Mobile Application Development

Produced by Dave Drohan (david.drohan@setu.ie)
Department of Computing & Mathematics
South East Technological University
Waterford, Ireland

Updated & Delivered by Gongzhe Qiao (003969@nuist.edu.cn)
Department of Computer Science
Nanjing University of Information Science and Technology
Nanjing, China

nuist.edu.cn setu.ie

Introducing Kotlin

Overview – Part 2

Agenda

- What is Kotlin?
- Context - Java Vs JVM
- Kotlin History
- Milestones
- General Overview & Features
- Kotlin & IntelliJ IDEA
- Conclusion

Agenda

- What is Kotlin?
- Context - Java Vs JVM
- Kotlin History
- Milestones
- General Overview & Features
- Kotlin & IntelliJ IDEA
- Conclusion

Kotlin Overview

Kotlin

Based on : https://en.wikipedia.org/wiki/Kotlin_(programming_language)

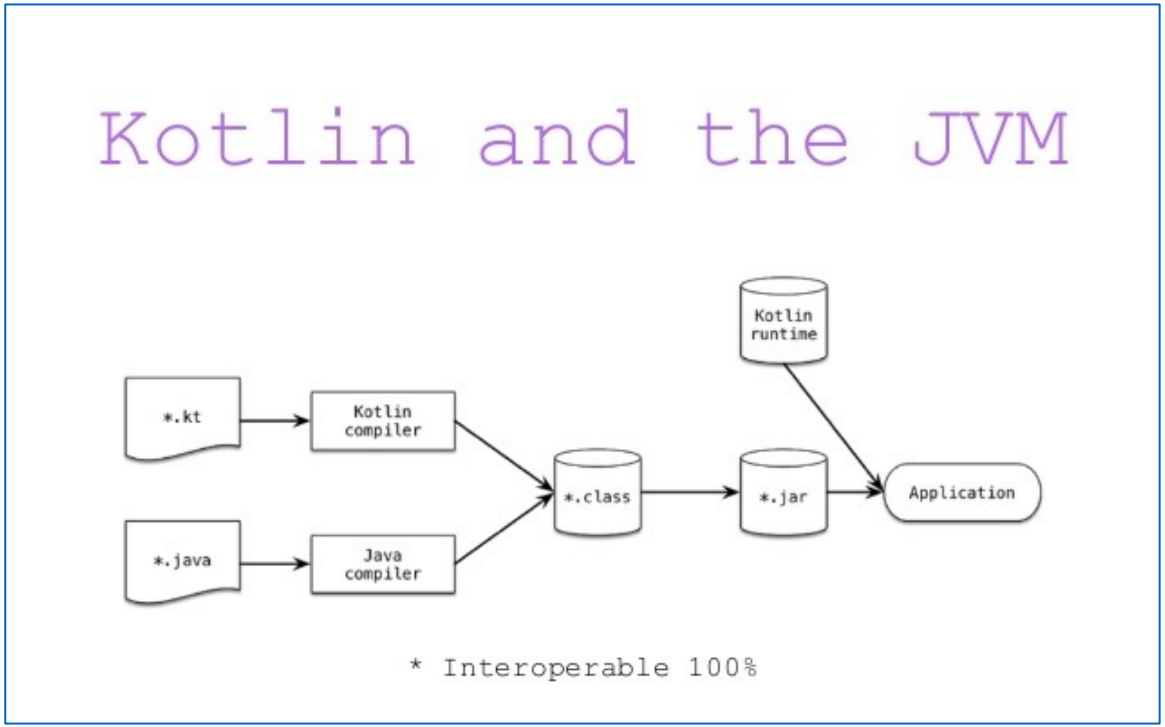
- runs on Java Virtual Machine (JVM).
- is an evolution of the Java syntax but is more concise and has cleaner syntax.
- is not syntax compatible with Java; but is interoperable with Java.
- relies on some Java Class Libraries
  e.g. Collections framework.
- is a statically-typed programming language.
- offers null safety.

