as *"one small part of the considerably greater whole"*,[262] constituting only 0.4% of the entire 2.86 million lines of Java API code. According to Harper & Row, even the small amount copied may represent *"'the heart'"*[263] of the copyrighted work's originality and expression, and therefore copying that small amount cannot qualify as a fair use. However, Google did not copy those 11,500 lines for their creativity and expression, but because of their *"user interface"* function that allows programmers to use what they already have mastered in a new environment, that is, smartphones, and *"it would have been difficult, perhaps prohibitively so, to attract programmers to build its Android smartphone system without them"*.[264]

The SCOTUS does not agree with the CAFC's opinion that Google should have copied only 170 lines of code which are essential to the Java language, as Google wanted to enable programmers to apply their already acquired Java API knowledge in Android. The SCOTUS concluded *"the declaring code was the key that it needed to unlock the programmers' creative energies. And it needed those energies to create and to improve its own innovative Android systems."*[265] Both the amount and the substantiality factors weigh in favour of fair use.

---

[259] See *infra* 8., pp. 88-89
[260] SCOTUS, *Google LLC v. Oracle America, Inc*., Majority opinion, April 5, 2021, p. 28, quoting Leval: *Toward a Fair Use Standard*, Harv. L. Rev, 1990
[261] *Ibid*.
[262] SCOTUS, *Google LLC v. Oracle America, Inc*., Syllabus, April 5, 2021, p. 3
[263] SCOTUS, *Google LLC v. Oracle America, Inc*., Majority opinion, April 5, 2021, p. 28, quoting SCOTUS, *Harper & Row v. Nation Enterprises*, 1985
[264] *Ibid*., p. 29
[265] *Ibid*., p. 30

In the fourth fair use factor, the effect of the copying on the market, or the value of the copyrighted work, the SCOTUS did not find the Java SE platform to be a market substitute for Android. If unsolicited, *"making a film of an author's book may (...) mean potential or presumed losses to the copyright owner."*[266] However, *"the public benefits the copying will likely produce"* must be balanced with the losses the copyright owner has suffered.[267]

The SCOTUS concluded that Android caused damage neither to actual nor potential markets for the Java SE platform, as Sun would not have been able to successfully compete in those markets regardless of Google copying the Java API. The SCOTUS reiterated that the evidence showed how Sun was beset by business challenges in developing a mobile phone product, and that *"devices using Google's Android platform were different in kind from those that licensed Sun's technology"*[268] in a way that Sun's technology could be found in *"feature phones"*, some of which lacked touchscreen, while others lacked a QWERTY keyboard.

Moreover, the SCOTUS established that Oracle could even benefit from Google's Java API reimplementation in the different market. The SCOTUS emphasised that a new interface attracts new users due to its *"expressive qualities"*[269] such as improved functionality or its visual characteristics, still, over time it may hold value because users are *"just used to it"*[270] and have learned how to work with it. It concludes that there is *"no reason to believe that the Copyright Act seeks to protect third parties' investment in learning how to operate a created work"*.[271]

The SCOTUS recognised the potential harm to the public, should the enforcement of Oracle's copyright be allowed. The enforcement of Oracle's copyright cannot be allowed as it would limit *"the future creativity of new programs"* due to *"the costs and difficulties of producing alternative APIs with similar appeal to programmers"*.[272]

6.4.2. The Supreme Court of the United States: Dissenting opinion

Justice Thomas, with whom Justice Alito joined, wrote the dissenting opinion which held that Oracle's code is copyrightable and that Google's use was not fair. Justice Thomas emphasised that by doing so, Google earned *"tens of billions of dollars"* while *"erasing"* 97.5%

[266] *Ibid.*
[267] *Ibid.*, p. 31
[268] *Ibid.*, p. 32
[269] *Ibid.*, p. 34
[270] *Ibid.*
[271] *Ibid.*
[272] *Ibid.*

of the Java SE's value during licencing negotiations for Amazon's Kindle platform and ultimately became *"the owner of the largest mobile operating system in the world"*.[273]

On the Java declaring code copyrightability issue, the definition in 17 U.S. Code §101 does cover declaring code, for it contains *"sets of statements that indirectly perform computer functions by triggering prewritten implementing code".*[274] It also satisfies *"the general test for copyrightability"* as it is *"an original work of authorship"* that is *"fixed in (...) tangible medium of expression"* which, according to Feist, *"possesses at least some minimal degree of creativity."*[275]

It does not qualify as an uncopyrightable method of operation merely because it is functional, as the declaring code and the implementing code are inextricably bound together, which was recognised by the SCOTUS majority. Without declaring code, implementing code has no function, and *vice versa* is also true. Without declaring code, every Java program would have to be made *ab ovo*. Justice Thomas argues that by enacting 17 U.S. Code §101, which protects computer code that is used both directly and indirectly to bring about a certain result, the Congress *"rejected any categorical distinction between declaring and implementing code"*,[276] protecting both implementing code that gives orders to the computer directly and declaring code that incorporates implementing code, thus giving its orders indirectly. Justice continues to explain that the term "method of operation" should include functions and ideas that are being implemented, e.g., *"math functions, accounting methods, or the idea of declaring code"*,[277] however not the Java API declaring code *per se*: *"Oracle cannot copyright the idea of using declaring code, but it can copyright the specific expression of that idea found in its library."*[278]

The merger doctrine is inapplicable as there were *"innumerable ways"* for Sun to write the declaring code, although there might have been only one for Google to copy. Justice emphasised that both Apple and Microsoft created their own declaring code.

Justice Thomas opines that the majority created a distinction between declaring and implementing code which *"makes it difficult to imagine any circumstance in which declaring code will remain protected by copyright"*.[279]

---

[273] SCOTUS, *Google LLC v. Oracle America, Inc*., Dissenting opinion, April 5, 2021, p. 1
[274] *Ibid.*, p. 4
[275] *Ibid.*, pp. 4-5
[276] *Ibid.*, p. 6
[277] *Ibid.*, pp. 6-7
[278] *Ibid.*, p. 4
[279] *Ibid.*, pp. 7-8

Regarding the fair use question, Justice Thomas concluded that the majority erred when it made, contrary to the Copyright Act, a distinction between declaring and implementing code, hence establishing that declaring code is not copyrightable, which has resulted in the majority's decision that all fair use factors favour Google.

When analysing the nature of copyrighted work factor, the majority created the already mentioned distinction and tied declaring code to uncopyrightable ideas, recognising its value only in the aspect of its use and familiarity to programmers. In the Justice's view, it is actually the implementing code that *"conveys no expression to programmers"*, making it less close to the *"core of copyright"* than the declaring code.[280] Moreover, the implementing code may also be tied to *"the division of computing tasks"* as an idea that cannot be copyrighted. *"We have not discounted a work of authorship simply because it is associated with noncopyrightable ideas."*[281]

Justice Thomas also did not agree that the value of declaring code is tied to the effort programmers had to invest in learning it. As a comparison, Justice Thomas explained that actors and singers learning and rehearsing a Broadway musical script does not give other theatres the right to copy the script merely because it is more efficient than asking them to learn a new one or encouraging them to switch theatres. As the declaring and implementing code are inextricably bound, *"the value of (...) implementing code thus is directly proportional to how much programmers value the associated declaring code"*.[282] Justice Thomas opined that the majority had conflated transformative and derivative use, for, by that logic, a film based on a book could be considered transformative even if made unlicensed by a movie studio.

Regarding the market effects, Justice Thomas established that Google's and Oracle's business models differ; whereas Oracle was charging device manufacturers to use the Java platform, Google earned revenue by selling advertisements, while releasing Android to manufacturers for free, making the manufacturers stop paying to embed the Java platform. Furthermore, Justice Thomas established that before Android, *"nearly every mobile phone on the market contained the Java platform"*[283] which proves the value of Java SE to smartphone developers at the time, including Google. After Android was released, Oracle's ability to license Java SE disappeared.

[280] *Ibid.*, p. 10
[281] *Ibid.*
[282] *Ibid.*, pp. 10-11
[283] *Ibid.*, p. 12

Justice Thomas does not share the majority's worries that Oracle enforcing its copyright could harm the public and create a lock-in effect as, at the time this opinion was released, only 7.7% of active Android devices still ran the versions of Android which had been discussed during the proceedings. The power of Oracle's lock-in is again disputed as both Apple and Microsoft have made their own mobile operating systems without using the Java API. However, potential monopolisation may arise on Google's part as it controls the most popular mobile operating system worldwide. *"And if companies may now freely copy libraries of declaring code whenever it is more convenient than writing their own, others will likely hesitate to spend the resources Oracle did to create intuitive, well-organized libraries that attract programmers and could compete with Android."*[284]

Regarding the purpose and character of the use, Justice Thomas concluded the use is overwhelmingly commercial. As to its transformativeness, the examples written in 17 U.S. Code §107, although not exhaustive, are illustrative, and there is no resemblance when comparing them to Google's use of the Java API. Google did not use the Java API to *"teach or reverse engineer to ensure compatibility"*,[285] it used the declaring code for the same purpose and in the same way Oracle had. Justice Thomas interprets the majority has changed the definition of transformative in the computer code context by saying transformative use is the kind of use that helps *"create new products"* and he opines that the *"new definition eviscerates copyright"*.[286]

Justice Thomas concludes that *"a work that simply serves the same purpose in a new context (…) is derivative, not transformative"* and therefore, according to 17 U.S. Code §106(2), Oracle has *"the exclusive rights (…) to prepare derivative works"*.[287]

Regarding the amount and substantiality of the portion used, Justice Thomas established Google did copy verbatim the heart of the Java API as a copyrighted work, specifically, the declaring code, as that part of the Java API is why programmers wanted to use the Java platform and, in the end, why Google copied it. Justice Thomas concludes that the copying was both qualitatively and quantitatively substantial, for copying verbatim the heart of Java API *"made Android a 'market substitute' for 'potentially licensed derivatives' of Oracle's Java platform"*.[288]

---

[284] *Ibid.*, p. 14
[285] *Ibid.*, p. 16
[286] *Ibid.*, pp. 16-17
[287] *Ibid.*, p. 17
[288] *Ibid.*, p. 18, quoting SCOTUS, *Campbell, aka Skyywalker, et al. v. Acuff-Rose Music, Inc.*, March 1994

7. European perspective

In today's world that persistently grows ever more interconnected, the importance of being compliant with national laws coincides with the growth of software companies that are offering their products and services worldwide. When it comes to the copyright protection of computer programs, the EU law and the U.S. law are rather similar, however, their differences should not be overlooked as when applied to the same facts of a case, they might lead to diametrically opposite results that in one legal system grant copyrights while in the other render a work uncopyrightable.

It is only a matter of time until the EU is also faced with the questions arising from the Google v. Oracle case. As of now, the Court of Justice of the European Union ("CJEU") had no opportunity to adjudicate a similar case. What answers might the CJEU give to those questions may only be deduced from analysing the EU *acquis communautaire* ("EU *acquis*").[289]

7.1. Comparison of the relevant EU and U.S. copyright concepts

Both the EU and the U.S. are WTO and WIPO members, thus the Berne Convention, the TRIPS Agreement and the WIPO Copyright Treaty all find application in their legislation. However, the U.S. and the EU copyright law have been developing independently long before the existence of these international treaties. The U.S. copyright legislation is based on common law principles foreign to EU copyright law and therefore, to discuss the facts of the Google v. Oracle case and the arguments on which the U.S. courts based their rulings from the European perspective, the most relevant differences between these two systems shall be presented in short.

The EU copyright law has developed in the European continental legal tradition where it is by and large referred to as author's rights (French: *Droit d'auteur*, German: *Urheberrecht*) as it emphasises the recognition due to the author. In the EU author's rights system, there are two equally important components to author's rights: the moral rights and the economic rights of the author. The aim of moral rights is to protect the author's personal and immaterial ties to his work. They are therefore connected to the author as a natural person and are untransferable, with the exception of being inheritable. The aim of the economic rights is to protect the author's

[289] The definition of the EU *acquis communautaire* as per EUR-Lex: "*The EU's 'acquis' is the body of common rights and obligations that are binding on all EU countries (...) [that] comprises: (...) legislation adopted in application of the treaties and the case law of the Court of Justice of the EU (...) [and] international agreements concluded by the EU (...).*" Available at: https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=LEGISSUM:acquis, last accessed April 30, 2022

economic interests with regard to his work. These may be transferred from the author, partially or in their entirety, usually as a means to profit from his work.

The U.S. copyright system puts emphasis on the author's economic rights, especially the right to copy or otherwise distribute the work, hence author's rights in the common law system are named copyrights. As the U.S. never fully transposed the Berne Convention with regards to moral rights, these seem to be marginalised, while the author's economic rights seem predominant in this common law copyright system.[290] Moral rights have traditionally been protected through *"judicial interpretation of several copyright, trademark, privacy, and defamation statues, and through 17 U.S.C. §106A, known as the Visual Artists Rights Act (...)."*[291] The Visual Artists Rights Act explicitly granted two moral rights only to authors of visual works: the moral right of attribution which grants the right to claim authorship, and the moral right of integrity, that is, the right to *"object to any distortion, mutilation, or other modification of (...) the work, which would be prejudicial to his honor or reputation"*.[292]

In the U.S. copyright system, the author may not only be a natural person, but also a legal person e.g., a production company. Article 2(1) of the EU Computer Programs Directive[293] stipulates that only the natural person may be considered an author of a computer program except when the legislation of the Member State recognises that the legal person may also be considered an author of a computer program.[294]

In the EU, author's rights last from the moment a work's creation for the lifetime of the author plus 70 years after the author's death. Similarly, in the U.S. copyright system, a work of authorship is protected from the moment it is fixed in a tangible medium of expression for the lifetime of the author plus 70 years after the author's death. An exception is made for works for hire and anonymous and pseudonymous works for which the copyright duration is 95 years from the first publication or 120 years from creation, whichever is shorter.[295] Unlike in the EU,

---

[290] *"The U.S. Supreme Court has (...) articulated that the underlying goals of copyright are economic."* Hayes, B. S.: *Integrating Moral Rights into U.S. Law and the Problem of the Works for Hire Doctrine*, Ohio State Law Journal, Vol. 61, Number 2, 2000, p. 1030

[291] Rosenblatt, B.: *Moral Rights Basics*, Harvard Law School, available at: https://cyber.harvard.edu/property/library/moralprimer.html#:~:text=In%20the%20United%20States%2C%20the,of%20who%20owns%20the%20work, last accessed on April 30, 2022

[292] Article 6 bis., the Berne Convention for the Protection of Literary and Artistic Works, Paris Act, 1971

[293] Directive 2009/24/EC of the European Parliament and of the Council of 23 April 2009 on the legal protection of computer programs

[294] Article 2(1) of the Computer Programs Directive: *"The author of a computer program shall be the natural person or group of natural persons who has created the program or, where the legislation of the Member State permits, the legal person designated as the rightholder by that legislation."*

[295] For more information on duration of copyright in the U.S. see The United States Copyright Office: *Duration of Copyright*, Circular 15A, available at: https://www.copyright.gov/circs/circ15a.pdf, last accessed April 30, 2022

in the U.S. copyright owners are required to register their work with the United States Copyright Office before filing a copyright infringement suit in a U.S. court.

Both in the EU and the U.S., a work has to meet the originality criteria, also referred to as the threshold of originality, so that it may enjoy copyright protection. Firstly, the threshold of originality in the EU requires a work to be the *"author's own intellectual creation"*[296] and likewise in the U.S. it has to be an *"original work of authorship"*.[297] Furthermore, a work also needs to be creative to meet the threshold, however, the EU and the U.S. copyright systems' criterion for creativity is slightly different. In the EU, if a work is *"dictated by technical considerations, rules or constraints which leave no room for creative freedom"*[298] it does not meet the threshold of originality, while in the U.S., a work needs to *"display some minimal level of creativity"* to be deemed original.[299] From this it would follow that the EU criterion for creativity is of a binary nature; a work is either creative or not, it is either dictated by technical considerations, rules or constraints or not. On the other hand, the U.S. criterion of originality implies the existence of the gradation of creativity, that a work may reach different levels or degrees of creativity, although the minimal level of creativity is sufficient for a work to be considered original and therefore copyrightable.

The other important difference between the U.S. copyright law and the EU author's rights law is that the author's rights law does not recognise the fair use doctrine. The author's rights law only recognises enumerated exceptions and limitations to copyright in the Information Society Directive[300] that allow for the use of the author's work which would otherwise constitute copyright infringement.

Both systems still have gaps regarding the copyright protection of computer programs. For instance, it should be noted that, unlike the U.S., the EU legislator is yet to lay down a definition of a computer program. On the other hand, the U.S. legislator is yet to regulate what an interface is and the status of a computer program's preparatory design material and how these are protected.

