

---

# Eine Vergleichsstudie: Java vs. Kotlin in Bezug auf Einsteiger in der Android-App-Entwicklung

**Franz Lea**

FH JOANNEUM Kapfenberg  
Graz, Austria  
lea.franz@edu.fh-joanneum.at

**Wieser Stefanie**

FH JOANNEUM Kapfenberg  
Graz, Austria  
stefanie.wieser2@edu.fh-joanneum.at

**Trummer Julia**

FH JOANNEUM Kapfenberg  
Graz, Austria  
julia.trummer@edu.fh-joanneum.at

**Abstract**

UPDATED—May 18, 2020. This paper presents the results of a comparative study about Kotlin and Java as programming languages for realizing Android Applications. In particular we cover different fields like readability, coding costs, and code structure. Furthermore, we undertook a closer look at the new features of Kotlin. Conclusively we will go over which coding language fits beginners.

**Author Keywords**

Java; Kotlin; Application Development; Android; Beginner;

**Einleitung**

Derzeit werden immer mehr Android-Applikationen mit Kotlin entwickelt. Diese Programmiersprache ist im Gegensatz zu Java relativ neu und wurde erstmals 2016 veröffentlicht. Zugleich ist Kotlin seit dem ersten Release als Open-Source-Software erhältlich und zielt im speziellen JVM (Java Virtual Machine) und Android an.[4] In diesem Paper vergleichen wir Java mit Kotlin in den Aspekten Aufwand und Struktur. Darüber hinaus gehen wir auch auf einige Sprachfeatures die Kotlin bietet ein. Zur Unterstützung unserer Vergleichsstudie haben wir uns eine App ausgedacht und in beiden Programmiersprachen (Java und Kotlin) umgesetzt. Ziel dieser Studie ist herauszufinden ob die Android Entwicklung in Java oder Kotlin einfacher für Einsteiger ist.

---

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).

Copyright held by the owner/author(s).  
CHI'20, April 25–30, 2020, Honolulu, HI, USA  
ACM 978-1-4503-6819-3/20/04.  
<https://doi.org/10.1145/3334480.XXXXXXX>



**Figure 1:** JetBrains Kotlin Logo.  
Photo: © JetBrains s.r.o.

## Kotlin

Kotlin ist gleich wie Java eine objektorientierte Programmiersprache. Ein in dieser Sprache geschriebener Code kann in Bytecode für die JVM (Java Virtual Machine) kompiliert werden. Mit Kotlin können große Teile der Java-Boilerplate-Codes<sup>1</sup> verhindert werden.[5]

## Java

Java ist eine objektorientierte Programmiersprache und besteht aus den JDK (Java-Entwicklungstools) und der JRE (Java-Laufzeitumgebung). Grundsätzlich wird diese Programmiersprache für die Android Entwicklung verwendet, weil sie sehr bekannt ist.

## Lesbarkeit

Viele Entwickler meinen, dass Kotlin eine bessere Syntax als Java hat und deshalb besser zu lesen ist. Nun ein Beispiel zu dieser Aussage: Wenn man versucht einen fremdsprachigen Text zu lesen, ist es schwierig einen Satz zu verstehen ohne, dass man die Bedeutung der einzelnen Wörter kennt. Versteht man aber mehrere Wörter und den Kontext ist, es einfacher den Text zu lesen. Deshalb hat die Wahl der Sprache keinen Einfluss auf die Lesbarkeit, solange der Kontext verständlich ist. Was die Lesbarkeit betrifft gibt es für Programmiersprachen keine objektiven Metriken.

## Sprachfeatures

Kotlin bietet viele neue sprachenspezifische Features, die es von Java unterscheiden und Beginners den Einstieg erleichtern. Im Folgenden wird auf vier davon aus der Prototyp Applikation eingegangen.



**Figure 2:** Java Technologies Java Logo © Java Technologies.

### Null Sicherheit

Die größten Fehlerquellen in Java-Systemen sind fehlende Null-Checks und die daraus entstehenden Null-Pointer Exceptions zur Laufzeit. In Kotlin können diese Fehler bereits zur Kompilierzeit aufgezeigt werden durch die klare Trennung von Referenzen, die Null sein können und Referenzen, die es nicht sein können. Variablen müssen dafür explizit als Nullable gekennzeichnet werden durch das Anhängen eines Fragezeichens an den Datentyp bei der Variablendeklaration. Der Safe-Call-Operator (?.) kann dafür verwendet werden, dass immer entweder ein Ergebnis oder Null (ohne Null-Pointer Exception) zurückgegeben wird. [2]

```
gpsText?.text = slide.GPS
```

### Datenklassen

Um Klassen als Datencontainer verwenden zu können, braucht es in Java min. einen Konstruktor und Getter/Setter. In Kotlin kann mit dem Keyword "data" eine Datenklasse erstellt werden, die ohne diesen Boilerplate-Code auskommt.

```
data class Slide(val id:Int, var title:String = "untitled", ...)
```

Der Compiler generiert hier Getter und Setter sowie die Methoden copy(), toString(), hashCode() und equals() automatisch im Hintergrund. [3]

### Objekt Deklaration und Singletons

In Kotlin gibt es das Sprachkonstrukt der Objekt Deklaration, mit dem das Singleton Pattern einfacher und kürzer als in Java implementiert werden kann und somit nur mehr eine Instanz einer Klasse erstellt werden kann. Dafür wird das Keyword "class" mit dem Keyword "object" ausgetauscht.

```
object Slideshow {...}
```

Einer Objekt Deklaration können Methoden und Properties genau wie einer Klasse hinzugefügt werden. [2]

<sup>1</sup>Boilerplate-Code: Sind häufig verwendete und meist unveränderte Code-Snippets mit gleicher Funktion.

