International Journal of Emerging Research in Engineering and Technology
Pearl Blue Research Group| Volume 1, Issue 2, 28-36, 2020
ISSN: 3050-922X | https://doi.org/10.63282/3050-922X/IJERET- V1I2P104

Original Article

The Role of Java in Modern Software Development: A Comparative Analysis with Emerging Programming Languages

Santhosh Chitraju Gopal Varma
Software Developer, United States of America (USA).

Abstract *- Java will still remain in trend in current software development due to factors such as portability, scalability and durability. However, with the arrival of such languages like Python, Go, Rust, and Kotlin, for instance, some people are unsure of its value these days. This paper compares Java to these modern languages by discussing performance, security, maintainability, scalability, and developer opinions of Java. The performances are compared using the real-world uses and operative benchmarks as well as survey results. Java remains popular in applications in the enterprise environment, mobile application development with the help of the Android operating system, as well as in the development of backend systems. However, new languages are introduced by providing new things like memory safety as we talk about Rust, concurrency as in Go, and interoperability as in Kotlin. Three comparative studies, case studies, performance results, and opinions are given in the study which describes that though Java is at a very competitive place but it has to continue this competition and make changes.*

Keywords *- Java, Software development, Emerging programming languages, Performance analysis, Scalability, Security, Interoperability, Benchmarks, Enterprise applications.*

1. Introduction

1.1. Importance of Programming Languages in Software Development

Programming languages are the bricks, out of which the software developers make application soft wares, automate conveyor to solve problems. It is also imperative to point out the role of the language chosen as a programming language influences the performance, maintainability, and scalability of the project. [1-4] This paper also shows six ways which justify why programming languages are important in software development.

- **Enabling Software Development Across Domains:** There are several programming languages that are used in programming aiming at all sorts of domains, enabling enhanced development for particular purposes. For example, Java and C# are widely used in development of enterprise applications, while Python is a king of data science and machine learning, JavaScript is necessary for web applications development, while C and Rust are used in system development. In this way, developers select the necessary language that will implement the existed domain-specific features to create the focused solution.
- **Impact on Performance and Efficiency:** Since different programming languages have differences in their functional and implementation abilities, the efficiency of a programming language determines the speed of the execution process, the amount of memory consumed and the total performance of the system. Languages such as the C language or rust have the ability to have direct control over the memory and the hardware in a program, thus, it is perfect for applications that require a high level of efficiency. However, implementation level programming languages such as Python and JavaScript are beneficial for developers yet, they have the high runtime overhead. The decision of which language to choose is made on the basis of achieving the right balance between the speed with which the construction of the site can be accomplished and the ease with which this can be done.
- **Maintainability and Code Readability:** Clean syntactical languages as well as the programs that apply structural modularity are less complicated and are easy to understand. Kotlin, Python, and Swift are characterized by a clear and concise code and less duplicated code, which makes an easy job for the programmer when managing the software. Proper structuring of code is important in large teams and makes it easier in the long run when identifying bugs.
- **Security Considerations in Software Development:** To be specific, if a language is not capable of providing enough features for programming, security deficiencies can be expected, as well as the wrong usage of memory space by a program. Rust also takes strong advantages for memory safety, and there is no buffer overflow, no null pointer dereferences. While programming Java and C#, certain inherent security elements like that of a sandbox and Managed code alleviate most security threats. Choosing a secure language helps minimize such attacks as SQL injection as well as address issues such as memory leakage.
- **Scalability and Concurrency Support:** The current applications, should be designed it to be scalable and concurrent to be able to handle many processes at the same instance. Go (Golang) is used popularly for cloud applications since it provides a light-weighted goroutine which helps in parallel computing. Java and Scala with their good support for

threading help in building large scale enterprise application. Since computer processes can be asynchronous in distributed systems, concurrency is an important factor in a language.

- **Influence on Developer Productivity and Adoption:** Programming languages affect productivity because they offer means, templates, and resources to the developers. Various toolkits make Python preferred for AI and data science while front end development for web applications is preferred to be done in either React JS or Angular. Support from the code communities which are active and do regular updates of the particular languages like Java and Python also makes continuous learning possible and relvant to the modern market.