---

[296] CJEU, *Infopaq International A/S v. Danske Dagblades Forening*, see *infra* fn. 303
[297] 17 U.S. Code §102(a)
[298] CJEU, *Football Dataco Ltd and Others v Yahoo! UK Ltd and Others*, C-604/10, 1 March 2021, para 39, available at: https://curia.europa.eu/juris/liste.jsf?&num=C-604/10, last accessed on May 25, 2022
[299] SCOTUS, *Feist Publications, Inc., Petitioner v. Rural Telephone Service Company, Inc*., para 44, see *supra* fn. 202
[300] Directive 2001/29/EC of the European Parliament and of the Council of 22 May 2001 on the harmonisation of certain aspects of copyright and related rights in the information society

7.2. The relevant provisions of the EU *acquis communautaire*

Before discussing questions arising from the Google v. Oracle case, the relevant provisions of "*international agreements concluded by the EU"* and *"legislation adopted in application of the treaties"* as well as *"the case law of the Court of Justice of the EU"* shall be presented. As of now, the EU *acquis* does not regulate software interface technology, so the author applied the provisions of international treaties and EU directives in combination with the CJEU case law to extrapolate the most probable result of the Google v. Oracle case if it had been adjudicated by the CJEU. It should be noted that the EU *acquis* is not clear regarding the questions raised in the case at hand and that the possible outcome of the CJEU preliminary ruling in such a case is merely an opinion of the author of this paper.

The three most important international agreements for the case at hand are: the Berne Convention for the Protection of Literary and Artistic Works ("Berne Convention"), the Agreement on Trade-Related Aspects of Intellectual Property Rights ("TRIPS") and the WIPO Copyright Treaty. The two EU directives applicable to software protection, that are relevant to the Google v. Oracle case are: the Directive 2001/29/EC of the European Parliament and of the Council of 22 May 2001 on the harmonisation of certain aspects of copyright and related rights in the information society ("Information Society Directive") and the Directive 2009/24/EC of the European Parliament and of the Council of 23 April 2009 on the legal protection of computer programs ("Computer Programs Directive").

7.3. Relevant case law of the Court of Justice of the European Union

As already mentioned, there is no equivalent judgement in the EU case law that approximates the factual situation and the legal questions raised in the Google v. Oracle case. However, the author of this paper finds that the CJEU preliminary ruling in Bezpečnostní softwarová asociace - Svaz softwarové ochrany v. Ministerstvo kultury ("BSA")[301] contains important legal reasoning that may indicate how the CJEU might adjudicate in a case similar to the Google v. Oracle case. It provides two tests: a test determining whether a part of a computer program is given copyright protection under the Computer Programs Directive or the Information Society Directive, and a test determining when the idea and the expression of a

[301] CJEU, C-393/09 *Bezpečnostní softwarová asociace – Svaz softwarové ochrany v Ministerstvo kultury*, 22 December 2010, available at: https://curia.europa.eu/juris/document/document.jsf;jsessionid=7B4FF3B6625A7E16FF16F3D02D27476A?text=&docid=83458&pageIndex=0&doclang=EN&mode=lst&dir=&occ=first&part=1&cid=510780, last accessed April 30, 2022

part of a computer program, which is protected under the Information Society Directive, have become indissociable, that is, when they have merged. The other relevant CJEU preliminary rulings that were considered in the analysis of the application of the EU *acquis* to the case at hand are: UsedSoft GmbH v. Oracle International Corp ("UsedSoft"),[302] Infopaq International A/S v. Danske Dagblades Forening ("Infopaq"),[303] and Eva-Maria Painer v. Standard Verlags GmbH and Others ("Painer").[304]

7.4. Applying the EU *acquis communautaire* to the Google v. Oracle case

The research conducted for the purposes of this paper did not result in finding a case in the CJEU case law that had resolved a dispute with a similar subject matter or questions raised in the Google v. Oracle case. Although the CJEU is yet to adjudicate such a case, the author of this paper attempted to subsume the subject matter and related questions under the EU *acquis*. The goal is to consider the questions the CJEU might be faced with and how the CJEU might weigh the key arguments the parties have presented in the Google v. Oracle case.

It should be noted that the arguments the parties put forward in the Google v. Oracle case were tailored to the U.S. legal framework and that the author of this paper attempted to adapt those arguments to the EU legal framework. The hypothetical EU member state's court referral of the matter to the CJEU has been written with this purpose in mind.

To this end, it shall be assumed that the hypothetical EU member state's legislator has duly transposed all relevant directives into its national law. The EU member state being hypothetical, its hypothetical court shall not refer to specific articles of its own national laws that are equivalent to the corresponding articles in the directives but shall rather directly invoke the relevant articles of the directives. The same is assumed for all relevant international treaties.

---

[302] CJEU, C-128/11 *UsedSoft GmbH v. Oracle International Corp*, 3 July 2012, available at: https://curia.europa.eu/juris/document/document.jsf?text=&docid=124564&pageIndex=0&doclang=EN&mode=lst&dir=&occ=first&part=1&cid=512034, last accessed April 30, 2022

[303] CJEU, Case C 5/08 *Infopaq International A/S v. Danske Dagblades Forening*, 16 July 2009, available at: https://eur-lex.europa.eu/legal-content/en/TXT/?uri=CELEX:62008CJ0005, last accessed April 30, 2022

[304] CJEU, C-145/10 *Eva-Maria Painer v. Standard Verlags GmbH and Others*, 1 December 2011, available at: https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX%3A62010CJ0145, last accessed April 30, 2022

7.4.1. The hypothetical EU member state's court referral of the matter to the CJEU

The hypothetical EU member state's court ("the referring court") determined that the main points of contention between the parties in the Google v. Oracle case are the copyrightability of the Java API's declaring code and of the Java API's SSO.

As the Java API is a type of software interface, the referring court applied the definition of the term interface from the 10th recital of the Computer Programs Directive. It stipulates that an interface is a part of a computer program and is given copyright protection under the said Directive. Given that the nature of the Java API as a software interface was not contested by the parties, this understanding was applied *prima facie* to discuss the copyrightability of its individual elements.

Regarding the Java API's SSO, the referring court determined the Java API's SSO consists of the non-literal elements of the Java API that make its overall structure and organisation. The referring court did not find an international agreement or EU law recognising a concept of copyright protection specific to SSO in computer programs or interfaces. Article 9(1) of the TRIPS Agreement and Article 2 of the WIPO Copyright Treaty both state that copyright protection is given to the expression of literary works while it is not afforded to ideas and procedures as such. By virtue of Article 1(1) of the Computer Programs Directive, copyright protection is given to the computer programs as literary works within the meaning of the Berne Convention. Even though Article 1(1) does not explicitly mention interfaces, the referring court has for the purposes of this proceeding concluded that interfaces are also protected by copyright as literary works. By virtue of Article 1(2) of the Computer Programs Directive, ideas and principles which underlie interfaces are not afforded copyright protection under that Directive.

The referring court has decided that the Java API's SSO may not be given copyright protection on the grounds that it equates to an idea of the organisation of an interface and has no literal expression in the program's source or object code. In the same way that other literary works' internal SSO e.g., the organisation of chapters of a book, may not enjoy copyright protection, the Java API's SSO also may not be given copyright protection.

Regarding the Java API's declaring code, the referring court did not find an international agreement or EU law directly stipulating matters related to software interface technology or to the library API as a type of software interface technology. Therefore, there are no provisions that are directly applicable to the Java API and its components that would

ultimately indicate under which law they might be given copyright protection. After the referring court found there is no directly applicable law to the subject matter i.e., the Java API's declaring code, it proceeded to interpret the current positive law related to the subject matter to resolve the dispute at hand. The referring court was unable to determine without a doubt how the international agreements and the EU positive law should be interpreted and applied to the subject matter. Namely, there is no definition of a computer program in the EU *acquis*, only the definition of an interface. However, it should be noted that not all types of interfaces enjoy protection under the Computer Programs Directive, as was determined in BSA. The nature of the Java API's declaring code proved to be crucial for the application of the appropriate Directive as the threshold of originality criteria are different for works protected under the Information Society Directive and the Computer Programs Directive.

To ascertain the nature of the Java API's declaring code and hence which EU directive would be applicable to the subject matter, the referring court decided to stay the proceedings and to refer questions to the Court of Justice for a preliminary ruling:

1. Which directive is applicable to the subject matter, the Information Society Directive or the Computer Programs Directive?

The referring court has *prima facie* determined that, in the light of the 10th recital of the Computer Programs Directive, the Java API is an interface and therefore considered a *"part (...) of the program which provide[s] for [an] (...) interconnection and interaction between elements of software and hardware"*. From this reasoning, it would follow that if an interface were afforded copyright protection under that Directive, the same should be afforded to its components. In this particular case, the referring court found that the Java API is a type of software interface. Therefore, by *argumentum a maiore ad minus*, both statements are also applicable to the Java API's declaring code ("declaring code") as it is a component of the Java API.

However, according to BSA, not all types of interfaces may be afforded protection under the Computer Programs Directive. In particular, the CJEU found that a graphics user interface ("GUI") as a type of interface constitutes an exception to the rule and is not protected under the Computer Programs Directive, but rather under the Information Society Directive. The CJEU determined that the GUI *"does not enable the reproduction of (...) [a] computer program, but merely constitutes one element of [a] program by means of which users make use*

of the features of that program [which means] (…) that that interface does not constitute a form of expression of a computer program within the meaning of Article 1(2) of [the Computer Programs Directive] and that, consequently, it cannot be protected specifically by copyright in computer programs by virtue of that directive".[305]

The referring court opines that the CJEU has thus presented a test for determining the nature of an interface in order to ascertain under which of the two directives an interface ought to be given copyright protection. The referring court applied the test to the declaring code as its nature is the point of contention. It is the understanding of the referring court that the declaring code is the main component enabling the interconnection between the Java API's implementing code and an independently written program using that implementing code. However, when observing the declaring code isolated from the rest of the Java API, that is, disconnected from the library and the implementing code, it is clear that the declaring code would not enable the execution of a program on its own. The referring court opines that if one were to replace every instance of the declaring code with the implementing code it refers to, the program would be successfully executed. After subjecting the declaring code's characteristics to the test, the referring court could not come to an unambiguous conclusion regarding the nature of the declaring code and thus which directive is applicable.

Namely, the referring court recognises that different parts of a computer program, or rather of an interface, serve different functions. Some are indispensable to the execution of the program for *"enabl[ing] the reproduction of [a] computer program"*, while others allow the users to *"make use of the features of [a] program"*.[306] The referring court recognises that the declaring code may serve both of these functions simultaneously.

The referring court next pondered the question of who the users of the declaring code are so as to clarify the nature of the declaring code. While a programmer is using the Java API to facilitate the writing process of his program, he is making use of the declaring code to establish the needed interconnections between particular points in a program's source code and the Java API's implementing code. During that process, the nature of the declaring code resembles that of a user interface and a programmer using the declaring code might therefore be qualified as a user of that interface. After a program has been written, during its execution,

[305]CJEU, *Bezpečnostní softwarová asociace - Svaz softwarové ochrany v. Ministerstvo kultury*, 22 December 2010, paras 41 - 42
[306] *Ibid*.

the nature of the declaring code is that of a software interface, enabling the bilateral communication between that program and the Java API's implementing code.

Throughout the Google v. Oracle case, the courts have referred to the programmers as *users* of the Java API.[307] Keeping the distinction between a user interface[308] i.e., GUI, and a software interface[309] i.e., API, in mind, the question arises whether the programmers using the declaring code to write their own programs might be considered *users,* especially from the perspective of Oracle who is in a licensor – licensee relationship with the programmers that are using the Java API. Furthermore, it is unclear whether the term *user* in the context of the Computer Programs Directive and in the context of BSA has the same meaning and whether *end-users* interacting with a user interface (GUI) and *programmers* using a software interface (API) are both to be considered *users*.

From all this, three options arise, either:

a) the Java API's declaring code is a user interface akin to a GUI when used by programmers, and is protected under the Information Society Directive,

b) the Java API's declaring code is a software interface and therefore an element of the program which engenders the reproduction of the program and is protected under the Computer Programs Directive, or

c) the Java API's declaring code is both a software and a user interface, and both directives are applicable.

[307] *"Unlike many other computer programs, the value of the copied lines is in significant part derived from the investment of users (here computer programmers) who have learned the API's system."* SCOTUS, *Google LLC v. Oracle America, Inc.*, Syllabus, April 5, 2021, p. 3

[308] The definition of the term "user interface" as per TechTarget: *"The user interface (UI) is the point of human-computer interaction and communication in a device. This can include display screens, keyboards, a mouse and the appearance of a desktop. It is also the way through which a user interacts with an application or a website."*, available at: https://www.techtarget.com/searchapparchitecture/definition/user-interface-UI, last accessed April 30, 2022

[309] See *supra* fn. 50

2. If the Computer Programs Directive is applicable to the subject matter, will the merger doctrine as elaborated in BSA *mutatis mutandis* be applicable regardless of Article 1(3) of the Computer Programs Directive?

The Information Society Directive is a *lex generalis* in relation to the Computer Programs Directive, that is, the latter Directive is a *lex specialis* in relation to the former. The CJEU confirmed this in UsedSoft.[310] Advocate General Sharpston elaborated her opinion on this matter as follows: *"The Court has (...) stated that Directive 2009/24 constitutes a lex specialis in relation to the provisions of Directive 2001/29. In my view, that statement must be read as meaning that the provisions of Directive 2009/24 take precedence over those of Directive 2001/29, but only where the protected material falls entirely within the scope of the former."*[311]

According to the *lex specialis derogat legi generali* principle, this would mean that after the specific subject matter has been stipulated in a *lex specialis* it cannot be stipulated by the *lex generalis*, or rather, that *the lex generalis* shall not be applicable to the specific subject matter that has been regulated by a *lex specialis*. However, in the related, more general questions to that subject, the *lex generalis* shall be applicable.

Provided that the Java API's declaring code is a form of expression of a computer program that engenders its reproduction and is therefore given protection under the Computer Programs Directive, the question of the interpretation of Article 1(2) and (3) of that Directive arises. Namely, Article 1(2) stipulates that ideas and principles which underlie any element of a program are not protected under that Directive. Article 1(3) stipulates that *"[a] computer program shall be protected if it is original in the sense that it is the author's own intellectual creation"*. The second sentence of the Article 1(3) excludes all other possible criteria which could determine computer programs' eligibility for protection.

---

[310] *"It is apparent from Article 1(2)(a) of Directive 2001/29 that the directive 'leave[s] intact and ... in no way affect[s] existing ... provisions [of European Union law] relating to ... the legal protection of computer programs' conferred by Directive 91/250, which was subsequently codified by Directive 2009/24. The provisions of Directive 2009/24 (...) thus constitute a lex specialis in relation to the provisions of Directive 2001/29 (...)."* CJEU, *UsedSoft GmbH v. Oracle International Corp*, 3 July 2012, para 51

[311] Sharpston, E.: *Opinion of Advocate General Sharpston in Case C-355/12 Nintendo Co. Ltd and Others v. PC Box Srl and 9Net Srl*, 19 March 2013, available at: https://eur-lex.europa.eu/legal-content/GA/ALL/?uri=CELEX:62012CC0355 , last accessed April 30, 2022

The referring court has taken the liberty of applying the teleological method of interpretation of the Computer Programs Directive and the EU *acquis*, taking into account specifically the BSA case.

Firstly, it should be noted that Infopaq[312] expanded the application of the Computer Programs Directive criterion of originality by giving copyright protection to every work of authorship *"that it is its author's own intellectual creation"*. In BSA, the CJEU has developed another criterion of originality that discerns whether a work may be protected under the ordinary law of copyright[313] by virtue of the Information Society Directive, which depends on whether the expression of a work and the work's idea have become indissociable. In the U.S. legal framework, this is also known as *the merger doctrine*. BSA posits that the criterion of originality is not met where the expression of the components is, in essence, dictated by their technical function as the different methods of implementing an idea are in that case so limited that the idea and the expression have become indissociable. [314] BSA further elaborates that the author has to have the ability to express his creativity in an original manner.[315]

On the one hand, Article 1(3) clearly states that only one criterion is applied when examining the originality of a computer program, precluding all other criteria. On the other hand, Article 1(2) states that ideas and principles which underlie a program cannot be copyrightable.

The referring court opines that ideas and principles cannot be given copyright protection under the EU *acquis*, specifically under Article 9(2) of the TRIPS Agreement and Article 2 of

---

[312] In Infopaq, the CJEU established that computer programs, databases and photographs are considered to be original and therefore afforded copyright protection provided that they are their author's own intellectual creation: *"Similarly, under Articles 1(3) of Directive 91/250, 3(1) of Directive 96/9 and 6 of Directive 2006/116, works such as computer programs, databases or photographs are protected by copyright only if they are original in the sense that they are their author's own intellectual creation."* CJEU, *Infopaq International A/S v. Danske Dagblades Forening*, para 34;
The CJEU then proceeded to interpret the EU *acquis* in a way that extends that criterion of originality to every work of authorship and gives it copyright protection under the same criterion:
"In those circumstances, copyright within the meaning of Article 2(a) of Directive 2001/29 is liable to apply only in relation to a subject-matter which is original in the sense that it is its author's own intellectual creation." CJEU, *Infopaq International A/S v. Danske Dagblades Forening*, para 37

[313] *"In that regard, it is appropriate to ascertain whether the graphic user interface of a computer program can be protected by the ordinary law of copyright by virtue of Directive 2001/29."* CJEU, *Bezpečnostní softwarová asociace - Svaz softwarové ochrany v. Ministerstvo kultury*, 22 December 2010, para 44

[314] *"As the Advocate General states in Points 75 and 76 of his Opinion, where the expression of those components is dictated by their technical function, the criterion of originality is not met, since the different methods of implementing an idea are so limited that the idea and the expression become indissociable."* Ibid., para 49

[315] *"In such a situation, the components of a graphic user interface do not permit the author to express his creativity in an original manner and achieve a result which is an intellectual creation of that author."* Ibid., para 50

the WIPO Copyright Treaty. From this, the referring court has concluded by deductive reasoning that if an expression of any literary work would equate to its underlying idea or principle, that expression could not be afforded copyright protection under the *lex generalis* and, by *argumentum a maiore ad minus*, also under the *lex specialis*. From this, it would follow that regardless of whether the declaring code is given protection under the *lex specialis* or under the *lex generalis* i.e., "the ordinary law of copyright", the criterion of originality that discerns whether an idea and an expression have become indissociable, as introduced by BSA, should be applicable to the case at hand.

