Intro to Kotlin

BCS421 - Mobile Development

By Dr. Moaath Alrajab
Computing Division
School of Engineering Tech.

SUNY The State University of New York

Farmingdale State College
State University of New York

Introduction to Kotlin

The next big thing in the Java world

What is Kotlin?

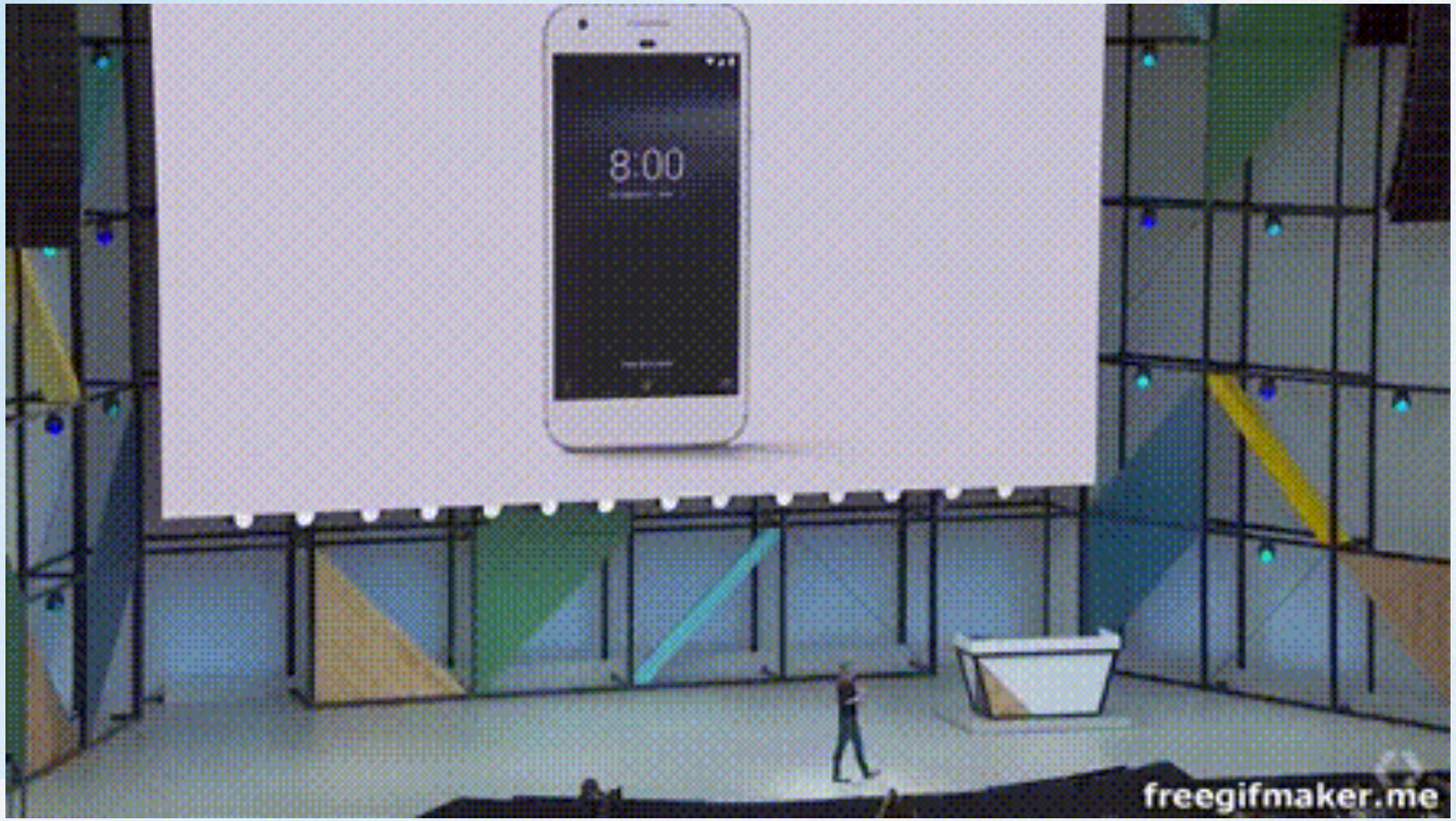
- A (fairly) new statically-typed language from JetBrains (creators of IntelliJ Idea, ReSharper, PyCharm and other IDEs and IDE extensions)
- Tries to fix many of Java's shortcomings
- Compiles to JVM bytecode, JavaScript (!) and Kotlin native (no VM)
- Created with focus on Java interoperability – Kotlin and Java classes can be used together in a project (but compilation is much faster than Scala's)

What is Kotlin?

