WIRTSCHAFTS
UNIVERSITÄT
WIEN VIENNA
UNIVERSITY OF
ECONOMICS
AND BUSINESS

BIS Seminar Paper

Java 8 to Java 17: Overview, Changes, Outlook, Suggestions

Lukas Artwohl

Date of Birth: 24.07.2000
Student ID: 11844808

Subject Area: Information Business

Studienkennzahl: 033 561

Supervisor: Prof. Dr. Rony G. Flatscher

Date of Submission: 01.06.2022

Institute for Information Systems and Society, Vienna University of Economics and Business, Welthandelsplatz 1, 1020 Vienna, Austria

Declaration of Authenticity

I assure:

- to have individually written, to not have used any other sources or tools than referenced and to not have used any other unauthorized tools for the writing of this seminar paper.
- to never have submitted this seminar paper topic to an advisor neither in this, nor in any foreign country.
- that this seminar paper matches the seminar paper reviewed by the advisor.

Vienna on June 1st, 2022

(Lukas Artwohl)

Contents

Abstract

The importance of Java in computer science undisputed. Changes that happened within the programming language do get overlooked often times though. This paper will analyze how the language has changed, especially between version 8 and 17 while also giving a general overview of the language and how the language will evolve in the future. The Java Module System, the introduction of licensing costs as well as significant performance and security improvements are only a couple of those changes that show how much Java 8 differs from Java 17. Documenting these changes as well as pointing out the differences between the versions could be a good assistance for programmers trying to figure out which version to use.

1 Introduction

This seminar paper discusses the evolution of the Java programming language, and in particular how the language changed from Java version 8 to Java version 17. In addition to that the paper evaluates how the programming language could evolve in the future. This means that we will first get an overview of the programming language and its use cases in general and subsequently we will be comparing Java 8 and Java 17 as well as how the changes over the versions inbetween influenced the programming language.

1.1 Research Question

The research questions addressed in this paper are "How has the Java programming language changed between the version 8 and the version 17?" and "How will Java continue to grow and evolve in the future?". Because of these research question the main emphasis in this paper lies on the new features and functions that got added with different versions of Java and how these have influenced users decisions to use a certain version of Java. Additionally the paper also focuses on differentiating version 8 and version 17 in particular as well as what could be added to the programming language in the near future.

To get a coherent overview of the topic of Java there will also be subsequent research questions such as "What are the main points that define the Java programming language and what are its main use cases?", "Which features were added to the programming language over the years?" as well as "What are the main reasons of users to use a certain version of Java?". All of these questions are obviously needed to round up the topic of Java 8 to Java 17 and make everything more clear.

1.2 Research Method

This paper is based on existing literature about Java as well its official documentation and particularly the release notes of the different versions. The release notes are the main sources used to analyze the different versions and what make them stand out. Literature used in this paper contains both books as well as websites (and even blogs).

1.3 Structure of Thesis

This paper starts of by introducing the general aspects of the Java programming language (nevermind the versions). This means that we will first look

at features that make the programming language widely used and popular and subsequently we will analyze the main fields in which Java is used.

Thereafter we will be having a deeper look into version 8 of the programming language, one of the most popular, if not the most popular version to date. We will be looking at which features already existed as well as which new features added with Java 8 make the version so special.

The next section is concerned with analyzing the changes made to the language from version 9 up to version 17. Here we will be looking at each versions new as well as removed or deprecated features.

After having established those differences we will start to really analyze the main differences between Java 8 and Java 17 and why users are using one of those 2 versions.

Having completed the past of the programming language we will move over to the future of it. In particular we will be looking at upcoming features of new versions as well as suggestions on how the use cases of the programming language could be further expanded.

Finally we will conclude the paper and also give an outlook on what further research could be done for this topic.

2 Overview of the Java SE

In this chapter we will at first be looking at some details around the programming language and, subsequently, at the key features of the Java SE as well as its key use cases. As the Java SE is a programming language that has existed since 1995, it has by now grown to being used in many different fields. Hence why we will only go over the most significant use cases and features.

"Java is a high-level, class based, object-oriented programming language that is designed to have as few implementation dependencies as possible."[3] This means that the programming language contains natural language parts and is, in general, more understandable to humans and also has a much higher grade of abstraction. This grade of abstraction is also the reason why low-level languages do mostly run more efficient, as they are primarily designed to optimize the workflow of computers.[2]

The project of creating Java initially started in 1991 with the first public implementation happening in 1996. This was Java version 1.0 and it was back then called JDK 1.0. In the following years JDK 1.1 got released. Thereafter the following versions were renamed to J2SE 1.2, J2SE 1.3 and J2SE 1.4, which was released in 2002. In 2005 there was a jump to release J2SE 5.0. After this version it got once again renamed to Java SE, which is the current name of the programming language versions. The releases of the versions Java SE 6 to Java SE 9 were irregular. From Java SE 10, which got released in the September after the release of Java SE 9, on the releases of the Java SE versions happen biyearly in March and September. The Java SE versions which receive Long-Term Support are also chosen irregularly with Java SE 8, Java SE 11, Java SE 17 and Java SE 21 (which will be released in September 2023). The amount of time passing between the versions is 4 years between versions 8 and 11, 3 years between versions 11 and (most probably) 2 years between versions 17 and 21. It has now been fixed that long-time support versions are to be released every second year.[3]

With the first versions, which started with 1 all having relatively few new features and improvements between them, the versions from Java 5 onward all had much broader updates in comparison with the predecessor. This is also when the expansion of the programming language really started. Java 5 was also initially named Java 1.5 but later renamed to Java 5 because of these big updates as well as the much higher maturity.[36]

2.1 Key Features

We will now move on to some features and characteristics that are reasons for Java to be used in almost every sector of computer science.