3. If the merger doctrine as elaborated in BSA is applicable, by what criteria should it be judged whether the idea and the expression have become indissociable?

In BSA, the CJEU instructed the Czech national court that when determining whether a GUI may be given protection under the Information Society Directive, the Czech national court would have to examine *"the specific arrangement or configuration of all the components"*[316] of a GUI to determine which meet the criterion of originality. Considering that the Java API as a whole is not at issue, but the declaring code as a component thereof is, the referring court has found that examining the declaring code by comparing it to the implementing code not to be as useful. Namely, while the implementing code resembles a computer program that performs its assigned functions and does not have the characteristics of an interface, the declaring code has the characteristics of both a software and a user interface, thus leaving the referring court without a relevant comparison that would help it to determine whether the declaring code meets the criterion of originality.

The next criterion the Czech national court was instructed to examine is the technical function of the GUI's components to discern which meet the criterion of originality. Should a component be *"dictated by its technical function, the criterion of originality is not met"*.[317] As the term "technical function" was not defined in the preliminary ruling, the referring court cannot differentiate between the declaring code and the implementing code as the functionality of both would *prima facie* seem to be dictated by their technical function. For this reason, the referring court has little use of the BSA-developed test for determining whether an idea and an expression have become indissociable.

[316] *Ibid*., para 48
[317] *Ibid.*

Provided that the BSA case is applicable to the subject matter, it would follow that the Painer v. Standard Verlags GmbH and Others ("Painer") case and its criterion of originality[318] should also be applicable as the BSA case and the Painer case both discuss criteria of originality with regards to the ordinary law of copyright. BSA stipulates that an author has to have the ability to express his creativity in an original manner to pass the threshold of originality, while Painer furthers that criterion by adding the element of "free and creative choice". Namely, Painer stipulates that a work is original if an author was able to express his creative ability by making free and creative choices when creating the work.[319] The referring court has thus recognised the relevance of Painer when determining whether the author's work may be considered his own intellectual creation.

During the Google v. Oracle case, it was established that Sun's engineers were indeed making free and creative choices while creating the declaring code.[320] If free and creative choices were made while creating the declaring code, the work would be considered copyrightable.

However, Google has challenged that finding by stating that, while Google did have the right, under the Article 5(3) of the Computer Programs Directive, to write an original expression of the Java API's implementing code, it could not put that implementing code to use because the declaring code is intertwined with the Java API's organisation of packages, classes and methods, and the rules of the Java language itself. Therefore, Google argues that it was made impossible to write a different method of implementing the idea behind the declaring code as it is dictated by its technical function, and thus, the number of different methods of writing that declaring code were so limited that its idea and expression have become virtually indissociable. Nonetheless, the referring court established that Google could have created its own declaring code but chose not to.[321] As the reason for using the declaring code was not

---

[318] *"[A] portrait photograph can, under that provision, be protected by copyright if, which it is for the national court to determine in each case, such photograph is an intellectual creation of the author reflecting his personality and expressing his free and creative choices in the production of that photograph"* CJEU, *Eva-Maria Painer v. Standard Verlags GmbH and Others*, 1 December 2011, para 152

[319] *"(…) the author was able to express his creative abilities in the production of the work by making free and creative choices." Ibid*., para 89

[320] *"The evidence showed that Oracle had 'unlimited options as to the selection and arrangement of the 7000 lines Google copied.' Appellant Br. 50. Using the district court's "java.lang.Math.max" example, Oracle explains that the developers could have called it any number of things, including "Math.maximum" or "Arith.larger." This was not a situation where Oracle was selecting among preordained names and phrases to create its packages."* CAFC, *Google LLC v. Oracle America, Inc.*, May 9, 2014, p. 30

[321] *"Google's basic objective was not simply to make the Java programming language usable on its Android systems. It was to permit programmers to make use of their knowledge and experience using the Sun Java API when they wrote new programs for smartphones with the Android platform. In principle, Google might have*

connected to the technical function of the declaring code but for the reason of attracting programmers to use the Android API, the referring court established that Google's argument is a moot point in determining whether the idea and the expression of the declaring code have become indissociable.

As the referring court lacked the necessary and sufficient criteria to determine if the expression of the declaring code is dictated by its technical function, it concluded that the expression of the declaring code is most probably dictated by its technical function. Contrary to this finding, the referring court also established that Oracle did have the opportunity to make free and creative choices while creating the declaring code, regardless of the fact that Google did not write its own declaring code afterwards.

Consequently, as the referring court lacks the criteria by which it is to resolve this contradiction, it fails to interpret the EU *acquis* and asks the CJEU to provide clear guidance and criteria on the following issues:

a) What does the term "technical function" mean?

b) By what other criteria should it be judged whether the different methods of implementing an idea have become so limited that the idea and the expression of that idea have become indissociable?

c) Can the merger of an idea and an expression arise when an author has created a work by making free and creative choices?

4. Regarding the application of Article 5(3)(i) of the Information Society Directive, what would the criteria be for determining whether Google's use of the Java API's declaring code falls under the incidental inclusion of a work exception?

In the light of the *lex specialis derogat legi generali* principle, it is the understanding of the referring court that if the *lex specialis* does not regulate all the aspects regarding the specific subject matter then the provisions of the *lex generalis* are to apply.

Article 4(1)(a) of the Computer Programs Directive prohibits the unwarranted *"reproduction of a computer program by any means and in any form, in part or in whole"*. It

created its own, different system of declaring code." SCOTUS, *Google LLC v. Oracle America, Inc*., Majority opinion, p. 30

is the understanding of the referring court that the copying and integrating a part of one computer program's source code into another computer program is encompassed in Article 4(1)(a) as a restricted act. The Computer Programs Directive does lay down an exception to such an act in Article 5(1)[322] which may be interpreted as being in line with Google's argument that the declaring code had to be copied verbatim due to the organisation of packages, classes and methods, and the rules of the Java language. As the referring court already established that using the Java API's declaring code was not necessary for Google to use its version of the Java API's implementing code, the exception under Article 5(1) is not applicable to the case at hand.

However, as in Article 5 of the Computer Programs Directive the exceptions to the restricted acts from the Article 4 are not explicitly limited to those mentioned in Article 5, it is the understanding of the referring court that the exceptions enlisted in Article 5 of the Information Society Directive are also to be applicable to the restrictions enlisted in Article 4 of the Computer Programs Directive. Therefore, regardless of whether the Computer Programs Directive, the Information Society Directive or both directives are to be applied to the subject matter, in this case, the Article 5(3)(i) of the Information Society Directive may be considered as an exception to the rightsholder's copyright.

Provided that the declaring code does not fall into the category of uncopyrightable creations due to the limited number of ways in which its underlying idea may be expressed, the use of the declaring code as a copyrighted work may be considered an incidental inclusion of a work[323] pursuant to Article 5(3)(i) of the Information Society Directive. If so, the use of the declaring code would constitute an exception or limitation to the copyright connected to that declaring code.

After comparing the 37 Java API and Android API packages at issue, the referring court established that only 3% of the source code is identical. Those identical lines of source code constitute the declaring code.

The referring court recognises that, according to settled CJEU case law, *"the provisions of a directive which derogate from a general principle established by that directive must be*

[322] Computer Programs Directive, Article 5(1): *"(...) the acts referred to in [Article 4(1)(a)] shall not require authorisation by the rightholder where they are necessary for the use of the computer program by the lawful acquirer in accordance with its intended purpose."*

[323] The author holds the concept of the incidental inclusion of a work to be rather similar to the concept of *de minimis* copyright exception in the U.S. legal framework.

interpreted strictly".[324] However, pursuant to the wording of Article 5(3)(i) of the Information Society Directive, the referring court is unable to determine without a doubt what the criteria for determining whether the inclusion of a work may be deemed incidental are, so that afterwards a strict or narrow interpretation of this exception may be performed.[325]

The criterion for determining the incidental inclusion of the declaring code might be that of quantity, quality, or both. If the quantity of the declaring code used in the Android API were to be taken as the sole criterion, it is not clear in the context of integrating parts of one source code into another, how much of the source code may be copied from the original work so as to constitute incidental inclusion of a work. If the quality of the declaring code used in the Android API is the sole criterion, it is unclear which factors should be taken into consideration, e.g., the nature of the declaring code, the relationship between the declaring code and the Android API, and must the inclusion of the declaring code be deemed unintentional. Should both criteria be applicable, it is unclear whether they are to be given equal weight or is one of the criteria of greater significance.

5. May Google's use of the declaring code be considered in the public interest pursuant to the 14th recital in the preamble to the Information Society Directive and therefore deemed as an exception to the copyright of the declaring code's rightsholder?

In order to justify its copying of the declaring code, Google invoked the 14th recital in the preamble to the Information Society Directive, which states: *"This Directive should seek to promote learning and culture by protecting works and other subject-matter while permitting exceptions or limitations in the public interest for the purpose of education and teaching."* Google argued that by copying and integrating the declaring code into the Android API, it

[324] Case C-5/08, *Infopaq International A/S v. Danske Dagblades Forening*, 2009, para 56; see also Case C-476/01 *Criminal proceedings against Felix Kapper*, 29 April 2004, para 72, and Case C-36/05 *Commission of the European Communities v Kingdom of Spain*, 26 October 2006, para 31

[325] Contrary to this, the German Federal Court of Justice ("the Bundesgerichtshof") in Case I ZR 177/13 *Möbelkatalog* developed its own criteria and test for the purposes of interpreting Article 5(3)(i) of the Information Society Directive, that is, for determining whether an inclusion of a work may be considered incidental. The Bundesgerichtshof deemed its interpretation of Article 5(3)(i) to be *acte clair* and therefore did not refer the question regarding the interpretation of Article 5(3)(i) of the Information Society Directive to the CJEU.
For more information see Nordemann, J. B.: *Incidental Inclusion of Works – Mere Incidental Relevance of the Exception according to the German Bundesgerichtshof*, available at: http://copyrightblog.kluweriplaw.com/2015/09/24/incidental-inclusion-of-works-mere-incidental-relevance-of-the-exception-according-to-the-german-bundesgerichtshof/, last accessed April 30, 2022

fostered learning and education[326] as one of the ultimate goals of the Directive. Not allowing Google to copy the declaring code would also prevent the programmers from using and transplanting their knowledge from one API to another, which would ultimately harm the public interest as it would hinder the development of the IT industry and computer science.

It is the understanding of the referring court that Google's interpretation of the 14th recital in the preamble to the Information Society Directive is inconsistent with Article 5(3)(a) of the Directive, which clearly states that, as an exception, a work may be used *"for the sole purpose of illustration for teaching or scientific research, as long as the source, including the author's name, is indicated, unless this turns out to be impossible and to the extent justified by the non-commercial purpose to be achieved"*. Google's use was neither intended for scientific research nor as an illustration for teaching purposes and it did not indicate who the rightsholder of the declaring code is. It should also be noted that Google's use of the declaring was decidedly commercial.

7.4.2. The hypothetical CJEU preliminary ruling

1. & 2. In the opinion of the author of this paper, it is most likely that the CJEU would answer the first and the second question together, given the close connection between the two issues posed by these questions. The CJEU would hold that the referring court had correctly interpreted the *lex specialis derogat legi generali* principle regarding the relationship between the Computer Programs Directive and the Information Society Directive. The CJEU would find that the Computer Programs Directive is applicable to the declaring code according to the two BSA v. Czech Republic criteria for distinguishing whether a part of a computer program, an interface, may be protected by copyright in computer programs by virtue of the Computer Programs Directive. These criteria require that a part of a computer program *"does not enable the reproduction of a computer program"* while constituting *"one element of that program by means of which users make use of the features of that program"*.[327]

[326] *"Finally, given programmers' investment in learning the Sun Java API, to allow enforcement of Oracle's copyright here would risk harm to the public. Given the costs and difficulties of producing alternative APIs with similar appeal to programmers, allowing enforcement here would make of the Sun Java API's declaring code a lock limiting the future creativity of new programs."* SCOTUS, *Google LLC v. Oracle America, Inc*., Majority opinion, April 5, 2021, p. 34

[327] CJEU, *Bezpečnostní softwarová asociace – Svaz softwarové ochrany v Ministerstvo kultury*, 22 December 2010, para 41

These criteria may be expressed as the following two questions:

a) Does a part of a computer program enable the reproduction of a computer program?

b) Is that part of a computer program one element of a program by means of which users make use of the features of that program?

When the characteristics of the declaring code were considered in the context of these two questions, the answer to the first question would be positive because the declaring code creates an interconnection between the implementing code and a program making calls to the implementing code. BSA did not establish that a part of a computer program needs to be essential to enable the reproduction of a computer program, therefore any part of a computer program that does in any way enable its reproduction is protected under the Computer Programs Directive.

The answer to the second question would also be positive because the declaring code is one element of the Java API by means of which programmers make use of the features of the Java API, the implementing code. The programmers may be considered users since they are the direct users of the declaring code and the method calls, unlike the end-users of a program that was written to use the Java API. The end-users who are not programmers themselves, apart from installing the JRE on their computers, usually have no knowledge of the functioning of the declaring code and only use the declaring code indirectly, while using a program that has been written to use the Java API.

From this it would follow that if the answer to both questions is positive, the part of the computer program at issue is to be given copyright protection under the Computer Programs Directive. If the answer to the first question is negative and to the second positive, the part of the computer program at issue is to be given copyright protection under the Information Society Directive. In the case that the answer to the first question is positive and to the second negative, that part of a computer program is to be given copyright protection under the Computer Programs Directive, because not all elements of a computer program are means by which users make use of the features of that program, but merely engender its reproduction. Should the answer to both questions be negative, it is likely that these two questions would be unsuitable as the work in question would not be part of a computer program and other criteria would need to be applied to determine whether the work may be given copyright protection.

As the answer to both questions is positive when it comes to the declaring code, the declaring code is afforded copyright protection under the Computer Programs Directive. This is without prejudice to the provisions of the Information Society Directive that would be applicable as *the lex generalis*, which pertains to matters of the ordinary law of copyright. Given that the Computer Programs Directive provides for no test suitable to determine whether an idea and its expression have become indissociable, from this it would follow that CJEU would apply the BSA-developed test, originally intended to establish whether a merger has occurred in a work protected by the ordinary law of copyright, to the declaring code, as it is the only such test provided by the CJEU case law.

After having found the Information Society Directive applicable and the application of the BSA-developed test to establish whether merger has occurred merited, the CJEU would next examine the application of the test itself.

3. Regarding the third question, it is the author's opinion that most likely the CJEU would hold that the expression of the declaring code and the underlying idea did not merge.

First, the CJEU would consider the BSA-developed test for establishing whether merger has occurred, as it is the only such test provided by the CJEU case law. It would find this test not as suitable for computer programs as it is for other types of works which are regulated by the ordinary law of copyright.

Namely, in BSA the CJEU determined that a work cannot meet the criterion of originality if the expression of its components is dictated by their technical function and consequently, the expression and the underlying idea are indissociable. From a legal standpoint, the term "technical function" is rather vague. In the context of computer programs and interfaces, it is hard to argue that, *stricto sensu*, computer programs and their components are not dictated by their technical function, considering that they are utilitarian by their very nature.

While deliberating which criteria to include in the test that could determine whether the expression of a computer program and an underlying idea have merged, the CJEU would also have to take into account the Painer criterion of originality that stresses the importance of the ability of an author to creatively express himself through his work by making free and creative choices. During the Google v. Oracle case, it was established that Sun's engineers were indeed

making free and creative choices while creating the declaring code.[328] The BSA criterion denies the ability of an author to creatively and freely express himself if the expression of his work is dictated by its technical function. Therefore, by *argumentum a contrario*, the author can creatively and freely express himself if his work is not dictated by its technical function.

As computer programs, as well as interfaces, are by their very nature dictated by their technical function, the author of a computer program would appear to never be able to express himself freely and creatively. This stands in contradiction to Article 1(1) of the Computer Programs Directive which explicitly gives protection to computer programs as literary works, thereby creating a paradoxical situation. The author opines that the only logical solution the CJEU might have to resolve this apparent paradox is to disregard the BSA test and apply solely the Painer-derived criteria in order to afford the declaring code the protection provided by Article 1(1) of the Computer Programs Directive.

Google's argument that the idea and the expression of the declaring code have merged would most probably not be accepted by the CJEU as during the Google v. Oracle case it has been proven that Apple and Microsoft have both successfully developed their own declaring code.[329]

As the declaring code most likely would not be deemed to have merged with its underlying idea and thus would be considered copyrightable, the possible exceptions and limitations to its author's copyright are to be considered next.