- Works seamlessly with IntelliJ Idea, just a bit less seamlessly with Eclipse
- The Java interoperability is real – even complex applications, relying on annotation processing
- It's gaining more and more traction and is being adopted increasingly often, especially in the recent year
- Has very good suport from JetBrains, an established company, who use Kotlin to develop their own products

Google and Android

At Google I/O 2017, Google announces that Kotlin would receive first-class suport for Android development



What does Kotlin feature?

- Null safety (drastically limiting the number of NPEs);
- If, try-catch, when expressions;
- Extension functions;
- Var and val keywords;
- Inline functions;
- Named function parameters;
- Multi-value return functions;
- Semi-colons are optional

More things…

- Functional programming constructs (like higher order functions)
- Smart casting
- Async/await
- **Data classes**
- Properties (C#- like)
- No checked exceptions
- Operator overloading
- … this is just scratching the surface

Let’s dive deeper and explore Kotlin!

Typical Kotlin Program

Developers commonly use an ***Integrated Development Environment*** (***IDE***) to write Kotlin programs. Android Studio is the most popular IDE for writing Android apps with Kotlin.

A developer creates and runs a typical Kotlin program in the following way:

- A developer uses an IDE to type Kotlin code into a source file with a .kt file extension.
- The IDE compiles Kotlin source files into ***bytecode***, which are binary instructions stored in a file with a .class file extension.
- The program runs on a ***Java Virtual Machine*** (***JVM***), software that executes bytecode.
- A Kotlin program must have a function called main.

Comments and semicolons

A ***comment*** is any text in a program intended for humans and ignored by the compiler. Kotlin uses the // operator to produce a single-line comment and /* */ operators to produce multi-line comments.

Kotlin allows statements to be terminated with a semicolon, but terminating semicolons are not necessary. Only when two statements reside on the same line must a semicolon separate the two statements.

Good practice is to avoid placing two statements on the same line.

Variables and data types

- Kotlin uses two different keywords for declaring two different types of variables:
  - A regular variable is declared with the ***var*** keyword. Ex:

    var score: Int
    declares a regular variable named score with the Int (integer) data type.

  - A read-only variable is declared with the ***val*** keyword. Ex:

    val maxPages: Int = 10
    declares a read-only variable maxPages that is always 10.

Variables and **data types**

Data type	Description	Example
`Byte`, `Short`, `Int`, `Long`	Integers that range in size from 1 to 8 bytes.	`var highScore: Int = 950` `val lightYears: Long = 640000000000`
`Float`, `Double`	Floating-point numbers with decimal places. Float is 4 bytes, Double is 8 bytes.	`var totalPrice: Float = 12.55f` `val pi: Double = 3.14159265359`
`Boolean`	`true` or `false` values	`var amHungry: Boolean = true` `val amThirsty = false`
`Char`	Single character delimited with 'single' quotes	`var letterGrade: Char = 'A'` `val answer = '?'`
`String`	Group of characters delimited with "double" quotes	`var name: String = "Joshua"` `val quote = "Shall we play a game?"`
`Array`	List of items, typically of the same data type	`var aveTemperatures = arrayOf(50.3, 56.2, 49.8, 60.5)` `val teams = arrayOf("Broncos", "Cowboys", "49ers")`

Naming variables

A name created for an item like a variable or function is called an ***identifier***. Kotlin imposes the following rules for identifiers:

- An identifier can be any combination of letters, digits, or underscores.
- An identifier may not start with a digit.
- An identifier may not be a reserved word like fun, var, or if.
- An identifier is case sensitive.

