



Introduction

CSD3156 Mobile and Cloud Computing

Chen Kan

Spring 2025

Acknowledgment

First part of this course borrows materials from the following source:

CSC2007 & ICT2105 & INF2007 Mobile Application Development

Jeannie S. Lee, Tan Chek Tien, Ng Pai Chet

Overview

Introduction to

Android/Platform/Apps

Kotlin

Jetpack Compose

Mobile Device Popularity

- There are approximately **6.84 billion** smartphones in the world.
- Worldwide smartphone users have **increased year-over-year** by at least **5%** over the last five years.
- There are upwards of **10.47 billion** IoT connections worldwide
- **China** has the most smartphone users in the world
- **Germany** has the most smartphone usage per capita
- **College graduates** are most likely to own a smartphone
- **Android** is the leading mobile operating system worldwide

Why is mobile interesting?

Increasing mobile adoption worldwide

Recent developments in computing happened in the mobile domain

E.g. VR HMDs, cars, wearables, mobile devices


First handheld mobile phone from Motorola


Demoed by Dr. Martin Cooper (Motorola), weighs 4.4lb (2kg!)


First commercial phone: Motorola DynaTAC 8000x (1983)

Even the Oculus Quest 3 VR HMD is mostly a phone on the inside

How much you can earn as Android Developer?


 NODEFLAIR

 [+ Contribute](#) [Sign In](#)

CompaniesJobsInterviews **NEW**SalariesReviewsBlogTools **NEW** For Employer

Android Developer Salary in Singapore

Updated 03 January 2025

 [FREE Android Developer Resume Examples](#)

Median Base Salary

\$S\$7,500

/mo


The median salary for Android Developer in Singapore is **\$S\$7,500** per month.

Base Salary Range





\$S\$4,300 - \$S\$15,000

/mo

The average Android Developer salary range in Singapore is from **\$S\$4,300** to **\$S\$15,000** per month.

 [Read the latest Asia Tech Salary Report 2024](#)


Recently Submitted Salaries

Grab Android Senior \$S\$12,100 / Monthly  <small>6 months ago</small>	NTUC FairPrice Android Mid \$S\$5,700 / Monthly  <small>8 months ago</small>	Grab Android Lead \$S\$17,500 / Monthly  <small>10 months ago</small>	Optimum Solutions Android Senior \$S\$9,000 / Monthly  <small>10 months ago</small>	Pace Enterp Android Si \$S\$7,500 / M
--	--	---	---	---

[https://nodeflair.com/salaries/singapore-android-developer-salary#:~:text=The%20average%20Android%20Developer%20salary,to%20\\$%2415%2C000%20per%20month.](https://nodeflair.com/salaries/singapore-android-developer-salary#:~:text=The%20average%20Android%20Developer%20salary,to%20$%2415%2C000%20per%20month.)

iOS Developer Salary in Singapore

Updated 03 January 2025

 [FREE iOS Developer Resume Examples](#)

Median Base Salary

\$S\$7,500 /mo

The median salary for iOS Developer in Singapore is **\$S\$7,500** per month.

Base Salary Range


\$S\$4,000 - \$S\$15,000 /mo

The average iOS Developer salary range in Singapore is from **\$S\$4,000** to **\$S\$15,000** per month.

 [Read the latest Asia Tech Salary Report 2024](#)


Recently Submitted Salaries

99 Group
iOS | Senior

\$S\$10,000 / Monthly 

7 months ago

99 Group
iOS | Senior

\$S\$10,000 / Monthly 


8 months ago

Screening Eagle Technologies
iOS | Mid

\$S\$6,800 / Monthly 

10 months ago

Binance
iOS | Mid


\$S\$6,700 / Monthly 

11 months ago

Grab
iOS | Senior

\$S\$11,700 / M

<https://nodeflair.com/salaries/singapore-ios-developer-salary>



NODEFLAIR

+ Contribute

Companies

Jobs

Interviews

NEW


Salaries

Reviews

Blog

Tools

NEW




Grab

Android Developer | Senior

Weighted Average Salary

\$S10,617 / mth

12 Salaries




ByteDance

Android Developer | Junior

Weighted Average Salary

\$S8,075 / mth

4 Salaries




Grab

Android Developer | Mid

Weighted Average Salary

\$S7,886 / mth

4 Salaries




MyDoc | Simplify Healthcare

Android Developer | Senior

Weighted Average Salary

\$S6,600 / mth

4 Salaries




Shopee

Android Developer | Senior

Weighted Average Salary

\$S8,775 / mth

4 Salaries




Carousell

Android Developer | Senior

Weighted Average Salary

\$S9,833 / mth

3 Salaries




Grab

Android Developer | Lead

Weighted Average Salary

\$S17,000 / mth

2 Salaries




Optimum Solutions

Android Developer | Senior

Weighted Average Salary

\$S8,600 / mth

2 Salaries



NODEFLAIR

+ Contribute

Companies

Jobs

Interviews

NEW


Salaries


Reviews

Blog


Tools


NEW



NODEFLAIR



[+ Contribute](#)