4. When weighing the quality and quantity criteria for determining whether the use of the declaring code may be qualified as incidental under Article 5(3)(i) of the Information Society Directive, it is likely the CJEU would give more weight to the quality criterion, while the quantity criterion might even be disregarded entirely.

The CJEU might decide that deliberating on the quantity criterion is a moot point as the subject matter of the case at hand is specifically the declaring code and not the Java API as a whole. For this reason, it is unclear whether the CJEU would take into account the 37 Java API packages as a whole and give any weight to the fact that only 3% of the source code was copied.

[328] See *supra* 6.2.2. p. 45

[329] *"Instead of creating its own declaring code — as Apple and Microsoft chose to do — Google copied verbatim 11,500 lines of Oracle's declaring code."* SCOTUS, *Google LLC v. Oracle America, Inc*., Dissenting opinion, April 5, 2021, p. 4

The CJEU would, however, take into account the quality criterion. The author of this paper speculates that although the CJEU would not consider the expression and underlying idea of the declaring code to have merged, it might take into account the nature of the declaring code as a part of an interface which is, as Google would claim, integral to communication between the Java API implementing code and another program. Although the declaring code would be considered the part of the Java API that engenders the reproduction of a computer program, the CJEU might see the declaring code as not being as essential to that function, but instead merely facilitating the process of writing a program. The reason for this is that, at least in theory, a programmer might copy the needed Java API implementing code and paste it in the place of the method call, a shorthand of the declaring code.

The CJEU might take into account whether Google intentionally integrated the declaring code, which might weigh against the incidental use of a work.

While determining whether Google's use of the declaring code meets the criteria for incidental use from Article 5(3)(i), the CJEU would also have to consider the 44th recital in the preamble to the Information Society Directive which stipulates that *"the scope of certain exceptions or limitations may have to be even more limited when it comes to certain new uses of copyright works"* and in particular *"in the context of the new electronic environment"*. Because the use of the Java API's declaring code in the Android operating system intended for smartphones may be considered a new use of a copyrighted work in the context of the new electronic environment, the CJEU would have to interpret the scope of Google's incidental use in an even more limited way.

After careful deliberation, the CJEU would either outright decide whether Google's use may be qualified as incidental under Article 5(3)(i) of the Information Society Directive or provide the referring court with a test or criteria to determine whether that use may be considered incidental. For the sake of argument, it shall be assumed that CJEU decided that Google's use qualifies as incidental under Article 5(3)(i). As the referring court did not ask the CJEU for an interpretation of Article 5(5) of the Directive ("three-step test"), it is uncertain whether the CJEU would, apart from Article 5(3)(i), also deliberate on Article 5(5). It is possible that it would be left to the referring court to deliberate on the questions of fact in the Google v. Oracle case and subsume the facts of the case under the three-step test as the criteria of the three-step test are considered an *acte claire*. For this reason, the application of Article 5(5) is analysed in section 7.4.3. of this paper.

5. The CJEU would probably decide that Google misinterpreted the 14th recital in the preamble to the Information Society Directive and that it may not be deemed as an exception to the copyright of the declaring code's rightsholder. Whereas the Directive does permit *"exceptions or limitations in the public interest for the purpose of education and teaching"*, it also seeks to *"promote learning and culture by protecting works and other subject-matter"*.

The purpose of the Android operating system or the Android API is not primarily educational and, in the light of the 42nd recital in the preamble to the Information Society Directive,[330] the nature of the activity of developing the Android API as well as programming by using the Android API is not strictly non-commercial.

Furthermore, the Information Society Directive has an exclusive list of exceptions and limitations to copyright and the exception closest to Google's use of the declaring code is the exception in Article 5(3)(a). The CJEU would find that the referring court correctly interpreted Article 5(3)(a) when it established that Google's use, not being strictly educational and not being strictly non-commercial, would not constitute an exception under that Article.

7.4.3. The application of the preliminary ruling by the hypothetical referring court

After the CJEU issues the preliminary ruling, the referring court is to apply the CJEU's interpretation of the EU *acquis* and subsume the facts of the case under it. As the question regarding Article 5(3)(i) of the Information Society Directive would be the only one in which the CJEU might leave it up to the referring court to subsume the facts of the case under the rule of law, it is the only question discussed in this section.

Provided that Google's use of the declaring code is qualified as incidental under Article 5(3)(i) of the Information Society Directive, the referring court would apply the three-step test from Article 9(2) of the Berne Convention, which has been transposed to Article 5(5) of the Information Society Directive, and determine whether its three criteria are cumulatively met.

It should be noted that the aim of Article 9(2) of the Berne Convention is to outline a standard for its signatories when introducing possible exceptions and limitations to copyright into their national copyright laws. For this reason, a citizen of a signatory to the Berne

[330] The 42nd recital in the preamble to the Information Society Directive: "When applying the exception or limitation for non-commercial educational and scientific research purposes, including distance learning, the non-commercial nature of the activity in question should be determined by that activity as such. The organisational structure and the means of funding of the establishment concerned are not the decisive factors in this respect."

Convention may not invoke Article 9(2) but may invoke an exception to the copyright which has been laid down in the national copyright law of the signatory on the basis of Article 9(2). Similarly, the aim of Article 5(5) of the Information Society Directive is to supplement the prescribed exceptions and limitations in Article 5(3)(i) of that Directive so that exceptions may be transposed into the national law of the respective EU member state.

Professors Hugenholtz and Okediji interpret the criteria of the three-step test in Article 9(2) of the Berne Convention as follows:

a) in *"certain special cases"*, which would mean that they are *"not overly broad"*,

b) those cases *"may not conflict with a normal exploitation of work or other subject matter"*, which would mean that they *"do not rob right holders of a real or potential source of income that is substantive"*, and

c) they *"do not unreasonably prejudice the legitimate interest of the rightsholder"*, which would mean that they *"do not do disproportional harm to the right holders"*.[331]

Although this interpretation is intended for Article 9(2) of the Berne Convention, the meaning assigned to its wording should be equally applicable to Article 5(5) of the Information Society Directive.

Under the condition that the CJEU qualifies Google's use as incidental under Article 5(3)(i) of the Directive, it is likely the referring court would not deliberate on the first of the three steps while considering whether Google's use may pass the three-step test under Article 5(5) of the Information Society Directive. The author of this paper opines that if the CJEU qualifies Google's use as incidental in its preliminary judgement, it would already be considered a *certain special case* under the first step, in which case the referring court would proceed to deliberate on the second step. Alternatively, if the CJEU provides the referring court only with the criteria whether the use of a work may be deemed incidental, it would be left to the referring court to conduct the analysis whether or not Google's use may be considered incidental in the light of Article 5(3)(i), thereby having to also deliberate on the first of the three steps. In this case, it shall be assumed that the CJEU found Google's use of the declaring code to be incidental.

[331] Hugenholtz, P.B.; Okediji, R. L.: *Conceiving an International Instrument on Limitations and Exceptions to Copy-Right*, Amsterdam Law School Legal Studies Research Paper No. 2012-43, Institute for Information Law Research Paper No. 2012-37, March 06, 2008, p. 25

While determining whether Google's use of the declaring code meets the criteria enumerated in Article 5(5), the referring court, like the CJEU, would also have to consider the 44th recital in the preamble to the Information Society Directive. Because the use of the Java API's declaring code in Android smartphones may be considered a new use of a copyrighted work in the context of the new electronic environment, the referring court would interpret steps two and three of Article 5(5) as even more limited than originally intended.

Regarding the second step, Google's use is proven to be commercial and therefore would be seen as conflicting with a normal exploitation of the declaring code. This is apparent from the fact that Oracle was deprived of real and potential profit from licensing the Java API and offering services like maintenance and support. Google's use pushed Java API out of the market and caused Oracle to suffer losses, having to cut its prices by up to 97.5%[332] when striking new deals. By using the declaring code without its rightsholder's consent, Google infringes copyright while realising profit and causing damages to Oracle.

Regarding the third step, it could be argued that Google did unreasonably prejudice the legitimate interest of the rightsholder as causing the loss of real and potential profits also caused a decline in relevance of the Java API due to its popularity among programmers falling. In its majority opinion, the SCOTUS opined that *"programmers learning the Java language to work in one market (smartphones) are (...) able to bring those talents to the other market (laptops)"*,[333] implying that Android actually contributed to popularising the Java API. In doing so, the SCOTUS lost sight of the fact that Java and Android API are not cross-compatible as only 37 out of 166 packages in Java and 167 packages in Android API have the same declaring code. This would mean that it is not easy for programmers who have learned only how to use Android API method calls to bring their talent to desktop and laptop computers and start using Java API method calls as they would still need to learn the remaining 130 method calls, the functions of the associated implementing code and study the documentation of the associated API packages. Therefore, it is very likely that the referring court would not see

---

[332] *"The jury heard evidence that Amazon, which had entered into a license to use Java for its Kindle tablet device, switched to Android for the subsequently released Kindle Fire and then used the existence of Android to leverage a steep discount from Oracle on the next generation Kindle."* CAFC, *Oracle America, Inc. v. Google, Inc*., March 27, 2018, p. 7;
"But after Google released Android, Amazon used the cost-free availability of Android to negotiate a 97.5% discount on its license fee with Oracle." SCOTUS, *Google LLC v. Oracle America, Inc*., Dissenting opinion, April 5, 2021, p. 12

[333] *"[P]rogrammers learning the Java language to work in one market (smartphones) are then able to bring those talents to the other market (laptops)."* SCOTUS, *Google LLC v. Oracle America, Inc*., Majority opinion, April 5, 2021, p. 33

Google's use of the declaring code as meeting the criteria set out in steps two and three of Article 5(5).

This highlights the contrasting way in which the U.S. and the EU legal systems view the use of a copyrighted work in a new electronic environment: The U.S. judiciary considers the use of the declaring code in smartphones transformative and therefore fair, as was established in the SCOTUS majority opinion, while the EU judiciary might view it as even more detrimental to the interests of the rightsholder.

8. Conclusion

When the U.S. enacted the Computer Software Copyright Act in 1980 and became the first country to unambiguously grant copyright protection to computer programs, it caused a ripple effect, leading to changes in the copyright protection of computer programs worldwide.[334] In 1994, the WTO TRIPS Agreement gave computer programs the same level of copyright protection as was afforded to other literary works, while in 1996, the WIPO Internet Treaties[335] proceeded to regulate the copyright protection of computer programs in greater detail. By enacting the Computer Programs Directive in 1991, the European Union also opted to use copyright as the optimal means of computer program protection, and by extension, all the EU member states did so too.

Ever since the beginning of the Google v. Oracle case, hopes were high that it would advance legal thought on the protection of computer programs and shed light on certain legal questions that remained ambiguous, in particular regarding the copyrightability of API. The *"copyright case of the century"*[336] might have led to a disruption affecting not only the U.S. legal system, but also echoing worldwide and, once again, influencing international agreements and national legal systems in the field of copyright protection.

At first glance, by ruling in favour of Google, it seems that the SCOTUS aimed to maintain the *status quo* of the current legal framework of software protection by neither expanding the scope of copyright protection of computer programs nor reducing it. Nevertheless, its effects are very much disruptive as they bring into question the suitability of copyright as a legal framework for the protection of computer programs and give cause to consider the alternatives thereto.

In their brief[337] to the SCOTUS in support of Google, 78 computer scientists as *amici curiae* argued that ruling in favour of Google is a ruling in favour of software innovation and *"the Progress of Science and useful Arts"*[338] as it would allow for continuing the current API

---

[334] Katulić, T., *op. cit*., fn. 2, p. 241

[335] The WIPO Copyright Treaty and the WIPO Performances and Phonogram Treaty are known together as the WIPO Internet Treaties. According to WIPO, available at: https://www.wipo.int/copyright/en/activities/internet_treaties.html, last accessed December 17, 2021

[336] Lemely, M., *op. cit*., fn. 6

[337] *Motion for Leave to File Brief of 78 Amici Curiae and Brief of 78 Amici Curiae Computer Scientists In Support Of Petitioner*, February 25, 2019, available at: https://www.supremecourt.gov/DocketPDF/18/18-956/89487/20190225134131839_18-956_Oracle_v__Google_Computer_Scientists_Amicus_Motion_Brief_FILE.pdf, last accessed December 15, 2021

[338] See *supra* fn. 258

reimplementation practice. The free use of API,[339] unencumbered by copyright and a need to prove fair use,[340] has become a standard industry practice, giving rise to the development of operating systems, programming languages, the Internet and cloud computing. The *amici* emphasised that when developing the Java API, Sun itself followed that practice by reimplementing the math API from the C language. A ruling in favour of Oracle would have *"threaten[ed] to upend decades of settled expectations across the computer industry"* that API is not copyrightable.[341] According to the *amici*, API copyrightability would lead to the monopolisation of standard APIs which, in turn, would lead to less competition and innovation,[342] increased software prices for end-users due to API licencing fees, fewer software product choices, and incompatible software. This would create a lock-in effect for end-users as companies would stop working toward interoperability between different services due to the fear of litigation with API copyright holders.

On the other hand, supporters of Oracle[343] argue that a ruling in favour of Google weakens the copyright protection of original software and opens the gates to companies such as Google to push other, weaker companies out of the market. Giving up copyright on a part of those companies' software *"would be far more destabilizing to the market than allowing Oracle to continue protecting its own IP"*.[344] This would lead to monopolisation and less innovation in the software industry because of *"a chilling effect on innovators who bear the cost of original content development"*.[345]

---

[339] Pro Google amici curiae on the nature of API opine that as API is not a computer program it is not copyrightable: "*Software interfaces, including those embodied in the Java Application Programming Interface (API) at issue here, are purely functional systems or methods of operating a computer program or platform. They are not computer programs themselves. Interfaces merely describe what functional tasks a computer program will perform without specifying how it does so*." *Motion for Leave to File Brief of 78 Amici Curiae and Brief of 78 Amici Curiae Computer Scientists In Support Of Petitioner*, p. 3, see fn. 337

[340] "*By creating standard specifications for computer programs to communicate with each other, uncopyrightable software interfaces have promoted competition in personal computing*." See also *ibid*., p. 24 "*Relying on fair use is no answer. A fair use standard creates uncertainty because it depends on fact-intensive, case-by-case determinations which can result, as demonstrated by this case, in lengthy and prohibitively expensive litigation. (…) Conditioning API reimplementation on fair use would impede innovation and competition almost as much as denying reimplementation outright*." *Ibid.*, p. 17

[341] *Ibid*. p. 2

[342] "*Uncopyrightable software interfaces address network effect barriers by enabling startups to plug into existing systems and grow through cumulative improvements*." *Ibid*., p. 22; Therefore, small businesses and start-up companies could not afford to pay API licence fees or would need a long time to invent their own, equally efficient API.

[343] Stricklett, S. G.: *Google v. Oracle: An Expansive Fair Use Defense Deters Investment In Original Content*, available at: https://www.ipwatchdog.com/2020/01/19/google-v-oracle-expansive-fair-use-defense-deters-investment-original-content/id=117951/, last accessed December 15, 2021

[344] *Ibid*.

[345] *Ibid*.

Both sides are claiming that the other party winning the case leads to the stifling of software innovation in the future, and both sides might be right. *"Software developers and investors greatly value clarity in making the difficult, time-sensitive decisions involved in designing products and platforms."*[346] By trying to perpetuate the *status quo*, the SCOTUS has left the legal scholars and courts, as well as programmers, investors and their lawyers, with many questions unanswered.

The SCOTUS gave its verdict on two groups of questions: the copyrightability and fair use of the Java API. The SCOTUS explained that by the Java API it specifically referred to the Java API's declaring code and its structure, sequence and organisation. The SCOTUS also included the implementing code in the scope of the Java API but did not adjudicate it, as it was not a point of contention. Without ruling on the copyrightability of the Java API, the SCOTUS responded positively to the software industry's plea for the continuance of the free Java API reimplementation practice by ruling that the use of Java's declaring code and SSO does constitute fair use. It did not address the question of the fair use of the SSO in any further detail, but it did elaborate on the fair use of the declaring code.

In elaborating on the fair use of the declaring code regarding the second fair use factor, the nature of copyrighted work, the SCOTUS established a distinction between the copyright protection of declaring and implementing code by stating that the declaring code is *"further from the core of copyright"* than other types of code.[347] It quoted the Campbell case to explain that *"some works are closer to the core of intended copyright protection than others, with the consequence that fair use is more difficult to establish when the former works are copied"* and corroborated this logic by quoting other cases that contrast *"fictional short stories with factual works"* as well as *"creative works with bare factual compilations"*.[348]

However, not giving a yes or no answer to the question of the API copyrightability, while implying that it might not be copyrightable by stating that it is by its very nature *"further from the core of copyright"*, only deepens the existing legal uncertainty regarding the nature of computer programs and copyright as a valid means of their protection.

