International Journal of Scientific Engineering and Science
Volume 8, Issue 5, pp. 52-55, 2024. ISSN (Online): 2456-7361

Modern Approaches to Backend Development in Java and Kotlin: A Comparative Analysis

Glumov Konstantin
Senior Software Engineer, Alfa-Russia, Perm
Email address: kglumov@alfabank.ru

***Abstract*—** *The study is devoted to a comparative analysis of modern approaches to backend development in Java and Kotlin. The main characteristics and innovations introduced into the programming world with the advent of Kotlin from JetBrains in 2011 are considered, as well as an overview of Java as an established development tool introduced by the Green Team in 1995. The work is based on an analysis of the range of Kotlin applications, its flexibility, security, and multiplatformity, in comparison with the object-oriented, multiplatform approach and Java multithreading. The study aims to identify the key advantages and disadvantages of both programming languages from the point of view of backend development, taking into account aspects such as syntactic efficiency, security, integration, and versatility*.

***Keywords*—** *Backend, backend development, Java, Kotlin, programming languages, programming, IT*.

I. Introduction

In the era of digital transformation and increasing demands for software functionality, security, and scalability, choosing the right programming language becomes a key factor for project success. Java, being one of the most popular and time-tested programming languages, has long dominated backend development due to its versatility, powerful tools, and extensive community. However, the introduction of Kotlin in 2011 from JetBrains brought a significant change to the technology selection landscape. Designed to improve performance and ease of development, Kotlin quickly gained acceptance among programmers due to its modern features such as shorter and more expressive syntax, enhanced security, and interoperability with Java.

The purpose of this paper is to conduct a comparative analysis of Java and Kotlin in terms of their application in backend development. The historical development of both languages, their main characteristics, advantages, and disadvantages, as well as the spheres of application in modern software development are considered [1,2].

The research is based on literature analysis, a comparison of existing research papers, and a study of practical examples of development in Java and Kotlin. The paper utilizes data from official sources, including JetBrains and Green Team documentation, as well as analytical reports and studies conducted in programming. The methodology includes qualitative analysis of syntactic constructs, type safety, multithreading, generality, and other critical aspects of programming in both languages.

II. Literature Review

The widespread use of Kotlin for its flexibility, security, and multiplatform nature, and Java's well-established position in the programming world for its multithreading, object-oriented, and code portability are noted. Scholars such as Blinova A. V., emphasize the ease of perception and great flexibility of Kotlin, while other researchers point to Java's established position as a robust and scalable solution for backend development. Emphasis is placed on the deep integration of Kotlin with Java, allowing developers to combine the benefits of both languages in a single project

This enriches the technology stack and increases development efficiency. In addition, the articles emphasize the significant reduction of boilerplate code when using Kotlin, which simplifies the development and maintenance of applications. However, despite Kotlin's high degree of compatibility and capabilities, Java continues to be the preferred choice in some scenarios due to its stability, wide community, and extensive ecosystem. Thus, the scope of this article will take a closer look at the choice between these programming languages based on a comparative analysis of their functionality, performance, security, and usability in the context of backend development.

In turn, this topic will be discussed in detail within this paper.

III. Materials and Methods

In 2011, JetBrains made an important contribution to the programming world by introducing Kotlin, a programming language focused on object-oriented and statically typed programming. Designed to be compatible with JVM (Java Virtual Machine) and JavaScript, Kotlin quickly gained recognition in the professional community. Its importance was confirmed by Google's decision to officially endorse Kotlin as one of the preferred languages for Android application development, alongside the established Java.

JetBrains' 2019 survey showed a wide range of Kotlin development applications, from Android and JVMs to Native and JavaScript. The language has formed the basis of many popular applications such as Pinterest, Trello, and CarLens, demonstrating its flexibility and power in creating a variety of software. Kotlin continues to be at the forefront of technological advancements, providing developers with convenient and efficient tools to realize their creative and professional ambitions.

