Kotlin for Android:

A suitable replacement for Android development at Heritage College?

Maxwell Haley

Systems IV

Ronald Patterson

March 13th, 2018

Table of Contents

[Executive Summary 3](#__RefHeading___Toc5850_961208968)

[Introduction 1](#__RefHeading___Toc3995_3478243197)

[1 – What Is Kotlin? 1](#__RefHeading___Toc3551_3478243197)

[2 – Monetary Cost Of Kotlin 2](#__RefHeading___Toc3553_3478243197)

[3 – Tools For Educators 2](#__RefHeading___Toc3555_3478243197)

[4 – Java and Kotlin Syntax Comparison 2](#__RefHeading___Toc3557_3478243197)

[5 – Kotlin Performance 3](#__RefHeading___Toc3559_3478243197)

[6 – Kotlin Documentation Availability 3](#__RefHeading___Toc3561_3478243197)

[7 – Kotlin Staying Power 4](#__RefHeading___Toc3563_3478243197)

[8 – Kotlin and the Job Market 6](#__RefHeading___Toc3565_3478243197)

[Conclusion 8](#__RefHeading___Toc3567_3478243197)

[References 9](#__RefHeading___Toc3569_3478243197)

Table of Figures

[Screenshot 1: Questions tagged with Java as of March 11th, 2018 4](#Screenshot!0|sequence)

[Screenshot 2: Questions tagged with Java and Android as of March 11th, 2018 4](#Screenshot!1|sequence)

[Screenshot 3: Questions tagged with Kotlin as of March 11th, 2018 4](#Screenshot!2|sequence)

[Screenshot 4: Questions tagged with Kotlin and Android as of March 11th, 2018 4](#Screenshot!3|sequence)

[Screenshot 5: Stack Overflow 2017 list of most popular languages, sorted by % of developers that marked the language as liked. 5](#Screenshot!4|sequence)

[Screenshot 6: Most popular programming languages on GitHub, ranked by opened repositories. 6](#Screenshot!5|sequence)

[Screenshot 7: Projected % of developers adopting the Kotlin programming language, as per the Realm Report Q4. 6](#Screenshot!6|sequence)

[Screenshot 8: Indeed search results for Kotlin jobs near Gatineau, QC. 7](#Screenshot!7|sequence)

# Executive Summary

The Computer Science Program at Heritage College currently teaches students to develop applications for the *Android* mobile platform using the enterprise programming language *Java*. *Java* is taught to students starting in their first semester, and the mobile programming course builds on top of that with *Android* specific programming patterns.

A newer language designed by the popular software development company *JetBrains* called *Kotlin* has recently become a first-class programming language for the *Android* platform. *Kotlin* boasts reducing the amount of boilerplate needed per applications, as well as avoiding entire classes of errors. All while running in all the same locations that *Java* can. With both the ease of programming with *Kotlin* on top of avoiding common Ja*va* pitfalls, ***should the Computer Science Program switch teach Kotlin for Android instead of Java?***

There is no cost associated alongside programming with the *Kotlin* language. It leverages already existing tools—including *Android Studio*. *Android Studio* is the industry standard integrated development environment for *Android* application development. The newest version supports *Kotlin* by default. Both *Kotlin* and the recommended integrated development environments are both: Under a free software license, have no monetary cost for individuals, and no monetary cost for teams. *JetBrains* has a free plug-in for educators that allows for the creation of tutorials withing the *Android Studio* application. This means educators can teach from within the tools the students will be using.

*Kotlin* keeps true with it’s promise to remove boilerplate. A comparison of a standard *Java* class with an equivalent *Kotlin* class shows how much *Kotlin* chooses to abstract. A developer only has to define the name of the class, and the attributes contained within. All other boilerplate code such as constructors, getters/setters, and commonly overridden methods—such as the equals method—are implicitly generated for the developer. There are also plug-ins available that abstract several *Android* specific snippets. For example, it allows directly calling views, and several helper functions for *SQLite*.

Current published benchmarks by third party show some possible performance costs involved in using *Kotlin* for *Android*. In some instances, *Kotlin* has shown a 300% decrease in compilation performance. However, other benchmarks have shown an increase in compilation time on partial builds. Ultimately, it should be decided by the department if ease-of-code is worth the possibility of worse performance.