---

[346] Menell, P. S.: *Rise of the API Copyright Dead?: An Updated Epitaph for Copyright Protection of Network And Functional Features of Computer Software*, Harvard Journal of Law & Technology, Vol. 31, Spring 2018, p. 472

[347] "*In our view, (…) the declaring code is, if copyrightable at all, further than are most computer programs (such as the implementing code) from the core of copyright*." SCOTUS, *Google LLC v. Oracle America, Inc*., April 5, 2021, p. 24

[348] SCOTUS, *Campbell, aka Skyywalker, et al. v. Acuff-Rose Music, Inc.*, March 1994, p. 586, available at: https://supreme.justia.com/cases/federal/us/510/569/case.pdf, last accessed December 15, 2021

The U.S. Copyright Act[349] does not protect procedures and methods of operation, yet it 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"* without any consideration for the fact that by definition a computer program is always going to be a set of functional procedures and methods of operation.[350] The U.S. Copyright Act neither defines what exactly the terms "procedure" and "method of operation" stand for, nor does it define the term "interface". It is not easy to grasp the logic of how one method of operation, the declaring code, is further from the core of copyright than another method of operation, the implementing code, when it seems that both are already light-years away. Except for *"computer games and audiovisual entertainment software"* virtually all computer programs *"are utilitarian in nature and, besides the fact that their source code can be printed out on paper and resemble a text, have very little in common with literary works (...)"*.[351] Computer programs *"are not works of art, their purpose in and of themselves is not to convey an original communication by the author. Instead, they are highly sophisticated tools that allow everyday use of computers in the digital environment (...)"*.[352] Therefore, to determine which part of a program is to be afforded legal protection, by discussing which part conveys an original message and therefore represents a more creative expression of its programmer, versus which part is entirely utilitarian in its nature, seems beside the point.

In the first ruling of the District court, judge Alsup noticed the existence of a paradox in the filtration step of the abstraction-filtration-comparison test that is meant to help courts distinguish copyrightable from uncopyrightable elements of a program. Specifically, judge Alsup referred to the uncopyrightability of the elements dictated by efficiency.[353] If an expression of an idea embodied in an element of a program is made so efficient, that a certain idea can only be feasibly expressed in one or two ways, that element of a program is uncopyrightable. *"Paradoxically, this meant that non-efficient structures might be copyrightable while efficient structures may not be."*[354] This paradox, instead of fostering the *"Progress of Science and Art"*[355] might in theory lead to programmers writing less efficient

[349] 17 U.S. Code §101
[350] According to Perlmutter, S. in *Google vs. Oracle: Implications for Software Innovation?*[Video], [21:10 – 21:57], uploaded by the Center for Strategic & International Studies, April 29, 2021, available at: https://www.csis.org/events/google-vs-oracle-implications-software-innovation, last accessed April 30, 2022
[351] Katulić, T., *op. cit*., fn. 2, p. 260
[352] *Ibid.*
[353] The elements dictated by efficiency are explained in 5.2.4. under a)
[354] District court, *Oracle America, Inc. v. Google Inc*., May 31, 2012, p. 22
[355] See *supra* fn. 258

code to protect their intellectual property. In its amicus brief in support of Oracle, the U.S. Government argued that contrary to the efficiency criterion of the filtration step of the abstraction-filtration-comparison test, *"[t]he Copyright Act as a whole makes clear that computer programs can be protected by copyright, refuting any suggestion that the functional character of computer code suffices to bring it within Section 102(b)."*[356]

In the EU, the Computer Programs Directive elaborates that *"[i]deas and principles which underlie any element of a computer program, including those which underlie its interfaces, are not protected by copyright under this Directive."*[357] In other words, the Directive differentiates an interface and its underlying idea more directly than the U.S. Copyright Act does, and this kind of phrasing might have served the SCOTUS to come to a different conclusion regarding the question of the copyrightability of the declaring code. Justice Thomas emphasised that Java declaring code is a part of a copyrightable interface, however, the ideas that underlie the Java declaring code are not: *"Oracle cannot copyright the idea of using declaring code"*.[358]

Given that the SCOTUS has created the distinction between code that is closer and code that is *"further from the core of copyright"*, it would be unsurprising to see other courts in the U.S. follow suit in the future. This problem does not exist only for API developers, but also for developers of other types of software. This ruling has opened the gates for other courts to also examine and determine which part of the code does and which does not enjoy copyright protection on the basis of whether the judge or the jury are convinced of a certain part of code being *"expressive"*[359] or closer to *"the core of copyright"*. The SCOTUS did state that the declaring code and its underlying idea have merged but offered only a meagre explanation as to the "whys and hows" of the analysis that led it to that conclusion, which was pointed out by Justice Thomas in his Dissenting opinion.[360] If that had been the case, it would have shined a light on that issue and have been of great help for legal scholars and other courts in the future.

Apart from the SCOTUS creating a more fragmented puzzle for legal scholars and courts to analyse, it did little to address the more practical and pressing issues programmers

---

[356] Brief for the United States as Amicus Curiae, p. 12, available at: https://www.supremecourt.gov/DocketPDF/18/18-956/117359/20190927165110897_18-956%20Google.pdf, last accessed December 15, 2021

[357] See Article 1(2) and 10th recital in the preamble to the Computer Programs Directive

[358] SCOTUS, *Google LLC v. Oracle America, Inc*., Dissenting opinion, April 5, 2021, p. 7

[359] *"[Fair use can be applied to help the Court] distinguish between expressive and functional features of computer code where those features are mixed."* SCOTUS, *Google LLC v. Oracle America, Inc.*, Majority opinion, April 5, 2021, p. 17

[360] See 6.4.2., pp. 57-60

encounter daily. Namely, while creating new programs this legal uncertainty causes them to risk serious damage to their employer or to their very own company. To decide whether to use another company's proprietary code on the basis of the Google v. Oracle case, a few questions[361] should be answered first:

If API is not the type of proprietary software being used, how does one determine whether a certain part of proprietary code is closer to the core of copyright, like implementing code, or further from it, like declaring code? Although the SCOTUS omitted defining implementing code in its fair use analysis, Justice Thomas did demonstrate in his analysis of the nature of copyrighted work that the features which make declaring code uncopyrightable, like its connectedness to the idea of *"the overall organization of the API"*,[362] are also present in the implementing code. This is because it may be associated with *"the division of computing tasks"*,[363] which is an idea unto itself. Are programmers to hire a legal expert at this point to help them determine with more certainty the nature of the desired part of the proprietary code? Could any legal expert answer this question with any certainty?

If API is the type of proprietary software being used, or a conclusion may be reached for some other type of software being used that it is *"further from the core of copyright"*, could programmers rely on the SCOTUS ruling in the Google v. Oracle case and consider it fair use? However, even if the fair use is obvious, and they would win the case when sued, could the programmer or his employer endure the length of a trial and the related legal costs? Even if the law is on their side, would paying for a licence be more cost-effective? If fair use is not all that obvious, would it be worth the price of a licence to take the risk and roll the dice of legal ambiguity? The court might simply not find the new program transformative enough and label it a derivative work, or the proprietary program's owner might be in a better market position to compete with their new program.

Alternatively, when the programmer or his employer ponder the scope of their proprietary program's protection, would they have to consider whether the initial value of a creatively expressive program might evolve over time in a way that causes it to become valuable primarily due to its popularity and wide adoption? And does this mean that as its popularity rises over time, its protection deteriorates in an inversely proportional extent due to

[361] Similarly, Hansen, C.: *Google v. Oracle: SCOTUS Sides with Google on Fair Use, But Is the Ruling Narrower Than It Seems?*, available at: https://www.jdsupra.com/legalnews/google-v-oracle-scotus-sides-with-4450778/, last accessed December 15, 2021
[362] SCOTUS, *Google LLC v. Oracle America, Inc*., Syllabus, April 5, 2021, p. 3
[363] SCOTUS, *Google LLC v. Oracle America, Inc*., Dissenting opinion, April 5, 2021, p. 10

its wide adoption? However, copyright would not punish an author of a popular book in the same fashion, quite the opposite. Connected to a program's popularity is the *"programmers' investment in learning"* to use that program, much like the Java API, and *"to allow enforcement of (…) copyright here would risk harm to the public."*[364] The question is how popular should a program be, how wide the public, i.e., how numerous the programmers who are using it, for the program to lose its copyright protection, and at what moment does that exactly happen?[365] Justice Thomas used an example of actors learning a Broadway musical script to illustrate that the value of declaring code should not depend on the time invested in learning how to use it.[366]

If API is a type of code that the programmer or his employer seeks to protect, can they continue to rely on its copyrightability? Does it make sense to put it under a proprietary licence? Even so, how likely is the other party's unauthorised use of their API to be considered fair? Is it sensible to file a lawsuit in that case? Should they abandon the copyright to their API altogether and decide to put API under an open-source licence, growing their business around offering services? Can they even put the API under an open-source licence, as they would need to have a copyright on an API to be able to opt for an open-source licence? Are the answers to those questions the same for every type of API? Is there any difference between the protection of the library API, like the Java API, and the Web API?

One may answer these questions differently when considering them from the perspective of a single programmer, a small software company, or from the perspective of a Big Tech giant. At this point, legal protection seems unaffordable for single programmers and small tech companies, while at best uncertain for Big Tech companies. The duration and expenses of litigation, coupled with the current legal ambiguity surrounding these questions, clearly demonstrate that only Big Tech has the endurance to finance and engage in a prolonged legal battle, without endangering their business in the meantime. This gives rise to an undesirable phenomenon causing individual programmers to turn to selling their skills as a service, instead of innovating. Specifically, regarding the API technology, *"[t]ech companies*

[364] SCOTUS, *Google LLC v. Oracle America*, Inc., Majority opinion, April 5, 2021, p. 34
[365] The answer to this question might be similarly confusing as the answer to Eubulides of Miletus' Sorites paradox on how many grains are needed to create a heap. For more information on Sorites paradox, visit Stanford Encyclopedia of Philosophy, available at: https://plato.stanford.edu/entries/sorites-paradox/, last accessed December 15, 2021
[366] "*[A] theater cannot copy a script—the rights to which are held by a smaller theater—simply because it wants to entice actors to switch theaters and because copying the script is more efficient than requiring the actors to learn a new one*." SCOTUS, *Google LLC v. Oracle America, Inc*., Dissenting opinion, April 5, 2021, p. 11

may be less incentivised now to innovate and develop APIs, knowing that bigger companies can appropriate their content rather than pay for it (like Google did with Java SE).”[367]

While not yet having been considered before the CJEU, some of these issues are already present in the EU and mirror the ambiguities found in the U.S. copyright system. When these issues finally arise before the CJEU, one should bear in mind that the EU *acquis* does not have a tool at its disposal such as the fair use doctrine. Namely, as the EU criterion of creativity seems to be of a binary nature, it appears that the CJEU would not differentiate between levels or degrees of a work's creativity to determine the possible application of certain exceptions or limitations of copyright in a particular case. The Information Society Directive lays down the enumerated exceptions to copyright and determines when those exceptions are to apply to a copyrighted work. Should the language of the Directive not be clear enough, it is up to the CJEU to interpret the meaning of an exception's wording. This would be done by referring to the preamble of the Directive, Article 5(3) of the Directive that stipulates the exceptions as well as Article 5(5) which transposes the Berne three-step test, and finally weighing them both individually and together to reach a final judgement in each particular case.

When determining whether an exception may be applicable, the criteria of Article5(5) of the Directive need to be met cumulatively. Contrary to this, when weighing whether an exception to copyright applies to a copyrighted work, the U.S. courts apply four fair use factors which do not have to be cumulatively met. Rather, the four fair use factors are weighed individually and are then balanced against each other. Different weight may be given to different factors depending on the circumstances of each case. This creates legal ambiguity as the parties in a lawsuit cannot always foresee how each individual factor would be weighed in a certain case. Google's legal consultants, that presumably believed Google's use of Sun's declaring code would be fair, could not have predicted that the decisions of the courts on the four fair use factors would swing like a pendulum; from the District court deciding that all four factors favour Google to the CAFC deciding that two factors favour Oracle, one factor favours Google while one factor is equivocal at best, and then back to the SCOTUS deciding again that all four factors favour Google. Prof. Nimmer opined: *“When courts bring all four fair use factors in one direction artificially, that is bad for copyright doctrine. (…) I would be much happier if the [SCOTUS] [had] started off with [the second factor, the nature of the copyrighted work] and then moved to [the third factor, the amount and substantiality of the*

[367] Frankel, B.*: Google v. Oracle: bad news for API developers*, available at: https://www.worldipreview.com/article/google-v-oracle-bad-news-for-api-developers, last accessed December 15, 2021

portion used] (…) and then said 'we agree with the [CAFC] that there are two other factors that move in the opposite direction, but in the context of computer software we balance the factors by concluding that this is fair use' and that would have been a very salutary development to the law of fair use. Unfortunately, the majority felt the need to align all four factors in favour of fair use and it does create dangers for the future."[368]

The legal ambiguity in the U.S. copyright system seems even greater when taking into consideration that even if a work does meet the threshold of originality, it nonetheless might be considered not to have a high degree of creativity. This may cause it to be deemed *"further from the core of copyright"* which would result in it being afforded less copyright protection and the scope of fair use being expanded when it comes to such a work.

Even though it appears that the fair use doctrine creates legal ambiguity, it could also be argued that the CJEU and the EU member states' courts not recognising the fair use doctrine creates legal ambiguity, albeit for different reasons. Namely, the main disadvantage of the EU copyright system is its rigidity when it comes to the enumerated exceptions and limitations to copyright, especially regarding computer programs. Considering the rapid development of the technology and market it could be argued that the fair use doctrine would better equip the courts to respond in a timely manner. Moreover, the EU member states' courts must rely on the rudimentary EU *acquis* when adjudicating issues of the copyright of computer programs, whereas it seems that the U.S. case law has tackled such issues more frequently and has produced more specific rules and tests to be applied in such cases.

The EU member states' courts might consider the EU *acquis* to be clear enough and decide that it is not necessary to stay the proceedings and refer questions to the CJEU and consequently may adjudicate differently in similar cases. This would not only result in legal ambiguity but also in the fragmentation of the single market, making the EU less attractive for investment and program development. Even if the EU member states' courts were likely to refer questions to the CJEU for a preliminary ruling, it still might take years for an individual case to be referred to the CJEU, depending on the speed of the courts of each EU member state.

[368] Nimmer, D. in *Google vs. Oracle: Implications for Software Innovation?*[Video], [50:39 – 51:53], uploaded by the Center for Strategic & International Studies, April 29, 2021, available at: https://www.csis.org/events/google-vs-oracle-implications-software-innovation, last accessed April 30, 2022; For more on the lack of predictability of fair use, see Nimmer, D.: *"Fairest of them All" and Other Fairy Tales of Fair Use*, Law and Contemporary Problems, Vol. 66, 263-288, Winter 2003

These issues that would be faced in the EU demonstrate that the EU copyright system is as out of phase with the protection of computer programs as the U.S. copyright system is, for similar reasons.

When considering that the nature of works protected by copyright is primarily expressive while that of computer programs is primarily functional, it becomes clear that the two copyright systems have adopted a legal fiction in which methods of operation are not copyrightable, yet computer programs are. The CJEU has inadvertently demonstrated through its case law how the EU copyright system is unfit for the protection of computer programs. It is oblivious to the existence of the creativity – functionality conundrum in the copyright protection of computer programs because in other copyright cases such a conundrum does not arise. As previously stated, in Football DataCo, if a work is *"dictated by technical considerations, rules or constraints which leave no room for creative freedom"*[369] it is not copyrightable. It would be hard to imagine that any computer program would be written without a programmer keeping the rules of the language or technical considerations in mind.

As the term "creative" is not defined in the EU *acquis* and is therefore vague, it could be said that both a poem and computer program are written creatively, however, it could be argued that the aim of the first is to elicit emotions in human beings while the aim of the second is to make a computer behave in a certain way. A programmer's freedom could be better described as the freedom to make choices of a functional nature in order to cause a computer to produce a desired result in the most efficient way. Needless to say, there is no need for efficiency when writing a poem. In Painer, a work is defined as an "*intellectual creation of the author reflecting his personality and expressing his free and creative choices in the production"* of that work.[370] Even though the expression of a programmer's *"free and creative choices"* would be an important criterion for the CJEU should it adjudicate in the Google v. Oracle case whether the idea and the expression of an idea have become indissociable in the declaring code, the first part of the Painer criterion which poses that a work is reflecting its author's personality clearly could not be applied to a programmer as an author. A programmer strives for efficiency while writing programs, which leaves no room for him to express his personality. The best programmers are those whose programs are written in the most efficient way possible, not those whose program is encumbered with the unnecessary content that would hinder a computer program from successfully completing a task. Clearly, the catch-all legal framework of

[369] CJEU, *Football Dataco Ltd and Others v Yahoo! UK Ltd and Others*, C-604/10, 1 March 2021, para 39, available at: https://curia.europa.eu/juris/liste.jsf?&num=C-604/10, last accessed on May 25, 2022
[370] CJEU, *Eva-Maria Painer v. Standard Verlags GmbH and Others*, 1 December 2011, para 94

copyright is a poor fit for computer programs as copyrighted works, offering neither sufficient protection nor incentive for development.

This calls for a reform of the copyright system in the U.S. and the EU as the problems and ambiguity that currently exist in the U.S. regarding the protection of computer programs, affecting both individuals and companies, represent a mirror image of the same issues existing in the EU which are yet to be tackled by the CJEU.

