Developing Android Apps with Kotlin

NTG Clarity Networks Inc.

- Alhaytham Attia
  - Technical lead, mobile development
  - Skills: Android, iOS and Flutter

Ground Rules

- Session break is about 30 minutes.
- Check your setup few minutes before we start.
- Keep your mic muted unless you are talking.
- Interrupt me whenever you want to ask.
- Careful what you wish for. Questions may turn into assignments.
- Bring your drinks and snacks with you.
- Learn well

Why?

- Easy
- Free
- Reach
- Market
- Career

What?

- Developing Android apps with
  - Kotlin
    - Open source
    - Statically typed programming language
  - Android Studio IDE
    - Unified environment for all Android devices
    - Visual layout editor
    - Code completion
    - Emulator

Environment setup

- Requirements
  - A computer running a 64-bit version of
    - Windows (8, 10, or 11),
    - Linux,
    - macOS (10.14 Mojave or later),
    - or ChromeOS.
  - Internet access for your computer.
- Installation
  - Download Android Studio
  - Open the downloaded file and follow the setup wizard

Environment setup

Kotlin Basics

- Package specification
  - At the top of the source file
  - Full names
  - Default package
- Imports
  - Default imports
  - Scope: package, class

```
package com.ntgclarity.basics

import androidx.appcompat.app.*
import android.os.Bundle
import kotlin.text.* // Default

// com.ntgclarity.basics.log
fun log() { /* ... */ }


// com.ntgclarity.basics.Basics
class Basics {
    // ... ...
}
```

Kotlin Basics

- Program entry point
  - `main` function
- Print to the console
  - `print` & `println`

```
fun main(args: Array<String>) {
    println("Hello world")
}

fun sum(a: Int, b: Int): Int {
    print("He")
    print("llo")
    return a + b
}
```

Data Types

- Declaration: `var` vs `val`
- Everything is an object
- Integer
  - Byte, Short, Int, Long
- Literals
  - 123, 123L, 0x0F, 0b00001011
- Floating point
  - Float, Double

```
val i = 1 // Int
val l = 3000000000 // Long
val l2 = 1L // Long
val b: Byte = 1

val pi = 3.14 // Double
val pi = 3.14f // Float

// Readability with underscores
val million = 1_000_000
```

Data Types

- Arithmetical operations
  - `+, -, *, /, %`
  - Division
- Bitwise operations
  - `shl, shr, ushr`
  - `and, or, xor`
  - `Inv`
- Assignment
  - Unsigned integers
  - Numbers comparison

```
// Division
val i = 5 / 2 // 2
val l = 5L / 2 // 2L
val d = 5 / 2.toDouble() // 2.5

// Bitwise
val i = 5 shl 1 // 101 => 1010
val i = 5 shr 1 // 101 => 0010
val i = 5 and 2 // 101 010 => 000
val i = 5 or 2 // 101 010 => 111
```

Data Types

- Boolean
  - `true` and `false`
  - Operations: `||`, `&&`, `!`
- Char
  - Literals: `'1'`, `'\t'`, `'\u0041'`
  - `digitToInt()`

```
val t: Boolean = true
val f: Boolean = false
val a = t && f
val o = t || f


val c1 = 'a'
val c2 = '2'
val i = c2.digitToInt() // Be careful
```

Data Types

- String
  - Immutable
  - Concatenation
  - Indexing
  - Literals
    - escaped strings
    - raw strings
  - Template expressions

```
val s = "Hello" + 1 // => Hello1
val o = "Hello"[4] // indexing
val e1 = "Hello" // escaped
val e2 = "Hello \nWorld!" // escaped
val r = """
    for (c in "Hello")
        print(c)
""" // raw
val e3 = "$e1 length is
${e1.length}" // template. Guess?
```

Data Types

- Arrays
  - `Array`
  - `ByteArray`, `ShortArray`, `IntArray`
  - `get` & `set`

```
val a = arrayOf("a", 2) // [a, 2]
val ia = intArrayOf(2, 1) // [2, 1]

val first = a[0] // get
a[0] = "first" // set. Guess?
```

Null safety

- Null reference
  - Billion Dollar Mistake
  - Null reference exception
- Nullable vs non-null references
  - Safe call operator: `?.`
  - Safe call chains
  - LS safe call

```
val s1: String = null // Compilation error
val s1: String = "Hello" // OK
val s2: String? = null // OK

val l1 = s1.length // Safe
val l2 = s2.length // Compilation error
val l2 = s2?.length // Safe too
val l = person?.name?.length // OK

person?.name?.first = "Ahmed" // OK
```

Null safety

- Elvis operator: `?:`
- not-null assertion operator: `!!`
  - A non-null type or NPE

```
val s: String? = null // OK
val l = s?.length ?: -1

val l = s!!.length // NPE
```

Conditions

- `if-else` statement

```
if (count == 42) {
    // ...
} else if (count < 42) {
    // ...
} else {
    // ...
}
```

Functions

- `fun` keyword
- Usage
- Single expression

```
fun double(x: Int): Int {
    return 2 * x
}

val result = double(2)

fun double(x: Int): Int = x * 2
```

Functions

- Default arguments
- Named arguments

```
fun read(
    b: Boolean,
    off: Int = 0,
    len: Int = 5,
) {}

read(b: false, 1, len: 2)
```

Functions

- Local functions
  - functions inside functions
- Member functions
  - functions in a class

```
fun foo(x: Int) {
    fun bar(y: Int, z: Int) {
    }

    bar(5, 6)
}
```

References

- Download and install Android Studio: https://developer.android.com/codelabs/basic-android-kotlin-compose-install-android-studio
- Kotlin
  - Basics: https://kotlinlang.org/docs/basic-syntax.html
  - Data types: https://kotlinlang.org/docs/basic-types.html
  - Null safety: https://kotlinlang.org/docs/null-safety.html
  - Functions https://kotlinlang.org/docs/functions.html
- Kotlin Playground: https://play.kotlinlang.org