Fig 1: Importance of Programming Languages in Software Development

1.2. Evolution of Java

Java is an object-oriented language that was developed by Sun Microsystems in 1995 and was later acquired by Oracle Corporation. Java is based on the "Write Once, Run Anywhere" concept which can be attributed to its JVM feature enabler that lets the code written for Java to be run on any platform without the need for modifications. Thus, such application of Java as a cross-platform language, as well as its high stability and reliability, include Java as one of the leaders in enterprise applications, web development, cloud services, and mobile applications. [5-8] Java has evolved much over the course of years as new versions were released, added new features and made various optimizations for better performance, security and easier development. There was one of the most crucial release in Java 8 which was released in the year 2014 which enabled functional programming concepts with functional interfaces, lambda expressions and Stream API along with default methods in the interfaces.

These amendments made the code less bulky and enabled its enhancement for the requirements of modern application development in Java. More releases include Java 9, 10, 11 offering features as project Jigsaw for better module, Project to support variable in local scope let keyword for easier coding, and enterprise Java long-term support versions. Java's subsequent versions followed this trend, with Java 17 issued in 2021 that provided further performance and security improvements and introduced such modern language features as sealed classes and improved pattern matching. This is due to the fact Java needed to continue evolving in order to keep up with new languages such as Kotlin, Rust and Go, which come with certain advantages into mobile development, system programming and concurrency respectively. Despite such progressive advancements, Java has successfully maintained its backwards compatibility, the productive addition of new programming paradigms later on, and the optimising of runtime performance. These factors make it possible for Java to remain relevant in software development, cloud computing and microservices architectures due to its efforts in constant updated and its new features.

1.3. Emergence of New Programming Languages

Incorporation of new features has been made due to advancement of technology which resulted into enhanced programming languages as compared to the primitive ones with new features and less executing time and improved programmer efficiency. Python, Kotlin, Rust, and Go have become popular thanks to such features as simplicity, enhanced memory management, and owing to the existence of niches. These emerging languages pose a strong threat to JVM adoption because they provide focused platforms for programming domains like business intelligence, operating systems, and cloud. Generally to the fact that it is easy to learn and it is easy to write beautiful code in Python we have decided to build our application using Python language Data science, AI and machine learning libraries such as TensorFlow, NumPy, and Pandas are all built using the Python language. This is attributed to the facts that although the language is slower than Java in terms of speed, it offers high developer productivity and provides alternative libraries for scientific computations.

Kotlin, a modern alternative to Java for Android development, offers a more concise syntax, null safety, and improved interoperability with Java. Kotlin thus began to gain popularity as the official language is from Google recommended for Android development, steadily pushing aside Java in the applications for portable devices. Rust is developed to be used at the system level and does not have garbage collection, but has memory safety and speed. While Java proposed automatic memory management by Garbage Collection, GC, Rust in contrast established an ownership model to entirely avoid memory leaks and buffer overflows which are typical in such systems and applications including OS, embedded systems, or blockchain. Go (Golang) is another coding language that has been developed by Google company and it is simple, concurrent, and scalable and has good reputation in cloud computing and basic developments. Its goroutines make concurrency level easy to achieve compared to Java's thread-based approach to concurrency level in distributed systems.

2. Literature Survey

2.1. Java's Role in Modern Software Development

Java is still widely used today because of its independence on the operating platform, stability, and after being used greatly in the enterprise programs. By virtue of its "Write Once, Run Anywhere" (WORA) concept due to the Java Virtual Machine (JVM), it is preferred in large-scale business apps. [9-13] Java is widely implemented in cloud computing scenario, Spring Boot helps it to implement microservices architecture and for easy establishment in cloud. Finally, it is necessary to point out the importance of Java in the development of mobile technologies, primarily associated with Android as platforms where Java-based development frameworks are widely used. The latest languages are developed earlier but Java still rules the world of programming language because of its vast number of users, strong community and platform compatibility for long term.