Kotlin stands out from other programming languages for its ability to radically reduce the amount of template code required. This is achieved by reducing the need to write redundant code

International Journal of Scientific Engineering and Science
Volume 8, Issue 5, pp. 52-55, 2024. ISSN (Online): 2456-7361

that is mandatory in other languages, thereby simplifying the development process.

One of the key advantages of Kotlin is its security. The language is designed to minimize common sources of errors, such as exceptions caused by accessing null-valued variables. In Kotlin, every variable is non-null by default, which greatly reduces the likelihood of such exceptions.

Kotlin demonstrates exceptional versatility, allowing its use in both Android app development and server-side development. The language supports a wide range of platforms including JVM, Android, JavaScript, iOS, Linux, Windows, Mac, and even embedded systems such as STM32, making it an ideal choice for multi-platform projects. This versatility makes Kotlin an important tool in the arsenal of the modern developer, able to serve a variety of programming needs and challenges.

In turn, if we talk about such programming language as Java, it was developed by the Green Team (James Gosling, Mike Sheridan, Patrick Naughton) in 1995 as an object-oriented multiplatform free programming language similar to C++, with extended and simplified features. Java has become one of the most popular and widely used programming languages because it is easy to learn, use, compile, and debug. Figure 1 shows the popularity graphs of these programming languages.