When widening the scope of consideration from issues faced by individuals or companies in the U.S. and the EU to a more global perspective, additional undesirable effects are noticeable, firstly the monopolisation of the global market by bad-faith actors is made only easier by the legal ambiguity, which allows for non-competitive practices such as *"embrace-extend-extinguish"*. Secondly, the ambiguity may lead IT companies to seek safe havens in permissive, if not more predictable jurisdictions, or apply undesirable practices when conducting business. These anti-competitive practices are already under scrutiny, and Big Tech companies are subject to investigations[371], fines[372] and new proposed legislation both in the U.S.[373] and the EU[374].

In the opinion of the author of this paper, copyright is burdened with years upon years of legal thought derived from the protection of other works, rendering it difficult to apply copyright to computer programs as a novel object of copyright protection. Creativity can only ever play second fiddle to efficiency when it comes to programming, yet when considering other works of authorship creativity is the first violin. *"The fact that computer programs are primarily functional makes it difficult to apply traditional copyright concepts in that technological world."*[375]

---

[371] *Antitrust: Commission opens investigation into possible anticompetitive conduct by Google in the online advertising technology sector*, EU Commission Press release, available at: https://ec.europa.eu/commission/presscorner/detail/en/ip_21_3143, last accessed December 15, 2021; see also *Investigation of Competition in Digital Markets, Majority Staff Report and Recommendations, Subcommittee on Antitrust, Commercial and Administrative Law of the Committee on the Judiciary*, the U.S. House of Representatives, available at: https://fm.cnbc.com/applications/cnbc.com/resources/editorialfiles/2020/10/06/investigation_of_competition_in_digital_markets_majority_staff_report_and_recommendations.pdf, last accessed December 15, 2021

[372] *Antitrust: Commission fines Google €1.49 billion for abusive practices in online advertising*, EU Commission Press release, available at: https://ec.europa.eu/commission/presscorner/detail/en/IP_19_1770, last accessed December 15, 2021

[373] Feiner, L.: *Lawmakers unveil major bipartisan antitrust reforms that could reshape Amazon, Apple, Facebook and Google*, available at: https://www.cnbc.com/2021/06/11/amazon-apple-facebook-and-google-targeted-in-bipartisan-antitrust-reform-bills.html, last accessed December 15, 2021

[374] The EU Commission: *Proposal for a Regulation of the European Parliament and of the Council on contestable and fair markets in the digital sector (Digital Markets Act)*, available at: https://eur-lex.europa.eu/legal-content/en/ALL/?uri=COM:2020:842:FIN, last accessed December 15, 2021

[375] SCOTUS, *Google LLC v. Oracle America, Inc*., Majority opinion, April 5, 2021, p. 35

The much-needed reform to the protection of computer programs cannot come about through the courts, on a case-by-case basis and through legal precedent. It should be kept in mind that the burden of the needed substantial change both in the U.S. and the EU cannot be left to the judicial system, as the courts are limited to a plaintiff's claim. While precedents are a major tool in the common law legal systems for developing legal thought and giving answers to the more nuanced legal questions of the law passed by the representative body, the U.S. courts cannot change the existing legal framework that serves as the basis for their decisions, and the same is true for the CJEU-issued preliminary rulings that serve to harmonise the EU *acquis*.

Solving this requires a thorough analysis of the current regulatory impact on the development of the software industry, public interest, and economy, among other things, before passing new regulations on the protection of computer programs. Perhaps the reason why the SCOTUS did not give the answer to whether declaring code is copyrightable or not, is because maintaining the *status quo* seems wise until the U.S. representative body once again forms a group of experts as it did in 1974 with the CONTU,[376] that would submit a new proposal on how to solve the software protection conundrum.

When creating new solutions for the protection of computer programs, finding the right balance between the galloping development of the IT industry and the protection of intellectual property is no easy task. Lawmakers should seek to enable the growth of innovation and services in the IT sector while levelling the playing field for both big and small participants in the IT market. One part of the solution lies in effective anti-trust legislation that is already under consideration in the U.S. and the EU. The other part lies in creating an appropriate legal framework for the protection of computer programs and providing tools for the defence of authors' rights, which are not only in theory, but also in practice equally accessible to every IT market participant.

The reform of software protection might come about as an extension to the current copyright framework on a national level, or on an international level, through an agreement regulating the copyright protection of computer programs. By this approach, the current grey areas of software protection could be addressed while still maintaining the legal fiction of the

[376] The National Commission on New Technology Uses of Copyrighted Works ("CONTU") proposal on the protection of programs with copyright was adopted and led to the amendment of the U.S. Copyright Act, ultimately making it the first law to give programs a copyright protection. For more information on CONTU, visit Digital Law Online, available at: http://digital-law-online.info/CONTU/contu1.html, last accessed December 15, 2021

copyrightability of computer programs despite them being, in essence, uncopyrightable methods of operation and processes. This may include the protection of software interfaces or other questions that were not addressed in this paper, such as *"[t]he loosening of decompilation regulations, shortening the term of protection of software to be more in line with rapid obsolescence and the short maintenance and support lifespan, and eligibility for application of orphan work provisions (…)."*[377]

While patents may, at first glance, offer an alternative solution to these issues, they are themselves fraught with ambiguities concerning computer programs, both within the EU and the U.S. legal systems,[378] the discussion of which is beyond the scope of this paper. Another possibility is the adoption of a 21st century, comprehensive *sui generis* solution to the protection of computer programs, a proposal discussed in several scientific papers.[379][380]

The author of this paper would wholeheartedly support an international initiative to gather and form a group of experts, which would devise a *sui generis* legal framework of computer programs protection for countries to adopt. The *sui generis* solution should take into account the importance of functionality and efficiency in computer programs and reward those who achieve the greatest degree of efficiency. The period of protection which is afforded to computer programs should be shortened, taking into account the rapid development and obsolescence of computer programs. The *sui generis* solution should level the playing field for every IT market participant. This would be well supplemented by the establishment of collective management organisations specifically for computer programs, both on a national and an international level. A model for these organisations could be the open source organisations, such as the FSF and OSI, that already manage licencing in a way similar to standard collective management organisations. [381] These would foster innovation and competition while providing protection to rightsowners, primarily small businesses and individual programmers.

Programmers and legal scholars worldwide hoped for the progress of software protection and some clarification and guidance. By not deciding on API copyrightability the

[377] Katulić, T., *op. cit*., fn. 2, p. 261
[378] Cupitt, P. L.; Thayer, L. J.: *A Comparison of the Patentability of Software in the United States and Europe: The key differences and a negative trend for computer-implemented inventions*, Computer Law Review International, Vol. 16, No. 3, 2015
[379] Vasudeva, V. N.: *A Relook at Sui Generis Software Protection Through the Prism of Multi-Licensing*, The Journal of World Intellectual Property Vol. 16, No. 1-2, 2013
[380] Flinders, M.: *Protecting Computer Software—Analysis and proposed alternative*, Journal of High Technology Law, Vol. 7, 2006
[381] Vasudeva, V. N., *op. cit.*, fn. 379, p. 94

SCOTUS missed an opportunity to lessen the legal entropy caused by many unanswered questions. However, this ruling just might have created enough momentum for the U.S. legislator to reform the legal framework for computer program protection. The efficient and progressive protection of software is not only of essence for the U.S., but to the whole world. The computer code we use on a daily basis knows no country, borders and already has the customs clearance. This creates a need for a progressive international treaty regulating the subject at hand. The Google v. Oracle case is a spark which may contribute to substantial changes to the protection of computer programs in U.S. law so that once again, the U.S. might be the driver of much-needed change on a global scale.

9. Abstract

This work aims to review the Google v. Oracle case and discuss the implications thereof on the legal protection of computer programs in the U.S. and the EU. Firstly, the basics of the Java and Android technologies, the history preceding the case, and the relevant U.S. legal doctrines and regulations are presented in short. Secondly, the case is reviewed by presenting the most important arguments of the parties and the diverging legal interpretation of both facts and law by the U.S. courts involved. Thirdly, the differences between the U.S. and the EU copyright systems are briefly presented to serve as the introduction to the analysis of how the EU courts might adjudicate such a case. Finally, in the conclusion, the current state of copyright protection of computer programs in the U.S. and the EU is analysed.

Keywords: Google, Oracle, Java, Android, API, declaring code, copyright, author's rights, fair use, Computer Programs Directive, Information Society Directive

Sažetak

Svrha ovog rada je razmotriti predmet Google protiv Oraclea i osvrnuti se na njegove implikacije na pravnu zaštitu računalnih programa u SAD-u te u EU-u. Prije svega su ukratko opisane osnove Java i Android tehnologija, povijest koja prethodi predmetu i važne pravne doktrine i propisi američkog prava. Nadalje, predmet se razmatra kroz prikaz najbitnijih argumenata stranaka te oprečnih tumačenja činjenica i prava od strane američkih sudova. Potom su ukratko opisane razlike između autorskopravnih sustava SAD-a i EU-a, što služi kao uvod u analizu mogućih odluka koje bi sudovi EU-a mogli donijeti u takvom predmetu. Konačno, u zaključku se razmatra trenutačno stanje autorskopravne zaštite računalnih programa u SAD-u te u EU-u.

Ključne riječi: Google, Oracle, Java, Android, API, Java deklaracija, copyright, autorsko pravo, pošteno korištenje, Direktiva o pravnoj zaštiti računalnih programa, Direktiva o određenim pravnim aspektima usluga informacijskog društva

10. References

a) Books and Articles

1. Cotton, I. W.; Greatorex, F. S.: *Data structures and techniques for remote computer graphics*, AFIPS (Fall, part I), 1968, available at: https://www.semanticscholar.org/paper/Data-structures-and-techniques-for-remote-computer-Cotton-Greatorex/060f4a49e8984f7b82efff4c5ca13166e0ee4811, last accessed December 15, 2021
2. Cupitt, P. L.; Thayer, L. J.: *A Comparison of the Patentability of Software in the United States and Europe: The key differences and a negative trend for computer-implemented inventions*, *Computer Law Review International*, Vol. 16, No. 3, 2015
3. Flinders, M.: *Protecting Computer Software—Analysis and proposed alternative*, Journal of High Technology Law, Vol. 7, 2006
4. Ghosh, R. A. (ed.): *Study on the: Economic impact of open source software on innovation and the competitiveness of the Information and Communication Technologies (ICT) sector in the EU*, UNU-MERIT, available at: https://szabadszoftver.kormany.hu/letoltesek/idegen-tanulmanyok/2006-11-20-flossimpact.pdf, last accessed December 15, 2021
5. Hayes, B. S.: *Integrating Moral Rights into U.S. Law and the Problem of the Works for Hire Doctrine*, Ohio State Law Journal, Vol. 61, Number 2, 2000
6. Hugenholtz, P.B.; Okediji, R. L.: *Conceiving an International Instrument on Limitations and Exceptions to Copy-Right*, Amsterdam Law School Legal Studies Research Paper No. 2012-43, Institute for Information Law Research Paper No. 2012-37, March 06, 2008
7. Katulić, T.: *Protection of Computer Programs in European and Croatian Law – Current Issues and Development Perspective*, Zbornik Pravnog fakulteta u Zagrebu, Vol. 65, No. 2, 2015
8. Level, P. N.: *Toward a Fair Use Standard*, Harvard Law Review, 1990
9. Menell, P. S.: *Rise of the API Copyright Dead?: An Updated Epitaph for Copyright Protection of Network And Functional Features of Computer Software*, Harvard Journal of Law & Technology, Vol. 31, Special Issue Spring 2018
10. Nimmer, D.: *"Fairest of them All" and Other Fairy Tales of Fair Use*, Law and Contemporary Problems, Vol. 66, 263-288, Winter 2003
11. Raymond, E. S.: *The Cathedral and the Bazaar*, O'Reilly, 1999
12. Vasudeva, V. N.: *A Relook at Sui Generis Software Protection Through the Prism of Multi-Licensing*, The Journal of World Intellectual Property, Vol. 16, No. 1-2, 2013
13. Wilkes, M.; Wheeler, D.; Gill, S.: *The Preparation of Programs for an Electronic Digital Computer*, Addison-Weasley Publishing Company, 1957

b) Regulations

1. *Agreement on Trade-Related Aspects of Intellectual Property Rights*, the World Trade Organisation, signed in Marrakesh, Morocco on 15 April 1994, available at: https://www.wto.org/english/docs_e/legal_e/27-trips.pdf, last accessed on December 15, 2021
2. *Berne Convention for the Protection of Literary and Artistic Works*, Paris Act, 1971, amended in 1979, available at: https://wipolex.wipo.int/en/text/283698, last accessed April 30, 2022
3. *Computer Software Copyright Act of the United States of the America*, 96th Congress, 1980
4. *Copyright Treaty*, the World Intellectual Property Organisation, signed in Geneva, Switzerland on December 20, 1996

5. *Council Directive 91/250/EEC of 14 May 1991 on the legal protection of computer programs*, the Council of European Communities, Official Journal L 122, 1991
6. *Directive 2001/29/EC of the European Parliament and of the Council of 22 May 2001 on the harmonisation of certain aspects of copyright and related rights in the information society*, the European Parliament and the Council of the European Union, Official Journal of the European Union, 2001, available at: https://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=CELEX:32001L0029:en:HTML, last accessed on December 15, 2021
7. *Directive 2009/24/EC of the European Parliament and of the Council of 23 April 2009 on the legal protection of computer programs*, the European Parliament and the Council of the European Union, Official Journal of the European Union, 2009, available at: https://eur-lex.europa.eu/legal-content/EN/ALL/?uri=CELEX%3A32009L0024, last accessed December 15, 2021
8. *Performances and Phonogram Treaty*, the World Intellectual Property Organisation, signed in Geneva, Switzerland on December 20, 1996
9. *Title 17 of the Code of Laws of the United States of America*, available at: http://uscode.house.gov/browse/prelim@title17&edition=prelim, last accessed December 15, 2021

c) Case Law

1. The Court of Justice of the European Union, *Bezpečnostní softwarová asociace – Svaz softwarové ochrany v Ministerstvo kultury*, C-393/09, 22 December 2010, available at: https://curia.europa.eu/juris/document/document.jsf;jsessionid=7B4FF3B6625A7E16FF16F3D02D27476A?text=&docid=83458&pageIndex=0&doclang=EN&mode=lst&dir=&occ=first&part=1&cid=510780, last accessed on April 30, 2022
2. The Court of Justice of the European Union, *Infopaq International A/S v. Danske Dagblades Forening*, C 5/08, available at: https://curia.europa.eu/juris/document/document.jsf;jsessionid=D204EF9A39546333B6F117DC48DE5D03?text=&docid=72482&pageIndex=0&doclang=EN&mode=lst&dir=&occ=first&part=1&cid=1250582, last accessed on April 30, 2022
3. The Court of Justice of the European Union, *UsedSoft GmbH v. Oracle International Corp*, C-128/11, available at: https://curia.europa.eu/juris/document/document.jsf?docid=124564&doclang=EN Last accessed April 30, 2022
4. The Court of Justice of the European Union, *Eva-Maria Painer v. Standard Verlags GmbH and Others*, C-145/10, 1 December 2011, available at: https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX%3A62010CJ0145, last accessed April 30, 2022
5. The German Federal Court of Justice ("Bundesgerichtshof"), *Möbelkatalog*, Case I ZR 177/13, available at: http://juris.bundesgerichtshof.de/cgi-bin/rechtsprechung/document.py?Gericht=bgh&Art=en&nr=71060&pos=0&anz=1, last accessed April 30, 2022
6. The United States Court of Appeals for the First Circuit, *Society of The Holy Transfiguration Monastery, Inc. v. Archbishop Gregory of Denver*, 2012
7. The United States Court of Appeals for the Federal Circuit, *Oracle America, Inc. v. Google, Inc.*, May 9, 2014, available at: https://law.justia.com/cases/federal/appellate-courts/cafc/13-1021/13-1021-2014-05-09.html, last accessed December 15, 2021
8. The United States Court of Appeals for the Federal Circuit, *Oracle America, Inc. v. Google, Inc.*, March 27, 2018, available at: https://law.justia.com/cases/federal/appellate-courts/cafc/17-1118/17-1118-2018-03-27.html, last accessed December 17, 2021

