universität innsbruck

Leopold-Franzens Universität Innsbruck

SE Specialisation Seminar
summer semester 2023

Kotlin Programming Language

Kristina Aleksandrova
Matrikel-Nr. 01437595

Instructor: Assoc.-Prof.
Dr. Cezary Kaliszyk

11.July 2023

Abstract

Kotlin, a modern programming language, has gained significant popularity among developers due to its versatility, efficiency, and seamless integration with existing Java codebases. This report provides an overview of Kotlin, including its brief history, vision, distinguishing features, differences from Java, limitations, and real-world applications. By understanding Kotlin's unique attributes and its role in contemporary software development, developers can make informed decisions about utilizing this language in their projects. This report aims to equip readers with the necessary knowledge to evaluate Kotlin's suitability for their programming needs and to appreciate its growing significance in the software development landscape.

Contents

1 Introduction

In the fast-paced world of software development, programming languages play a crucial role in shaping the way applications are built, maintained, and scaled. Kotlin, a statically typed programming language, has emerged as a compelling choice for developers seeking a modern alternative to traditional languages like Java. Developed by JetBrains, the creators of popular IDEs such as IntelliJ IDEA, Kotlin was designed to address the shortcomings of existing languages while seamlessly integrating with the Java ecosystem [1].

This report serves as an introduction to Kotlin, exploring its origins, unique features, and real-world applications. By delving into Kotlin's history, vision, and distinguishing characteristics, we can gain a deeper understanding of why this language has garnered such widespread acclaim and adoption.

With its concise syntax, type safety, and interoperability with Java, Kotlin offers developers an efficient and expressive programming experience [7]. By leveraging Kotlin's powerful features, developers can enhance productivity, reduce boilerplate code, and improve overall code quality. Moreover, Kotlin's compatibility with existing Java codebases allows developers to leverage their investments in Java while benefiting from Kotlin's modern language features.

However, like any programming language, Kotlin has its limitations and trade-offs. This report will highlight these limitations to provide readers with a balanced perspective on the language's capabilities and potential constraints.

Furthermore, we will explore real-world applications of Kotlin across diverse industries, including mobile development, server-side programming, and multi-platform development. By examining successful use cases, we can gauge Kotlin's versatility and assess its suitability for various software projects.

In conclusion, this report aims to provide a comprehensive overview of Kotlin, enabling developers to make informed decisions about its adoption and integration into their development workflows. Whether you are a seasoned developer or someone exploring new programming languages, understanding Kotlin's history, vision, distinguishing features, differences from Java, limitations, and real-world applications will equip you with the necessary knowledge to navigate the evolving landscape of software development.

2 Related Works

For the compilation of this report on Kotlin programming language, various resources were consulted to ensure accuracy and completeness of information. The primary sources utilized include online materials available on the internet and the official Kotlin website and documentation.

Websites such as blogs, forums, and technology news platforms were explored to gather insights into the community's experiences, opinions, and discussions surrounding Kotlin. These online resources offered valuable perspectives, examples, and practical use cases, which aided in presenting a comprehensive view of the language.

However, one of the most significant sources for this report was the official Kotlin website [10]. The Kotlin website serves as a central hub for all official information, documentation, tutorials, and resources related to the language. The documentation provided by JetBrains, the developers of Kotlin, proved to be an invaluable resource for understanding the language's syntax, features, and usage. It offered detailed explanations, code samples, and practical illustrations to support the understanding of Kotlin's concepts and principles.

3 Kotlin Programming Language

Kotlin is a statically typed programming language. It is widely used in various domains, including Android app development, backend development, web development, and more. It has a growing ecosystem of libraries, frameworks, and tooling, making it a versatile language for building robust and efficient software applications.

3.1 History

Kotlin was conceived and developed by a team of programmers at JetBrains, led by Dmitry Jemerov [9]. The team wanted to create a modern programming language that could improve upon the shortcomings of existing languages, particularly Java, and enhance productivity for developers.