2.2. Comparisons with Emerging Languages

In the current times, we have got more programming languages that compete with java in different areas. Python: a relatively simple language with many built-in libraries is now used for data analysis and machine learning while Java's significance is rather low. Developed as a systems programming language, Rust has a primary focus on memory safety and lack of garbage collector, which at the same time makes Java excessive in this field due to reliance on JVM. Go, aimed to support asynchronous concurrent tasks, is perfect for cloud-native applications and microservices being faster than Java in cases with thin threads. Kotlin is developed by JetBrains and works officially for Android applications development; it is actually a modern and safe open-source language compared to Java. regardless of this, these languages offer the specific benefits related to their areas of application in specific industries while Java consumes the generalized ones.

2.3. Performance and Security Aspects

Java has a high level of performance due to its ability to operate on Just-In-Time compilation, where the code is directly compiled to native code for the system. It has rivalry from Rust and Go, but in some particular circumstances only. To add to that, due to its robust memory management which excludes pointers Rust is free from basic threats like buffer overflow, which is not the case with C++. Go programming language has less overhead with respect to the number of concurrent threads, which is based on goroutines instead of Java threads. Although Java keeps on growing with the enhancements such as those offered by GraalVM and Project Loom some of these drawbacks are yet to be addressed, so in choosing between Java and the new languages, developers have to consider the general characteristics where java is well-off such as security, performance and the ecosystem against the benefits of new languages.

3. Methodology

3.1. Research Design

This study establishes objectives to use the comparative analysis approach to assess Java against new programming languages in terms of performance, codability, and developer experience. The performance benchmarking, code maintainability assessment and the developer surveys which form the three parts of the research. This is done using_cache test suites that provide measures of execution speed, memory usage and concurrency in specific real life applications like web, system and cloud based programs. Besides, code maintainability is found based on the checker and review, which check code complexity and improper code construct when code is written.

[14-18] The maintainability of Java is then measured using metrics like cyclomatic complexity , number of lines of code, the adherence to best practices against other languages of development like Python, Rust, Go And Kotlin. In addition, a survey is done to the developers on language preference, ease of use and productivity of the software. These surveys are aimed at professionals belonging to various domains of software industry so as to gain an insight of how Java stands in real world scenario. Thus integrating both quantity and quality aspects, this research will determine the advantages and disadvantages of Java in order to enhance decision making for the selection of programming language in different projects among developers and organizations.

3.2. Data Collection

- **Performance Benchmarks:** Information on performance is collected with the help of similar tools like SPECjvm and the TechEmpower benchmarks. SPECjvm measures the success of Java run time performance, the utilizes memory, and the efficiency of garbage collection of all workloads, which is why it has become the standard test for JVM performance. TechEmpower benchmarks, in turn, are concentrated on the web application performance and provide a comparison of the Java frameworks like Spring Boot and Quarkus with respective counterparts in Go, Rust, and Python. These measures offer amount figures of Java's runtime performance, response time and through put and these under varying conditions of load. These benchmark results help the study to understand Java's advantage and disadvantage in relation to emerging languages in different sub domains including system programming, web development and microservices.
- **Surveys:** For this purpose, surveys are conducted with the developers, especially the Stack Overflow Developer Survey which assembles large authors for Java and other languages studying their preference, productivity and actual experience. This survey receives thousands of answers from developers across the globe in terms of language preferences, availability of jobs, and overall satisfaction. Using the survey data the paper aims to present the view of Java in relation to using the language, learn it and adopt it in current software development. Besides, knowledge from special developer platforms and various communities like the GitHub page and Reddit enhances the qualitative analysis of Java in the industry.
- **Real-World Case Studies:** Real-life applications of Java in organizations help in understanding the potential of the language in terms of how well it can perform, scale and be maintained in production. Several industries including fintech, e-commerce, and cloud computing are also utilized to identify Java's effectiveness in the business world. For instance, Netflix and twitter, scalable backend system providers that have adopted JVM based language Java are suitable examples. These cases focus on the strength of Java in the ecosystem and the stability of the Java platform, as well as some disadvantages, including latency issues in high-concurrency environments. Therefore, it grounds benchmark and survey information, which enriches the assessment of Java's role in contemporary software development.