9. The United States Court of Appeals for the Ninth Circuit, *Atari Games Corp. v. Nintendo of America,* March1990, available at: https://casetext.com/case/atari-games-corp-v-nintendo-of-america, last accessed December 15, 2021
10. The United States Court of Appeals for the Ninth Circuit, *Disney Enterprises., Inc. v. VidAngel, Inc.*, 2017
11. The United States Court of Appeals for the Ninth Circuit, *Dr. Seuss Enterprises., L.P. v. Penguin Books USA, Inc.*, 1997, available at: https://casetext.com/case/dr-seuss-enterprises-lp-v-penguin-books-usa-inc, last accessed December 15, 2021
12. The United States Court of Appeals for the Ninth Circuit, *Fisher v. Dees*, 1986
13. The United States Court of Appeals for the Ninth Circuit, *Johnson Controls v. Phoenix Control Systems*, October 3, 1989, available at: https://casetext.com/case/johnson-controls-v-phoenix-control-systems, last accessed December 15, 2021
14. The United States Court of Appeals for the Ninth Circuit, *Monge v. Maya Magazines, Inc.*, 2012
15. The United States Court of Appeals for the Ninth Circuit, *Wall Data Incorporated, v. Los Angeles County Sheriff's Department, A Division Of The County Of Los Angeles; County Of Los Angeles*, 2006
16. The United States Court of Appeals for the Second Circuit, *Kregos v. Associated Press*, 1993
17. The United States Court of Appeals for the Second Circuit, *Computer Associates International, Inc. v. Altai*, 1992, available at: https://caselaw.findlaw.com/us-2nd-circuit/1233733.html, last accessed December 15, 2021
18. The United States Court of Appeals for the Third Circuit, *Whelan Assocs., Inc. v. Jaslow Dental Laboratory, Inc.*, 1986, available at: https://casetext.com/case/whelan-associates-v-jaslow-dental-laboratory, last accessed on December 17, 2021
19. The United States District Court for The Northern District of California, *Oracle America, Inc. v. Google Inc.*, May 31, 2012, available at: https://docs.justia.com/cases/federal/district-courts/california/candce/3:2010cv03561/231846/1202, last accessed December 15, 2021
20. The United States District Court for The Northern District of California, *Oracle America, Inc. v. Google Inc., Order Denying Rule 50 Motions*, June 8, 2016, available at: https://www.leagle.com/decision/infdco20160609998, last accessed December 15, 2021
21. The United States District Court for The Northern District of California, *Oracle America, Inc. v. Google Inc., Special verdict form, Jury Verdict - Document #1982.* May 26, 2016, available at: https://www.courtlistener.com/docket/4177532/1982/oracle-america-inc-v-google-inc, last accessed December 15, 2021
22. The Supreme Court of the United States, *Baker v. Selden*, October 1879, available at: https://www.law.cornell.edu/supremecourt/text/101/99, last accessed December 15, 2021
23. The Supreme Court of the United States, *Campbell, aka Skyywalker, et al. v. Acuff-Rose Music, Inc.*, March 1994. Available at: https://www.law.cornell.edu/supremecourt/text/510/569, last accessed December 15, 2021
24. The Supreme Court of the United States, *Feist Publications, Inc., Petitioner v. Rural Telephone Service Company, Inc.*, March 27, 1991, available at: https://www.law.cornell.edu/supremecourt/text/499/340, last accessed December 15, 2021
25. The Supreme Court of the United States, *Google LLC v. Oracle America, Inc.*, April 5, 2021, available at: https://www.supremecourt.gov/opinions/20pdf/18-956_d18f.pdf, last accessed December 15, 2021
26. The Supreme Court of the United States, *Harper & Row v. Nation Enterprises*, 1985, available at: https://supreme.justia.com/cases/federal/us/471/539/, last accessed December 17, 2021

27. The Supreme Court of the United States, *Sony Corp. of America v. University City Studios*, 1982

d) Filings

1. Department of Justice of the United States: *Brief for the United States as Amicus Curiae*, available at: https://www.supremecourt.gov/DocketPDF/18/18-956/117359/20190927165110897_18-956%20Google.pdf, last accessed December 15, 2021
2. Google, LLC: *GOOGLE INC.'S ANSWER to Complaint with Jury Demand, COUNTERCLAIM against Oracle America, Inc. by Google Inc*., filed by Google, Inc. on April 10,2010, available at: https://docs.justia.com/cases/federal/district-courts/california/candce/3:2010cv03561/231846/32, last accessed December 15, 2021
3. Google, LLC: *Google Inc.'s Answer to Plaintiff's Amended Complaint for Patent and Copyright Infringement and Amended Counterclaim*, filed by Google on November 10, 2010, available at: https://cases.justia.com/federal/district-courts/california/candce/3:2010cv03561/231846/51/0.pdf?ts=1376938370, last accessed December 15, 2021
4. Oracle America, Inc.: *Amended Complaint for Patent And Copyright Infringement*, filed by Oracle on October 27, 2010, available at: https://cases.justia.com/federal/district-courts/california/candce/3:2010cv03561/231846/36/0.pdf?ts=1288282135, last accessed December 15, 2021
5. Oracle America, Inc.: *COMPLAINT (with jury demand) for Patent and Copyright Infringement against Google Inc*., filed by Oracle America, Inc. on December 8, 2010, available at: https://docs.justia.com/cases/federal/district-courts/california/candce/3:2010cv03561/231846/1, last accessed December 15, 2021
6. 78 Computer Scientists: *Motion for Leave to File Brief of 78 Amici Curiae and Brief of 78 Amici Curiae Computer Scientists In Support Of Petitioner*, February 25, 2019, available at: https://www.supremecourt.gov/DocketPDF/18/18-956/89487/20190225134131839_18-956_Oracle_v__Google_Computer_Scientists_Amicus_Motion_Brief_FILE.pdf, last accessed December 15, 2021

d) Patents and Works Registered with the U.S. Copyright Office

1. Oracle, Inc.: *Java 2 standard edition 1.4.*, copyright registration in 2002, Copyright Catalog, the U.S. Copyright Office, available at: https://cocatalog.loc.gov/cgi-bin/Pwebrecon.cgi?Search_Arg=TX0006196514&Search_Code=REGS&PID=ssHB3uJGf4NsmyKlPRytdQcTqSm-NS&SEQ=20210627073715&CNT=25&HIST=1, last accessed December 15, 2021
2. Oracle, Inc.: *Java 2 standard edition, version 5.0. By Sun Microsystems, Inc., Comsys, the Carl Group, PrO Unlimited, ZAO Elbrus MCST, TelTech International Corporation*., copyright registration in 2004, Copyright Catalog, the U.S. Copyright Office, available at: https://cocatalog.loc.gov/cgi-bin/Pwebrecon.cgi?Search_Arg=TX0006143306&Search_Code=REGS&PID=55I_hpsI8vTGym6afeAw6FQ9j6REAg_&SEQ=20210627074330&CNT=25&HIST=1, last accessed December 15, 2021
3. Oracle, Inc.: Patent No. 6,192,476, *Controlling Access To A Resource* ("the '476 patent"), available at: https://patft.uspto.gov/netacgi/nph-Parser?Sect1=PTO1&Sect2=HITOFF&d=PALL&p=1&u=%2Fnetahtml%2FPTO%2Fsrc

hnum.htm&r=1&f=G&l=50&s1=6,192,476.PN.&OS=PN/6,192,476&RS=PN/6,192,476, last accessed December 15, 2021
4. Oracle, Inc.: Patent No. 6,910,205, *Interpreting Functions Utilizing A Hybrid Of Virtual And Native Machine Instructions* ("the '205 patent"), available at: https://patft.uspto.gov/netacgi/nph-Parser?Sect1=PTO1&Sect2=HITOFF&d=PALL&p=1&u=%2Fnetahtml%2FPTO%2Fsrchnum.htm&r=1&f=G&l=50&s1=6,910,205.PN.&OS=PN/6,910,205&RS=PN/6,910,205, last accessed December 15, 2021
5. Oracle, Inc.: Patent No. 5,966,702, *Method And Apparatus For Preprocessing And Packaging Class Files* ("the '702 patent"), available at: https://patft.uspto.gov/netacgi/nph-Parser?Sect1=PTO1&Sect2=HITOFF&d=PALL&p=1&u=%2Fnetahtml%2FPTO%2Fsrchnum.htm&r=1&f=G&l=50&s1=5,966,702.PN.&OS=PN/5,966,702&RS=PN/5,966,702, last accessed December 15, 2021
6. Oracle, Inc.: Patent No. RE38,104, *Method And Apparatus For Resolving Data References In Generate Code* ("the '104 patent"), available at: https://patft.uspto.gov/netacgi/nph-Parser?Sect1=PTO1&Sect2=HITOFF&d=PALL&p=1&u=%2Fnetahtml%2FPTO%2Fsrchnum.htm&r=1&f=G&l=50&s1=RE38,104.PN.&OS=PN/RE38,104&RS=PN/RE38,104, last accessed December 15, 2021
7. Oracle, Inc.: Patent No. 6,061,520, *Method And System for Performing Static Initialization* ("the '520 patent"), available at: https://patents.google.com/patent/US6061520A/en, last accessed December 15, 2021
8. Oracle, Inc.: Patent No. 6,125,447, *Protection Domains To Provide Security In A Computer System* ("the '477 patent"), available at: https://patft.uspto.gov/netacgi/nph-Parser?Sect1=PTO1&Sect2=HITOFF&d=PALL&p=1&u=%2Fnetahtml%2FPTO%2Fsrchnum.htm&r=1&f=G&l=50&s1=6,125,447.PN.&OS=PN/6,125,447&RS=PN/6,125,447, last accessed December 15, 2021
9. Oracle, Inc.: Patent No. 7,426,720, *System And Method For Dynamic Preloading Of Classes Through Memory Space Cloning Of A Master Runtime System Process* ("the '720 patent"), available at: https://patents.google.com/patent/US7426720, last accessed December 15, 2021

e) Web Sources:

1. Altexsoft: *What is API: Definition, Types, Specifications, Documentation*, available at: https://www.altexsoft.com/blog/engineering/what-is-api-definition-types-specifications-documentation/, last accessed December 15, 2021
2. Amadeo, R.: *Android N switches to OpenJDK, Google tells Oracle it is protected by the GPL*, available at: https://arstechnica.com/tech-policy/2016/01/android-n-switches-to-openjdk-google-tells-oracle-it-is-protected-by-the-gpl/, last accessed December 15, 2021
3. Anderson, T.: *Giving away software makes good sense for Sun*, available at: https://www.theguardian.com/technology/2008/feb/21/opensource.sunmicrosystems, last accessed December 15, 2021
4. The Android Open Source Project, visit: https://source.android.com/, last accessed December 15, 2021
5. The Apache Software Foundation: *Apache Harmony*, available at: http://harmony.apache.org/, last accessed December 15, 2021
6. The Apache Software Foundation, available at: https://www.apache.org/foundation/, last accessed December 15, 2021
7. The Apache Software Foundation: *Open Letter to Sun Microsystems*, available at: https://www.apache.org/jcp/sunopenletter.html , last accessed December 15, 2021

8. The Apache Software Foundation: *The ASF Resigns From the JCP Executive Committee*, available at: https://blogs.apache.org/foundation/entry/the_asf_resigns_from_the, last accessed December 15, 2021
9. ARN: *IN PICTURES: Remember this? The rise and fall of Sun Microsystems*, available at: https://www.arnnet.com.au/slideshow/334210/pictures-remember-rise-fall-sun-microsystems/, last accessed December 15, 2021
10. Baeldung: *What Is the Difference Between DVM and JVM?*, available at: https://www.baeldung.com/java-jvm-vs-dvm, last accessed December 15, 2021A.
11. Bansal, A.: *View Bytecode of a Class File in Java*, available at: https://www.baeldung.com/java-class-view-bytecode, last accessed December 15, 2021
12. Bergia, A.: *Stack Based Virtual Machines – 1*, available at: https://andreabergia.com/stack-based-virtual-machines/, last accessed December 15, 2021
13. Bloch, J.: *A Brief, Opinionated History of the API*, recorded at Q Con, August 8, 2018, available at: https://www.infoq.com/presentations/history-api/, last accessed December 15, 2021
14. Brodkin, J.: *Sun wanted up to $50 million from Google for Java license, Schmidt says*, available at: https://arstechnica.com/tech-policy/2012/04/sun-wanted-up-to-50-million-from-google-for-java-license-schmidt-says/, last accessed December 15, 2021
15. Brodkin, J.: *Newly revived patent gives Oracle extra ammunition in Google trial* Available at: https://arstechnica.com/tech-policy/2012/04/newly-revived-patent-gives-oracle-extra-ammunition-in-google-trial/, last accessed December 15, 2021
16. The Cambridge Dictionary: Meaning of taxonomy in English, available at: https://dictionary.cambridge.org/dictionary/english/taxonomy, last accessed December 15, 2021
17. Center for Strategic & International Studies: *Google vs. Oracle: Implications for Software Innovation?*[Video], uploaded by Center for Strategic & International Studies, April 29, 2021, available at: https://www.csis.org/events/google-vs-oracle-implications-software-innovation, last accessed April 30, 2022
18. Confluence: *The Apache Harmony library*, available at: https://cwiki.apache.org/confluence/display/harmony, last accessed December 15, 2021
19. DejaCode: *Linux Syscall Exception to GPL*, available at: https://enterprise.dejacode.com/licenses/public/linux-syscall-exception-gpl/, last accessed December 15, 2021
20. Devopedia: *Standard Library*, available at: https://devopedia.org/standard-library, last accessed December 15, 2021
21. Digital Law Online: *Final Report of the National Commission on New Technology Uses of Copyrighted Wo*rks, available at: http://digital-law-online.info/CONTU/contu1.html, last accessed December 15, 2021
22. Dignan, L.: *Oracle buys Sun; Now owns Java; Becomes a hardware player*, available at: https://www.zdnet.com/article/oracle-buys-sun-now-owns-java-becomes-a-hardware-player/, last accessed December 15, 2021
23. Duhaime.org: *Scenes A Faire Definition*, available at: https://www.duhaime.org/Legal-Dictionary/Term/ScenesAFaire, last accessed December 15, 2021
24. Elgin, B.: *Google Buys Android for Its Mobile Arsenal*, available at: https://web.archive.org/web/20060427095759/http://www.businessweek.com/technology/content/aug2005/tc20050817_0949_tc024.htm, last accessed December 15, 2021
25. Encyclopedia.com: *Program Specification*, available at: https://www.encyclopedia.com/computing/dictionaries-thesauruses-pictures-and-press-releases/program-specification, last accessed December 15, 2021

26. Encyclopedia.com: *The Great Internet Explosion*, available at: https://www.encyclopedia.com/science/encyclopedias-almanacs-transcripts-and-maps/internet-explosion, last accessed December 15, 2021
27. EUR-Lex: *Acquis*, available at: https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=LEGISSUM:acquis, last accessed April 30, 2022
28. European Commission Press release: *Antitrust: Commission fines Google €1.49 billion for abusive practices in online advertising*, available at: https://ec.europa.eu/commission/presscorner/detail/en/IP_19_1770, last accessed December 15, 2021
29. European Commission Press release: *Antitrust: Commission opens investigation into possible anticompetitive conduct by Google in the online advertising technology sector*, available at: https://ec.europa.eu/commission/presscorner/detail/en/ip_21_3143, last accessed December 15, 2021
30. European Commission: *Proposal for a Regulation of the European Parliament and of the Council on contestable and fair markets in the digital sector (Digital Markets Act)*, available at: https://eur-lex.europa.eu/legal-content/en/ALL/?uri=COM:2020:842:FIN, last accessed December 15, 2021
31. Fathima, S.: *Functional Programming VS Object Oriented Programming (OOP) Which is better….?*, available at: https://medium.com/@shaistha24/functional-programming-vs-object-oriented-programming-oop-which-is-better-82172e53a526 , last accessed December 15, 2021
32. Feiner, L.: *Lawmakers unveil major bipartisan antitrust reforms that could reshape Amazon, Apple, Facebook and Google*, available at: https://www.cnbc.com/2021/06/11/amazon-apple-facebook-and-google-targeted-in-bipartisan-antitrust-reform-bills.html, last accessed December 15, 2021
33. Frankel, B.: *Google v. Oracle: bad news for API developers*, available at: https://www.worldipreview.com/article/google-v-oracle-bad-news-for-api-developers, last accessed December 15, 2021
34. The Free Dictionary: *Embrace, extend, extinguish*, available at: https://idioms.thefreedictionary.com/Embrace%2c+extend%2c+and+extinguish, last accessed December 15, 2021
35. Free Software Foundation Europe: *Software Patents in Europe*, available at: https://fsfe.org/activities/swpat/swpat.en.html, last accessed on December 12, 2021
36. Foojay.io: *Technology Compatibility Kit*, available at: https://foojay.io/pedia/tck/, last accessed December 15, 2021
37. FOSSA: *All About Copyleft Licenses*, available at: https://fossa.com/blog/all-about-copyleft-licenses/, last accessed on April 30, 2022
38. FOSSA: *All About Permissive Licenses*, available at: https://fossa.com/blog/all-about-permissive-licenses/, last accessed December 15, 2021
39. GeeksforGeeks: *How is Java platform independent?*, available at: https://www.geeksforgeeks.org/java-platform-independent/, last accessed December 15, 2021
40. GeeksforGeeks: *Introduction of System Call*, available at: https://www.geeksforgeeks.org/introduction-of-system-call/, last accessed December 15, 2021
41. GeeksforGeeks, *What is DVM (Dalvik Virtual Machine)?*, available at: https://www.geeksforgeeks.org/what-is-dvmdalvik-virtual-machine/, last accessed December 15, 2021
42. Stricklett, S. G.: Google v. Oracle: *An Expansive Fair Use Defense Deters Investment In Original Content*, available at: https://www.ipwatchdog.com/2020/01/19/google-v-oracle-expansive-fair-use-defense-deters-investment-original-content/id=117951/, last accessed December 15, 2021