Arithmetic in string template

```
var x = 3
var y = 5

// Produces "3 + 5 = 8"
println("$x + $y = ${x + y}")

// Produces "Answer is 7"
println("Answer is ${x + 2 * (7 - y)}")
```

Arithmetic operator	Description	Example
+	Add	`// x = 3` `var x = 1 + 2`
-	Subtract	`// x = 2` `var x = 6 - 4`
*	Multiply	`// x = 6` `var x = 2 * 3`

/	Divide	`// x = 0`
		`var x = 1 / 2`
		
		`// y = 0.5`
		`var y = 1.0 / 2.0`

%	Modulus (remainder)	`// x = 0`
		`var x = 4 % 2`
		
		`// y = 0.5`
		`var y = 4.5 % 2`

Conditionals

An ***if statement*** executes a block of statements if a condition is true. An ***if-else statement*** executes a block of statements if the statement's condition is true and a different block of statements if the condition is false. Braces {} surround the statement blocks.

```
If (condition){
        // statements if true
} else{
        // statements if false
}
```

Example

```
fun main() {
    var ticketPrice = 0
    val customerAge = 35

    ticketPrice = if (customerAge < 18) {
        println("You are under 18")
        15
    } else {
        println("You are at least 18")
        20
    }

    println("Ticket price is $$ticketPrice")

    val extraFee = if (customerAge > 60) 2 else 5
    println("You must pay $" + (ticketPrice + extraFee))
}
```

Farmingdale State College
State University of New York

Conditions – in and when

```
homeTeam = 2
decision = if (homeTeam in 1..5) 1 else 0
```

```
homeTeam = 5
decision = if (homeTeam !in 1..5) 1 else 0
```

```
val coinType = when (coinValue) {
    1 -> "Penny"
    5 -> "Nickel"
    10 -> "Dime"
    25 -> "Quarter"
    in 50..70 -> "Magical"
    else -> "Unknown"
}
```

Assume coinValue is 25

Conditions – in and when

```
homeTeam = 2
decision = if (homeTeam in 1..5) 1 else 0
```

Decision is 1

```
homeTeam = 5
decision = if (homeTeam !in 1..5) 1 else 0
```

Decision is 0

```
val coinType = when (coinValue) {
    1 -> "Penny"
    5 -> "Nickel"
    10 -> "Dime"
    25 -> "Quarter"
    in 50..70 -> "Magical"
    else -> "Unknown"
}
```

When coinValue is 25 then coinType is "Quarter"

Kotlin Hard vs Soft keywords

- Hard keywords are always interpreted as language keywords and cannot be used as identifiers while soft keywords (including modifiers) can be used as identifiers (depends on the context).

Hard Keywords	Soft Keywords
as is used for type casts. specifies an alias for an import **as?** is used for safe type casts. **break** terminates the execution of a loop. **fun** declares a function. **is** checks that a value has a certain type. is used in when expressions for the same purpose.	**by** delegates the implementation of an interface to another object. delegates the implementation of the accessors for a property to another object. **delegate** is used as an annotation use-site target. **get/set** declares the getter and setter of a property. is used as an annotation use-site target. **inline** tells the compiler to inline a function and the lambdas passed to it at the call site.

https://kotlinlang.org/docs/keyword-reference.html#hard-keywords

Loops

- while loop
- do-while loop
- for loop

```
var count = 1
while (count <= 4) {
    println(count)
    count++
}
println("Done!")
```

```
var count = 1
do {
    println(count)
    count++
} while (count <= 4)
println("Done!")
```

```
for (count in 5..10 step 2) {
    println(count)
}
```

```
println("Counting up:")
for (count in 1..4) {
    println(count)
}

println("Counting down:")
for (count in 5 downTo 1 step 2) {
    println(count)
}
```

Conclusion

- Kotlin is low-cost: can be developed in the same IDE, in the same project as Java code
- Did I mention first-class Spring support?
- It's on track to become one of the leading JVM languages
- It's best suited to work with IntelliJ idea; it can be adapter to Eclipse, albeit with some shortcomings
- It's cool; allows bragging to friends about using the latest technologies

End of Intro To Kotlin

- Thank you
- Q/A