Unfortunately, *Kotlin* simply does not have the market presence that *Java* has. Documentation—both official and community—is far more abundant for *Java*. 2017 statistics from both *Stack Overflow* and *GitHub* show that *Kotlin* currently has little to no presence, especially compared to *Java* which commonly ranks high on most used programming language lists. On top of this, jobs being offered in the Ottawa/Gatineau area asking for *Kotlin* experience are few. Since *Kotlin* currently does not increase the skill-set of students in a way that increases their chance of getting a career after their educations, ***it is not recommended to use Kotlin for Android application development courses.***

# Introduction

The Heritage College Computer Science Program contains a course on mobile application development. Specifically, applications for the *Android* mobile platform. The course is currently taught using the popular enterprise programming language *Java,* which students are taught starting in the course *Programming I* during their first semester. While *Java* is still going strong in the *Android* world, there is a new programming language that is attempting to make waves. *Kotlin* is a language looking to improve upon *Java—*making programming for *Android* a bit more comfortable. Running on the same technologies as *Java*, *Kotlin* is looking to become an enterprise language alternative to *Java—*similar to the programming languages *Clojure* and *Scala*. *Kotlin* boasts being a far more productive language to type than *Java—*with specific focus in the *Android* realm*—*making it easier to program with. It contains several out-of-the-box improvements to the typical *Java* syntax, and has several improvements for *Android* specific hardships included in an official plug-in. This begs the question: Should the Computer Science Program start teaching *Kotlin* for *Android* as opposed to *Java* for *Android*? This research paper will look over an overview of the *Kotlin* programming language, the cost of setting up a *Kotlin* development environment, how *Kotlin* differs syntactically from *Java*, performance benchmarks comparing *Kotlin* to *Java*, documentation availability of *Kotlin* vs. *Java*, the staying power of *Kotlin*, employment opportunities opened to students because of their knowledge of *Kotlin*, and tools available for educators to teach *Kotlin* in a classroom.

# 1 – What Is Kotlin?

*Kotlin* is a programming language designed, and primarily developed by the software company *JetBrains. JetBrains* is a well known software development company. They have developed several successful integrated development environments such as the *InteliJ IDEA* for *Java* and *PyCharm* for *Python*. *JetBrains* also has a suit of plug-ins for *Microsoft’s Visual Studios* IDE*—*including the plug-in *ReSharper* that is regarded as an essential plug-in for serious *.NET* developers. Development of *Kotlin* began due to *JetBrains* need for a language that: ran via the *Java Virtual Machine,* fit their feature needs, and performed at a respectable speed. No existing language met that requirement, so work on their own internal programming language began [9]. *Kotlin* was announced in 2011, and was open-sourced in 2012 under the *Apache 2.0* free software license. *Kotlin* was built specifically to address the short-comings of *Java*, and runs in all environments that *Java* traditionally can. *Kotlin* can run in the *Java Virtual Machine*, can be *transpiled* into the *JavaScript* programming language for use in web applications, can be compiled against the native machine for use without the *Java Virtual Machine,* and can be used to develop applications for the *Android* platform [7]. The official *Kotlin* web page boasts that *Kotlin* can:

* *“...drastically reduce the amount of boilerplate code.”*
* *“Avoid entire classes of errors, such as null pointer exceptions.”*
* *“Leverage existing libraries...”*

As well, the page states that developers can: *“Choose any Java IDE or build from the command line”* [7].

# 2 – Monetary Cost Of Kotlin

Tooling for building *Android* applications using *Kotlin* is readily available. As of version 3.0, *Android Studio*—the official integrated development environment for *Android*—has native support for the *Kotlin* programming language. This means developers can develop *Android* applications using *Kotlin* without any additional set-up. *Android Studio* contains an action for converting already existing proper *Java* code to *Kotlin*. On top of that, *Android Studio* will suggest converting *Java* to *Kotlin* if you attempt to paste a snippet of *Java* into a *Kotlin* file [6]. The popular *Java* integrated development environment *Eclipse* can also handle *Kotlin,* albeit necessary to download an—also free—plug-in. A trend with both the *Kotlin* programming language and the listed integrated development environments, are that they are *free*. Both in cost, but also in licensing. *Eclipse* and *Android Studio* are both available to all to download for no cost. The source code for *Eclipse,* and *Android Studio* are under the *Eclipse Public License* and *Apache 2.0* respectively*.* Both of these licenses are free software licenses—which is appropriate for a language that is also under a free software license. Neither of these integrated development environments are mandatory to program for *Android* using *Kotlin*. Any text editing program can be used to write *Kotlin.* This, combined with zero monetary investment, makes it extremely accessible to begin programming in *Kotlin*.