Kotlin

Based on : https://en.wikipedia.org/wiki/Kotlin_(programming_language)

- runs on Java Virtual Machine.
- is an evolution of the Java syntax but is more concise and has cleaner syntax.
- is not syntax compatible with Java; but is interoperable with Java.
- relies on some Java Class Libraries
  e.g. Collections framework.
- is a statically-typed programming language.
- offers null safety.

Runs on Java Virtual Machine



Kotlin compiles to JVM ByteCode (like Java)

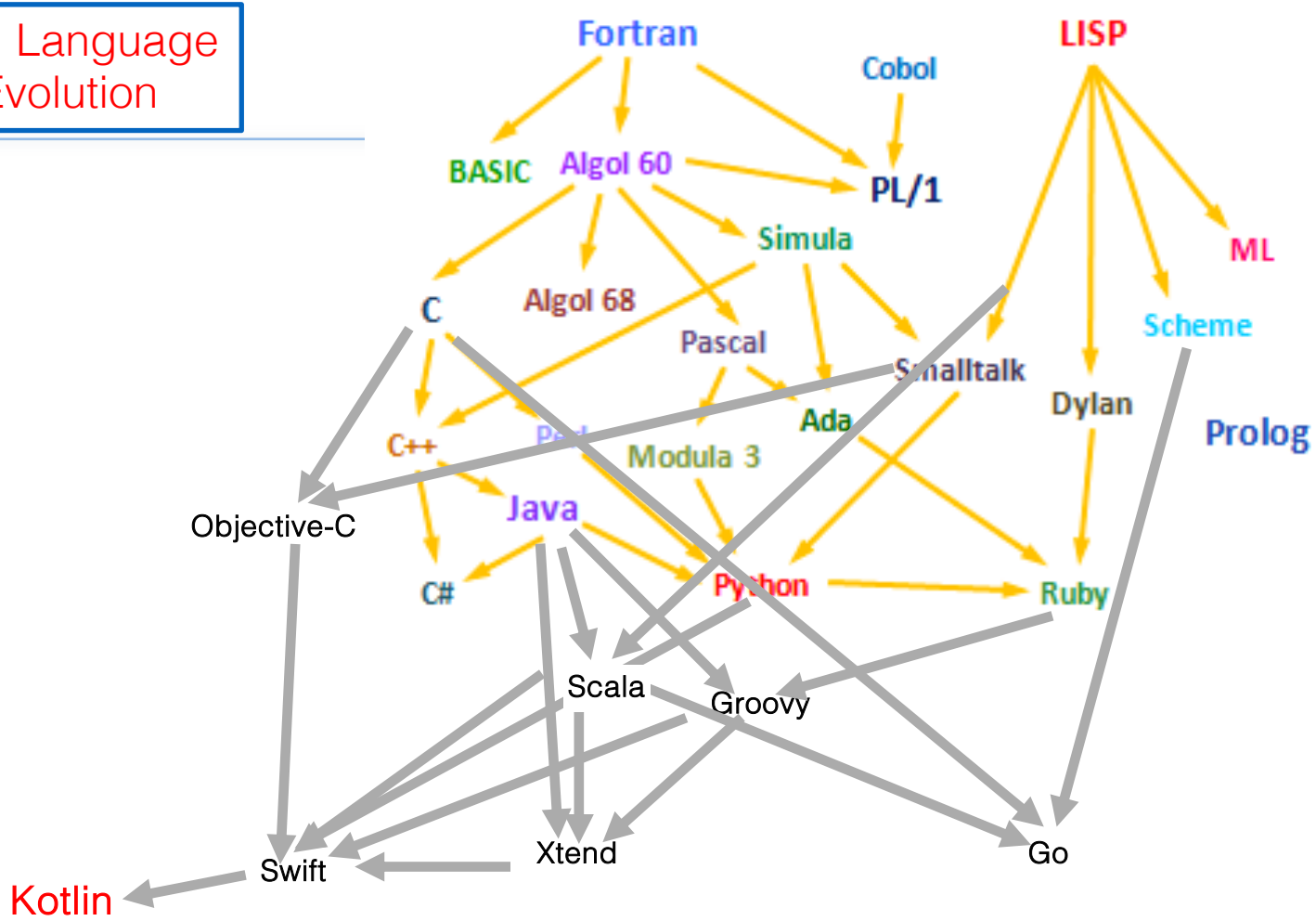
Note: Kotlin also compiles to JavaScript