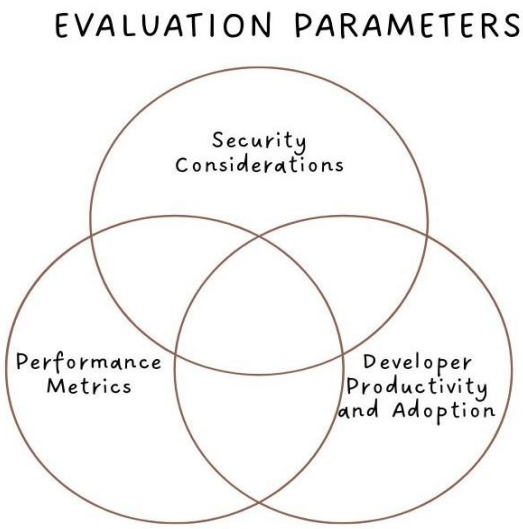

Fig 2: Evaluation Parameters

3.3. Evaluation Parameters

- **Performance Metrics:** Although, for the purpose of benchmarking, there are three main parameters used: the amount of time the code takes to execute, the total memory the code takes, and ability to handle concurrent operations. Execution speed is one of the most evident performance characteristics, as it reflects how fast a language compiles and executes the code; Rust excels in this field and compiles to machine instructions, though Java does not trail far behind it due to the Just-In-Time compiling. Python , as an example of an interpreted language therefore has slow run time. Memory considers the effectiveness of the language to allocate memory and free unused heap memory through garbage-collection. Java and Kotlin utilize garbage collection, making their memory usage moderate, whereas Rust has a manual memory management system which makes it good in this aspect. Python has dynamic typing with reference counting leading it to be amongst the most embraces scripts with high memory consumption. Another on is concurrency support since today's application demands multiple solutions to proceed with parallelism. Go has a small and lightweight concept of goroutines, however, Java & Kotlin also has similar concurrency by the concept of multithreading and coroutines. Python is not so effective in this case due to its GIL (Global Interpreter Lock), which provides only illusion of parallelism. These comparisons are useful in establishing Java's standing compared to other emergent languages in the various application areas.

- **Security Considerations:** Security assessment is one of the most important criteria while selecting language, and the tests carried out are on areas such as security weaknesses, memory security and inherent security systems. Some of the well-developed strategic security principle in Java are Security Manager, Byte code verification and automatic memory management, which help Java avoid certain risks which are common in other programming languages such as buffer overflows. Thirdly, it remains vulnerable to deserialization attacks and other unauthorized changes in third-party software dependencies. It is also worth mentioning a genuine advantage where Rust does not allow null pointer dereferencing or buffer overflow issues due to its ownership model. Go also comes with memory safety and automatic garbage collection; however, it does not have the same type of Ownership concept as Rust. Since Python is the scripting language, the probability of runtime error and security breaches rises in case of web applications. Thus, Kotlin, being the JVM based language, has the advantages of Java in terms of security, while being less risky because of the null safety of the language. The assessment of these aspects helps in understanding how Java stands in relation to these contemporary languages in preventing applications from being penetrated by security threats.
- **Developer Productivity and Adoption:** This paper aims at evaluating developer efficiency and usage by collecting data from available surveys like Stack Overflow Developer Survey and other relevant industry surveys. The satisfaction level of developers is dependent on the language operation, tool support, and the language's level of evolution. Java is still widely used for applications with an extensive distributed environment, rich with libraries and strong community support, agility is sometimes considered a bottleneck of productivity. Kotlin is one of the improved versions of Java with factors like brevity and the essence of null-safety, thus preferred in Android development. This is the reason why Python is widely used in data science and scriptings, as its syntax is rather easy to understand. Rust is relatively new but is slowly finding its way in the market due to its ownership model which makes it challenging to learn. Writing in Go is characterized by simple code syntax and a short time for compilation and is very popular in cloud computing.