# 3 – Tools For Educators

*JetBrains* has released a tool specifically for educators. A free plugin for either *Android Studio* or *IntelliJ IDEA* called *EduTools* can be used to easily create interactive lessons. This tools allows educators to design exorcises for students to be completed within the development environment that students will have to become comfortable with. It allows inserting partially completed code with specific placeholder locations meant to be filled out by students. Educators can develop automated tests to check the answers given by students for correctness. Courses can be exported for use by other educators and can be tied into the IT focused education platform *Stepik* [5]. While this tool doesn’t negate the need for a lesson plan to be created, it does take some weight of the teachers shoulders as far as *how* to introduce the language.

# 4 – Java and Kotlin Syntax Comparison

Developers who are familiar with *Java*, or any object-oriented programming language, will find getting started typing *Kotlin* very simple. As with *Java*, a *Kotlin* code-base is spread across multiple classes—with each class containing some amount of properties, methods, comments, etc. Data types remain the same as they are in *Java*—as do Boolean-statements, and loops. Methods have a return type, and none-or-many parameters. All of these core concepts are the same as they are in *Java*. The largest difference between *Kotlin* and *Java* is the actual syntax. *Kotlin* is extremely terse, requiring significantly fewer lines-of-code than an equivalent *Java* application. A *Java* class containing constructors, properties, accessors, mutators, and various default method overrides—equals, toString, hashCode, etc—can easily reach up to hundreds lines-of-code. Since most typical *Java* classes will require all these to fulfill its purpose, the tedious tasks is often automated away by the development environment. *Kotlin* removes the verbosity, instead choosing to make assumptions:

data class User (var name: String,  
 var age: Int,   
 var password: String)

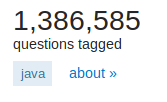
The above code handles all of the functionality our standard *Java* class must explicitly contain, including overriding some default methods [8]. There are several other examples of *Kotlin* code being much more efficient to write, with the explicit declarations of *Java* replaced with *Kotlin’s* assumptions. *Kotlin* also includes several concepts that either do not exist in *Java*—or have only recently been added—such as: extension methods, the safe call operator, and higher-order functions [10]. *Kotlin* also shines when it comes to *Android* development. With the inclusion of the *Kotlin Android Extensions*, and *Anko* plug-ins, *Kotlin* simplifies most hardships a developer would have with *Java* for *Android*. Calling views directly without having to find by identifier, helper functions for *SQLite,* instant *toasts*/*alerts*/*snackbars*, etc [8]. *Kotlin’s* terse syntax is quicker to parse—as well as more efficient to type—than *Java*’s verbose syntax. This makes the *Kotlin* programming language inherently quicker to write and maintain than *Java*.

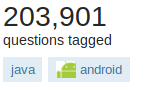
# 5 – Kotlin Performance

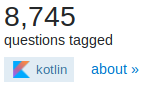
*Kotlin* is inherently more efficient to write, but it’s compile time efficiency has come into question. Does the additional layer of *Kotlin* cause an increase in compile time compared to *Java*? A series of blog posts by *Android* developer *Christophe Beyls* titled *Exploring Kotlin’s Hidden Costs* contains the results of several benchmarks ran against *Kotlin* and *Java*—specifically inside the *Java Virtual Machine*. These benchmarks returned the following results: Local functions, null safety operations, indirectly accessing ranges, non-primitive ranges, range iteration with explicit steps, and the use of indices on custom collections did not result in a significant performance loss with *Kotlin* compared to *Java*. However, *Kotlin’s* Varags and Spread Operator doubles performance expense due to a redundant array copy, delegate properties resulted in an approximate 10% performance loss, and making a forEach call against a range resulted in a **300%** performance cost [2]. *Evan Tatarka*—software engineer at the software development agency *Willowtree*—took these benchmarks and ran them against the *Android* *Runtime.* The conclusion was that these costs found on the *Java Virtual Machine* also exists on the *Android Runtime* [3]. *Aj Alt* of *Keepsafe Engineering* ran build benchmarks to compare *Kotlin* to *Java* on the *Java Virtual Machine*. He found that on a clean build, *Java* was more optimized than *Kotlin* between 10% and 15%. However, on partial builds—which are arguably more common—*Kotlin* was as fast or **faster** than *Java* [1]. In the end, it’s a game of trade-offs. Is the decrease in code production thanks to *Kotlin’s* abstractions worth the possible increase in build times?