Kotlin

Based on : https://en.wikipedia.org/wiki/Kotlin_(programming_language)

- runs on Java Virtual Machine.
- is an evolution of the Java syntax but is more concise and has cleaner syntax.
- is not syntax compatible with Java; but is interoperable with Java.
- relies on some Java Class Libraries
  e.g. Collections framework.
- is a statically-typed programming language.
- offers null safety.

JVM Language Evolution

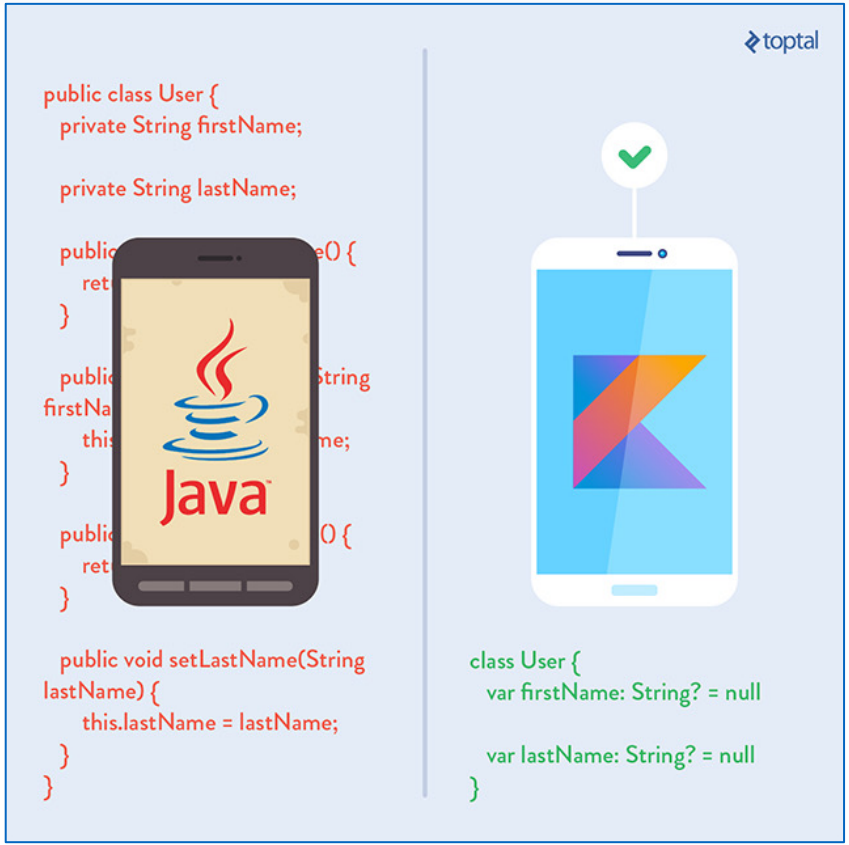


Kotlin (Concise) Vs Java (Overly Verbose)

Rough estimates indicate approximately a 40% cut in the number of lines of code.

https://kotlinlang.org/docs/reference/faq.html

Kotlin (Concise) Vs Java (Overly Verbose)



Kotlin drastically reduces the amount of boilerplate code you have to write.

The less code you write, the fewer mistakes you make, the less to test, the better the execution.