2.1.1 Object-Orientation

First of all, as already mentioned earlier, Java is based on object-oriented programming (OOP). This means that everything that is implemented in Java is treated as an object. This means that both variables and methods are treated as objects. This means that all objects communicate in the same way and are also treated the same way. A method is for instance as much of an object as a class. Exceptions from being objects are primitive data types, because this increases the performance.[26]
Two other key principles of OOP are encapsulation, which allows access control. This means that we are able to only show what we want to who we want. The second principle is abstraction, which means that the program hides unnecessary information from the user and only shows what is really relevant. Other key features of OOP are Inheritance and Polymorphism.[26]

2.1.2 Platform independence trough Interpreters

Java instantly executes the high-level programming code without the need of compiling it into machine instructions. While using interpreters is typically slower than using compilers, compilers make up the need to convert the code into instructions that only fit the used architecture of the processor. This means that such compiler languages are less portable as they are "bound" to a processor architecture.[35]

What makes Java special in this case is, that it uses both the fast compilers as well as the flexible interpreters. This is accomplished by using bytecode, which is the intermediate stage between the high-level programming language and machine instructions.[3]

2.1.3 Memory management

Java has multiple ways of managing memory. The first, and most well-known one is the so-called garbage collector. Garbage collectors work in a way that they collect and thereafter get rid of objects which can no longer be addressed, because they are not linked to other objects anymore. This means that garbage collectors work throughout the runtime of a program.[3]
In addition to that Java also uses exception handling during runtime. When such an exception is thrown, it is immediately discarded, which means that

it will not be passed to the system but rather it terminates the execution. This is also very useful as it can avoid damaging programs or systems.[14]

2.1.4 Simplicity and Dynamism

The syntax of Java is based on C++, but is designed much easier in a way that it is more tidy. This means that parts that were needed in C++ were removed in Java. Examples for that are a pointer syntax or header files.[14] Java is also designed dynamically in a way that developers of libraries and the language itself can always add functions to the respective packages or the language itself without users needing to adapt running programs.[14]

2.2 Key Use Cases

With Java being such a versatile programming language there is also a plethora of use cases of the language. We will now go over the most renowned ones.

2.2.1 Applications

Java is generally used for most types of applications. These include web- and mobile applications, Enterprise applications and also desktop GUI (graphical user interface) applications.[7]

In web applications Java can be used with servlets to create easy to use applications. The most prominent user of Java when it comes to web development is Amazon.[7]

The most notable use case of Java in mobile applications and general development for mobile devices is the operating system Android. Android is based on Java and thus deeply tied with the language. Additionally the feature of platform independence comes into play here as there are many different systems and hardware used in mobile devices. Apps such as Uber, Google or Netflix use Java software.[7]

Another very important use case is with enterprise applications. As Java is such an extensive and fast programming language it is very often used to build ERP (enterprise resource planning) software. Such programs need to process a huge amount of data as well as handle many processes at the same time. [7]

Lastly the GUI applications use Java because of JavaFX. JavaFX is a platform for building such applications and is supported on Windows, Linux as well as MacOS and different web browsers.[37]

What is also important to note here is that applications used for programming, such as IntelliJ Idea, Eclipse and other big IDE's, are also based on Java.[7]

2.2.2 Big Data

The prevalent topic of Big Data also uses the Java programming language. The most notable use case here is for Apache Hadoop, which is a framework used in Big Data, which is completely written in Java. Furthermore Java is used in Apache HBase or ElasticSearch. The efficient memory management give Java the edge over other programming languages.[7]

2.2.3 Embedded Systems

A use case of which not many would think of as relevant in the first place are embedded systems. This is actually a fundamental part of current technology. The most important system here are sim-cards which are based on the Java programming language. It is obvious that society would not be the same without sim-cards and its Java basis.[7]

3 Java 8

In this chapter we will be looking into Java SE version 8, which was the first LTS version of Java and is one of the first versions which was used for a long time and is even still used despite being over eight years old.[36]

We will look into the general scope of this version as well as into the new features that got added to Java SE 8. As of Java 8 there is insufficient documentation regarding deprecated features and functions hence why we will not go over features that were deprecated or removed in Java 8.

3.1 Features in Java 8

As this version of Java was one of the largest updates at the time, we will first look into the already existent features and subsequently into the newly added features of version 8.

3.1.1 Already existent Features

Before Java 8 was introduced there were not many features added with new versions with the exception of Java 5. Here some notable added features were the data type "Generics" which allowed for better abstraction. This means that users could decide the final data type when instantiating the object. Java 5 also introduced the option to now add annotations for both classes and methods. Additionally Java 5 introduced enumerations which create ordered lists. From Java 5 on it was also possible to use a different syntax for for-loops. This syntax is defined as "for(:)" whereas the traditional for-loop (which can still be used) is defined as "for(int i=0;i<50;i++)" (as an example).[10]

In Java 6 there were no new features added, which shows how inconsistent the scope of updates was before adding LTS versions. Java 6 only changed or improved some libraries.[11]

In Java 7 there was a notable feature added, that improved readability of numbers. This means that it was now possible to add underscores to a numeric data type in order to break large numbers up into chunks.[12]

3.1.2 New Features

Here we will look at the most notable added features, which are Lambda expressions, improved Type Inference as well as some Annotations.[4]

Java 8 brought one of the most useful features that is still prevalent today (even in other programming languages, in python for instance such expressions are very widely used), namely Lambda expressions. These expressions allow the programming of functions as so called one-liners. This means that, when a simple function is only to be used once, it can be written and instantly called within one line of code. Such expressions can be written everywhere in Java.[33]

Additionally there was another feature that was improved, type inference. Type inference enables programmers to drastically reduce code needed to instantiate objects. The programming language will here decide on the data type itself based on how the variable is called. An example for that is that when an integer variable gets defined and subsequently another object is instantiated by adding this integer variable to another variable, Java will automatically know that both the sum of the two variables as well as the variable to be added to the integer have to be integers as well, as objects of different data types cannot be added to one another.[38]

Another big improvement to the programming language was the introduction of a sophisticated date/time API. Before Java 8 there only was a class for date which did not include times or time zones or similar. The new API improved features for dates and also introduced features such as the time in general as well as time zones, formatting of times as well as a class to better connect the time and the date class. Java 8 also introduced support for the newest Unicode version - 6.2.0.[13]

Furthermore a new related feature to that is an enumerate function for weekdays as well as month. These respectively have seven and twelve values representing the names of weekdays and months. In addition to that this also includes some basic operations for these enumerations.[13]

Java 8 also introduced some improved security features such as TLS (Transport Layer Security) 1.2 being enabled as default as well as a new version of an access controller which checks some code without going through the entire stack looking for permissions. Additionally Java 8 now also offers improved algorithms for the generation of stronger passwords.[5]

Java 8 also improved the high entropy random number generation. This is crucial for cryptography as truly-random or pseudo-random numbers with the highest-possible entropy are needed to create safe keys or passwords.[5]

Since JavaFX was still part of Java with version 8, we will also go over how this has been improved with this release.