[Companies](#)
[Jobs](#)
[Interviews](#)
[NEW](#)
[Salaries](#)
[Reviews](#)
[Blog](#)
[Tools](#)
[NEW](#)





ByteDance
 Android Developer |  Junior


 Weighted Average Salary
S\$8,075 / mth


4 Salaries
 





ST Engineering
 Android Developer |  Junior


 Weighted Average Salary
S\$4,533 / mth


2 Salaries
 





Novade Solutions
 Android Developer |  Junior


 Weighted Average Salary
S\$5,500 / mth


1 Salaries
 





Carousell
 Android Developer |  Junior


 Weighted Average Salary
S\$6,500 / mth


1 Salaries
 




Concentrix Tigerspike
 Android Developer |  Junior


 Weighted Average Salary
S\$4,500 / mth


1 Salaries
 




Defence Science and Technology Agency (DSTA)
 Android Developer |  Junior


Weighted Average Salary
S\$4,400 / mth


1 Salaries
 





Dyson
 Android Developer |  Junior


Weighted Average Salary
S\$5,000 / mth

1 Salaries
 



Google
 Android Developer |  Junior

 Weighted Average Salary
S\$8,300 / mth

1 Salaries
 

https://nodeflair.com/salaries?page=1&user_submitted_only=false&positions%5B%5D=android-developer&seniorities%5B%5D=junior&countries%5B%5D=Singapore

**Mainframe
Computing
1950's**



**Mini
Computing
1960's**



**Personal
Computing
1970's**



**Desktop Internet
Computing
1980's**



**Mobile Internet
Computing
1990's**



Our world is changing: mobile internet

We can no longer imagine life without mobile phones

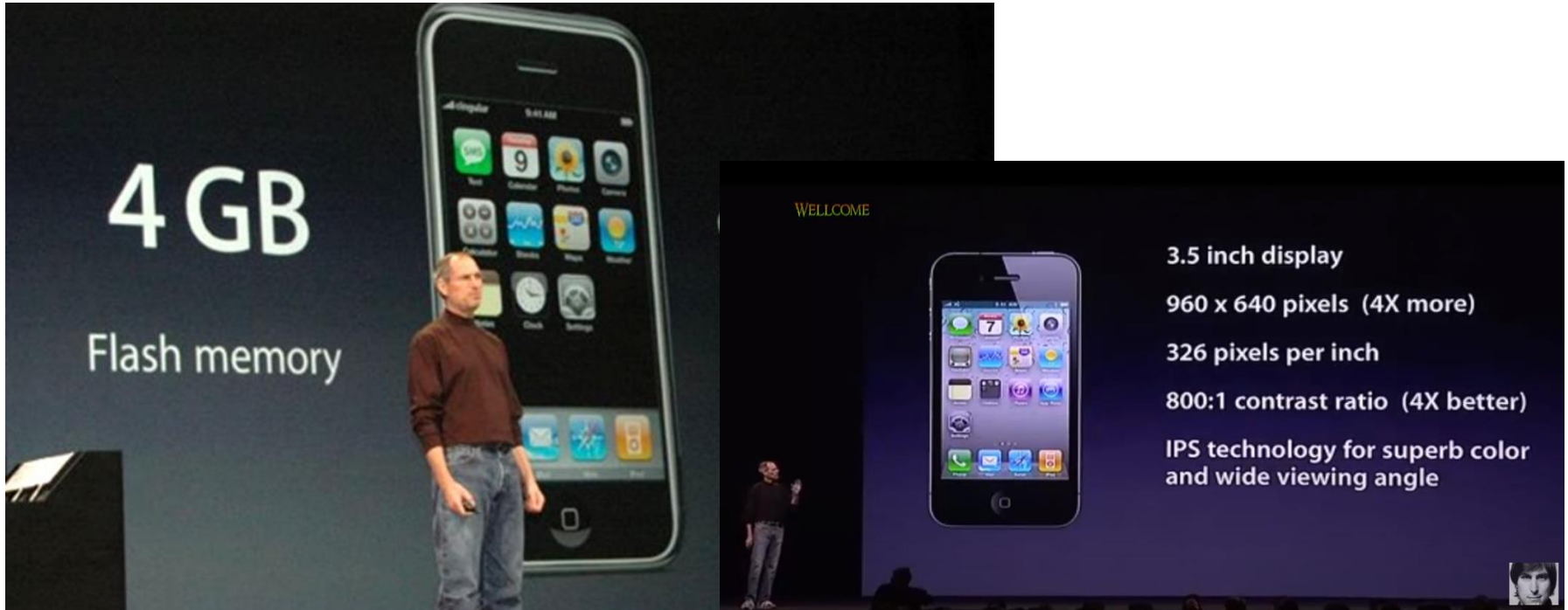
Making computers smaller was never a new idea, but...







