Android App Development with Kotlin

Variables, Constants and Types

1

1

Software

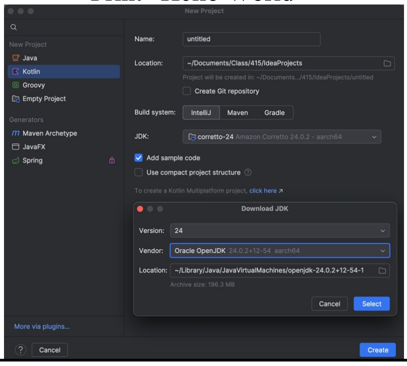
- IntelliJ (Download Community version)
  - https://www.jetbrains.com/idea/
- Any JDK would work
- I recommend **installing JDK through IntelliJ**

2

2

Create a new Kotlin Project

Print "Hello World"



Name: untitled

Location: ~/Documents/Class/415/IdeaProjects

Project will be created in: ~/Documents.../415/IdeaProjects/untitled

☐ Create Git repository

Build system: [✓ IntelliJ] Maven Gradle

JDK: corretto-24 Amazon Corretto 24.0.2 - aarch64

☑ Add sample code

☐ Use compact project structure

To create a Kotlin Multiplatform project, click here ↗

Version: 24

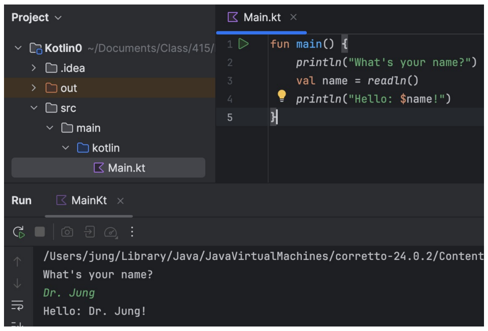
Vendor: Oracle OpenJDK 24.0.2+12-54 aarch64

Location: ~/Library/Java/JavaVirtualMachines/openjdk-24.0.2+12-54-1

Archive size: 196.3 MB

3

3



```kotlin
fun main() {
    println("What's your name?")
    val name = readln()
    println("Hello: $name!")
}
```

```
/Users/jung/Library/Java/JavaVirtualMachines/corretto-24.0.2/Content
What's your name?
Dr. Jung
Hello: Dr. Jung!
```

4

4

5

5

Kotlin Coding Conventions

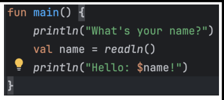
https://kotlinlang.org → Get Started → Basics → Coding Conventions

- https://kotlinlang.org/docs/coding-conventions.html
- Source file names: `.kt`
  - Script file names: `.kts`
- Naming rules (file, function, class etc): use **CamelCase**
  - https://en.wikipedia.org/wiki/Camel_case
- In Kotlin, semicolons are optional, and therefore **line breaks** are significant.

6

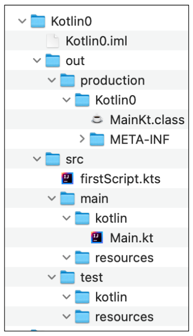
6

Kotlin Essentials



```
fun main() {
    println("What's your name?")
    val name = readln()
    println("Hello: $name!")
}
```

- Whenever you write a Kotlin application, you **must add a function to it called** `main`, which starts your application.
- When you run your code, the JVM looks for this function, and executes it.
  – You don't need `main()` in **Kotlin script**
- Compiles your Kotlin source code (`xxx.kt`) into JVM bytecode, then JVM runs `XxxKt.class`
  – JVM requires upper case first letter (Think about Java)

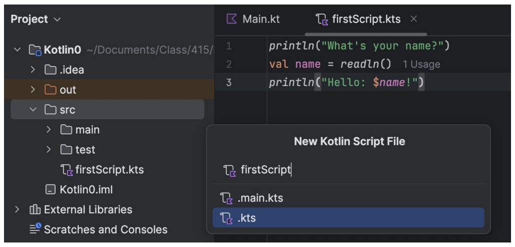


```
Kotlin0
    Kotlin0.iml
    out
        production
            Kotlin0
                MainKt.class
                META-INF
    src
        firstScript.kts
        main
            kotlin
                Main.kt
            resources
        test
            kotlin
            resources
```

7

Kotlin Script

`src` (right click) → new → Kotlin Script



```
println("What's your name?")
val name = readln()  1 Usage
println("Hello: $name!")
```

You don't need `main()` in **Kotlin script**

8

8

9

9

```
var x = 10
while (x > 1){
    x--
    if (x < 3)
        println ("inside IF")
}
```

Output? How many times?

10

10

- Semicolon is **optional**
- Variable type specification is **optional**
  – **Statically typed**
  – **Type inference**

```
var x = 10
while (x > 1){
    x--
    if (x < 3)
        println ("inside IF")
}
```

- **Functional-style**
  – Languages like Java/C/C++/C# has more statements than expressions
  – Languages like **Haskell (F#, Rust)** has more expression than statements

Let's test it!