First of all Java 8 introduced a new theme as well as new UI controls, namely DatePicker and TreeTableView. Furthermore has the 3D graphics library been expanded widely. This library now includes shapes, lights, camera, material and more features.[5]

In addition to that has the WebView class been improved to have better support for HTML5, the latest version of the hypertext markup language used in web development. Finally does JavaFX now offer compatability with ARM-based processors. This was not a big change at the time of the release but as of today are ARM processors becoming more and more relevant.[5]

We will no go over what are the main reasons Java 8 was so popular. Firstly Java 8 enables users to develop applications necessary to build IoT (Internet of Things) devices, as it is secure, scalable and flexible. Furthermore Java 8 also improved the productivity of programmers by implementing Lambda functions which can drastically reduce the code necessary to write functions. The new and improved data and time API also allows a more modern and state-of-the-art build of applications. JavaFX, which was at the time still part of the Java JDK also had a major overhaul to modernize the front-end development of applications. Lastly the introduction of long-time support versions was also a major reason for developers to move to Java 8, simply because older versions would not be updated and supported as long. This is even a reason why people choose Java 8 over subsequent, newer versions.[28]

A major reason to be using Java 8 instead of older versions is the introduction of an API that allows users to use multiple cores of their central processing units (CPU) at once whereas Java previously only used one core. This API is called Stream. Java 8 also features a class that is called Optional with which you can decide whether some variables need to be defined or are optional for running the code. This optional call will of course also reduce null pointer exceptions as such errors will no be thrown for optional variables.[32]

Lastly a main reason why Java 8 is still used over versions 11 or newer is that these new versions are now licensed and need to be paid for if they are used commercially. This is obviously a major downside if you are either a very small organisation that needs Java but also if you are a big enterprise that uses Java however not to the extent where it pays off to have probably (for large enterprises) very high licensing costs.[29]

While most companies could possibly switch to the open source version of Java, namely the OpenJDK, there are known differences regarding the support of the open source version. As well as that existed differences that seemed significant in older versions, but are not anymore significant, that companies probably have not yet completely realised. Lastly are there multiple open source versions where users could possibly lose track of which to pick or even thinking that changing from closed source to open source is coupled with heavy-impact changes.[29]

Coupled with this are the known troubles that can arise when updating Java. Many companies are not willing to take the risk of updating to a newer version which does not offer significant changes, as the possibility of having to completely overhaul a whole program is always existent. This is especially troublesome if companies keep up with the semi-annual update schedule of Java. The possibility of everything stopping to work twice a year is a risk almost no companies are willing to take.[34]

4 Changes up to Java 17

In this chapter we will look into features that were added between Java 8 and Java 17 and subsequently into features added in Java 17. After that we will also look at features that were deprecated or removed as of Java 17. Finally we will outline the main differences between Java 8 and Java 17 and why so many people are still using Java 8 despite version 17 being much more expansive and new.

4.1 New Features

Here we will go over the most significant new features of each version coming after Java 8 up to Java 17.

4.1.1 Java 9

Starting with Java 9, there were not many added features. The most notable one of these is the so-called Java Module System which was also referred to as "Project Jigsaw". Oracle Inc.[20] defined this modular system as follows:

> "The Java Platform module system introduces a new kind of Java programing [sic!] component, the module, which is a named, self-describing collection of code and data. Its code is organized as a set of packages containing types, i.e., Java classes and interfaces; its data includes resources and other kinds of static information. Modules can either export or encapsulate packages, and they express dependencies on other modules explicitly."

Additionally the try-with-resources statement was improved for final variables.[20]

4.1.2 Java 10

Java 10 was another version with many updates and added features.
To start with there was the adding of the "var" data type. While with this you can define any variable as var, Java still infers the data type the variable actually is. If a variable that is actually a String (i.e. a word) was defined as var Java would still define it as String. This means that this is purely for adding simplicity for coders. If someone would still want to define an integer as a String, they will need to implicitly use String as var would define the integer as an integer.[8]

In Java 10 there were also some added functions that prove to be very useful. To start we have a .copyOf function for each a list, a set as well as a map. This function creates an unmodifiable copy of variables of said types. Additionally before Java 10, JMX passwords were present in clear text. With the introduction of JDK 10 that changed in a way that now the hash value of this password will be present. Another thing connected to hash values is the introduction of better support for TLS (Transport Layer Security - a major tool used for e-mail communication) session hash as well as the corresponding master secrets.[21]

Furtheremore another improvement made was regarding the bytecode generation of for-loops. This allows a better translation from the Java Code used in the for-loops to a computer-readable language.[21]

4.1.3 Java 11 (LTS)

We will now look at the first long-term support version that followed the Java 8 long-time support version. Long-time support versions are usually coupled with plenty of new features but also quite a few features which were removed. These will however be examined in the following section "Removed or Deprecated Features".

A notable change here is that from this version of Java on, JavaFX is no longer part of the JDK but is now a separate download, the same applies for Java Mission Control.[22]

With the new Java 11 there also came support for newer versions of Unicode (versions 9 and 10). This change meant that there were now over 16,000 new characters. Some of those characters are fundamental for the 4K video format as well as Bitcoin. Java 11 also standardized a HTML client that was introduced in Java 9. This also means that a similar but older version of this client was removed.[22]

Java 11 also improved the String class by improving whitespace handling with the .strip, .stripleading and .striptrailing functions. Additionally there is now also a function that allows to check whether a string is blank, which checks if the variable has either no characters or only whitespaces. Furthermore there is now a function that allows to repeat a string a defined number of times. Lastly there is now also the option to stream a string through the Stream API.[9]

Another thing that is important to mention here is a .toArray function that overloads the previously existent function with the same name. This function differs to the older version in a way that it transforms a collection into an array whereas the older version takes an array as an argument. Lastly the Lambda expressions that were added in Java 8 can now be used with the var data type that was added in Java 10. Previously variables of the var type could not be used. A restriction here is that if one variable is of type var, then all variables need to be of this type.[22]

There were also a few security features or functions added with this version. First of all the signature algorithm RSASSA-PSS was added. In addition to that Java 11 also added support for a newer version of TLS v1.3. Furthermore there were 2 AES encryption standards added with this version.[22]

4.1.4 Java 12

Java 12 was introduced in March of 2019 as the (partly) successor of Java 11.[36]

With this version came support for the latest Unicode 11 version. This version meant that close to 700 new symbols would now be available in Java. Most of those symbols are however not relevant for new emerging technologies.[23]

Additionally Java 12 brought a new way of formatting numbers. From this version on it will be possible to format numbers much shorter. An example for that would be that 1 million can now be shown as "1M" instead of "1_000_000" (here the underlines represent the Java syntax for thousands separators).[23]

Java 12 also added some security features. First of all the Java Flight Recorder (JFR) received some new functions: It is now possible to record TLS handshakes with the JFR. It was from here on also possible to record modifications of security properties.[23]

Lastly this version also improved the switch statement. Using switch as an expression would for example shorten /simplify the code it is used on.[23]