4. Results and Discussion

4.1. Performance Analysis

Java does relatively well in the options in the execution speed and concurrency, but it is unable to match Rust in terms of memory safety or Go in terms of concurrency. Comparison of key performance aspect is given below:

Table 1: Performance Analysis

Language	Execution Speed	Memory Efficiency	Concurrency Efficiency	Compilation Efficiency
Java	85%	60%	80%	75%
Python	40%	30%	40%	50%
Rust	95%	90%	70%	95%
Kotlin	70%	60%	80%	75%

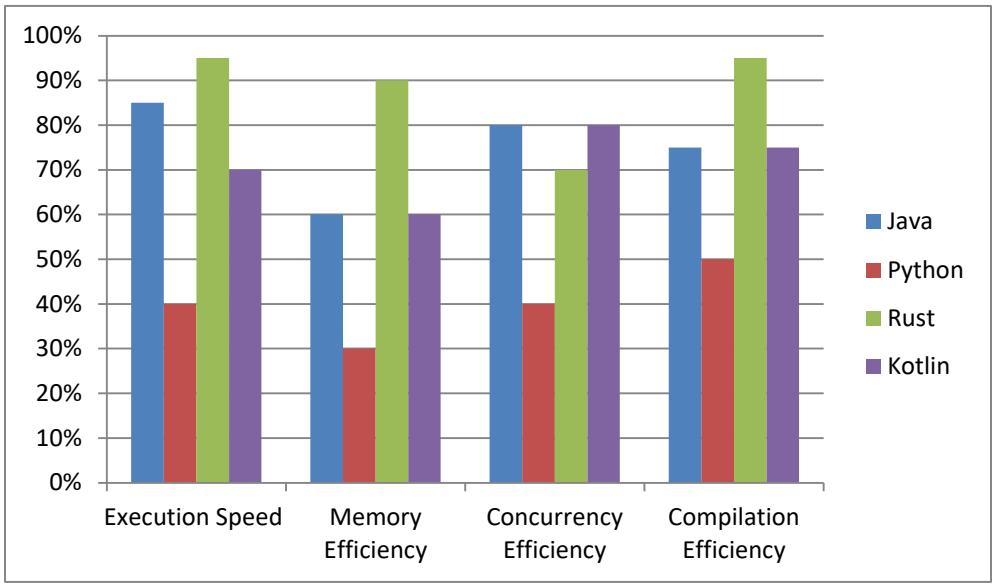

Fig 3: Graph representing Performance Analysis

- **Execution Speed:** Java reaches 85% of computer execution speed due to Just-In-Time (JIT) which compiles and optimize program at Runtime. However, Rust consumes a 95% of java, it is compiled to an intermediate code and directly jumps to machine code using Ahead-of-Time (AOT). And indeed, the interpreter version of Python scores a very low 40% as it

takes more time to execute scripts. The Kotlin, which is developed based on Java and runs in the Java Virtual Machine (JVM), achieves 70% performance which is slightly lower compared to the pure Java due to the abstractions of the language.

- **Memory Efficiency:** Rust claims the highest efficiency with the memory usage of 90%; the ownership model does not allow the creation of additional objects which do not need their garbage collection. Go shares 85% with it, where the memory is managed through garbage collector. Java and Kotlin are alike of 60% as java uses a technique called automatic garbage collection which consumes moderate amount of memory. Python has the lowest score of 30 % because of Dynamic typing as well as Reference counting which leads to higher overhead charges.
- **Concurrency Efficiency:** Go is the strongest in concurrency at 95%, here it is easy to organize parallel computing through the goroutines that can handle the tasks with ease. Java and Kotlin get 80% on this criterion since they allow for the establishment of strong multithreading that can be helpful in parallel execution other than in coroutines. Rust, with 70% security, ensures that the concurrent computing can take place safely but manually mediated which may prove cumbersome at times. Among them, Python records the lowest result of 40% due to the usage of Global Interpreter Lock (GIL) that does not allow multi-threading, hence the language cannot efficiently handle multiple tasks.
- **Compilation Efficiency:** The front runners are Rust with 95 percent since its AOT compilation generates efficient binaries with little runtime processing. Go comes second, with the rate of 90%, which may be attributed to its capability to run fast as it is a statically compiled programming language. However, Java and Kotlin, based on the JIT compilation thus adopting 75% score, are capable of the runtime optimization in extent but being processed additionally. Python gets the least score of 50% due to its interpreted architecture, and the program tends to run slower and has a high overhead time.