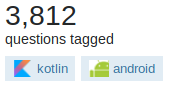
# 6 – Kotlin Documentation Availability

Both *Java* and *Kotlin* have full documentation pages readily available. *Java* has the huge *Java Platform Standard Edition API Specifications*, while *Kotlin* has a full reference style documentation page online. The *Kotlin* documentation is arguably easier to parse through due to it’s size, as well as the fact that it presents high level concepts to the readers upfront. Ideas such as: classes, objects, functions, etc. are given right away—and lower level info such as each class in the standard library is hidden a layer deeper. However, *Java’s* age lends it an advantage. Any search for issues regarding *Java* for *Android* will be met with a solution, due to the fact that *Java* has had far more years of developers hacking on it. Using the popular Q&A forum *Stack Overflow* shows that there are *1,386,585* questions tagged with *Java* [Screenshot 1], and *203,901* questions with both the tags *Java* and *Android* [Screenshot 2]*.* Whereas only *8,450* questions are tagged as *Kotlin* [Screenshot 3], and only *3,812* tagged with both *Kotlin* and *Android* [Screenshot 4]. Of course, *Kotlin’s* documentation available online will increase over time, assuming it stays relevant.

  
Screenshot 1: Questions tagged with Java as of March 11th, 2018

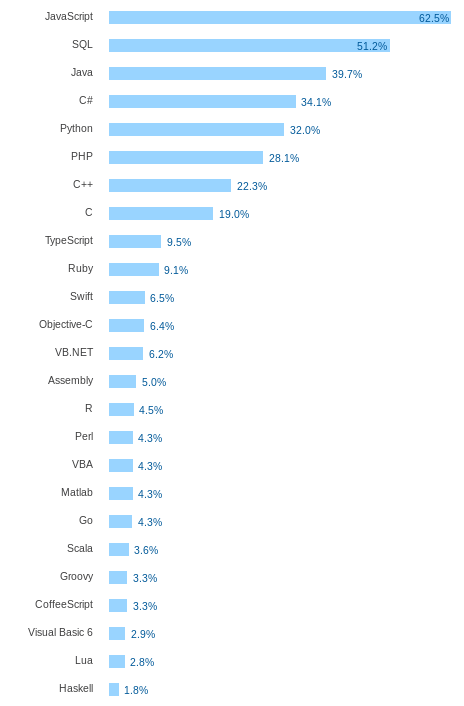
  
Screenshot 2: Questions tagged with Java and Android as of March 11th, 2018

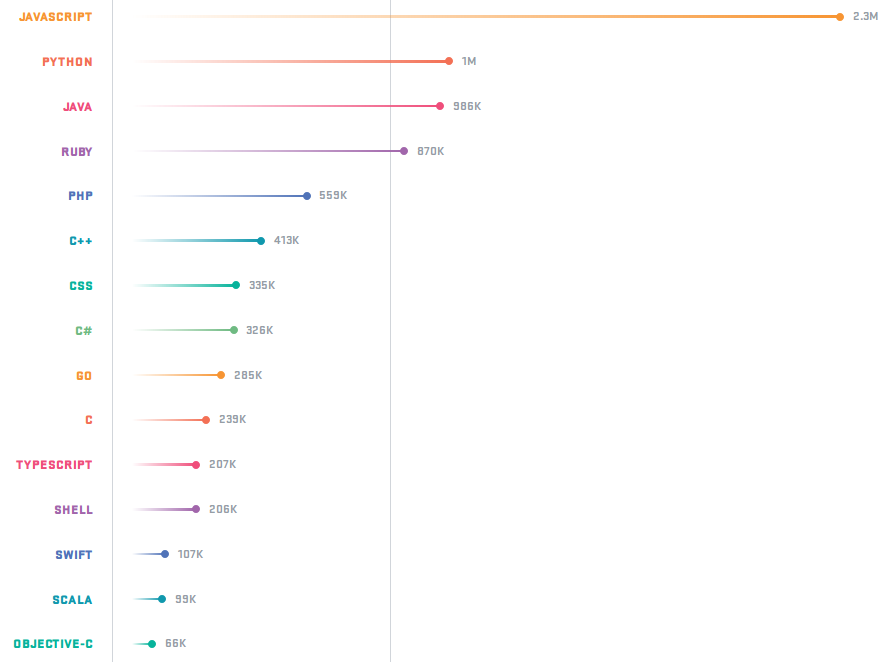
  
Screenshot 3: Questions tagged with Kotlin as of March 11th, 2018

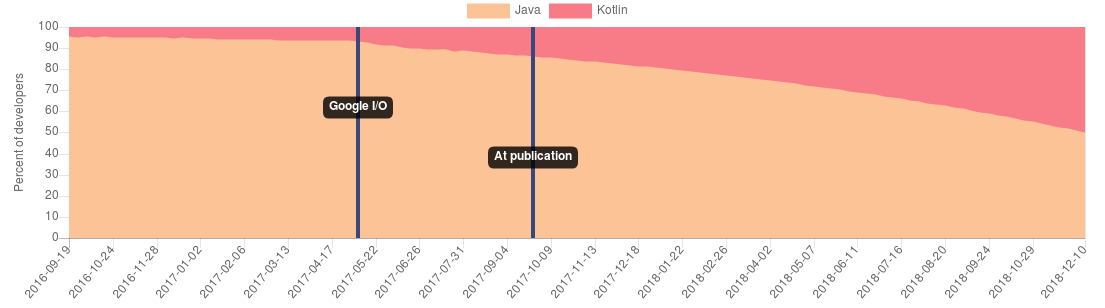
  