4.1.5 Java 13

As for almost every new Java version, Java 13 introduces support for a new Unicode version, which is 12.1. This Unicode version added over 500 new characters. Java 13 also introduced a preview of text blocks. These are used to assign a string to a variable that is more than one line long. Such a vari-

able can be defined by using three double-quotes (""") both at the beginning and at the end of the string value. Previously such assignations were not easily possible. Before Java 13 we would have needed to use concatenations of multiple strings to assign such a long string. Java 13 also further improved the switch expressions which were introduced with Java 12.[24]

Another new feature that could be very useful for many is the ability to return unused memory to the operating system (OS) if more memory is needed elsewhere. Coupled with that Java 13 also introduced an option to set the maximum heap memory size beforehand by calling "-XX:SoftMaxHeapSize=<bytes>". This number of bytes can also be changed during runtime. Furthermore will Java also use more memory than allowed if necessary, to avoid errors caused by having too little memory. In addition to that the total maximum amount of memory has also been quadrupled from 4TB to 16 TB. These are all features that are coupled with the Z Garbage Collector (ZGC).[24]

Some added security features are for example that TLS certificates must now mach existent ones to allow establishing TLS connections via the TLS handshake. Java 13 updates also added 2 different certificates and 5 in total.[24]

While this update seems to not have brought many changes there have been a total of over 200 changes in documents of Java APIs and over 500 changes in contexts of those APIs. This shows that while Java 13 did not introduce many new features it still maintained and/or improved the already existent APIs.[15]

4.1.6 Java 14

Java 14 once again did not bring many new features or enhancements. The most notable one is the adaption of currency formats to fit accounting. This means that negative amounts can now be shown in brackets instead of with a leading -. This is the standard in accounting and thus a big improvement for people using Java for such purposes. Furthermore Java 14 introduced an experimental early version of the Z Garbage Collector for both Windows and MacOS. The ZGC has already been mentioned before with Java 13 and brings a few improvements in regard to efficiency with the programming language.[25]

As with every new Java there also came new security improvements. Java

14 disables some weak curves of protocols such as TLS, CertPath and Signed JAR, which are used for establishing safe connections with the help of cryptography. In total Java 14 disables 47 of those curves that were deemed weak. The Java Flight Recorder (JFR) mentioned in Java 12, is now able to constantly monitor activities due to the introduction of a new API. This is now activated as a default JFR setting.[25]

4.1.7 Java 15

We will now look into the new features added with Java 15, which is one of the bigger updates compared to recent ones. First of all Java now supports Unicode 13.0 which adds over 5,900 characters. This is one of the biggest updates of Unicode in Java since the programming language was created.[16]

With Java 15 also came a new function called isEmpty for character sequences. This function can, as the name implies, return boolean values based on whether a character sequence contains characters or not. Java 15 also introduced hidden classes. This means that when someone tries to get the class name "/" is returned unlike with normal classes. Additionally calling descriptorString will return a dot (".") instead of a description of such a hidden class. Lastly getNestMembers will not throw an exception when someone is not part of the members but rather return the name of the host as well as members' names.[16]

Java 15 also added text blocks, which allows users to input strings that are longer than one line and also automatically formats this block in a reasonable way.[16]

We will now look at the added security features and functions of Java 15. A new function that allows users to check whether a certificate has been revoked has been added to jarsigner and is called revCheck. Additionally jarsigner as well as keytool are now able to put out warnings if weaker algorithms are used in cryptography. Furthermore has the TLS signature configuration been expanded on both the client side and the server side. TLS 1.3 introduced the possibility to point out certificate authorities (CA) that are supported on different endpoints. This is once again also for both the client and the server side.[16]

4.1.8 Java 16

Java 16 introduced the option to access native code through a new Foreign Linker API. This is also coupled with another new API that allows Java to access memory that is not allocated to Java but rather to a different heap. All this improves binding to native libraries. Java 16 also made the encapsulation of the JDK's internal elements stronger which means that code containing calls to those elements could now fail.[17]

Java 16 also improved the Date Time API further by introducing the possibility to define time spans in a day, such as "in the morning" or "at evening". This adds to already existent am and pm, a way to define whether it is before or after midday.[17]

The ZGC was also once again improved with Java 16. ZGC is now capable of processing multiple thread stacks at the same time which also reduces pauses of the ZGC.[17]

We will now look at the added security features of Java 16. First of all it is now possible to sign JAR files using the RSASSA-PSS algorithm as well as the EdDSA algorithm. The SHA-3 algorithm is now also supported by the providers SUN, SunRsaSign, SunEC as well as SunPKCS11. Furthermore SunJSSE now supports TLS with the EdDSA algorithm. Additionally the maximum size of the TLS or DTLS handshakes has also been adapted to 32kb. The maximum length of a certificate in TLS and DTLS is now 10. The encoding of Application-Layer-Protocol-Negotiation values has also been improved within TLS. The sealed classes which were introduced in Java 15 have also been previewed again in Java 16.[17]

Java 16 also made pattern matching a feature for the instanceOf operator. This "allows common logic in a Java program to be expressed more concisely and safely, namely the conditional extraction of components from objects".[17]

4.1.9 Java 17 (LTS)

We will no move on to Java 17, the next long-time support version of the Java programming language and a key version for this paper.

The first new feature added are so called sealed classes, which were available as a preview feature in Java 15. Thereafter they were improved and once again available for preview in Java 16. With Java 17 now came the official release and introduction of these classes. Java 17 also introduced pat-

tern matching for switch expressions as well as more features for patterns as a preview. A new API was introduced for users to access large icons with higher quality as previously. This API is, as of Java 17, only optimized on Windows.[18]

Pseudo-random number generators and random number generators are of huge importance, not just for general coding but for cryptography in particular. Java 17 improved those pseudo-random number generators by adding new interfaces and implementations as well as new algorithms.[18]

Another important new feature, especially for ease of coding, is the improvement of error messages by adding the source of where the error was thrown. An important update is that Java 17 introduced an early access version of Java for ARM computer chips, prevalent in new Apple Macs. It is now also possible to convert primitive data types as well as byte arrays to the hexadecimal format. In addition to that there are also some functions to format those converted hexadecimal strings.[18]

We will now look at the added security features of Java 17.

First of all the SunJCE provider does now support the AES encryption algorithm for data encryption and decryption as well as key wrapping and unwrapping. SunPKCS11 now supports better handling of native resources through new attributes through which it can for example be specified how often tokens are destroyed. SunPKCS11 also added some encryption and key generation algorithms with Java 17.[18]

Java 17 also introduced configurations for TLS, namely being able whether to allow/enable extensions used within a client or even a whole server. This can however impact the TLS connection or even completely hinder it from establishing a connection.[18]

Java 17 is, not only because of being a long-time support version, a major update for the programming language. Java 17 brings many new features with previous updates rarely being this expansive.