43. GNU: *GCC, the GNU Compiler Collection*, available at: https://gcc.gnu.org/, last accessed December 15, 2021
44. GNU: *GNU Classpath*, available at: https://www.gnu.org/software/classpath/, last accessed December 15, 2021
45. GNU: GNU General Public Licence, version 2, available at: https://www.gnu.org/licenses/old-licenses/gpl-2.0.html, last accessed December 15, 2021
46. GNU: *What is Copyleft?*, available at: https://www.gnu.org/copyleft/, last accessed April 30, 2022
47. Google Git: *android / platform / dalvik / 2ad60cfc28e14ee8f0bb038720836a4696c478ad / . / README.txt*, available at: https://android.googlesource.com/platform/dalvik/+/2ad60cfc28e14ee8f0bb038720836a4696c478ad/README.txt, last accessed December 15, 2021
48. Gomes, L.: *Sun Microsystems' Rise and Fall*, available at: https://www.forbes.com/2009/03/18/sun-microsystems-internet-technology-enterprise-tech-sun-microsystems.html, last accessed December 15, 2021
49. Greig, J.: *8 of the most popular programming languages*, available at: https://www.techrepublic.com/article/8-of-the-most-popular-programming-languages/, last accessed December 15, 2021
50. Hansen, C.: *Google v. Oracle: SCOTUS Sides with Google on Fair Use, But Is the Ruling Narrower Than It Seems?*, available at: https://www.jdsupra.com/legalnews/google-v-oracle-scotus-sides-with-4450778/, last accessed December 15, 2021
51. Hoffman, C.: *Android is Based on Linux, But What Does It Mean*? Available at: https://www.howtogeek.com/189036/android-is-based-on-linux-but-what-does-that-mean/, last accessed December 15, 2021
52. Horcasitas, J.: *What Is a Kernel?* Available at: https://www.digitalocean.com/community/tutorials/what-is-a-kernel, last accessed December 15, 2021
53. Hossain, A.: *C++ vs Java: Basic Comparison, Key Differences, & Similarities*, available at: https://hackr.io/blog/cpp-vs-java, last accessed December 15, 2021
54. Information Week: *Sun Microsystems Falls Victim To Internet Bust And Strategy*, available at: https://www.informationweek.com/sun-microsystems-falls-victim-to-internet-bust-and-strategy/d/d-id/1019496, last accessed December 15, 2021
55. InfoQ: *Apache Harmony Finale*, available at: https://www.infoq.com/news/2011/11/apache-harmony-finale/, last accessed December 15, 2021
56. Investopedia: *Business-to-Business (B2B)*, available at: https://www.investopedia.com/terms/b/btob.asp, last accessed December 15, 2021
57. Java.com: *Learn About Java Technology*, https://web.archive.org/web/20111124090716/http:/www.java.com/en/about/, last accessed December 15, 2021
58. Java.com: *What is the difference between the JRE and the Java SE platform?, What is the difference between the JRE and the JDK ?*, available at: https://java.com/en/download/help/techinfo.html, last accessed December
59. The Java Community Process: *Becoming a JCP Member*, available at: https://jcp.org/en/participation/membership, last accessed December 15, 2021
60. The Java Community Process: *Executive Committee Info*, available at: https://jcp.org/en/participation/committee, last accessed December 15, 2021
61. The Java Community Process: *Java Specification Requests Overview*, available at: https://jcp.org/en/jsr/overview, last accessed December 15, 2021

62. The Java Community Process: *JCP Procedures Overview*, available at: https://jcp.org/en/procedures/overview, last accessed December 15, 2021
63. The Java Community Process: *Overview*, available at: https://jcp.org/en/introduction/overview, last accessed December 15, 2021
64. Java T Point: *Difference between JDK, JRE, and JVM*, available at: https://www.javatpoint.com/difference-between-jdk-jre-and-jvm, last accessed December 15, 2021
65. Java T Point: *History of Java*, available at: https://www.javatpoint.com/history-of-java, last accessed December 15, 2021
66. Java T Point: *Java Interpreter*, available at: https://www.javatpoint.com/java-interpreter, last accessed December 15, 2021
67. Java T Point: *What is Java SE?*, available at: https://www.javatpoint.com/java-se, last accessed December 15, 2021
68. Kaffe: *The Kaffe Virtual Machine*, available at: http://www.kaffe.org/, last accessed December 15, 2021
69. Kaspersky IT Encyclopedia: *Closed-source software (proprietary software)*, available at: https://encyclopedia.kaspersky.com/glossary/closed-source/, last accessed April 30, 2022
70. Kaufman, J. R.: *What Google v. Oracle means for open source*https://opensource.com/article/21/5/google-v-oracle, last accessed December 15, 2021
71. Kirkpatrick, D.: *Meanwhile, Back at Headquarters... Scott McNealy Made Sun the Hottest Server Company. Now He's Got to Prove That Java's More Than Just a Good Buzz. His Toughest Job's The One Ahead*, Fortune Magazine, October 13, 1997, available at: https://archive.fortune.com/magazines/fortune/fortune_archive/1997/10/13/232511/index.htm, last accessed December 15, 2021
72. Known Host: *What Does "Unix-Like" Mean?*, available at: https://www.knownhost.com/blog/unix-like-mean/, last accessed December 15, 2021
73. Lane, K.: *Intro to APIs: History of APIs*, available at: https://blog.postman.com/intro-to-apis-history-of-apis/, last accessed December 15, 2021
74. Lane, K.: *Intro to APIs: What Is an API?*, available at: https://blog.postman.com/intro-to-apis-what-is-an-api/ , last accessed December 15, 2021
75. LeMonica, M.: *Sun picks GPL license for Java code*, available at: https://www.cnet.com/news/sun-picks-gpl-license-for-java-code/, last accessed on December 15, 2021
76. Legal Information Institute: *De Novo*, available at: https://www.law.cornell.edu/wex/de_novo, last accessed December 15, 2021
77. Legal Information Institute: *Motion For Judgment As A Matter Of Law*, available at: https://www.law.cornell.edu/wex/motion_for_judgment_as_a_matter_of_law, last accessed December 15, 2021
78. Li60.zendesk: *Class-based vs. prototype-based languages*, available at: https://li60.zendesk.com/hc/en-us/articles/115002448751-Class-based-vs-prototype-based-languages, last accessed December 15, 2021
79. Lowe, D.: *Methods and Method Declaration in Java*, available at: https://web.archive.org/web/20180616174802/http://www.dummies.com/programming/java/methods-and-method-declaration-in-java/, last accessed December 15, 2021
80. Mahanta, S.: *Disclosure-Dedication Rule: An Effective Tool Against Infringement Claims Under the Doctrine of Equivalents*, available at: https://www.ipwatchdog.com/2020/07/13/disclosure-dedication-rule-effective-tool-infringement-claims-doctrine-equivalents/id=123275/, last accessed December 15, 2021

81. Niccolai, J.: *Google: Sun offered to licence Java for $100M*, available at: https://www.computerworld.com/article/2509401/google--sun-offered-to-license-java-for--100m.html, last accessed December 15, 2021
82. Nordemann, J. B.: *Incidental Inclusion of Works – Mere Incidental Relevance of the Exception according to the German Bundesgerichtshof*, available at: http://copyrightblog.kluweriplaw.com/2015/09/24/incidental-inclusion-of-works-mere-incidental-relevance-of-the-exception-according-to-the-german-bundesgerichtshof/, last accessed April 30, 2022
83. The Open Handset Alliance: *Industry Leaders Announce Open Platform for Mobile Devices*, available at: http://www.openhandsetalliance.com/press_110507.html, last accessed December 15, 2021
84. The Open Handset Alliance: *Members*, available at: http://www.openhandsetalliance.com/oha_members.html, last accessed December 15, 2021
85. OpenJDK: *GNU General Public License, version 2, with the Classpath exception*, available at: https://openjdk.java.net/legal/gplv2+ce.html, last accessed December 15, 2021
86. Open JFX: *Java FX*, available at: https://openjfx.io/, last accessed December 15, 2021
87. Opensource.com: *What is open source?*, available at: https://opensource.com/resources/what-open-source, last accessed December 15, 2021
88. The Open University: *Open Source User Guide*, available at: https://www.open.ac.uk/library-research-support/sites/www.open.ac.uk.library-research-support/files/files/Open%20Source%20Software%20Licences%20User%20Guide%20V4%20-%20clean%20copy(1).pdf, last accessed December 15, 2021
89. Oracle: *Expressions, Statements, and Blocks*, available at: https://docs.oracle.com/javase/tutorial/java/nutsandbolts/expressions.html, last accessed December 15, 2021
90. Oracle: *Introduction to Java TM Technology*, available at: https://www.oracle.com/java/technologies/introduction-to-java.html, last accessed December 15, 2021
91. Oracle: *Java EE at a Glance*, available at: https://www.oracle.com/java/technologies/java-ee-glance.html, last accessed December 15, 2021
92. Oracle: *Java™ Platform, Standard Edition 6 API Specification,* available at: https://docs.oracle.com/javase/6/docs/api/index.html, last accessed December 15, 2021
93. Oracle: *Oracle Java SE Licensing FAQ, What is the licensing for currently available Oracle Java SE releases?*, available at: https://www.oracle.com/java/technologies/javase/jdk-faqs.html, last accessed December 15, 2021
94. Oracle: *Variables*, available at: https://docs.oracle.com/javase/tutorial/java/nutsandbolts/variables.html, last accessed December 15, 2021
95. Oracle: *Welcome to the Java Community!*, available at: https://www.oracle.com/java/technologies/javacommunity.html, last accessed December 15, 2021
96. Oracle: *What Is a Class*?, available at: https://docs.oracle.com/javase/tutorial/java/concepts/class.html, last accessed December 15, 2021
97. Oracle: *What Is an Interface?*, available at: https://docs.oracle.com/javase/tutorial/java/concepts/interface.html, last accessed December 15, 2021

98. Oracle: *What Is an Object?*, available at: https://docs.oracle.com/javase/tutorial/java/concepts/object.html, last accessed December 15, 2021
99. Oracle: *What Is Inheritance?*, available at: https://docs.oracle.com/javase/tutorial/java/concepts/inheritance.html, last accessed December 15, 2021
100. Paul, R.: *Android fragmentation: something to fear?*, available at: https://arstechnica.com/information-technology/2010/06/ars-explains-android-fragmentation/, last accessed December 15, 2021
101. Paul, R.: *Java wars: IBM joins OpenJDK as Oracle shuns Apache Harmony*, available at: https://arstechnica.com/information-technology/2010/10/ibm-joins-openjdk-as-oracle-shuns-apache-harmony/, last accessed December 15, 2021
102. PC Mag: *Interface*, available at: https://www.pcmag.com/encyclopedia/term/interface, last accessed December 15, 2021
103. Quinn, G.: *Patent Misuse, Exploring the Basics*, available at: https://www.ipwatchdog.com/2011/11/18/patent-misuse-exploring-the-basics/, last accessed December 15, 2021
104. The Rapid API Blog: *API vs Library (What's the Difference?)*, available at: https://rapidapi.com/blog/api-vs-library/, last accessed December 15, 2021
105. Red Hat: *What is a virtual machine (VM)?*, available at: https://www.redhat.com/en/topics/virtualization/what-is-a-virtual-machine, last accessed December 15, 2021
106. Rooney, P.: *Schwartz: Sun is world's largest open source company*, available at: https://www.zdnet.com/article/schwartz-sun-is-worlds-largest-open-source-company/, last accessed December 15, 2021
107. Rosenblatt, B.: *Moral Rights Basics*, Harvard Law School, available at: https://cyber.harvard.edu/property/library/moralprimer.html#:~:text=In%20the%20United%20States%2C%20the,of%20who%20owns%20the%20work, last accessed on April 30, 2022
108. Roulo, M.: *Java's three types of portability*, available at: https://www.infoworld.com/article/2076944/java-s-three-types-of-portability.html, last accessed December 15, 2021
109. Sass, R.: *Top 9 GPL With the Classpath Exception Questions Answered*, available at: https://www.whitesourcesoftware.com/resources/blog/top-9-gpl-with-the-classpath-exception-questions-answered/, last accessed December 15, 2021
110. Schwartz, J.: *The Rise of JAVA – The Retirement of SUNW*, available at: https://jonathanischwartz.wordpress.com/2007/08/23/the-rise-of-java-the-retirement-of-sunw/, last accessed December 15, 2021
111. Schwartz, J.: *Congratulations Google, Red Hat and the Java Community!*, available at: https://web.archive.org/web/20101023072550/http:/blogs.sun.com/jonathan/entry/congratulations_google, last accessed December 15, 2021
112. Schwartz, M.: *Reverse-Engineering*, available at: https://www.computerworld.com/article/2585652/reverse-engineering.html, last accessed December 15, 2021
113. Simply Coding: *Java Class Libraries and Packages*, available at: https://simplycoding.in/java-class-libraries-and-packages/, last accessed December 15, 2021
114. Shankland, S.: *Sun, Microsoft settle Java suit*, available at: https://www.cnet.com/news/sun-microsoft-settle-java-suit/, last accessed December 15, 2021
115. Sharpston, E.: *Opinion of Advocate General Sharpston in Case C 355/12 Nintendo Co. Ltd and Others v. PC Box Srl and 9Net Srl*, 19 March 2013, available at: https://eur-

lex.europa.eu/legal-content/GA/ALL/?uri=CELEX:62012CC0355, last accessed April 30, 2022
116. Sources Select Resources: *Copyleft*, available at: https://www.sources.com/SSR/Docs/SSRW-Copyleft.htm, last accessed on April 30, 2022
117. Stallman, R.: *Android and Users' Freedom*, available at: https://www.gnu.org/philosophy/android-and-users-freedom.html, last accessed December 15, 2021
118. Stallman, R.: *Free but Shackled – The Java Trap*, available at: https://www.gnu.org/philosophy/java-trap.html, last accessed December 15, 2021
119. Stallman, R.: *Why Open Source Misses the Point of Free Software*, available at: https://www.gnu.org/philosophy/open-source-misses-the-point.en.html, last accessed December 15, 2021
120. Stanford Encyclopedia of Philosophy: *Sorites Paradox*, available at: https://plato.stanford.edu/entries/sorites-paradox/, last accessed December 15, 2021
121. Stanford Law School press: *Supreme Court, Finally, Takes Up 'Google v. Oracle'*, available at: https://law.stanford.edu/press/supreme-court-finally-takes-up-google-v-oracle, last accessed on December 15, 2021
122. T-Mobile*: Industry Leaders Announce Open Platform For Mobile Devices*, available at: https://www.t-mobile.com/news/press/industry-leaders-announce-open-platform-for-mobile-devices, last accessed December 15, 2021
123. Technical Guidance: *Google wanted to license Java from Sun*, available at: https://technical-guidance.com/article/google-wanted-to-license-java-from-sun1619020366, last accessed December 15, 2021
124. Technology and IP Law Glossary: *Clean Room*, available at: http://www.ipglossary.com/glossary/clean-room/#.YMzVbfkzY2w, last accessed December 15, 2021
125. Techopedia: *Platform*, available at: https://www.techopedia.com/definition/3411/platform-computing, last accessed December 15, 2021
126. Techopedia: *Software Library*, available at: https://www.techopedia.com/definition/3828/software-library, last accessed December 15, 2021
127. TechTarget: *User Interface (UI)*, available at: https://www.techtarget.com/searchapparchitecture/definition/user-interface-UI, last accessed April 30, 2022
128. TechTerms: *Programming Language*, available at: https://techterms.com/definition/programming_language, last accessed December 15, 2021
129. Thakur, D.: *What is Application Program and Application Software*?, available at: https://ecomputernotes.com/fundamental/terms/application-program, last accessed December 15, 2021
130. The University of Iowa: *Prologue on Program Specification*, available at: https://homepage.cs.uiowa.edu/~fleck/spec.html, last accessed December 15, 2021
131. The United States Copyright Office: *Copyright Basics*, available at: https://www.copyright.gov/circs/circ01.pdf, last accessed December 15, 2021
132. The United States Copyright Office: *Duration of Copyright*, Circular 15A, available at: https://www.copyright.gov/circs/circ15a.pdf, last accessed April 30, 2022
133. The United States House of Representatives: *Investigation of Competition in Digital Markets, Majority Staff Report and Recommendations, Subcommittee on Antitrust, Commercial and Administrative Law of the Committee on the Judiciary*, available at: https://fm.cnbc.com/applications/cnbc.com/resources/editorialfiles/2020/10/06/investigation_

of_competition_in_digital_markets_majority_staff_report_and_recommendations.pdf, last accessed December 15, 2021
134. Upcounsel: *Equitable Defenses: Everything You Need to Know*, available at: https://www.upcounsel.com/equitable-defenses, last accessed December 15, 2021
135. Valdellon, L.: *What is an SDK? Everything You Need to Know*, available at: https://clevertap.com/blog/what-is-an-sdk/, last accessed December 15, 2021
136. Vance, A.: *Sun ditches its dot in dot-com slogan*, available at: https://www.computerworld.com/article/2796937/sun-ditches-its-dot-in-dot-com-slogan.html, last accessed December 15, 2021
137. Vaughan-Nichols, S. J.: *Linus Torvalds on Android, the Linux fork*, available at: https://www.zdnet.com/article/linus-torvalds-on-android-the-linux-fork/, last accessed December 15, 2021
138. Vredenburg, A. C.; Poindexter, A. A.: *No Justice for Unclean Hands*, available at: https://www.fosterswift.com/communications-no-justice-clean-hands-doctrine.html, last accessed December 15, 2021
139. Webopedia: *Fork,* available at: https://www.webopedia.com/definitions/fork/, last accessed December 15, 2021
140. Wong, W.: *Sun vs. Microsoft: Clash of the titans*, available at: https://www.zdnet.com/article/sun-vs-microsoft-clash-of-the-titans-5000121284/, last accessed December 15, 2021