

Kotlin vs. Java

Group Members:

- Zhewei Wu
- Ko Yat Chan
- Hassan Yakefujiang
- Ryan Hoffman
- Ziming (Joanna) Fang

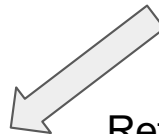
Date: Sep 15, 2020



Popularity of Programming Languages

Worldwide, Sept 2020 compared to a year ago:

Rank	Change	Language	Share	Trend
1		Python	31.56 %	+2.9 %
2		Java	16.4 %	-3.1 %
3		Javascript	8.38 %	+0.3 %
4		C#	6.5 %	-0.8 %
5		PHP	5.85 %	-0.5 %
6		C/C++	5.8 %	+0.0 %
7		R	4.08 %	+0.3 %
8		Objective-C	2.79 %	+0.2 %
9		Swift	2.35 %	-0.1 %
10		TypeScript	1.92 %	+0.1 %
11		Matlab	1.65 %	-0.1 %
12		Kotlin	1.61 %	+0.1 %



Reference: <http://pypl.github.io/PYPL.html>

Kotlin: The Rise of a Newborn Language

- Google created Android SDK in 2009 which allows developers to create apps using **Java**.
- Oracle ended up suing Google for patent and copyright infringement based on Google's use of Oracle's Java APIs.
- The lawsuit has been going on for about 10 years.
- In the meantime, Google tried to move away from Java as soon as possible by working with JetBrains, a company that created **Kotlin** in 2011.

ORACLE

vs

Google

What is Kotlin?

- An officially supported language for developing Android apps.
- A cross-platform, statically typed, general-purpose programming language with type inference
- Is the most strongly supported JVM language in the Android ecosystem.
- Is used for Android development, web development, server-side development and more



Features of Kotlin

1. Brevity
2. Interoperability
3. Inbuilt Null Safety
4. No Raw Types
5. No Checked Exceptions

Brevity

Java: 18 lines

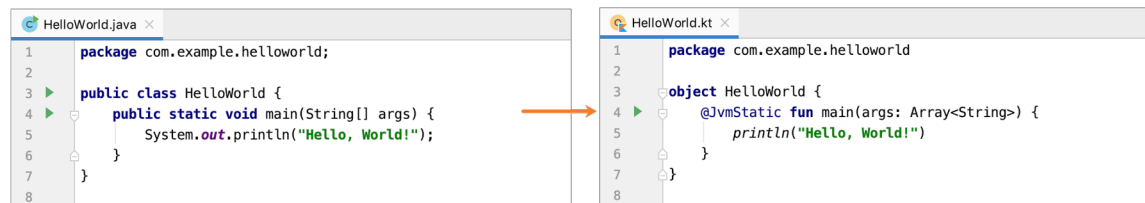
```
public class ClearBridge {  
  
    public static double calculate (double a, String op, double b) throws Exception {  
        switch (op) {  
            case "add":  
                return a + b;  
            case "subtract":  
                return a - b;  
            case "multiply":  
                return a * b;  
            case "divide":  
                return a / b;  
            default:  
                throw new Exception();  
        }  
    }  
}
```

Kotlin: 9 lines

```
fun calculate (a: Double, op: String, b: Double): Double {  
    when (op) {  
        "add" -> return a + b  
        "subtract" -> return a - b  
        "multiply" -> return a * b  
        "divide" -> return a / b  
        else -> throw Exception()  
    }  
}
```

Interoperability

- Easy conversion: Java → Kotlin with only one extension



```
1 package com.example.helloworld;
2
3 public class HelloWorld {
4     public static void main(String[] args) {
5         System.out.println("Hello, World!");
6     }
7 }
8
```

```
1 package com.example.helloworld
2
3 object HelloWorld {
4     @JvmStatic fun main(args: Array<String>) {
5         println("Hello, World!")
6     }
7 }
8
```

- Kotlin is 100% inter-operable with Java
 - Kotlin modules works within existing Java code
 - Java source code can be added to an existing Kotlin project

Null Safety

- **NullPointerExceptions** causes huge frustration for developers. It allows users to assign null to any variables but while accessing an object reference having null value raises a null pointer exception which user needs to handle.
- Unlike Java, all types are non-nullable in Kotlin by default. **Kotlin** also has a **safe** call operator, to avoid methods being called on objects with a **null** reference

```
var a: String = "abc"  
a = null // compilation error
```

```
var b: String? = "abc"  
b = null // ok
```

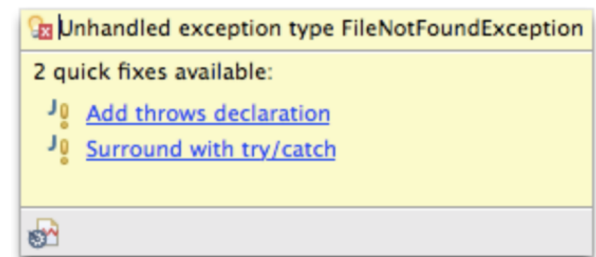

No Raw Types

- Kotlin doesn't have raw types; you have to specify the type parameter.
- In Java, using raw types such as “List” instead of “List<Integer>” can lead to `ClassCastException`.
- Java Raw Types are translated into Star Projections for interoperability.

An example: — `List` becomes `List<*>!`, i.e. `List<out Any?>!`.

No Checked Exceptions

- Kotlin removes this feature entirely
- What is Checked Exception
 - exceptions that get checked at compile-time.
 - must be handled with try-catch block or throws
 - Exp. FileNotFoundException, ClassNotFoundException
- Why is it problematic
 - Often unnecessary (empty catch blocks)
 - Inconvenient for developers



Java Background

- A class-based, object-oriented programming language
- first appeared in 1995, designed by James Gosling
- one of the most popular programming languages in use

Commonly used for:

- server-side language for most back-end development projects
- desktop computing, other mobile computing, games, and numerical computing



Major Similarities between Kotlin and Java

- Both are Statically Typed languages
 - Variable type is known at compile time as opposed to run time
- Both languages compile into Bytecode
 - Meaning that they can be executed by the Java Virtual Machine (JVM)
- Existing Java frameworks and libraries are available to both languages
- Entry point to a program written in either language is the 'Main' function
- They implement similar garbage collection algorithms

Major Differences between Kotlin and Java

1. Checked Exceptions (mentioned)
2. Null safety (mentioned)
3. Code Conciseness (Similar to Brevity)
4. Extension Functions
5. Higher-Order Functions and Lambdas

Difference #4.

Extension Functions

- Kotlin allows developers to extend a class with new functionality via extension functions.
- Creating an extension function is easy in Kotlin.

Difference #5.

Higher-Order Functions and Lambdas

- Kotlin functions are first-class. This means that they can be stored in data structures and variables.

To Summarize

For general-purpose programming, Java gains the upper hand.

On the flip side, more and more developers and organizations are adopting Kotlin for rapidly developing Android applications.

Discussion

Q: Do you think Kotlin will replace Java for Android app Development in the near future?