Screenshot 4: Questions tagged with Kotlin and Android as of March 11th, 2018

# 7 – Kotlin Staying Power

Does the *Kotlin* programming language have staying power? On *Stack Overflow*’s 2017 *Developer Survey*, *Kotlin* does not appear **anywhere**. It is does not appear on the list of loved language, wanted language, languages by occupation, or even on dreaded languages [12] [Screenshot 5]. The same can be said for *GitHub’s Octoverse 2017*, where *Kotlin* does not appear on the list of most used languages [4] [Screenshot 6]. However, data technology company *Realm* paints a more positive picture of *Kotlin.* Their Q4 2017 report shows *Kotlin’s* rate of growth speeding up after being officially brought to *Android* after *Google I/O 2107.* It jumps for 7.4% of developers using *Kotlin*, to 14.7% at the time the report was published. *Realm* forecasts that *Kotlin* will overtake *Java* in percentage of developers by **December 2018** [11][Screenshot 7]. However, *Realm’s* data-set is pulled from their own community of developers. As *Realm’s* audience is focused on up-and-coming technologies, the data-set may be skewed to appear as if the adoption rate is greater than it actually is.

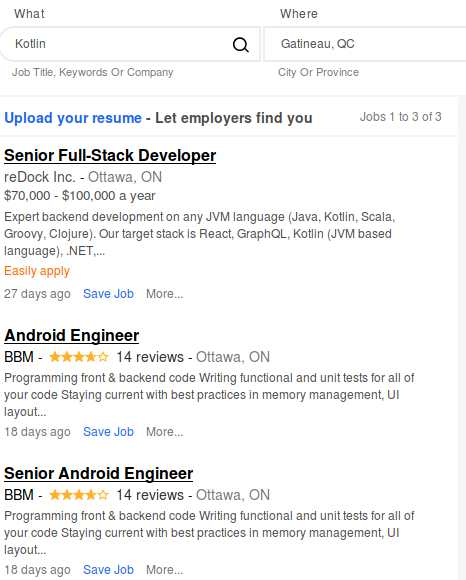
  
Screenshot 5: Stack Overflow 2017 list of most popular languages, sorted by % of developers that marked the language as liked.

  
Screenshot 6: Most popular programming languages on GitHub, ranked by opened repositories.

  
Screenshot 7: Projected % of developers adopting the Kotlin programming language, as per the Realm Report Q4.