Once, change the world, again



What goes on inside them?



Mobile Device Evolution

Formerly proprietary OS and apps, sparse information available

Focused on single use-case, either calls or note taking

Rarely networked, no third-party application development

Improvement in hardware, battery and processing power and incorporation of many features e.g. GPS, camera, cellular modem, display

Third party application distribution service in 2008, Apple App Store

OS fragmentation and cross platform frameworks using HTML5 and javascript

Multiple use cases

Communication Device or Swiss Army Knife?



V.S.



Banking, PoGo, WhatsApp, txtng, selfies, maybe phone calls...

Mobile Resource Constraints

Mobile devices are resource-poor relative to static devices

Limited processing power and CPU

Limited RAM, disk, screen size, battery ...

Connectivity highly variable: Disconnection, low bandwidth, volatility

Location changes often

Higher security risk - More likely to be stolen/lost, many attack surfaces

M. Satyanarayanan, Fundamental Challenges in Mobile Computing (1995)

Mobile Contexts

Location changes often

Users subject to disturbances

Lower attention span

Intermittent interaction

No full concentration



Hardware



Mobile processors usually based on the ARM RISC architecture

32 & 64-bit processors

Lower cost, heat and power consumption

Chip designs are licensed

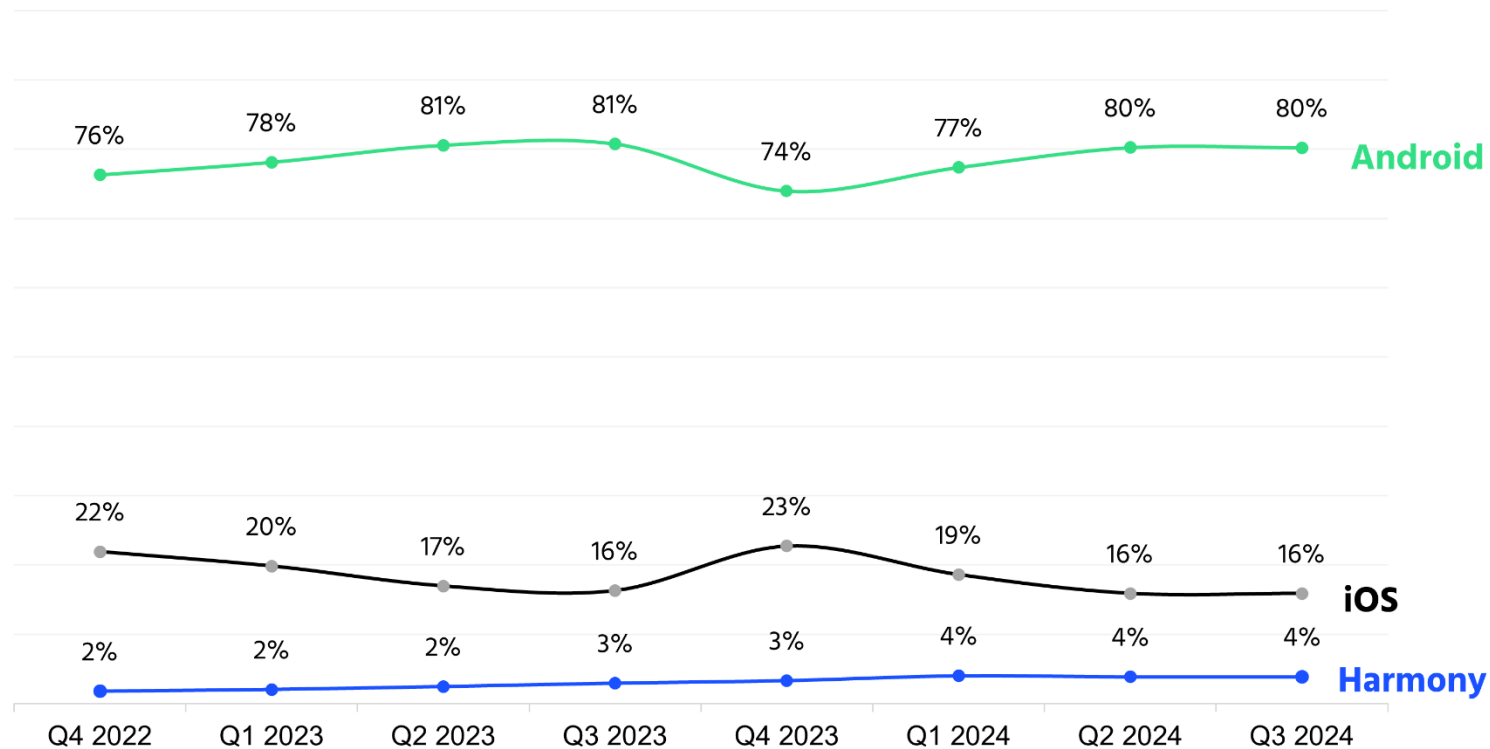


Some companies that make chips that implement an ARM architecture:

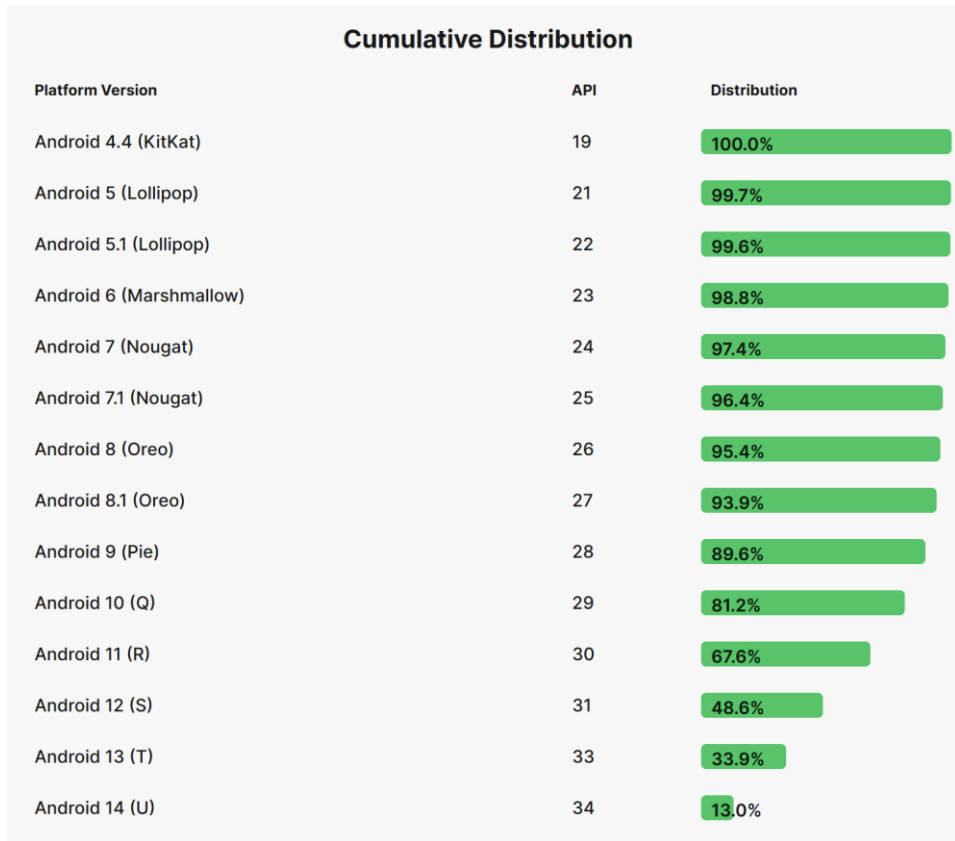
Apple, Analog Devices, Broadcom, Qualcomm, Samsung, ST Microelectronics, Texas Instruments

There are CISC architecture mobile devices too!

Global Smartphone Sales Share by Operating System



Android Fragmentation Cumulative Distribution



Find it in in
Android
Studio's **Create**
New Project
wizard

Android



Linux + Framework + JVM

OS developed by Google, open sourced (AOSP)

Based on the Linux kernel, middleware and libraries written in C/C++

Application framework and libraries in Java/Kotlin

Java libraries based on OpenJDK (Formerly Apache Harmony)

Dalvik as process VM, now Android Runtime (ART) since Lollipop (5.0)

Yearly version releases, Code changes rapidly!