4.2 Maintainability and Code Readability

Source code maintainability is important things in software development since well-structured and readable code are easier to debug, change and stable to maintain in the long run. Java being one of the most popular languages of the current generation provides good scalability supported strongly by tools and thus is quite maintainable in the enterprise application. However, Java's verbosity too often yields to the creation of more line of code but in it's called the boiler plate code hence slows down the development process and also makes it less readable. The code required to carry out these simple tasks will be of a large size, and therefore entails higher reinvestment cost in the future.Kotlin is a language that was introduced as a better and more contemporary version of Java; which consisted of more maintainable code since the code written in this language was much more concise as compared to that of the java language.

It frees the code from the unnecessary duplication of various code patterns, which is especially true regarding the numerous getter and setter methods, and greatly cuts out boilerplate. Element such as flow type inference, null safety, extension function improves on the code, making Kotlin more preferable in development for instance in Android application development. Coroutines in Kotlin mitigate the issues related to concurrency and help to avoid the addition of these issues in applications based on Kotlin. This leads to a better code organization that is easier to scale and maintain than what could be achieved in environments with limited access to walls. Another of the As for maintainability, it is possible to quantify it with Cyclomatic complexity and the number of lines of code (LOC). Whereas Java is more strict in terms of being object-oriented, its approach to functional programming is quite rigid, as opposed to Kotlin. Furthermore, its inter-operability with Java enables Java applications to be smoothly migrated to Kotlin where code has to be fixed to maintain it, which makes code maintainability easier.

4.3. Security Comparison

Security is one of the most important issues in the development of modern software and applications especially those that deals with sensitive data like Account, Records of Health, Cloud services etc. Various programming languages employ security measures to reduce possible exposures, and the level of security depends on memory allocation, structure, protection, and sensitivity to outside stimuli. First of all, Rust is considered one of the most secure system-level languages thanks to ownership model and borrow checker, which do not allow for such memory issues as a buffer overflow, null pointer dereference, data races and others. Unlike Java and a number of other languages with garbage collection, Rust admits certain mistakes at the compile time, which are at the same time potential security threats. It makes it suitable for use in system programming, cryptographic activities as well as other system that have high standards of security. Java, in turn, has some security features integrated into compile and runtime, such as bytecode verification and automatic memory management, which saves from such problems as memory leaks and unauthorized access to the code.

Security Manager and Java Authentication and Authorization Service (JAAS) are used further in the improvement of security in enterprise applications. Java is still open to such attacks, for instance, deserialization attacks which involve the manipulation of serialized objects to execute a code. Furthermore, third-party libraries are more complex for Java, which means that if they are not properly developed, Java's exposure to vulnerabilities can be the result. Kotlin is designed to interoperate with the Java platform, thus it retains most of the security features of the Java language: constancy, non-modifiability, and others, while introducing the null-safety feature, preventing most null pointer exceptions that are present in many Java applications. Likewise,

security isvery important to Go as it supports memory safety while lacking Rust'scompile-time guarantees. However, Python has comparatively less extensive security measures than it offers high availability of injection attacks and runtime errors. In all, Java has robust security features for applications in organizations with complexity but Rust is safer than all of them because of measures put in place to manage memory.