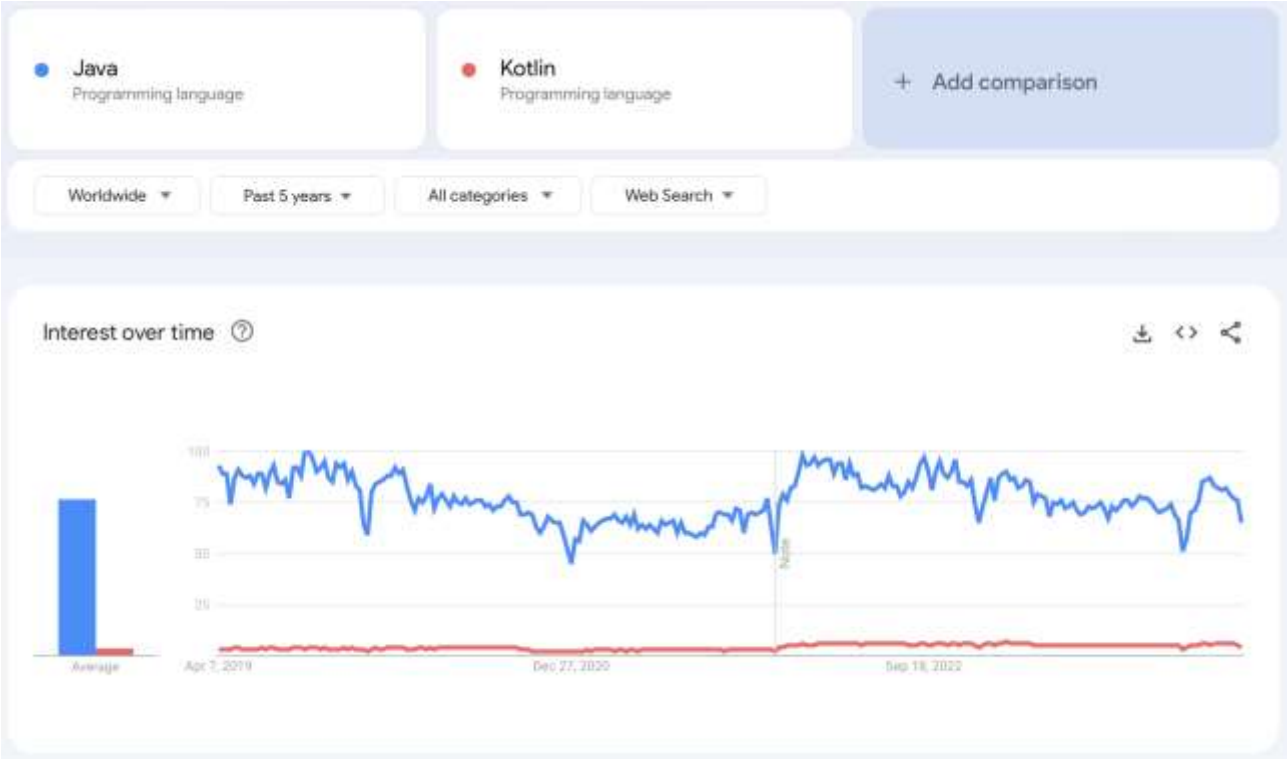

Figure 1. Graph of the popularity of programming languages.
Source: https://trends.google.ru/trends/explore?date=today%205-y&q=%2Fm%2F07sbkfb,%2Fm%2F0_lcrx4&hl=ru

A programming language that provides code portability between different computing platforms offers considerable flexibility to developers. This portability allows the same program to function on a variety of systems without the need for specific adaptations for each of them. This approach, taken as a basis in object-oriented programming, contributes not only to the orderly structuring of program code but also to the simplification of its maintenance and modification processes. In addition, the ability to reuse objects in different projects significantly increases the development efficiency.

Multithreading, which provides the ability to perform multiple tasks simultaneously, proves to be critical in areas such as game application development. This aspect is key to creating responsive and high-performance software solutions.

Java has a unique characteristic of being platform-specific independent, which is achieved by compiling to bytecode that is universal to any Java virtual machine. This aspect, along with the object-oriented approach, provides Java with a high level of security through mechanisms such as encryption-based authentication and malware protection. In addition, a strict type system and error-checked compilation contribute to creating robust and stable applications by minimizing the risks of erroneous states in program code.

In the context of Java, the architectural neutrality of compiler-generated object files represents a significant achievement, allowing programs to run on multiple processor architectures without the need for code modification. Additionally, Java's high performance is due to the use of Just-In-Time (JIT) compilers, which optimizes code execution at runtime.

In turn, when considering these programming languages, it becomes apparent that they have distinctive features. For example, demonstrating the differences in server-side development approaches between Kotlin and Java sheds light on their structural simplicity and developer accessibility.

International Journal of Scientific Engineering and Science
Volume 8, Issue 5, pp. 52-55, 2024. ISSN (Online): 2456-7361

Consider an example that includes HTTP server implementations in both Kotlin and Java (Figs. 2, 3). These examples not only emphasize the syntax of the languages but also demonstrate their different approaches to building web servers, thus offering insight into their suitability for server-side development.

Kotlin simplifies server-side development through its concise syntax and integration with platforms such as Ktor, allowing developers to define routes and process requests more simply. Let's look at implementing a simple REST API in Kotlin using Ktor:

```
import io.ktor.server.engine.*
import io.ktor.server.netty.*
import io.ktor.application.*
import io.ktor.response.*
import io.ktor.routing.*

fun main() {
    embeddedServer(Netty, port = 8080) {
        routing {
            get("/") {
                call.respondText("Hello, Backend World!")
            }
        }
    }.start(wait = true)
}
```

Figure 2. A program written in the Kotlin programming language

This example demonstrates the brevity of Kotlin and the ease of setting up a basic web server with a minimal set of templates, making it accessible to developers and providing robust capabilities for more complex applications.

In contrast, implementing the same functionality in Java, especially with the Spring Framework, requires more detailed customization but offers extensive customization and control, which appeals to developers who require a full feature set for large-scale applications.

```
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RestController;

@SpringBootApplication
@RestController
public class HelloWorldApplication {

    public static void main(String[] args) {
        SpringApplication.run(HelloWorldApplication.class, args);
    }

    @GetMapping("/")
    public String hello() {
        return "Hello, Backend World!";
    }
}
```

Figure 3. A program written in the Java programming language

This Java example reflects the language's traditional verbosity as well as its flexibility for developing complex server systems. The Spring Framework's extensive features support a wide range of server-side development tasks, from simple API services to complex enterprise-level applications.