First Android

Very wise to support Java

Qualcomm MSM7201

ARMv6 instruction set

528MHz x 1CPU, sequential 8-stage pipeline

TSMC 90nm

Even running map is not smooth

Moore's Law happened



Android

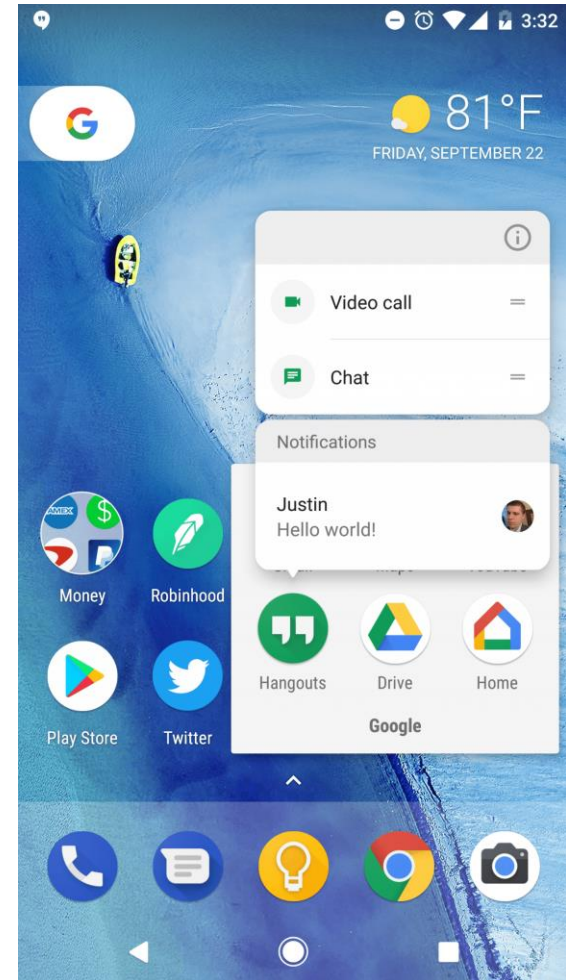
Languages: Java, Kotlin, C/C++

Supports 64 or 32bit ARM architecture, x86 also

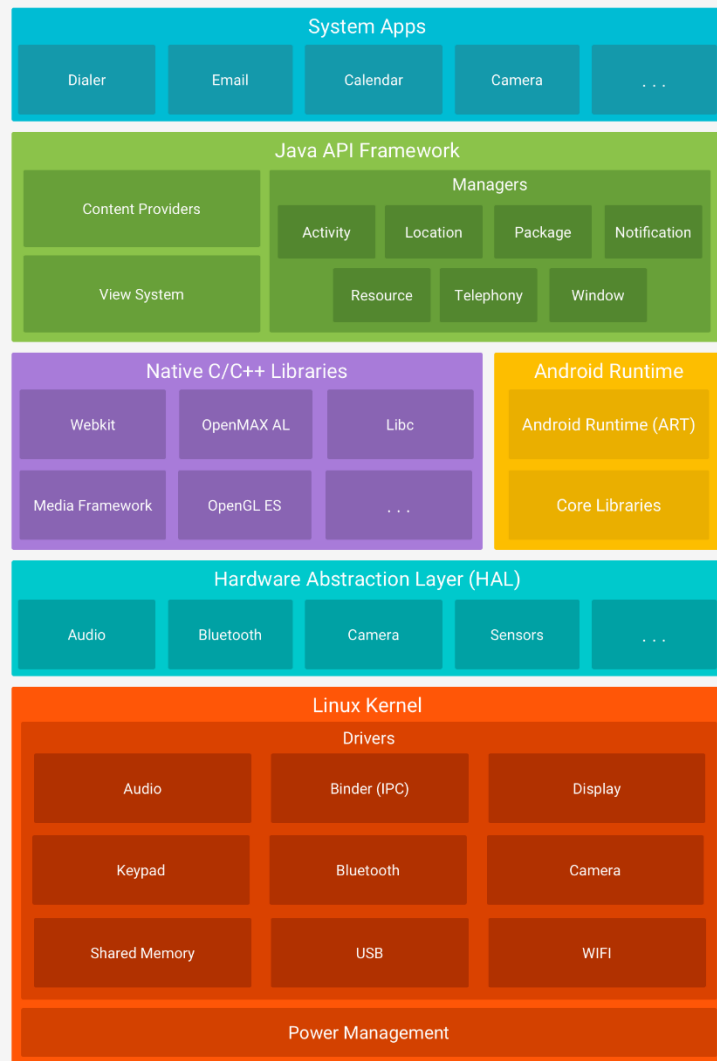
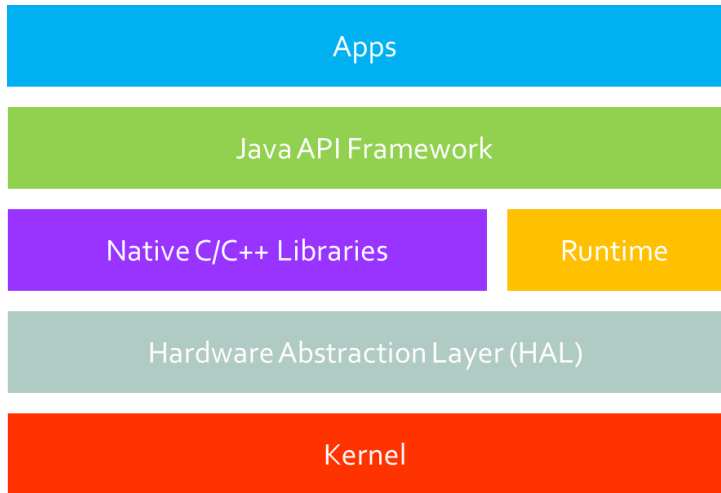
Compile to dex bytecode to run on Android
Runtime (ART) virtual machine