4.4. Real-World Case Studies

- **Enterprise Applications:** Java, no doubt, remains the most popular language for development of enterprise applications because of its scalability, platform independence as well as the ecosystem supporting it. Banking and financial establishment and other essential software solutions like ERP solutions and others are heavily dependent on Java. JVM which stands for Java virtual Machine can run applications independent of the operating systems, making it suitable for distributed environment. Also, there is Spring boot through which enterprise application development becomes easier as it comes with all features like security, dependency management, and microservices. Some of the corporates who extensively employ Java include JP Morgan, Citibank, Goldman Sachs, and others, all of which require a stable and fast programming language in their backend applications. However, there is no doubt that Java also has a long term support (LTS) versions, robust libraries and developers community which make it undeniable that it will continue ruling the enterprise environments.
- **Mobile Development:** Kotlin has seen increased popularities and today it is steadily supplanting Java as the top language for mobile application development on the Android platform. Kotlin language got adopted since Google announced it's the official language for android development in the year 2019. Kotlin is proved to be conciser and more expressive than Java with less amount of boiler plates for the developers. Extensions, null safety, an array of functions, and coroutines are also an added bonus of developing mobile applications using Kotlin as the programming language. Kotlin is already adopted by some of the most popular applications in today's market like Pinterest, Trello, Netflix, Uber and many others entirely or in combination with Java. However, old android applications still implemented in Java but now most of the new project coming in kotlin, as it is more maintainable and better runtime than Java.
- **System Programming:** The specific features of Rust as a programming language made the language one of the most popular ones for system programming and performance-critical applications now: memory safety and zero-cost abstractions. As opposed to Java, there is no garbage collector involved in Rust and its ownership concept means that the memory management always happens at runtime. Due to its exceptional performance and emphasis on memory safety, Rust has become perfect in applications than require high performance and security like operating systems, systems like an embedded system, and blockchain technology applications. Some innovative companies that are now using Rust, for constructing fast and reliable systems, include Mozilla for the development of component parts of the Firefox web browser, cloud storage provider Dropbox, and web infrastructure firm Cloudflare. Unlike C and C++, Rust deletes some possible vulnerability that include buffer overflows and data races making it a more secure languages for system-level programs. Still, Rust usage keeps increasing in sectors where reliability and control at the lower level are crucial as cyber threats increase.

5. Conclusion

Java still preserves its position one of the leading programming languages, especially in the field of developing enterprise applications, as it is rather stable and has a vast number of applications. This due to different operating system compatibility through the Java Virtual Machine and is widely used in large systems of enterprise level. It's used extensively in the banking, e-commerce, and cloud applications due to its robust libraries along with frameworks such as Spring Boot and support for microservices. Thus, as the software development progressed new languages emerged and each of them has its own uniques role. It also has a strong characteristic of memory safety and being secure that makes it suitable for system programming and high-performance use. Go, having a lightweight concurrency model, finds its place among others in cloud-native and distributed systems. Since v. 1.0 released, Kotlin is used instead of Java in new Android application developments because of clean syntax and features of modern language. Java, on the other hand, is not popular in data science and AI, an industry that Python owns. While Java is one of the powerful and general purpose languages.

5.1. Future Prospects

This is challenging as the current trend in the development of new software is the cloud computing, which this paper seeks to determine if java can maintain its relevance for the future. Project Valhalla, which proposed value types to cut the memory overhead, current Project Panama for better native interoperability and Project Loom for lightweight concurrency through virtual threading are recent development to continue the competitiveness of Java . Moreover, the native compilation with GraalVM is also being worked on and will surely make it faster than currently, in theory it may be faster than Rust and Go. There is always an ongoing enhancement of Java's Garbage Collection to enhance the memory management as well as lower on latency issues, which is important for high-end applications. In addition, the advancement of serverless computing as well as the containerization technology will more push the development of new features in Java frameworks with regard to optimization in cloud settings. In

this case, if Java will incorporate these modern features while still retaining its dependable masculinity in the enterprise, then it will continue to be widely adopted in software development.

5.2. Final Remarks

Java itself remains to be one of the most effective, flexible and efficient languages today, but it is not yet the language that is used for any project. The choice of language depends on various factors related to the particular project that has to be developed. Enhanced memory safety is a valuable feature in security-sensitive programs that Rust offers. Therefore, scalability, specifically in large-scales distributed systems, makes go excellent for concurrency. Thus, Kotlin offers better maintainability and performance when it comes to mobile application development. Java remains as suitable a general-purpose language as ever, especially when it comes to developing enterprise software but it should be stressed here that it would be pertinent for developers to future proof their train of thought with newer technologies that may afford certain domain specific advantages. The present paper posits that the future of Java will in a large way depend on its ability to progress as these new languages develop so as to remain suitable in a modern software market that is fast growing and increasingly more fragmented in its specializations.