# **8 – Kotlin and the Job Market**

The Computer Science Program is specifically tailored for students to be able to enter the job force after graduating. Specifically: focusing on public sector jobs and local private industries. In order for *Kotlin* to be worth teaching, it must increase the chances of students being hired. To determine this, the output of two large job posting aggregation sites—*Indeed* and the *Government of Canada’s Job Bank*—has been looked at to determine positions asking for *Kotlin* as either a main language, or a bonus language. A search on *Indeed—*performed on the March 11th, 2018—found only three job postings looking for experience with *Kotlin* [Screenshot 8]. A *Job Bank* search performed the same day found **no** job postings for *Kotlin* around Gatineau, Quebec. This, unfortunately, means *Kotlin* is currently not a sought after skill. While *Kotlin* usage may increase with time as the previously mentioned *Realm* report has foreseen, currently there is no evidence of *Kotlin* increasing a students chance of becoming employed.

  
Screenshot 8: Indeed search results for Kotlin jobs near Gatineau, QC.

# **Conclusion**

Based on the evidence given above, the Computer Science Program should **not** switch to *Kotlin* for *Android* development. *Kotlin* contains cleaner syntax, specialized tools for educators, and performance is arguably on par with *Java*. However, the uncertainty as to the staying power of *Kotlin—*as well as the lack of employment opportunities—bars *Kotlin* from becoming an independent language to teach *Android*. While *Kotlin* might be nice to teach alongside *Java*—especially since at this point students have had a lot of exposure to *Java*—it does not have the influence to warrant ejecting *Java* for *Android* as the primary language. Should *Kotlin’s* popularity rise and we see the adoption of *Kotlin* from the public sector or local private industries, then *Kotlin* should be seriously considered as a replacement for *Java*. Until that point, *Java* must remain the primary *Android* platform development language.

# References

[1] Alt, A. J. “Kotlin vs Java: Compilation Speed.” *Keepsafe Engineering*, 8 Sept. 2016, https://medium.com/keepsafe-engineering/kotlin-vs-java-compilation-speed-e6c174b39b5d.

[2] Beyls, Christophe. “Exploring Kotlin’s Hidden Costs — Part 1.” *Christophe Beyls*, 5 June 2017, https://medium.com/@BladeCoder/exploring-kotlins-hidden-costs-part-1-fbb9935d9b62.

[3] Engineer, Evan TatarkaSoftware. *Kotlin’s Hidden Costs - Android Benchmarks*. https://willowtreeapps.com/ideas/kotlins-hidden-costs-android-benchmarks. Accessed 3 Feb. 2018.

[4] “GitHub Octoverse 2017.” *GitHub Octoverse 2017*, https://octoverse.github.com/. Accessed 1 Mar. 2018.

[5] “Kotlin Edu: Learn and Teach Kotlin with Easy and Professional Tools.” *JetBrains*, https://www.jetbrains.com/education/kotlin-edu/. Accessed 3 Feb. 2018.

[6] “Kotlin for Android - Kotlin Programming Language.” *Kotlin*, https://kotlinlang.org/docs/reference/android-overview.html. Accessed 3 Feb. 2018.

[7] “Kotlin Programming Language.” *Kotlin*, https://kotlinlang.org/. Accessed 3 Feb. 2018.

[8] *Kotlin VS Java: Basic Syntax Differences*. http://yalantis.com/blog/kotlin-vs-java-syntax/. Accessed 3 Feb. 2018.

[9] Krill, Paul. “JetBrains Readies JVM-Based Language.” *InfoWorld*, 22 July 2011, https://www.infoworld.com/article/2622405/java/jetbrains-readies-jvm-based-language.html.

[10] Lima, Pedro. “Kotlin vs. Java: First Impressions Using Kotlin.” *ArcTouch*, 3 May 2017, https://arctouch.com/2017/05/kotlin-vs-java/.

[11] *Realm Report Q4 2017*. https://realm.io/realm-report/. Accessed 10 Feb. 2018.

[12] “Stack Overflow Developer Survey 2017.” *Stack Overflow*, https://stackoverflow.com/insights/survey/2017/?utm\_source=so-owned&utm\_medium=social&utm\_campaign=dev-survey-2017&utm\_content=social-share. Accessed 1 Mar. 2018.