4.2 Removed or Deprecated Features

In this section we will go over the features that were removed from Java 10 up to Java 17. Java 9 will not be shown here because of insufficient documentation of said removed features.

4.2.1 Java 10

With the first big update since Java 8 there were also a few features for which the support was omitted or which were completely removed.

First of all two function previously used for DOM (Document Object Model - a fundamental part of web development) manipulation, namely the "com.sun.java.browser.plugin2.DOM" and "sun.plugin.dom.DOMObject" have been removed and were replaced by a different function.[21]

The update also removed old versions of the "LookAndFeel" libraries which are used for GUI (graphical user interface) development.[21]

4.2.2 Java 11 (LTS)

As already mentioned earlier the older version of the HTML client was removed with Java 11. Another minor change was the removal of some fonts, the fonts from the Lucida family. A distinct thing that was also already mentioned earlier was the removal of Java Mission Control and JavaFX from the JDK. These are now separate downloads.[22]

4.2.3 Java 12

A class that was deprecated in Java 11, the SecurityWarning class, was removed in Java 12 as it was unused. Java 12 also removed a few file management methods, namely the finalize Methods for both the FileInputStream and FileOutputStream. These 2 methods were deprecated in Java 9 and removed in Java 12 while being replaced by the Cleaner method. In addition to that the finalize methods for ZipFile, Inflator and Deflator were removed after also being deprecated in Java 9. The removal of this method in regard to ZipFile, Inflator and Deflator will however result in throwing the Throwing exception when finalize were to be called.[23]

With an update of Java 12 there were additional removals of certificates as they expired since the initial version of Java 12 was released. These were 3 different types of root certificates, namely two DocuSign certificates, two Comodo certificates as well as a T-Systems Deutsche Telekom certificate.[23]

4.2.4 Java 13

In Java 13 there were some removals within the Runtime class. Namely the traceInstructions and traceMethodCalls methods. These have been removed as the functionality has been replaced by similar methods within the Java

Virtual Machine Tool Interface (JVMTI).[24]

There have also been some security changes with the introduction of Java 13: Some methods for RSA-cryptography have been removed because they have been replaced by a different class. These methods were only still in Java for compatibility reasons. This impacts 5 methods within the SunJSSE provider. Lastly some certificates, similar to those removed in older Java versions have also been removed with Java 13.[24]

4.2.5 Java 14

Java 14 removed an older garbage collector, namely the Concurrent Mark and Sweep (CMS) garbage collector. This could have been done because of the introduction of the new ZGC garbage collector.[25]

In regard to security Java 14 removed the security.acl API and its classes as well as it deprecated some more (elliptic) curves, now for the SunEC provider, which earlier provided similar functionality as the TLS, CertPath or Signed JAR curves that were also disabled as of Java 14. Oracle also states that some curves will be replaced by more modern ones.[25]

The updates of Java 14 have as usual removed some certificates due to them expiring. Among those are Comodo and DocuSign Root CA certificates.[25]

4.2.6 Java 15

Most features removed in Java 15 have been deprecated in Java 11 or subsequent versions. A JavaScript Engine called Nashorn has for instance been deprecated in Java 11 and removed here in Java 15. Furthermore Java 15 also removed 2 certificates of both the Comodo and DocuSign Root CA certificates. With Java 15 also some SunEC curves used for cryptography have been disabled because of being out of date. In total those curves disabled were a total of around 50 curves.[16]

4.2.7 Java 16

Java 16 removed 5 1024-bit RSA keys which were deemed weak. In addition to that a plethora of elliptic curves have once again been removed by the SunEC provider with Java 16.[17]

Overall there have not been many removals of features or features that were deprecated with Java 16 that were significant. Rather there have been

some more security features that were removed mostly because of being unused or unsafe. With updates of Java 16 there have once again been Root CA certificates that have been removed but also some certificates that have bee added.[17]

4.2.8 Java 17 (LTS)

Java 17 removed the option to strongly encapsulate internals of Java and will now do this without the option of removing the encapsulation with only a few exceptions. Furthermore the Java 17 updates also removed some root certificates, as usual. These contain a Google GlobalSign root certificate as well as a IdenTrust root certificate.[18]

Other than that there have only been little features or functions that have been removed hence why we will now look at deprecated ones.

First of all the Applet API has been deprecated since most browser do no longer support Java plug-ins. Java 17 also deprecated the security manager, one of the major deprecations/removals of recent Java versions.[18]

5 Key Differences Between Java 8 and 17

We will no go over the key differences between Java version 8 and version 17. As described in the previous section there have been a plethora of changes between those versions. This section will point out the most significant ones. We will first look at which features were added or removed and subsequently how the security of the programming language has improved between both versions.

Before looking at the technical differences between the versions we will talk about the other changes that happened between Java 8 and Java 17.

First of all a key reason for many people to still use Java 8 is the fact that using the Oracle JDK is no longer free to use for commercial causes. This changed with Java 11, where the relevant licensing model got introduced. While the open source version of Java, namely Open JDK, is still free to use, many people using Oracle JDK would presumably not want to change to the open source version for different reasons. Important to note here is that there are only little differences between the 2 versions, which are pretty much insignificant.[29]

Furthermore it makes sense that organizations keep using one version longer rather than keeping up with the very frequent new updates as this can completely change how programs run, which means a lot of maintenance for companies. This is one of the main reasons why many organizations keep or kept using Java 8 instead of moving to Java 15 or similar. While Java 11 was another version for which long-time support is provided, this version never became as popular as Java 8.[29]

Corporations did not even migrate to Java 9 even though this version brought significant new features (more on that in section 5.1) because the process of updating everything to work with version 9 was too much work for too little return for those companies.[34]

What we notice now is that people using Java 8 are moving to Java 17, probably because the many new features outweigh the possible difficulties coupled with switching versions.[34]

Another reason for that is that Oracle does no longer provide free public updates for the commercial version of Java 8. These updates expired in March of 2022.[36]

5.1 Features

We will now look into the most significant features that were added between Java 8 and Java 17 but also at those removed that had the biggest impact.

A first big update that was shortly mentioned above was the introduction of the Java Module System with Java 9. Even though this was a big update it was still not worth the update from Java 8 for most people though.[34]

Java 10 also brought a new feature that is vastly used in today's world of programming, namely the data type var, with which every variable can be assigned. Java will in the background determine the actual data type like integer or string.[8]

From Java 11 on JavaFX is no longer part of the JDK, this has however not much of an impact on users since JavaFX is still easily downloadable but as a separate product. Java11 also brought the option to trim whitespaces within strings, which is very useful for data cleaning. Furthermore there was a new HTTP client introduced, which was huge for web development.[22]

Java 12 and Java 13 as well as Java 14 did not bring new features that would on its own make most people switch from the LTS version 8 to either of those.