Although the elementary "Hello, World!" Programs may seem too simplistic to reveal the full range of programming language capabilities, but these more complex examples highlight the key attributes of each language. In particular, the implementation of such programs in Kotlin demonstrates that the same goal can be achieved with fewer lines of code compared to Java, demonstrating the syntactic economy and high degree of expressiveness of the former.

When choosing a programming language to learn, it is advisable to consider your own goals and the objectives you have in mind. Java, characterized by a higher level of complexity, offers at the same time ease of learning thanks to its structured nature and extensive documentation. Kotlin, on the other hand, is characterized by its ease of comprehension and greater flexibility, which makes its use more widespread, although it involves a steeper learning curve [3].

Also, Kotlin, which has emerged on the programming scene as a more accessible and intuitive language compared to Java, frees developers from the need to use semicolons at the end of each line of code. This characteristic makes Kotlin syntax particularly transparent and makes the language easier to learn and understand. In contrast to Kotlin, Java is a more complex language with a high potential for syntax and logic errors, potentially increasing the time to debug and fix code. However, Kotlin stands out not only by simplifying code writing but also by reducing the likelihood of errors during compilation and execution due to its brevity and conciseness. Additionally, Kotlin simplifies the handling of mutable and immutable data structures, which extends its application to server-side development [4].

One of the key advantages of Kotlin is its deep integration with Java, allowing both languages to coexist and interact in the same project. As a result, Kotlin inherits and extends the functionality of Java, providing a high degree of compatibility with existing Java libraries and platforms. This feature makes the transition to Kotlin less costly in terms of time and resources, as developers can gradually integrate Kotlin into their projects without having to completely migrate from Java. In addition, learning Kotlin is relatively easy for those who are already familiar with Java, thanks to the availability of code conversion tools such as the Kotlin plugin for IntelliJ IDEA or Android Studio.

In a development context, using Kotlin usually results in shorter and more understandable code than Java for similar tasks. This compactness not only improves code readability but also makes it easier to debug. The perfect integration with Kotlin Integrated Development Environment (IDE) further simplifies the development process by making the code less error-prone and more efficient [5].

Kotlin addresses one of Java's most pressing problems, the problem of null pointer exceptions (NPEs), by providing an innovative solution that significantly reduces the probability of their occurrence. Inspired by best practices and elements from

other programming languages such as C# and Scala, as well as borrowing ideas from Pascal, Kotlin successfully integrates advanced concepts into its syntax. Its support for extended data structures and declarative style of variable declaration, where the data type follows the variable, further emphasizes its flexibility and power, providing developers with greater opportunities to create quality and robust software [6,7].

Additionally, Kotlin provides convenient facilities for working with classes of primitive methods. This includes the use of operators for basic operations on complex classes such as BigDecimal, as well as simple and straightforward ways to work with arrays.

In addition, it is possible to write single-line methods on a single line in Kotlin, which helps improve code readability and compactness. Contextual functions such as let, apply, also, with, and run open up new possibilities for functional programming, increasing the flexibility and expressiveness of the language.

While some may view these features as "sugarcoating", they greatly simplify the development process and reduce the likelihood of errors. It is also important to note that the need to adhere to code style in Kotlin becomes more urgent, but the development process becomes simpler and more efficient [8-10].

IV. Backend development approaches

The traditional monolithic approach involves creating a single, seamless application in which all backend components are integrated and interact directly with each other. This approach is typically used in small to medium-sized projects where scalability and flexibility requirements are not critical. The advantages of the monolithic approach include ease of deployment and debugging and relative ease in understanding the application architecture. However, it also has its disadvantages, such as limited scalability and difficulty in maintaining when code changes are required.

