Unit 1 — Your First Android App

Mobile Application Development — Lecture 1

Masoud Hamad

State University of Zanzibar (SUZA)

Semester II, 2025/2026

Outline

The Mobile Ecosystem

Android

- Linux kernel + ART runtime
- $>$ 70% global market share
- Open source (AOSP)
- Kotlin / Jetpack Compose

iOS

- Darwin / XNU kernel
- Apple hardware only
- Swift / SwiftUI

Cross-platform alternatives

Flutter (Dart), React Native (JS), Kotlin Multiplatform — write once, target multiple platforms with trade-offs in performance/UX fidelity.

Modern Android Stack

- **Language:** Kotlin (Google-preferred since 2019)
- **UI:** Jetpack Compose — declarative toolkit
- **IDE:** Android Studio (IntelliJ-based)
- **Build:** Gradle (Kotlin DSL)
- **Architecture:** MVVM + Unidirectional Data Flow
- **Libraries:** Jetpack (Room, Navigation, ViewModel, . . .)

Project Structure

- `app/src/main/java/` — Kotlin source
- `app/src/main/res/` — resources (drawables, strings, themes)
- `AndroidManifest.xml` — components, permissions
- `build.gradle.kts` — dependencies, build config
- `gradle/libs.versions.toml` — version catalog

SDK & API Levels

`minSdk` = oldest supported Android version; `targetSdk` = version you test against. Higher `targetSdk` unlocks newer APIs but enforces stricter behaviour.

Running an App

- **Emulator** — virtual device (AVD Manager)
- **Physical device** — USB debugging + ADB
- **Gradle tasks:** `assembleDebug`, `installDebug`, `test`

ADB basics

```
adb devices              # list connected devices
adb install app.apk      # install APK
adb logcat               # stream logs
```

Why Kotlin?

- **Null-safe** — nullability is part of the type
- **Concise** — less boilerplate than Java
- **Interoperable** — calls Java freely
- **Functional** — lambdas, higher-order functions, Flow
- **Coroutines** — clean asynchronous code

Variables & Types

```
val name: String = "Ali"        // immutable (read-only)
var age: Int = 25               // mutable
val pi = 3.14                   // type inferred (Double)
var score: Int? = null          // nullable type
```

Basic types: `Int`, `Long`, `Double`, `Float`, `Boolean`, `String`, `Char`

Control Flow

```
// if as expression
val max = if (a > b) a else b

// when (like switch)
val grade = when {
    score >= 80 -> "A"
    score >= 70 -> "B"
    else -> "F"
}

// loops
for (i in 1..10) println(i)
for (item in list) println(item)
while (count < 5) count++
```

Functions

```
fun add(a: Int, b: Int): Int = a + b

fun greet(name: String = "Student") {
    println("Hello, $name!")
}

// Higher-order function
fun transform(x: Int, op: (Int) -> Int): Int = op(x)
val doubled = transform(4) { it * 2 }  // 8
```

Classes & Data Classes

```
class Student(val name: String, var grade: Int) {
    fun promote() { grade++ }
}

val s = Student("Amina", 3)
s.promote()

data class Book(val title: String, val author: String)
```

`data class` auto-generates `equals()`, `hashCode()`, `toString()`, `copy()`.

Null Safety

```
var name: String? = null

val length = name?.length          // safe call -> null if name null
val len = name?.length ?: 0        // Elvis operator -> default 0
val forced = name!!.length         // NPE if null (avoid!)

if (name != null) {
    println(name.length)           // smart cast -> non-null
}
```

Collections

```
val list = listOf(1, 2, 3)                    // immutable
val mutable = mutableListOf("a", "b")
val map = mapOf("one" to 1, "two" to 2)
val set = setOf(1, 2, 3, 3)                   // {1, 2, 3}

// Functional operations
val doubled = list.map { it * 2 }             // [2, 4, 6]
val evens = list.filter { it % 2 == 0 }       // [2]
val sum = list.reduce { a, b -> a + b }       // 6
```

“Hello Compose”

```
class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContent {
            Greeting("Zanzibar")
        }
    }
}

@Composable
fun Greeting(name: String) {
    Text(text = "Hello, $name!")
}
```

Summary

- Kotlin is **concise, null-safe**, Google's preferred Android language
- Android Studio + Gradle form the core toolchain
- Jetpack Compose is the modern, **declarative** UI framework
- Everything you see on screen is just a `@Composable` function

Next lecture: Composables, layouts (Column, Row, Box) & Modifiers.

Questions?