With Java 15 came the official introduction of text blocks for strings, which allow multi-line strings to be easily assigned to variables while also having good formatting of those blocks.[16]

Throughout the versions from Java 8 to Java 17 the support for newer versions of Unicode also came. Some of those versions were of significant importance for new technologies. Some versions for example introduced symbols that are needed to develop Bitcoin or the 4K resolution. This can mean that some programmers or organizations would need to update to newer versions in order to develop such technologies.[22]

Throughout the versions APIs like the Date time API which was introduced in Java 8, were further improved.[17]

Error handling and exception throwing was also improved. The Null-Pointer Exception was for example enhanced in a way that it now shows in which line the error was thrown. This is obviously significant for everyday coding.[34]

The switch expressions were also finally improved to make them usable again without being cumbersome as those were not enhanced for a long time at first.[34]

Java also introduced a new garbage collector, namely the ZGC (Z Garbage

Collector), which is much more efficient by reducing downtime of the garbage collector. Furthermore the ZGC as well as a second garbage collector (called G1) is also able to return unused memory to the OS. This is significant for the speed of the programming language.[34]

We will now look at the features that were deprecated and/or removed between Java 8 and Java 17.

First of all features that were removed from the JDK are JavaFX and also Java Mission Control. While those were removed, they are still available as a separate download, they are simply not a part of the JDK anymore.[22]

With Java 14 a garbage collector, namely the CMS (Concurrent Mark and Sweep) garbage collector, has been removed. This is however not a big impact since the ZGC and the G1 garbage collectors are more efficient.[25]

Finally, in Java 17 the security manager was deprecated, which means that it will be removed in future versions of the programming language. It is however still available to be used.[18]

5.2 Security

We will now look at how the security of the programming language has improved from Java 8 through Java 17. Improved security is a key reason for switching to the newest versions, especially since Java 8 does no longer receive regular free updates for the commercial version.

With Java 11 came support for the current TLS version, namely TLS v1.3. This is a major upgrade since TLS is used almost everywhere where internet is involved nowadays. Furthermore a new RSA signature algorithm as well as 2 AES standards were added with this version[22]

In Java 12, the Java Flight Recorder was expanded so that it can now record TLS handshakes along with further features.[23]

Java 13 was also improved in regard to TLS. From this version on TLS certificates must be verified by matching existent ones to establish TLS connections. Furthermore Java 13 removed some RSA methods because they have since been replaced.[24]

The Java Flight Recorder was further improved with Java 14 to monitor activities more consistently.[25]

While Java 15 further improved TLS, Java 16 introduced the SHA-3 algorithm through multiple providers. Furthermore the TLS and DTLS handshake size was improved to a maximum of 32kb.[16][17]

Finally we will look at the new security features of Java 17. With this version came support for AES en- and decryption through the SunJCE provider. Furthermore the features of TLS were further expanded and improved.[18]

With the new versions being released, as well as with the subsequent updates of the versions the security certificates were also introduced but also removed. This is obviously critical since expired certificates are big security holes. In addition to that a few versions also removed weak curves, which are used for key generation with different key generation algorithms. Such curves were removed in Java 14, 15 and 17.[25][16][18]

To conclude this section we can clearly see that for security reasons alone it would be smart to change to Java 17 or even the newest version available (which is currently Java 18).

Java also got significantly faster, and more efficient through Java 17 which is of huge importance as well, since programs get larger and larger. The added features do also add many handy ways to code more efficiently as well as code newer technologies.[34][22]

6 Outlook up to Java 21 (LTS)

In this section we will look at what will come or has already been released after Java 17. We will at first look at the next version that has been released, namely Java 18, and what this version has brought. Subsequently we will look at what is most likely to be changed within the programming languages up to the next long-time support version, which is Java 21. Finally we will look at some suggestions on how Java could further be improved to expand its use or enhance current use cases.

6.1 Java 18

We will now look at what Java 18 has brought but also which features were deprecated or removed, similar to how we analyzed the new versions in section 4.

First of all Java 18 now enabled the character set (charset) UTF-8 as its default for all Java APIs. This is significant in a way that the charset used is now standardized which makes the interoperability between APIs but also switching between them much less cumbersome. Furthermore has Java 18 also introduced a new web server which is supposed to be very easy-to-use and minimal. This is obviously important for the development of web applications. Java 18 has also brought another enhancement related to web development, it has introduced an interface to use name and address resolution services outside of Java's built-in ones.[19]

The Vector API was also introduced to "express vector computations that reliably compile at runtime to optimal vector instructions on supported CPU architectures, thus achieving performance superior to equivalent scalar computations".[19] This further enhances the performance of the programming language. Performance has also been improved with the help of further expanding multiple garbage collectors, namely the ZGC, the SerialGC as well as the ParallelGC, to now being able to handle string deduplication.[19]

Java 18 has also further improved in regard to security features by adding new APIs and expanding already existent ones. These include APIs from the SunPKCS11 provider.[19]

Java 18 has obviously also removed some features. There have once again been root certificates which have been removed. Other than that there have not been significant removals or markings for removal.[19]

6.2 Outlook

We will no go over what is most likely to be introduced with Java 19 as well as Java 20 and Java 21. While there is already a pretty good idea of what is to come in Java 19, the following versions are not yet well documented/talked about. Hence why we will focus more on Java 19.

The signs are looking good that Java 19 will finally officially publish features that have long been expected and awaited. Most of those features have until now only been in the testing or incubator stages in the previous versions. Furthermore will Java 19 also likely introduce anticipated features as previews to be released in subsequent versions.[27]

The first anticipated feature is the ability to use code outside the Java runtime which can be made possible by accessing memory not assigned to Java. This feature has already been in the incubator in previous versions and is expected to be officially released in Java 19.[27]

Furthermore the vector API which has already previously been mentioned with Java 18, could see its official release with Java 19. The same is possible for pattern matching with switch expressions, which has also undergone multiple previews up until Java 18.[27]

There is also some speculation regarding features that are not as likely as the previously mentioned ones, to be implemented in Java 19. These include mostly previews as the features pending for official release have already been talked about. First of all the ability to "unify the treatment of reference and primitive types in generic code by allowing Java type variables to range over both kinds of types"[27] which is referred to as universal generics could find its way into Java 19 as a preview. This feature as well as another feature probably coming, namely value objects, are both part of Project Valhalla.[27]

Java 19 could also improve one of its garbage collectors, namely the G1 garbage collector to reduce the latency of the garbage collector be introducing so-called "region pinning". This would further improve the performance of the programming language. Reasons to keep updating Java as a user.[27]

While it is possible that not all of those features will be introduced in Java 19, this could mean that they will be implemented in Java 20 or even Java 21. What is known is that these features will not be forgotten just because they did not come with Java 19.[27]

Unfortunately there are currently very few speculations that go beyond Java 19 as it is not even fixed what Java 19 itself will bring to the programming language.