Development of Kotlin started in 2010 [2], and the language was initially built to run on the Java Virtual Machine (JVM). Kotlin was announced to the public at the JVM Language Summit in July 2011 [4]. JetBrains presented Kotlin as a pragmatic and concise language that aimed to improve Java development by adding modern features, better tooling, and seamless interoperability with existing Java codebases [11].

In February 2012, JetBrains open-sourced Kotlin under the Apache 2.0 license [2]. This move allowed the language to gain wider community involvement and receive feedback. Following its open-source release, Kotlin gained a small but dedicated following, primarily within the Kotlin developer community and among Java developers looking for a modern alternative. During this period, Kotlin continued to evolve with regular updates and improvements, addressing feedback and adding new language features.

In May 2017, at the Google I/O developer conference, Google announced first-class support for Kotlin on the Android platform [3]. Kotlin became an officially supported language for Android app development alongside Java. This endorsement significantly increased Kotlin's visibility and adoption among Android developers. Since the official support for Kotlin on Android, the language has seen a rapid rise in popularity.

Many developers and organizations have embraced Kotlin due to its concise syntax, null safety, interoperability with Java, and modern language features. Kotlin has continued to evolve with regular updates, adding new features and improvements based on community feedback.

3.2 Vision of Kotlin

The vision of the Kotlin programming language is to provide a modern, concise, and pragmatic language that enhances developer productivity, promotes code safety, and offers seamless interoperability with existing Java codebases.

The important aspects of Kotlin are:

1. Conciseness and Readability: Kotlin aims to minimize boilerplate code and reduce verbosity, allowing developers to express their intentions clearly and concisely. It provides a clean and expressive syntax, making code more readable and maintainable [1].

2. Interoperability: Kotlin is designed to seamlessly integrate with existing Java code. It can be used alongside Java in the same project, allowing developers to leverage their Java knowledge and reuse Java libraries and frameworks without any significant effort.

3. Safety and Reliability: Kotlin addresses common pitfalls and vulnerabilities in programming, offering features such as null safety and type inference. It aims to minimize runtime errors and provides compile-time checks to catch errors early in the development process, improving the overall reliability of software.

4. Tooling and IDE Support: JetBrains, the company behind Kotlin, provides excellent tooling and IDE support for Kotlin development. Kotlin is fully supported by JetBrains' IntelliJ IDEA, Android Studio, and other IDEs, offering features like code completion, refactoring, and debugging capabilities.

5. Industry Adoption: Kotlin aims to have broad adoption across different domains, including Android app development, backend development, and more. It strives to be a practical and viable choice for developers and organizations, offering productivity gains, improved code quality, and performance [8].

6. Community-Driven Development: Kotlin encourages community involvement and contributions to shape the language's evolution. The Kotlin community actively participates in providing feedback, reporting issues, and contributing to the language's development through open-source contributions [6].

3.3 Distinguishing Features

Kotlin is known for several distinguished features that set it apart from other programming languages.

Concise Syntax: Kotlin offers a concise and expressive syntax, reducing boilerplate code and making code more readable and maintainable. Features like type inference, smart casts, lambda expressions, and extension functions contribute to the brevity of Kotlin code.

```
fun main() {
  println("Hello world!")
}
```

Kotlin excels at reducing boilerplate code in projects, showcasing its efficiency by converting a basic Java class containing 96 lines of code into just a single line [5].