Microservice architecture is one of the most popular modern approaches to backend development. Using this approach, an application is broken down into small independent services, each of which performs a limited set of functions and interacts with other services through APIs. This approach provides high flexibility, scalability, and fault tolerance to the application because each service can be scaled and updated independently. However, a microservice architecture requires additional deployment, management, and monitoring efforts, and can increase the complexity of integration and testing.

Event-driven architecture focuses on handling real-time events and notifications. With this approach, the backend consists of independent components that respond to incoming events and handle them accordingly. This approach is particularly useful for applications that require asynchronous processing of large amounts of data or real-time interaction. However, implementing an event-driven architecture can be challenging due to the need to ensure reliable event delivery and processing, as well as potential scalability and state management issues [11].

V. Conclusion

Kotlin offers modern syntactic constructs, makes it easier to write code by reducing the amount of code, and improves application security. At the same time, Java retains its importance due to its broad support, scalability, and performance, especially in large and complex projects.

Both languages exhibit a high degree of interoperability and compatibility, allowing developers to flexibly utilize their advantages according to project objectives. The availability of skilled professionals and ease of integration with existing tools and libraries are also important factors in the choice.

Thus, the final choice between Kotlin and Java should be based on a comprehensive analysis and correspond to the strategic development goals of the project, its technical requirements, and resources. Further development of both languages is expected in the future, which will make them even more powerful tools in the hands of backend developers.

References

1. Kotlin Vs. Java: Which Is The Better For Android App Development?[Электронный ресурс] Режим доступа: https://www.elinext.com/blog/kotlin-vs-java/ .– (дата обращения 29.02.2024).
2. Java Vs Kotlin: A Comparative Analysis For Mobile Developers.[Электронный ресурс] Режим доступа: https://ranksol.com/java-vs-kotlin/ .– (дата обращения 29.02.2024).
3. Блинова А. В. Противостояние языков программирования Kotlin и Java в разработке мобильных приложений // Актуальные исследования. 2022. №37 (116). С. 38-40.
4. Why Choose Kotlin Over Java For Backend Development?[Электронный ресурс] Режим доступа: https://quokkalabs.hashnode.dev/why-choose-kotlin-over-java-for-backend-development .– (дата обращения 29.02.2024).
5. Kotlin vs Java при разработке backend-приложений .[Электронный ресурс] Режим доступа: https://deveducation.com/blog/kotlin-vs-java-pri-napisanii-backend-prilozheniy/ .– (дата обращения 29.02.2024).
6. Java vs Kotlin - Comparison & Use Cases for Elegant Android App Development .[Электронный ресурс] Режим доступа: https://www.tempest.house/blog-posts/java-vs-kotlin-comparison-use-cases-for-elegant-android-app-development .– (дата обращения 29.02.2024).
7. Java или Kotlin: что выбрать начинающему андроид-разработчику .[Электронный ресурс] Режим доступа: https://skillbox.ru/media/code/java_ili_kotlin_chto_vybrat_nachinayushchemu_android_razrabotchiku/ .– (дата обращения 29.02.2024).
8. Comparison of Kotlin and Java when writing backend applications .[Электронный ресурс] Режим доступа: https://devlister.com/tutorials/103-comparison-of-kotlin-and-java-when-writing-backend-applications.html .– (дата обращения 29.02.2024).
9. Сравнение Kotlin и Java при написания backend-приложений .[Электронный ресурс] Режим доступа: https://tproger.ru/articles/sravnenie-kotlin-i-java-pri-napisanija-backend-prilozhenij .– (дата обращения 29.02.2024).
10. Kotlin Vs Java: What Software Engineers Choose For Android Development.[Электронный ресурс] Режим доступа: https://dashdevs.com/blog/kotlin-vs-java-what-software-engineers-choose-for-android-development/ .– (дата обращения 29.02.2024).
11. Kotlin vs Java: Breaking Down the Differences for Business Leaders.[Электронный ресурс] Режим доступа: https://jaydevs.com/kotlin-vs-java/ .– (дата обращения 29.02.2024).