Journal of Advanced Zoology
ISSN: 0253-7214
Volume 45 Issue S-4 Year 2024 Page 73-76

Analysis Of Android Programming Languages Based On Several Factors

Shweta Bhoyate[1*]

[1*]*Computer Science Department, Indira College of Commerce and Science, Pune, Maharashtra, India.*
Email: shweta.bhoyate@iccs.ac.in

***Corresponding Author:** Shweta Bhoyate*
Email: shweta.bhoyate@iccs.ac.in

Abstract

Android is a popular choice for mobile operating systems. It was essentially designed for mobile touch-screen devices, such tablets and smartphones. Many programming languages are being used to design application for android operating systems. Android programming can be made incisive, cross-platform, and amusing with the help of Kotlin. Since the beginning, Java has been the main programming language for Android development and is still in high stipulation. Programming on a wider scale is made easier for android applications on Scala. It is a general-purpose, practical programming language. For creating rapid apps on any platform, Dart is a client-optimized programming language used for android application development. Here in this paper, we are going to scrutinize these languages based on several factors like implementation and effectiveness, cross-platform compatibility and android runtime.

Keywords: **Android, Kotlin, Scala, Java, Dart, scrutinize, implementation, effectiveness, compatibility, runtime**

I. INTRODUCTION

First, we will review each android application programming language and its features regarding the same.

II. JAVA

With Android application development, Java is one of the most often employed programming languages [1]. It is still habitually used today and has been the main language for Android development since the platform's float. Programming apps is made much elementary by the cluster of libraries and frameworks accessible in Java. Java operates in virtual machines; as a consequence, code does not need to be recompiled for each gadget. Java provides an immense array of libraries and frameworks that substantially streamline the process of developing apps. It comprises numerous admired development tools, inclusive of Eclipse, NetBeans, and much more. Java is the programming language of option for app developers who issue Java development services because of its powerful toolkit.

III. KOTLIN

Jet Brains, PhpStorm, Appcode, and other top IDEs, created Kotlin, a statically typed general-purpose programming language. It is a contemporary language for the JVM that was originally released by Jet Brains in 2011[1]. It is an object-oriented language considered better than Java, Kotlin can however interface with Java programs smoothly. Google financed Kotlin, which was announced one of the formal languages for Android development in 2017. Kotlin is a programming language that is reconcilable with Java. It is Java further with a few supplementary project-related frameworks and programs [6]. This authorizes the shift to Kotlin and enables for the appropriate synchronous utilization of the two programming languages.

IV. SCALA

Scala source code can run on any JVM since it compiles to Java bytecode. Furthermore, it proposes complete similarity with Java, permitting us to easily refer to Java from Scala and the other way round [4]. Several features missing from Java is added by Scala [8].
Google financed Kotlin, which was announced one of the formal languages for Android development in 2017. The incitement of a functional programming style is wholly an additional phenomenal benefit of adopting the Scala for application development. This intimates that there will be hardly any bugs in an application created with Scala. Furthermore, a small number of problems render into escalated productivity and better-quality products [5].

V. DART

Google fabricate Dart, an object-oriented, extensive-purpose, C-style programming language, in 2011. It is obtainable as an open-source project. The formation of front-end user interfaces for online and mobile applications is the objective of Dart programming. Arrays are not directly subsidized by the Dart language. It empowers optional typing, generics, arrays, and alternative collection-based data structures that are utilized to clone data structures.
The essence of the whole application are widgets. Widgets are the building chunks of Flutter; you can design bespoke ones or use the pre-made ones. The ultimate product will no doubt have less compatibility problems across various platforms and OS versions because widgets are a feature of the app rather than the particular platform.

VI. JAVA / KOTLIN / SCALA/ DART

I. Performance and Efficiency

Java's incorporated garbage collection technique for automatic memory management ensures reasonable memory use and protects averse to recurring memory leaks. The JVM's garbage collector may look after memory deallocation, relieving developers of the strain of managing memory manually [2].

Although Kotlin offers certain unique refinements to aid with memory management, the garbage collection mechanism is fundamentally carried over from the JVM. In Kotlin, for instance, null safety ideas are no segregated into the type system, diminishing the feasibility of null pointer exceptions, which can knock memory consumption and sturdiness [10].

Because Scala runs on the JVM, it uses memory management methods equivalent to those of Java. Since memory is required for all operation, memory management can be defined as granting OS resources and clearing up unemployed RAM to prevent out-of-memory issues [11].