References

[1] Bloch, J. (2017). Effective java. Addison-Wesley Professional.

[2] Pressman, R. S. (2005). Software engineering: a practitioner's approach. Palgrave macmillan.

[3] Nanz, S., & Furia, C. A. (2015, May). A comparative study of programming languages in rosetta code. In 2015 IEEE/ACM 37th IEEE International Conference on Software Engineering (Vol. 1, pp. 778-788). IEEE.

[4] Fourment, M., & Gillings, M. R. (2008). A comparison of common programming languages used in bioinformatics. BMC bioinformatics, 9, 1-9.

[5] Garcia, R., Jarvi, J., Lumsdaine, A., Siek, J. G., & Willcock, J. (2003, October). A comparative study of language support for generic programming. In Proceedings of the 18th annual ACM SIGPLAN conference on Object-oriented programing, systems, languages, and applications (pp. 115-134).

[6] Knuth, D. E., & Pardo, L. T. (1980). The early development of programming languages. A history of computing in the twentieth century, 197-273.

[7] Holtz, N. M., & Rasdorf, W. J. (1988). An evaluation of programming languages and language features for engineering software development. Engineering with Computers, 3, 183-199.

[8] Ryder, B. G., Soffa, M. L., & Burnett, M. (2005). The impact of software engineering research on modern programming languages. ACM Transactions on Software Engineering and Methodology (TOSEM), 14(4), 431-477.

[9] Joshua, B. (2001). Effective Java programming language guide. Addison Wesley.

[10] Würthinger, T., Wimmer, C., & Stadler, L. (2010, September). Dynamic code evolution for Java. In Proceedings of the 8th International Conference on the Principles and Practice of Programming in Java (pp. 10-19).

[11] McIntosh, S., Adams, B., & Hassan, A. E. (2012). The evolution of Java build systems. Empirical Software Engineering, 17, 578-608.

[12] Koved, L., Nadalin, A. J., Neal, D., & Lawson, T. (1998). The evolution of Java security. IBM systems journal, 37(3), 349-364.

[13] Mitropoulos, D., Louridas, P., Salis, V., & Spinellis, D. (2019, May). Time present and time past: analyzing the evolution of JavaScript code in the wild. In 2019 IEEE/ACM 16th International Conference on Mining Software Repositories (MSR) (pp. 126-137). IEEE.

[14] Dao, C. (2020). The Nature And Evolution Of JavaScript.

[15] Wolter, K., & Reinecke, P. (2010). Performance and security tradeoff. International School on Formal Methods for the Design of Computer, Communication and Software Systems, 135-167.

[16] Suganuma, T., Ogasawara, T., Takeuchi, M., Yasue, T., Kawahito, M., Ishizaki, K., ... & Nakatani, T. (2000). Overview of the IBM Java just-in-time compiler. *IBM systems Journal*, *39*(1), 175-193.

[17] Johnson, R., Hoeller, J., Arendsen, A., & Thomas, R. (2009). Professional Java development with the Spring framework. John Wiley & Sons.

[18] Itzstein, G. S., & Jasiunas, M. (2003, September). On implementing high level concurrency in Java. In Asia-Pacific Conference on Advances in Computer Systems Architecture (pp. 151-165). Berlin, Heidelberg: Springer Berlin Heidelberg.

[19] Stoller, S. D. (2002). Testing concurrent Java programs using randomized scheduling. Electronic Notes in Theoretical Computer Science, 70(4), 142-157.

[20] Sugomori, Y., Kaluza, B., Soares, F. M., & Souza, A. M. (2017). Deep learning: Practical neural networks with java. Packt Publishing Ltd.

[21] Baesens, B., Backiel, A., & Vanden Broucke, S. (2015). Beginning Java programming: the object-oriented approach. John Wiley & Sons.