IDE: Android Studio

Google Play Store



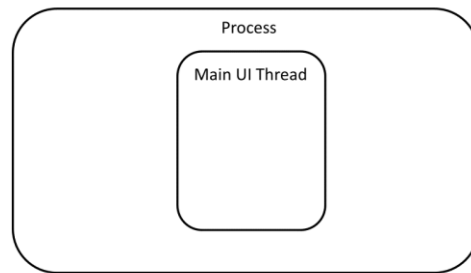
Android Architecture



Process vs. Threads

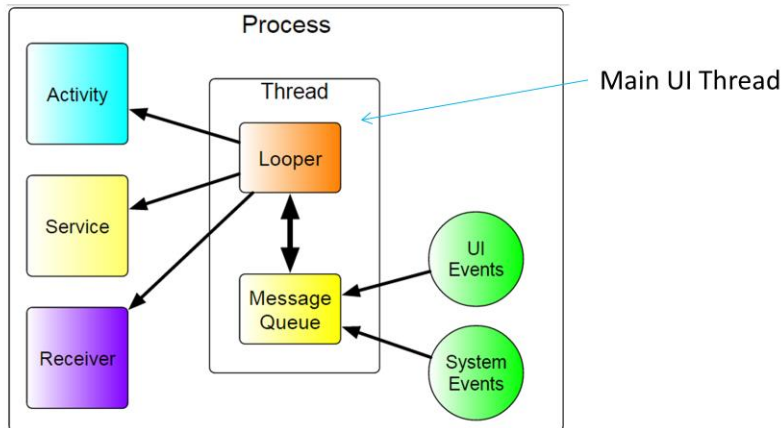
A thread is the entity within a process

App Process & Main UI Thread



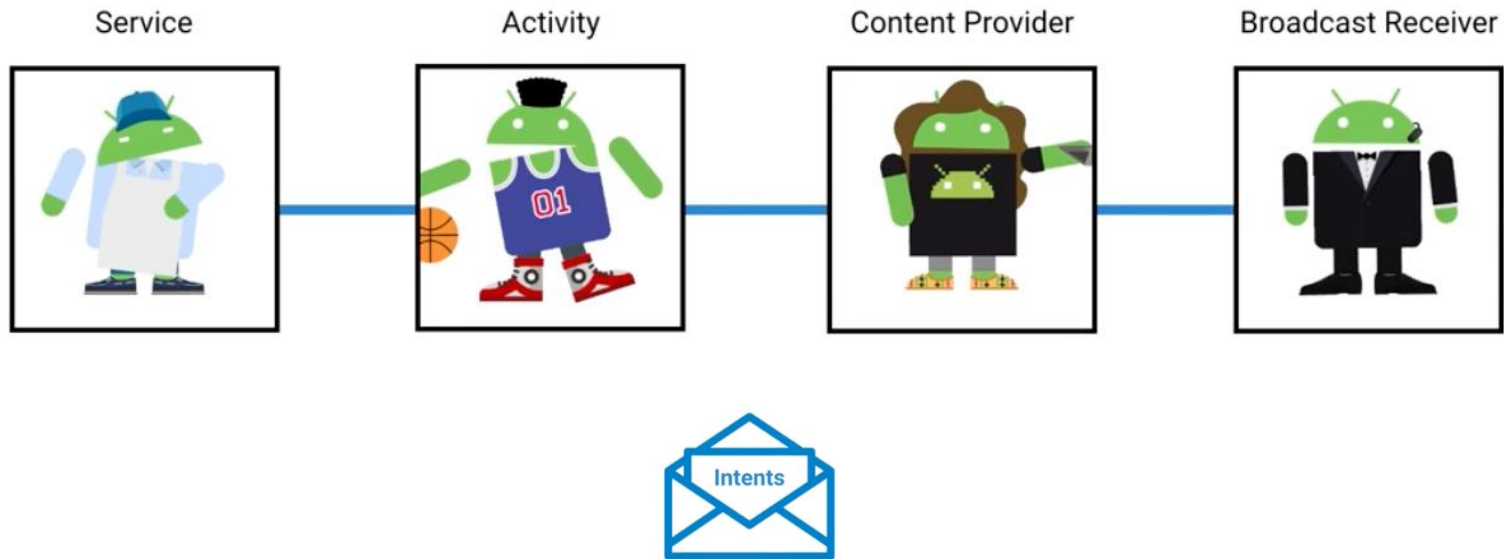
11

Main UI Thread (Detailed View)



Each process is started with a single thread, often called the primary thread, but can create additional threads from any of its threads.