11

11

`var` vs. `val`

```
var score = 10
// or var score: Int = 10
score = 11
println (score) // 11

val pi: Double = 3.14
val pi = 3.14
pi = 3.14 // ERROR: val cannot be reassigned
println (pi)
```

12

12

var vs. val

```
var score = 10
// or var score: Int = 10
score = 11
println (score) // 11

val pi: Double = 3.14
val pi = 3.14
pi = 3.14 // ERROR: val cannot be reassigned
println (pi)
```

- Mutating variables is a way of life in imperative style of programming. But that's **taboo in functional programming**
- Immutable variable (constant or value)
  - Use `val (Java's final)`

Let's test it!

13

13

var vs. val

```
var myArray = arrayOf(1,2,3)
myArray=arrayOf(4,5)    // re-assigned

val yourArray = arrayOf(1,2,3)
// you can change content of the object
yourArray[2] = 6
yourArray = arrayOf(4,5) // compile error
```

14

14

var vs. val

```
var myArray = arrayOf(1,2,3)
myArray=arrayOf(4,5)    // re-assigned

val yourArray = arrayOf(1,2,3)
// you can change content of the object
yourArray[2] = 6
yourArray = arrayOf(4,5) // compile error
```

- `val` only guarantees immutability of the reference and doesn't prevent the object from changing.
  - For example, `String` is immutable, but `StringBuilder` is mutable

Let's test it!

15

15

Equality Check

```
println ("hi" == "hi")
println ("hi" == "Hi")
println (null == "hi")
println ("hi" == null)
println (null == null)
```

In Java, `.equals()` vs. `==` ?

Output?

16

16

Equality Check

```
println ("hi" == "hi")
println ("hi" == "Hi")
println (null == "hi")
println ("hi" == null)
println (null == null)
```

In Java, `.equals()` vs. `==` ?

- Comparison of values, called **structural equality**
  - `equals()` method in Java
  - `==` operator in Kotlin
  - When `==` is used in Kotlin, it performs the null checks and then calls `equals()` method
- Comparison of references, called **referential equality**
  - `==` operator in Java
  - `===` in Kotlin
  - Compares references and returns `true` if the two references are identical (check if they are same instance)

Let's test it!

17

17

String Templates

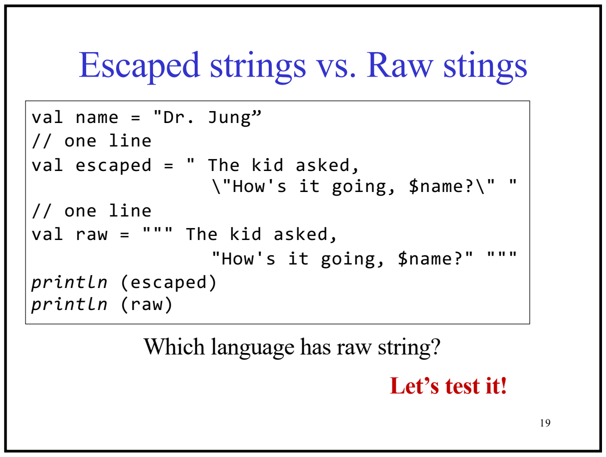
```
val price = 12.25
val taxRate = 0.08
// one line
val output = "the amount $price after
           tax comes to $${price * (1 + taxRate)} "
// one line
val disclaimer = "the amount is in US$,
                  that's right in \$only"
println (output)
println (disclaimer)
```

Output?

- Embedded values of expressions.
- `+` in Java
- `$, {}, """, \` in Kotlin

18

18



Escaped strings vs. Raw stings

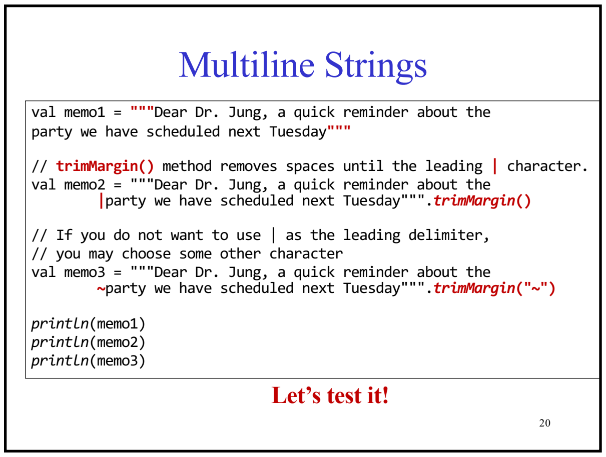
```
val name = "Dr. Jung”
// one line
val escaped = " The kid asked,
                \"How's it going, $name?\" "
// one line
val raw = """ The kid asked,
                "How's it going, $name?" """
println (escaped)
println (raw)
```

Which language has raw string?

Let’s test it!

19

19



Multiline Strings

```
val memo1 = """Dear Dr. Jung, a quick reminder about the
party we have scheduled next Tuesday"""

// trimMargin() method removes spaces until the leading | character.
val memo2 = """Dear Dr. Jung, a quick reminder about the
        |party we have scheduled next Tuesday""".trimMargin()

// If you do not want to use | as the leading delimiter,
// you may choose some other character
val memo3 = """Dear Dr. Jung, a quick reminder about the
        ~party we have scheduled next Tuesday""".trimMargin("~")

println(memo1)
println(memo2)
println(memo3)
```

Let’s test it!

20

20