6.3 Suggestions

In this section we will look at how Java could be enhanced or expanded to be more used in fields where the programming language has lost relevance but also how Java could be introduced to fields where only other programming languages like Python or C are used.

6.3.1 Mobile Development

We will start by analyzing why Java is struggling to keep up with different programming languages when it comes to mobile app development. Competitors for Android apps are mainly Kotlin. Kotlin was created by being compatible to Java 6. A main reason for the growing importance of Kotlin is Google, which is focusing on Kotlin instead of Java.[30]

A problem that Java has in comparison to Kotlin, that can however not be tackled is that Kotlin needs much less code than Java much like how Java needs less code than C or Python needs less than Java. This also makes the language easier to learn and to read. Kotlin, similar to Python, does not need semicolons, in contrast to Java. This is another reason that makes Java harder to read. Furthermore does Kotlin better handle the declaration and casting of data types by being a concise language.[30]

While Java is known for its wide compatibility and platform independence, Kotlin takes advantage of this by being used in parallel to Java and thus being able to use Java libraries. This is possible without much translation work or needing to know Java. In addition to that is Kotlin similar to Java while also being easier (as already mentioned earlier).[30]

One of the general big disadvantages of the Java programming languages are the so called null reference exceptions. Kotlin does not have this exception in most cases. While it is still possible to get such exceptions, for example through directly calling it, the possibility of it happening is slim to none.[30]

Java is of very little importance for mobile app development for iOS devices on the other hand. The reason for that is that Apple does not natively support Java. This means that a virtual machine would be needed to delevop such apps for iOS devices. While there are companies doing that, Java will most likely stay in limited relevance unless Apple will sometime enable Java. There is not much that can be done from the people developing Java to get more programmers to use Java for these apps.[1]

6.3.2 Possible Upcoming Use Cases

We will no look at emerging technologies where Java is not largely present and how this could change.

Data Science and Machine Learning
We will start by looking at how Java could gain importance in Data Science and why Python or even R are the preferred languages used or Data Science.

Once again is the complexity of Java, especially in comparison with Python, one of the main reasons of why Java is not as relevant as Python in this case. Python is not only much simpler but also much shorter and easy to learn for users. This is especially important in the case of machine learning, where better readable code plays a huge role.[31]

Many frameworks that are prevalent in Big Data are however based on Java. Apache Spark is for instance one of the main Big Data frameworks that is used in Python (where it is referred to as PySpark). Scalability plays a huge role with Big Data and this is where Java-based frameworks excel, in comparison to Python packages/libraries. Not only the scalability is an advantage here, Java is in general faster than Python.[31]

Here it becomes clear that really only the Syntax is the main disadvantage of Java in comparison to Python. So for people already knowing Java well and not knowing Python at all, it could definitely make sense to start building data science programs directly with Java, especially when it is foreseeable that the amount of data will need scalability, be it now or in the future.

Blockchain
Another emerging technology which will play a huge role in the near future is blockchain. “Blockchain is an open, distributed ledger that can record transactions between two parties efficiently, in a verifiable and permanent

way."[6]

Blockchain is currently mainly used by cryptocurrencies to document transactions with the respective currency. The idea of the distributed ledger can however be expanded to many fields where immutability of ledgers plays a big role. Even the security, which is coupled with the immutability and decentralization of blockchains, is enormous with this technology. More and more global players are starting to introduce it within their systems.[6]

As a blockchain is based on blocks that are interconnected to each other, an object-oriented programming language is needed to program this technology. As Java is one of the fastest programming languages and also has all the features needed for blockchain, it will definitely play a huge role when it comes to the development of blockchain. If the Java developers extend the programming language with a few APIs that directly help with programming this emerging technology, it can definitely establish itself as the main language used.[6]

7 Conclusion and further Research

The aim of this paper was to analyze the evolution of the Java programming language with an emphasis on the versions 8 and 17. The focus laid in particular on how these 2 versions differ and why people use the respective one.

We started by looking at the reasons for the popularity of Java such as the object-orientation, platform independence or even performance. Furthermore did it become clear that the use cases of the programming language are of vast importance in today's digital world. Applications, big data or embedded systems are all more important than ever.

Moving on to the analysis of Java 8 we found that the programming language at the time already had a huge number of features that are still prevalent today. We saw that most programmers needs were and still are satisfied by this version. The introduction of lambda expressions did allegedly bring one of the biggest new features added to date. This obviously further improved the likeability of this version. The fact that Java 8 also was the first long-time support version just added another reason for people to like this version.

By analyzing the subsequent versions of Java 8 we found that the general performance in regard to speed and compatibility increased largely. In addition to that did changes like the Java Modular System or new and improved garbage collectors (also influencing performance) coupled with a vast number of security improvements modernize the programming language even further. The introduction of support for the at the time newest Unicode versions enabled users to develop emerging technologies such as the cryptocurrency Bitcoin.

Another change to the programming system that happened between Java 8 and Java 17 was the introduction of a licensing model, urging users to keep using Java 8 as long as possible. The end of support for the version of Java 8 as well as the largely improved security portfolio that came since then will most probably mean the end of the popularity for version 8 in the near future.

The general future of the programming language does look quite good: There are big features to be introduced in the upcoming versions, most probably already in version 19, which will be released in September of 2022. Furthermore could Java establish itself as the main programming language used for blockchains. The growing importance of this technology will obviously also make Java even more popular.

This paper could be used as a basis to further analyze the described changes by showing their direct influence in Java code as well as a deeper look into different APIs that have changed the most since Java 8. Analyzing such changes more closely could also further encourage users of Java to change to Java 17 or a later version. In addition to that could the most notable features of Java be compared to other similarly popular programming languages, such as Python or C.

References

[1] CodeCondo. Is it possible to use java with ios? https://codecondo.com/is-it-possible-to-use-java-with-ios/. [Online; accessed 28-May-2022].

[2] Wikipedia contributors. High-level programming language — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Highlevel_programming_language&oldid=1078244053, 2022. [Online; accessed 6-April-2022].

