Madrid,
Spain 2024

INTERNAUKA
internauka.org

XLVI International Multidisciplinary Conference

PROSPECTS AND KEY TENDENCIES OF SCIENCE IN CONTEMPORARY WORLD

PROSPECTS AND KEY TENDENCIES OF SCIENCE IN CONTEMPORARY WORLD

Proceedings of XLVI International Multidisciplinary Conference

August, 2024

Madrid, Spain
2024

COMPARATIVE STUDY OF JAVA AND KOTLIN FOR ANDROID DEVELOPMENT

Nicolas Petrov
Student,
Netology,
Russia, Moscow

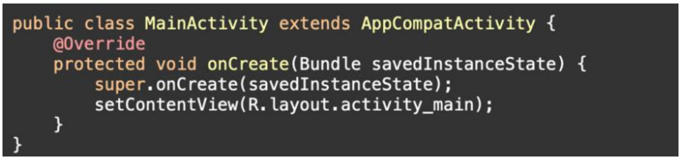
ABSTRACT

Android development has traditionally been dominated by Java, a language known for its reliability and extensive history. However, the emergence of Kotlin language created by JetBrains in 2011 and its subsequent adoption by Google* as the official language for Android development in 2017 changed the situation. The main purpose of this paper is to conduct a comparative analysis of Java and Kotlin, focusing on syntax, performance, developer productivity and community acceptance, and to develop an understanding of the strengths and weaknesses of both languages in the context of Android development.

Keywords: Java, Kotlin, programming, Android development, programming languages comparison, syntax.

The Java language, developed by Sun Microsystems (now taken over by Oracle), has been a cornerstone of Android development since the platform's inception. But in recent years, Kotlin has become a popular alternative. In 2017, Google* officially announced Kotlin as the language for Android development, which has led to a rise in its popularity and interest from the programming community. Kotlin, developed by JetBrains, was created as a more concise and type-safe language than Java. The language is fully compatible with Java, allowing Java developers to gradually adopt its use; in particular, the language is also embedded by Android, allowing for an existing Android application to implement new features in Kotlin without rewriting the entire application.

Java syntax is verbose and has a classic object-oriented approach.

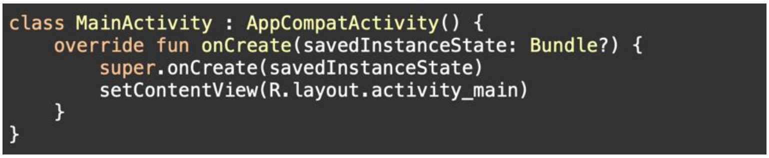


```java
public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
}
```

Image 1. Example of Java syntax

Java syntax, while rich and versatile, maintains a structured and predictable format that helps both novice programmers and experienced developers create efficient, readable, and reliable code. As the language continues to evolve, a thorough understanding of its syntax will remain fundamental to utilising its full potential in software development.

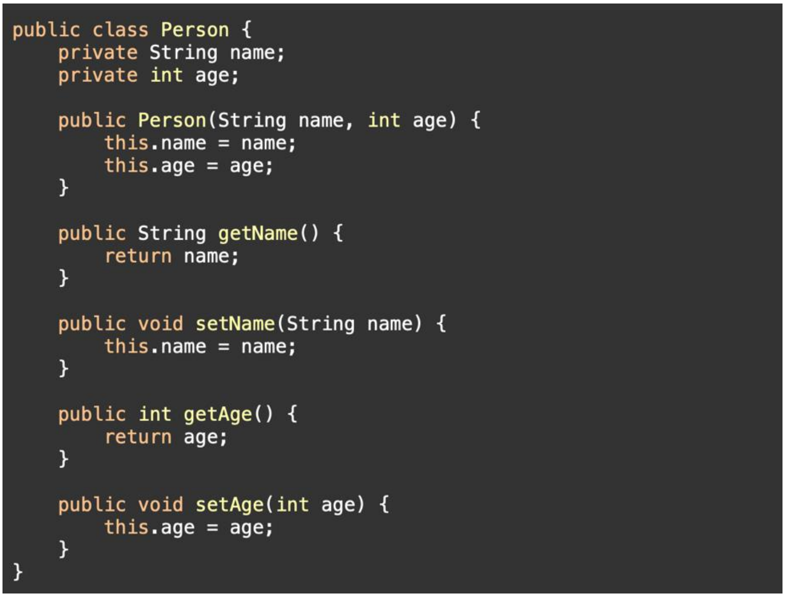
Kotlin syntax is concise and expressive, predominantly combines inheritance from two language branches: C/C++/Java and ML, and is designed to minimise the amount of templated code.



```
class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
    }
}
```