Null Safety: Kotlin provides built-in null safety features, addressing one of the most common sources of runtime errors in programming. It distinguishes between nullable and non-nullable types, reducing the occurrence of null pointer exceptions. Null safety is enforced at compile-time, ensuring safer code. In Kotlin, the type system distinguishes between references that can hold null (nullable references marked by "?") and those that cannot (non-null references). For example, a regular variable of type String cannot hold null:

```
var a: String = "abc" //means non-null by default
a = null // compilation error

var b: String? = "abc" // can be set to null
b = null // ok
print(b)
```

Smart Casts: Kotlin's smart casts allow automatic type casting when certain conditions are met. This eliminates the need for explicit type checks and casting, making the code cleaner and more concise. Smart casts contribute to improved readability and reduced developer effort. In most cases, you don't need to use explicit cast operators in Kotlin because the compiler tracks the is-checks and explicit casts for immutable values and inserts (safe) casts automatically when necessary:

```
fun demo(x: Any) {
  if (x is String) {
    print(x.length) // x is automatically cast to String
  }
}
```

Extension Functions: Kotlin supports extension functions, which allow developers to add new functions to existing classes without modifying their source code. This feature enables adding utility functions or extending the functionality of third-party libraries, promoting code reuse and enhancing expressiveness.

```
// Extension function on the String class
fun String.removeWhitespace(): String {
  return this.replace(" ", "")
}
```

Coroutines: Kotlin provides built-in support for coroutines, enabling asynchronous and concurrent programming. Coroutines simplify handling asynchronous operations by allowing developers to write sequential code that suspends and resumes execution, eliminating the need for callbacks or complex threading mechanisms.

```
fun main() = runBlocking { // this: CoroutineScope
  launch { // launch a new coroutine and continue
    delay(1000L) // non-blocking delay for 1 second
    println("World!") // print after delay
  }
  println("Hello") // main coroutine continues, previous is delayed
}
```

Interoperability with Java: Kotlin offers seamless interoperability with Java, allowing developers to leverage existing Java codebases, libraries, and frameworks. Kotlin code can call Java code directly and vice versa, making it easy to adopt Kotlin gradually or work on projects with mixed Java-Kotlin codebases.

```
// Person.kt
class Person(val name: String, val age: Int) {
  fun sayHello() {
    println("Hello, my name is $name and I am $age years old.")
  }
}

// JavaExample.java
public class JavaExample {
  public static void main(String[] args) {
    Person person = new Person("John", 25);
    person.sayHello();}}
```

Type Safety: Kotlin is a statically typed language that ensures strong type safety. It performs comprehensive type checks at compile-time, reducing the occurrence of type-related runtime errors and providing early detection of potential issues.

```
fun main() {
  val name: String = "John" // String assigned to name variable
  val age: Int = 30 // Int type assigned to age variable
  val result = name + age
}
// Compile-time error: Type mismatch
println(result)
```

Data Classes: Kotlin provides data classes, which are classes specifically designed to hold data with automatic generation of common methods such as `equals()`, `hashCode()`, `toString()`, and `copy()`. Data classes help reduce boilerplate code when working with simple data structures and automatically provide several useful features. It automatically generate getters for all properties. In this example, you can access the name and age properties directly.

```
val person = Person("John", 25)
println(person.toString()) // Output: Person(name=John, age=25)
```

Destructuring declarations: Data classes generate component functions that allow you to destructure objects into individual variables.

```
val person = Person("John", 25)
val (name, age) = person
println(name) // Output: John
println(age) // Output: 25
```

3.4 Comparison to Java

There are several reasons why Kotlin has gained popularity among Android developers as an alternative to Java [1]:

Reduced boilerplate code: Kotlin's concise syntax and features like data classes, smart casts, and extension functions significantly reduce the amount of boilerplate code needed to write Android applications.

Improved code safety: With its null safety feature, Kotlin reduces the risk of encountering `NullPointerExceptions` at runtime, leading to more stable and reliable applications.

Better support for functional programming: Kotlin provides first-class support for functional programming concepts such as lambdas, higher-order functions, and extension functions, making it easier to write clean and modular code.

Seamless Java interoperability: Kotlin code can be called from Java and vice versa, enabling developers to gradually migrate their existing Java codebases to Kotlin or leverage existing Java libraries and frameworks in Kotlin projects.