Dart objects are stored in heap. These objects are created using a class constructor. The Dart virtual machine (VM) is in-charge of superintending the heap's memory [12]. When an object is made, the Dart Virtual Machine (VM) assigns memory for it. When the object is no longer in use, it frees the memory.

II. Cross-platform Compatibility

Java is platform neutral. The motto "write once, run a anywhere" has been realistically procured, the research c claims. Applications may now operate fluently on a variety of operating systems because to Java's ground-breaking cross-platform compatibility [9].

One important use case for Kotlin cross-platform is code sharing between mobile platforms [7]. Building cross-platform mobile applications with Kotlin Multiplatform allows you to unite networking, data storage and validation, analytics, computations, including application functionality in projects for both iOS and Android [14].

With Scala, developers can design cross-platform applications that work on any operating system. As a result, they don't need to worry about hardware and software compatibility.

Building apps using Flutter permits you hold a single codebase to create apps for web, desktop, Android, and iOS [14]. This saves a lot of time and work when compared to creating separate native apps for every device because you can create your app logic and user experience only once.

III. JVM Runtime

Kotlin, however, sometimes overshadows Java in terms of performance. One way that Kotlin can oblique reduce runtime exceptions and strengthen application performance is through its null safety property. Performance can also be enlarged by using Kotlin's immutable data structures [3] [16].

Scala supports faster compilation with incremental builds [15].

Dart offers a quicker execution procedure, so it performs better. Because Java runs code more fast and uses statically typed syntax, it performs better than Dart [17].

A command line for Dart might make it run more quickly. However, it is completely negligible when combined with Java and JavaScript in a backend architecture [18].

VII. CONCLUSION

Through analysis, we were able to determine that each language, Java, Kotlin, Scala and Dart has pros and cons.

Java is a fantastic option for enterprise applications because of its performance and library environment.

Kotlin Application Deployment minimizes application size growth, compiles more quickly, and is lightweight.

Though Scala works well for tasks requiring a blend of OOP and functional programming languages, especially where developers must work with large amounts of data or intricate modelling.

As an online and mobile development tool, Dart excels due to its quick development, hot reloading, and flexibility, particularly when combined with Flutter.

So, application programmer can choose any of the programming language as per their requirement, taking into consideration various factors required by the application and available resources.

REFERENCES

1. https://pdfs.semanticscholar.org/c0ee/43434064520cdde7222318bf6c4d2db69177.pdf
2. https://arxiv.org/pdf/2003.12730.pdf
3. https://www.diva-portal.org/smash/get/diva2:1231573/FULLTEXT01.pdf
4. https://www.knowledgehut.com/blog/programming/scala-the-complete-guide
5. https://www.techavidus.com/blogs/benefits-of-java-for-app-development
6. https://www.linkedin.com/pulse/top-7-programming-languages-develop-native-android-apps-parthi-patel
7. https://krify.co/advantages-and-disadvantages-of-kotlin/
8. https://www.linkedin.com/pulse/exploring-pros-cons-scala-right-choice-your#:~:text=Higher%20Quality%20%E2%80%93%20Another%20great%20advantage,and%20a%20higher%20quality%20product.
9. https://medium.com/agileinsider/top-programming-languages-for-android-app-development-in-2023-3eb5d3e0e2f1

10. https://www.calibraint.com/blog/kotlin-vs-java-for-app-development#:~:text=Performance%20And%20Optimization,memory%20management%20and%20runtime%20performance.&text=Java's%20automatic%20memory%20management%2C%20through,and%20prevents%20common%20memory%20leaks.
11. https://towardsdatascience.com/decoding-scala-without-the-code-6db00f37c469
12. https://mobileacademy.io/introduction-to-dart-memory-management/
13. https://kotlinlang.org/docs/multiplatform.html
14. https://www.linkedin.com/pulse/dart-flutter-ideal-combination-cross-platform-4fywc
15. https://wezom.com/blog/scala-vs-java-development
16. https://www.freecodecamp.org/news/kotlin-vs-java-whats-the-difference/#:~:text=But%20Kotlin%20has%20some%20performance,also%20lead%20to%20improved%20performance.
17. https://www.redswitches.com/blog/scala-vs-java/#:~:text=Java%20is%20a%20widely%20used,syntax%2C%20and%20functional%20programming%20capabilities.
18. https://www.reddit.com/r/dartlang/comments/z79ntw/how_exactly_does_dart_compare_performancewise_to/