Easy(ish) learning curve for Java Developers

"Kotlin is approachable and can be acquired in a few hours by simply reading the language reference. It has a lean and intuitive syntax."

"Kotlin is also designed to have a gentle learning path for Java developers. Java programmers will find that most of the Kotlin syntax feels familiar."

https://dzone.com/articles/why-you-should-consider-kotlin-for-android-develop?fromrel=true

Kotlin

Based on : https://en.wikipedia.org/wiki/Kotlin_(programming_language)

- runs on Java Virtual Machine.
- is an evolution of the Java syntax but is more concise and has cleaner syntax.
- is not syntax compatible with Java; but is interoperable with Java.
- relies on some Java Class Libraries
  e.g. Collections framework.
- is a statically-typed programming language.
- offers null safety.

Kotlin / Java Interoperability

- Kotlin and Java are 100% interoperable; Java and Kotlin code can co-exist very well in the same project and compile perfectly.
- Kotlin can be called from Java and Java from Kotlin.
- Both .java and .kt files are compiled to .class bytecode.
- When a project containing both Java and Kotlin is compiled, it would be difficult to tell which parts were created in Java and which in Kotlin.
- You can start using Kotlin in an existing Java project, without having to convert the project to Kotlin (no need to bin all your Java code!)

https://dzone.com/articles/why-you-should-consider-kotlin-for-android-develop?fromrel=true

Kotlin

Based on : https://en.wikipedia.org/wiki/Kotlin_(programming_language)

- runs on Java Virtual Machine.
- is an evolution of the Java syntax but is more concise and has cleaner syntax.
- is not syntax compatible with Java; but is interoperable with Java.
- relies on some Java Class Libraries e.g. Collections framework.
- is a statically-typed programming language.
- offers null safety.

Kotlin and the Java Class Libraries

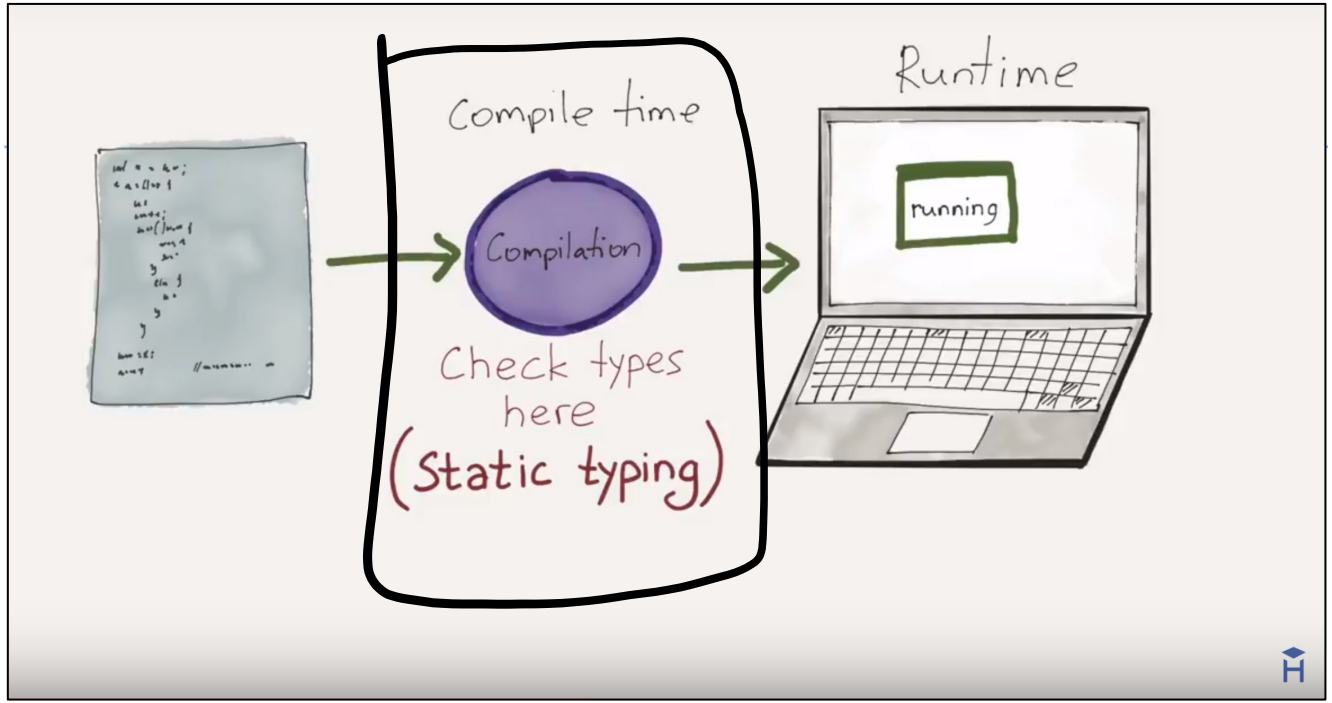
- Interoperability advantages:
  - you can use any of the vast number of Java Libraries and Frameworks in a Kotlin project.
  - Kotlin can also easily integrate with Maven, Gradle and other build systems.