Big 4



Activity

Primary class for user interaction

Usually implements a single focused task a user can do

One Activity usually corresponds to a single screen

Typical application may have many activities

Laid out as a stack

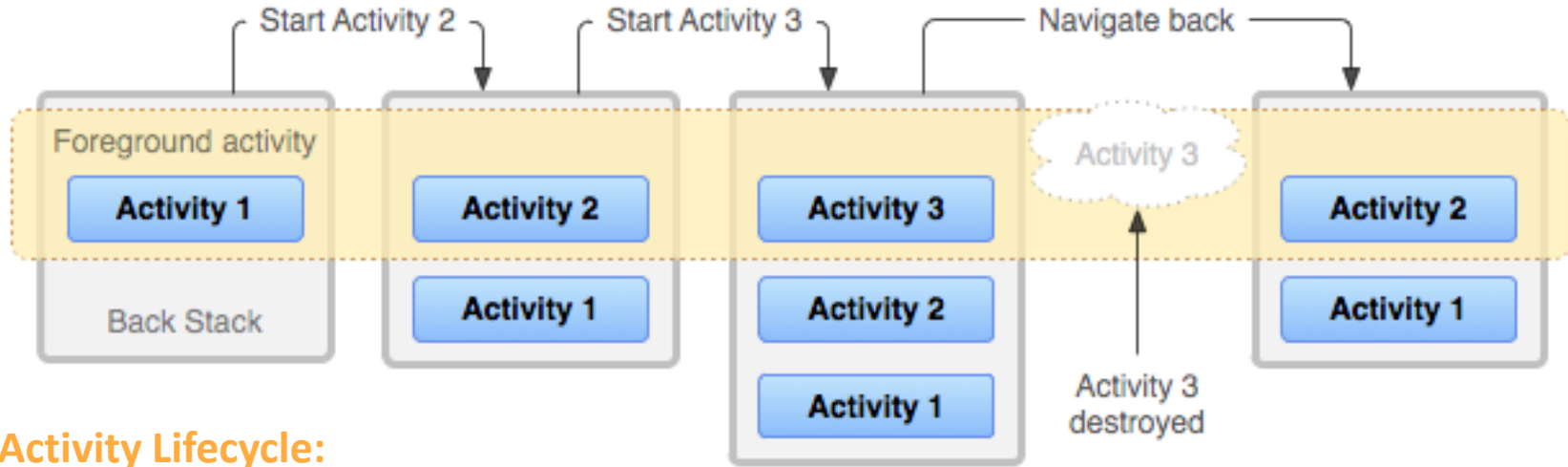
Activity on the top is visible/in foreground

Background activities are stopped, state retained

Back button resumes previous Activity in stack

Home button moves app and activities into background

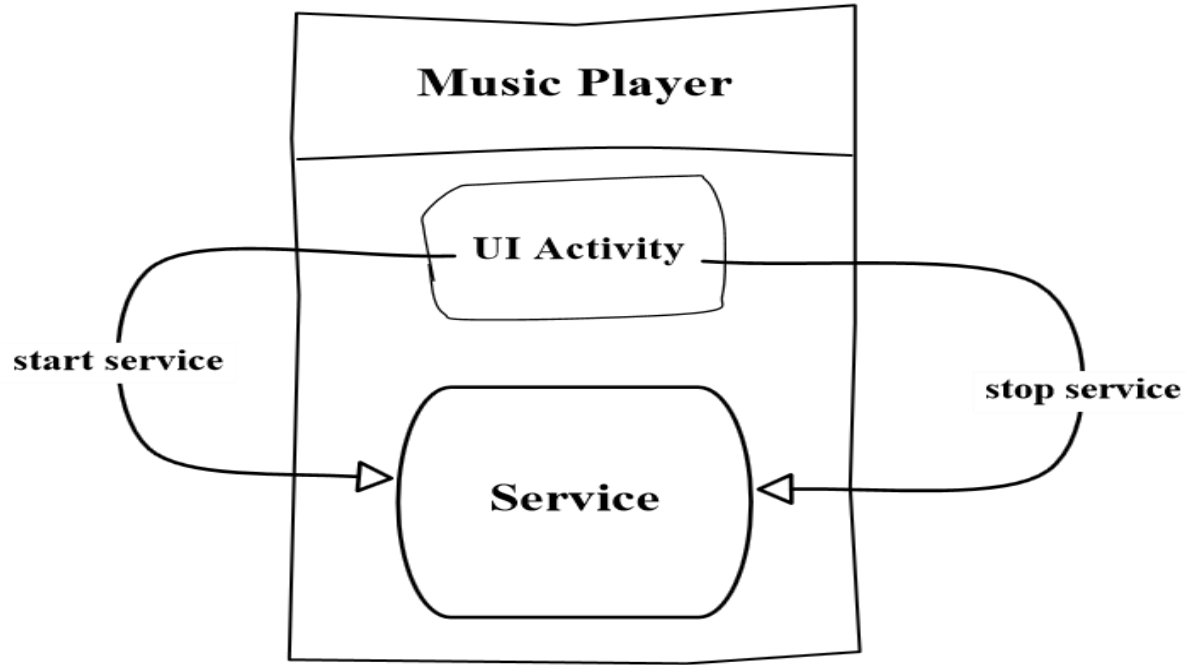
Activity Stack



Activity Lifecycle:

- An Android app consists of one or more activities, each representing a screen or a user interface component.
- The lifecycle of an Activity includes methods such as `onCreate()`, `onStart()`, `onResume()`, `onPause()`, `onStop()`, and `onDestroy()`.
- When an Activity is closed or destroyed, the `onDestroy()` method is called, indicating that the Activity is being terminated.