Image 2. Example of Kotlin syntax

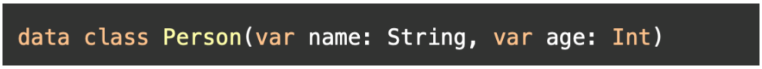
The object-oriented paradigm requires explicit declarations and strict typing, which, while ensuring type safety, can lead to long code bases.



```
public class Person {
    private String name;
    private int age;

    public Person(String name, int age) {
        this.name = name;
        this.age = age;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public int getAge() {
        return age;
    }

    public void setAge(int age) {
        this.age = age;
    }
}
```

Image 3. We define the Person class, which has two properties: name and age in Java

Kotlin supports both object-oriented and functional programming paradigms, allowing for more flexible and expressive designs.



```kotlin
data class Person(var name: String, var age: Int)
```

Image 4. We define a dataclass Person, which has two properties: name and age in Kotlin (data class is a special class type in the Kotlin programming language designed to represent small, simple data objects)

Kotlin greatly reduces the number of templates, making code easier to read and maintain. It implements null safety in the type system, which reduces the likelihood of NullPointerExceptions. Kotlin also allows developers to extend existing classes with new features without inheriting them.

Java is known for its stable and predictable performance thanks to the Java Virtual Machine (JVM). Kotlin is also compiled to bytecode that runs on the JVM, so its performance is usually on par with Java. Benchmarks show little difference in performance, with Kotlin sometimes showing slightly lower performance due to the additional layers of abstraction. However, the standard Kotlin library provides optimised implementations of some operations, such as collections and higher-order functions, which conversely can lead to better performance in certain scenarios.

Extensive Java documentation and significant community support allow developers to quickly find solutions to common problems. Java has a large and mature community with extensive libraries, frameworks, and tools. Its long history means there are plenty of resources for learning and troubleshooting.

The Kotlin community is new, but it is growing rapidly, especially after it was endorsed by Google* for Android development. JetBrains and Google* provide Kotlin developers with strong support and resources. Kotlin's interoperability with Java allows developers to use existing Java libraries and gradually migrate their codebase to Kotlin. However, Java has a more established community with many long-standing resources.

When choosing between Java and Kotlin for Android development, there are several factors to consider.

Java remains a robust and widely used language for Android development, offering broad community support and predictable performance. Kotlin offers modern language features, conciseness, and enhanced security, which can significantly improve developer productivity. In addition, Kotlin's excellent tooling support in Android Studio through more modern language features and IDE plugins specifically designed for Android development further enhances developer productivity. However, Kotlin has a more difficult learning curve for developers unfamiliar with functional programming concepts.

Both languages work similarly, with minor differences that rarely affect real-world applications. Both Java and Kotlin are viable options for Android development, each with their own strengths and weaknesses.

However, in my opinion, for new projects Kotlin seems to be a better choice because of its modernity. But if you are a novice programmer, it is better to choose Java at the start: this language is older and it is easy to find information on how to solve a particular problem due to its wide distribution and extensive documentation.

References:

1. JetBrains. (2011). Kotlin Documentation. [Online]. Available: https://kotlinlang.org/docs/reference/
2. Oracle. (2023). Java Documentation. [Online]. Available: https://docs.oracle.com/javase/
3. Google*. (2017). Kotlin for Android Development. [Online]. Available: https://developer.android.com/kotlin
4. Bloch Joshua. "Effective Java." Addison-Wesley, 2018.
5. Sierra Kathy and Bert Bates. "Head First Java." O'Reilly Media, Inc., 2005.
6. Horstmann Cay S., and Gary Cornell. "Core Java Volume I--Fundamentals." Prentice Hall, 2015.
7. Kotlin Documentation. "Basic Syntax." Kotlinlang.org, JetBrains. https://kotlinlang.org/docs/basic-syntax.html.
8. Maple S. and Zou Y. (2020). How Kotlin's Coroutines Improve Code Readability. In: 2020 IEEE 20th International Working Conference on Source Code Analysis and Manipulation (SCAM). IEEE, pp.43-47.

**At the request of Roskomnadzor, we inform you that a foreign person who owns Google information resources is a violator of the legislation of the Russian Federation* – ed. note