https://dzone.com/articles/why-you-should-consider-kotlin-for-android-develop?fromrel=true
https://www.xenonstack.com/blog/overview-of-kotlin-comparison-between-kotlin-java

Kotlin

Based on : https://en.wikipedia.org/wiki/Kotlin_(programming_language)

- runs on Java Virtual Machine.
- is an evolution of the Java syntax but is more concise and has cleaner syntax.
- is not syntax compatible with Java; but is interoperable with Java.
- relies on some Java Class Libraries
  e.g. Collections framework.
- is a statically-typed programming language.
- offers null safety.



STATIC TYPING
"Variable declarations are mandatory before usage, else results in a compile-time error"

Static Typing – Example

```
String greeting = "Hello!";
int someRandomInteger = 100;
double aDoubleVariable = 2.2;
```

A type is assigned to each variable.

In Java, if we don't assign a type, we get a compiler error → Java is statically typed.

Types determine the operations we can perform on the variables.

https://howtoprogramwithjava.com/dynamic-typing-vs-static-typing/

Static Typing – Example

```
fun main(args : Array<String>)
{
    var someRandomInteger = 100
    var aDoubleVariable = 2.2
    println (someRandomInteger)
    println (aDoubleVariable)
}
```

*In Kotlin, you don't have to (but still can!) specify the type of each variable explicitly, even though Kotlin **is** statically-typed.*

Here, Kotlin determines the type from the initialisation.

http://petersommerhoff.com/dev/kotlin/kotlin-for-java-devs/#type-inference

Static Typing – Example

```
fun main(args : Array<String>)
{
    var someRandomInteger : Int = 100
    var aDoubleVariable : Double = 2.2
    println (someRandomInteger)
    println (aDoubleVariable)
}
```

However, you can choose to explicitly define a data type.

Static Typing – Example

```
fun main(args : Array<String>)
{
    var someRandomInteger //compile error
    var aDoubleVariable : Double = 2.2
    println (someRandomInteger)
    println (aDoubleVariable)
}
```

With Kotlin, you have to <u>either</u> define a type or initialise the variable (kotlin then determines the type!).