Dateiname	LoC	Kotlin
		Methoden
Slide.kt	15	1
Slideshow.kt	31	6
MainActivity.kt	119	5
SharedPreferences.kt	16	2

**Figure 3:** In dieser Tabelle sind alle wichtigen Files, der in Kotlin implementierten Applikation samt Zeilenanzahl und Methodenanzahl enthalten.

Dateiname	LoC	Java
		Methoden
Slide.java	57	14
ObservedObject.java	16	2
Slideshow.java	56	7
MainActivity.java	133	7
SharedPref.java	17	2

**Figure 4:** In dieser Tabelle sind alle wichtigen Files, der in Java implementierten Applikation samt Zeilenanzahl und Methodenanzahl enthalten.

### Delegated Properties und Observer

Auch mit diesem Sprachfeature gibt es eine Erleichterung bei der Implementierung eines Design Patterns im Gegensatz zu Java. Anstatt einer eigener Implementierung des Observer Patterns kann das in Kotlin integrierte Observable Delegate verwendet werden. Dabei ruft die Delegated Property eine Callback Funktion mit drei Parametern bei einer Änderung auf.

```
private var currentSlideId: Int by
Delegates.observable(0) { prop, oldSlideId,
newSlideId -> .... }
```

Eine Delegated Property funktioniert prinzipiell wie eine reguläre Property, aber delegiert das Schreiben und Lesen von Werten an andere Funktionen. [1]

### Lines of Code (LoC)

Mittels dieser Metrik sollen die Umfänge der beiden Prototypen miteinander verglichen werden. Das Ziel ist es heraus zu finden welche der beiden Apps eine geringere Anzahl an Programmzeilen und Methoden hat.<sup>2</sup>

### Vergleich

Wie man deutlich an der Anzahl an Codezeilen sehen kann, hat der Java Prototyp um etwa ein Drittel mehr Zeilen als die in Kotlin implementierte Applikation. Desweiteren ist deutlich zu erkennen das die Anzahl an Methoden in der in Java implementierten Applikation, doppelt so hoch ist wie in der Kotlin App. Dieser eindeutige Unterschied im Umfang entsteht durch die in Kotlin integrierten Sprachfeatures. Einen wesentlichen Beitrag dazu leisten die in Kotlin schnell zu implementierenden Patterns wie Singleton und Observer und die Verwendung einer Data Class, bei der auf den Boilerplate-Code verzichtet werden kann. Ein weiterer Unterschied liegt bei dem Thema durchmischen einer ArrayList. Bei Kotlin gibt es bereits eine fertige Methode, die

<sup>2</sup>Klicken sie hier: <https://github.com/LeaFranz/KotlinJavaSlideshow>

man direkt verwenden kann. In der in Java implementierten Applikation braucht man dazu Collections und Comparater. Dadurch wurden aus einer einfachen Methode, eine sehr viel aufwändigere, die mit einigen Zeilen an zusätzlichen Code nur das gleiche Ergebnis erzielt.

### Fazit

Zusammenfassend kann man also sagen das die in Kotlin implementierte Applikation nicht nur um ein Drittel des Codes kürzer, sondern auch durch die einzelnen Sprachfeatures deutlich einfacher zu implementieren ist. Durch die Analyse der Prototypen anhand der verwendeten Metriken (Lesbarkeit, Sprachfeatures und Lines of Code) sind wir zu dem Schluss gekommen, dass wir einem Einsteiger für die Android App Entwicklung Kotlin empfehlen würden.

### REFERENCES

- [1] Husayn Hakeem. 2018. The magic in Kotlin: Delegates. (05 February 2018). <https://proandroiddev.com/the-magic-in-kotlin-delegates-377d27a7b531> Accessed: 18. May 2020.
- [2] Marcin Moskala and Igor Wojda. 2017. *Android Development with Kotlin*. Packt Publishing Ltd.
- [3] Stephen Samuel and Stefan Bocutiu. 2017. *Programming kotlin*. Packt Publishing Ltd.
- [4] Vasiliy. 2017. Kotlin vs Java The Whole Story. (07 September 2017). <https://www.techyourchance.com/kotlin-vs-java-whole-story/> Accessed: 18. May 2020.
- [5] Vasiliy. 2020. Was versteht man unter einer Boilerplate? (2020). <https://kulturbanause.de/faq/boilerplate/> Accessed: 18. May 2020.