Android Studio support: As both Android Studio and Kotlin are developed by JetBrains, the IDE provides excellent support for Kotlin development, including syntax highlighting, code completion, debugging, and refactoring tools.

3.5 Limitations

While Kotlin has gained popularity and addressed many limitations of other programming languages, it does have some limitations of its own.

1. Immaturity compared to Java: Kotlin is considered less mature than Java, which means it may have more bugs and frequent updates. This can lead to challenges in finding solutions for uncommon issues. In contrast, Java's extensive history and community support make it easier to find answers to a wide range of problems [13].

2. Performance: While Kotlin is generally fast and efficient in areas like incremental builds [12], it may not match the raw power of Java in developing high-performance Android applications. Since Kotlin builds on top of Java, the additional layer can impact performance. However, this drawback is often outweighed by the simplicity and ease of use that Kotlin offers, particularly for beginners [13].

3. Hiring challenges: Due to its relative newness and less maturity compared to Java, the pool of experienced Kotlin developers may be smaller. For larger companies seeking to hire a significant number of Kotlin developers, it could be challenging to find experienced individuals. Job portals like Indeed may list Kotlin developers, but identifying highly skilled candidates might be more difficult compared to hiring Java developers. However, for smaller companies looking to hire a few Kotlin developers, this may not be a significant issue [13].

3.6 Real-World applications

Kotlin has achieved significant success across a wide range of real-world applications. It has particularly excelled in the following domains: Android development, server-side development, multi-platform development, data science and machine learning, desktop application development, game development. Notably, a substantial 81% of Kotlin's usage is attributed to Android application development (Figure 1). Prominent examples of successful Android apps developed using Kotlin include Airbnb, Coursera, Uber, Netflix, Pinterest, Duolingo, and many more [8].

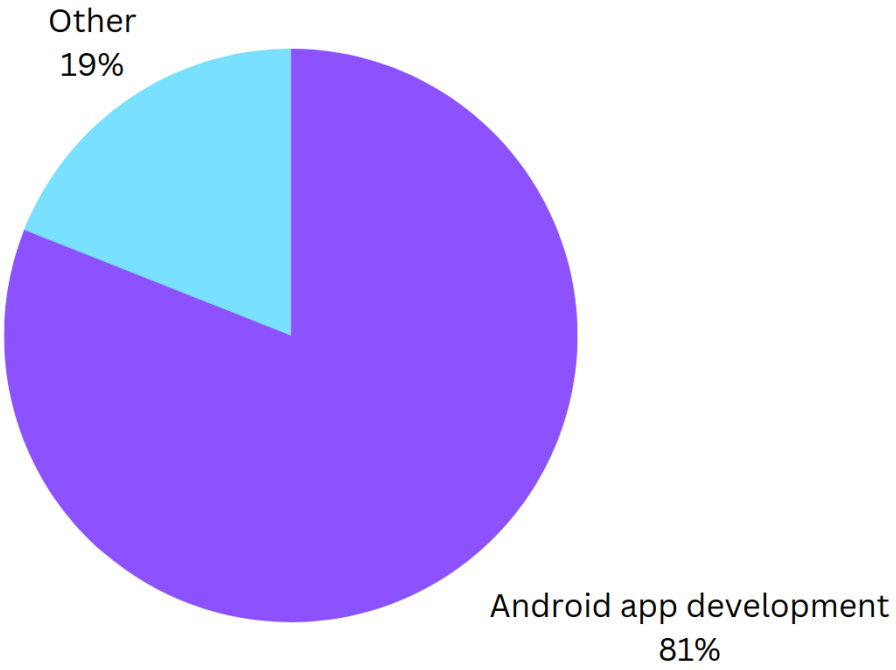

Figure 1: Tasks Kotlin is recommended for [8].