[3] Wikipedia contributors. Java (programming language) — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.phptitle=Java_(programming_language)&oldid=1080629199, 2022. [Online; accessed 6-April-2022].

[4] Oracle Corporation. Java programming language enhancements. https://docs.oracle.com/javase/8/docs/technotes/guides/language/enhancements.html#javase8. [Online; accessed 30-April-2022].

[5] Oracle Corporation. What's new in jdk 8. https://www.oracle.com/java/technologies/javase/8-whats-new.html. [Online; accessed 30-April-2022].

[6] Roman Golovin. Java and blockchain â€“ a match made in heaven. https://kodejava.org/java-and-blockchain-a-match-made-in-heaven/. [Online; accessed 28-May-2022].

[7] Software Testing Help. What is java used for: 12 real world java applications. https://www.softwaretestinghelp.com/real-world-applications-of-java. [Online; accessed 20-April-2022].

[8] Chua Hock-Chuan. Jdk 10 new features. https://www3.ntu.edu.sg/home/ehchua/programming/java/JDK10_NewFeatures.html. [Online; accessed 07-May-2022].

[9] Chua Hock-Chuan. Jdk 11 (18.9)(lts) new features. https://www3.ntu.edu.sg/home/ehchua/programming/java/JDK11_NewFeatures.html. [Online; accessed 15-May-2022].

[10] Chua Hock-Chuan. Jdk 5 new features. https://www3.ntu.edu.sg/home/ehchua/programming/java/JDK5_NewFeatures.html. [Online; accessed 30-April-2022].

[11] Chua Hock-Chuan. Jdk 6 new features. https://www3.ntu.edu.sg/home/ehchua/programming/java/JDK6_NewFeatures.html. [Online; accessed 30-April-2022].

[12] Chua Hock-Chuan. Jdk 7 new features. https://www3.ntu.edu.sg/home/ehchua/programming/java/JDK7_NewFeatures.html. [Online; accessed 30-April-2022].

[13] Chua Hock-Chuan. Jdk 8 new features. https://www3.ntu.edu.sg/home/ehchua/programming/java/JDK8_NewFeatures.html. [Online; accessed 30-April-2022].

[14] Cay S. Horstmann. *Core Java, Volume I: Fundamentals*. Oracle Press, 12 edition, 12 2021.

[15] Oracle Inc. Api differences between java se 12 (build 32) & java se 13 (build 33). https://cr.openjdk.java.net/~iris/se/13/latestSpec/apidiffs/overview-summary.html. [Online; accessed 18-May-2022].

[16] Oracle Inc. Consolidated jdk 15 release notes. https://www.oracle.com/java/technologies/javase/15all-relnotes.html#JSERN15. [Online; accessed 19-May-2022].

[17] Oracle Inc. Consolidated jdk 16 release notes. https://www.oracle.com/java/technologies/javase/16all-relnotes.html#JSERN16. [Online; accessed 19-May-2022].

[18] Oracle Inc. Consolidated jdk 17 release notes. https://www.oracle.com/java/technologies/javase/17all-relnotes.html#JSERN17. [Online; accessed 19-May-2022].

[19] Oracle Inc. Consolidated jdk 18 release notes. https://www.oracle.com/java/technologies/javase/18all-relnotes.html. [Online; accessed 27-May-2022].

[20] Oracle Inc. Java platform, standard edition java language updates. https://docs.oracle.com/javase/9/language/toc.htm#JSLAN-GUID-B06D7006-D9F4-42F8-AD21-BF861747EDCF. [Online; accessed 07-May-2022].

[21] Oracle Inc. Jdk 10 release notes. https://www.oracle.com/java/technologies/javase/10-relnote-issues.html. [Online; accessed 15-May-2022].

[22] Oracle Inc. Jdk 11 release notes. https://www.oracle.com/java/technologies/javase/11-relnote-issues.html. [Online; accessed 15-May-2022].

[23] Oracle Inc. Release notes for jdk 12 and jdk 12 update releases. https://www.oracle.com/java/technologies/javase/12all-relnotes.html. [Online; accessed 17-May-2022].

[24] Oracle Inc. Release notes for jdk 13 and jdk 13 update releases. https://www.oracle.com/java/technologies/javase/13all-relnotes.html#JSERN13. [Online; accessed 18-May-2022].

[25] Oracle Inc. Release notes for jdk 14 and jdk 14 update releases. https://www.oracle.com/java/technologies/javase/14all-relnotes.html#JSERN14. [Online; accessed 18-May-2022].

[26] Simon Kendal. *Object oriented programming using Java*. Bookboon, 2009.

[27] Paul Krill. Java 19 could be big. https://www.infoworld.com/article/3652336/java-19-could-be-big.html. [Online; accessed 27-May-2022].

[28] Caroline Kvitka. 8 reasons to love java 8. https://www.forbes.com/sites/oracle/2014/03/26/8-reasons-to-love-java-8. [Online; accessed 18-May-2022].

[29] Framework Training Limited. Why is java 8 more popular than java 14? https://www.frameworktraining.co.uk/blog/why-is-java-8-more-popular-than-java-14/. [Online; accessed 18-May-2022].

[30] Disha Misal. 5 reasons why developers choose kotlin over java. https://analyticsindiamag.com/5-reasons-why-developers-choose-kotlin-over-java/. [Online; accessed 28-May-2022].

[31] ProjectPro. Java vs python for data science in 2022-what's your choice? https://www.projectpro.io/article/java-vs-python-for-data-science-in-2021-whats-your-choice/433. [Online; accessed 28-May-2022].

[32] Raoul-Gabriel Urma. Introducing java 8. https://www.oreilly.com/content/introducing-java-8/. [Online; accessed 18-May-2022].

[33] W3Schools. Java lambda expressions. https://www.w3schools.com/java/java_lambda.asp. [Online; accessed 30-April-2022].

[34] Dariusz Wawer. Java 17 features: A comparison between versions 8 and 17. what has changed over the years? https://pretius.com/blog/java-17-features/. [Online; accessed 24-May-2022].

[35] Wikipedia contributors. Interpreter (computing) — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Interpreter_(computing)&oldid=1078926915, 2022. [Online; accessed 6-April-2022].

[36] Wikipedia contributors. Java version history — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Java_version_history&oldid=1080703318, 2022. [Online; accessed 6-April-2022].

[37] Wikipedia contributors. Javafx — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=JavaFX&oldid=1064852607, 2022. [Online; accessed 20-April-2022].

[38] Wikipedia contributors. Type inference — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Type_inference&oldid=1082307610, 2022. [Online; accessed 30-April-2022].