Static Typing – Example

```
fun main(args : Array<String>)
{
    var someRandomInteger : Int = 100
    var aDoubleVariable : Double = 2.2

    someRandomInteger = 2.65      //compile error
    aDoubleVariable = 233         //compile error

    println (someRandomInteger)
    println (aDoubleVariable)
}
```

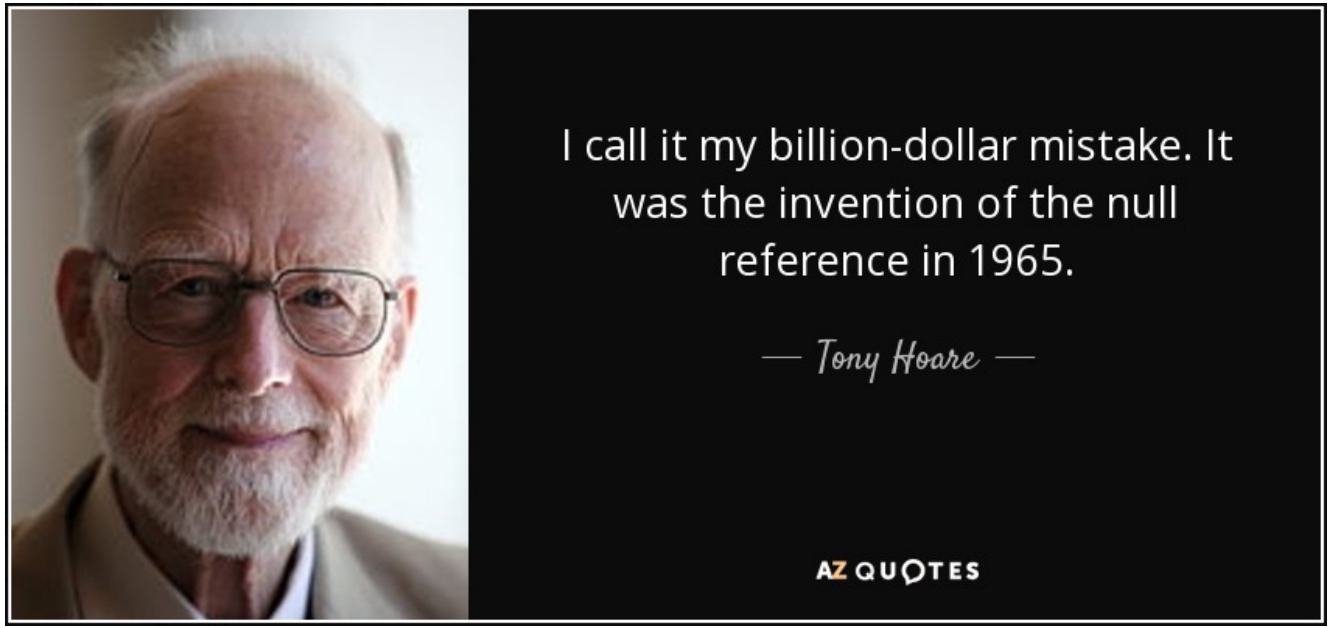
http://petersommerhoff.com/dev/kotlin/kotlin-for-java-devs/#type-inference

Kotlin

Based on : https://en.wikipedia.org/wiki/Kotlin_(programming_language)

- runs on Java Virtual Machine.
- is an evolution of the Java syntax but is more concise and has cleaner syntax.
- is not syntax compatible with Java; but is interoperable with Java.
- relies on some Java Class Libraries
  e.g. Collections framework.
- is a statically-typed programming language.
- offers null safety.

Aside - Null (The Billion Dollar Mistake)



- who developed the sorting algorithm: QuickSort

Kotlin and Null Safety

- Kotlin eliminates most sources of null references by making all types non-nullable by default — meaning that the compiler won't let you use a non-initialized, non-nullable variable.
- If you need a variable to hold a null value, you have to declare the type as nullable by adding a **question mark** after the type (more on this in later lectures).

```
1 var nonNullable: String = "My string" // needs to be initialized
2 var nullable: String?
```

Source: https://dzone.com/articles/kotlin-vs-java-first-impressions-using-kotlin-for

Agenda Recap

- ❑ What is Kotlin?
- ❑ Context - Java Vs JVM
- ❑ Kotlin History
- ❑ Milestones
- ❑ General Overview & Features
- ❑ Kotlin & IntelliJ IDEA
- ❑ Conclusion


Thanks.