4 Conclusions

In conclusion, Kotlin has emerged as a versatile and Java-compatible programming language that has gained significant traction in the software development community. Through its unique blend of features, Kotlin offers developers a modern and efficient programming experience while maintaining seamless interoperability with existing Java codebases.

Throughout this report, we explored the history, vision, distinguishing features, differences from Java, limitations, and real-world applications of Kotlin. It is evident that Kotlin's concise syntax, null safety, extension functions, and other language features contribute to improved code readability, reduced boilerplate code, and enhanced productivity.

Kotlin's success in Android development is undeniable, with a significant percentage of Kotlin usage being attributed to Android applications. Industry giants like Airbnb, Coursera, Uber, Netflix, Pinterest, and Duolingo have embraced Kotlin, highlighting its value in delivering high-quality, user-friendly Android apps.

The overall trajectory of Kotlin demonstrates its significance and potential impact on the future of software development. As more developers embrace Kotlin and its ecosystem continues to mature, we can expect even greater advancements, improved tooling, and expanded real-world applications.

For developers considering Kotlin for their projects, it is crucial to carefully evaluate its suitability based on project requirements, team expertise, and existing codebase considerations. The official Kotlin website, documentation, and the vibrant Kotlin community serve as valuable resources to navigate the language effectively.

In summary, Kotlin offers a compelling alternative to traditional programming languages, particularly for Android development, while also providing strong support for server-side, multi-platform, data science, desktop application, and game development. Its versatility, efficiency, and growing adoption make it a language worth exploring for developers seeking to enhance their productivity and deliver robust software solutions in today's fast-paced software development landscape.

Bibliography

[1] AppMaster. Kotlin: Understanding the Java alternative. https://appmaster.io/blog/kotlin-embracing-the-future-of-android-development, 2023. [Online; accessed 03-July-2023].

[2] The Kotlin Blog. Kotlin goes open source! https://blog.jetbrains.com/kotlin/2012/02/kotlin-goes-open-source-2/, 14.02.2012. [Online; accessed 03-July-2023].

[3] The Kotlin Blog. Kotlin on Android. Now official. https://blog.jetbrains.com/kotlin/2017/05/kotlin-on-android-now-official/, 17.05.2017. [Online; accessed 03-July-2023].

[4] The Kotlin Blog. Hello World. https://blog.jetbrains.com/kotlin/2011/07/hello-world-2/, 19.07.2011. [Online; accessed 06-July-2023].

[5] DigitalOcean. Kotlin data class. https://www.digitalocean.com/community/tutorials/kotlin-data-class, 03.08.2022. [Online; accessed 03-July-2023].

[6] Kotlin Foundation. Protect, promote and advance the development of the Kotlin programming language. https://kotlinfoundation.org/, 2023. [Online; accessed 03-July-2023].

[7] HIGHSOURCE. What is the Kotlin programming language and the advantages of working with it. https://highsource.sa/en/blogs/87, 2023. [Online; accessed 03-July-2023].

[8] InfoStride. Top 15 famous apps built with Kotlin. https://infostride.com/apps-built-with-kotlin/. [Online; accessed 03-July-2023].

[9] Dmitry Jemerov. Dmitry Jemerov. https://www.yole.page/. [Online; accessed 06-July-2023].

[10] JetBrains. Kotlin docs. https://kotlinlang.org/, 2023. [Online; accessed 03-July-2023].

[11] Analytics India Magazine. Ten years of Kotlin programming language. https://analyticsindiamag.com/ten-years-of-kotlin-programming-language/, 2023. [Online; accessed 03-July-2023].

[12] Medium. Kotlin vs Java: Compilation speed. https://medium.com/keepsafe-engineering/kotlin-vs-java-compilation-speed-e6c174b39b5d, 09.09.2016. [Online; accessed 03-July-2023].

[13] TechQuintal. Advantages and disadvantages of Kotlin. https://www.techquintal.com/advantages-and-disadvantages-of-kotlin/, 27.09.2022. [Online; accessed 03-July-2023].