Service Example



Intent

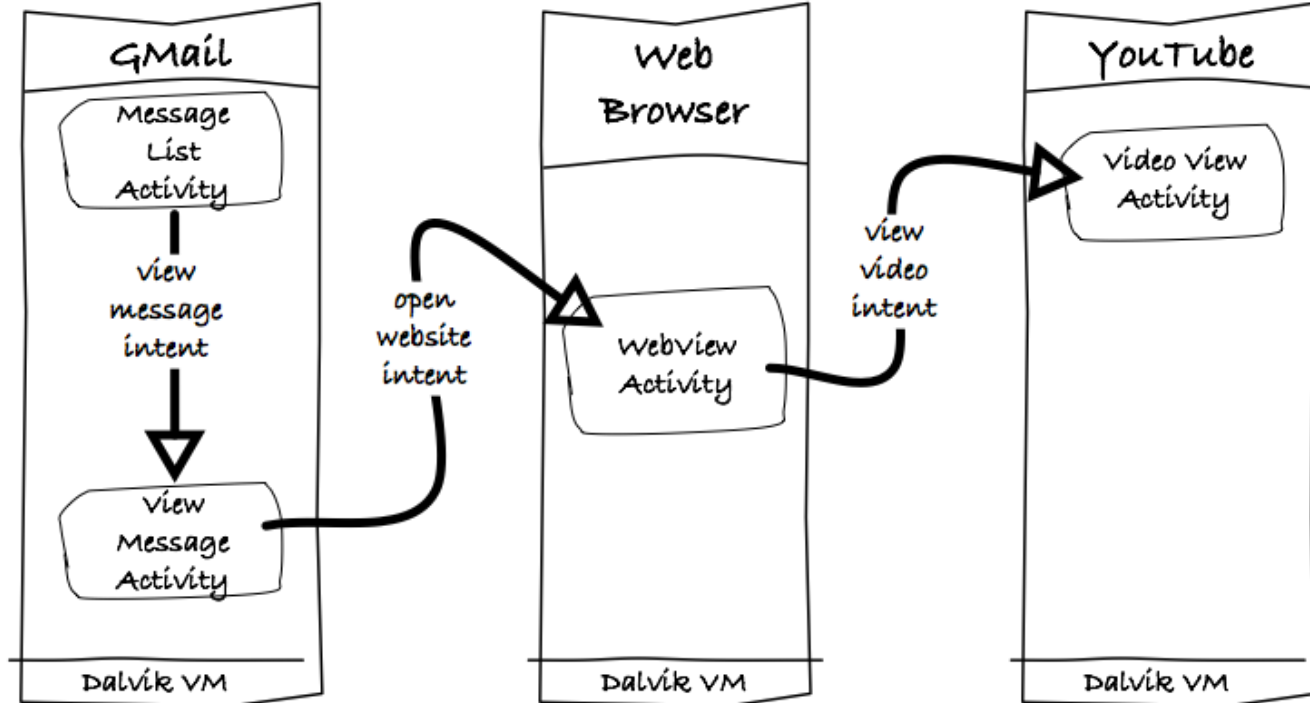
Intents are like events or messages

Used to start components or send broadcasts

Intents can be implicit or explicit

A form of IPC in Android

Intent Example



BroadcastReceiver

Component that listens for and responds to events (i.e. broadcast messages)

Can receive from system:

New phone call comes in

Battery level change

Can receive from other applications

Apps have to define their own

BroadcastReceiver

Events are represented by the Intent class

Then they are broadcast

BroadcastReceiver receives and then responds to Intent

Register for system or application events you want to receive

Publish-Subscribe model

ContentProvider

Store and share content with applications across application boundaries

Use database style interface

Handle inter-process communication (IPC)

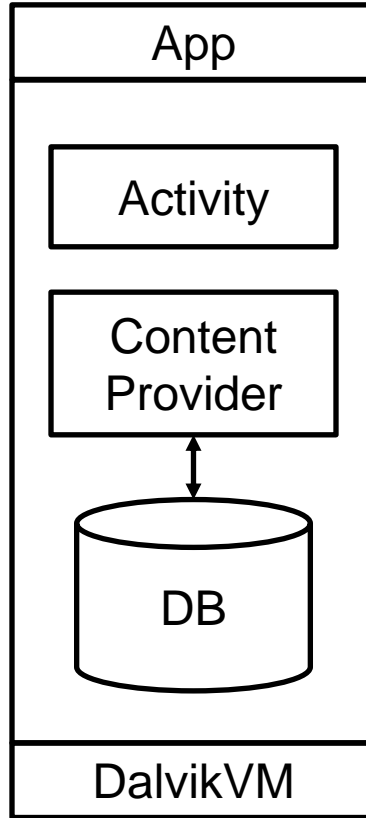
Examples of built in content providers:

Contacts

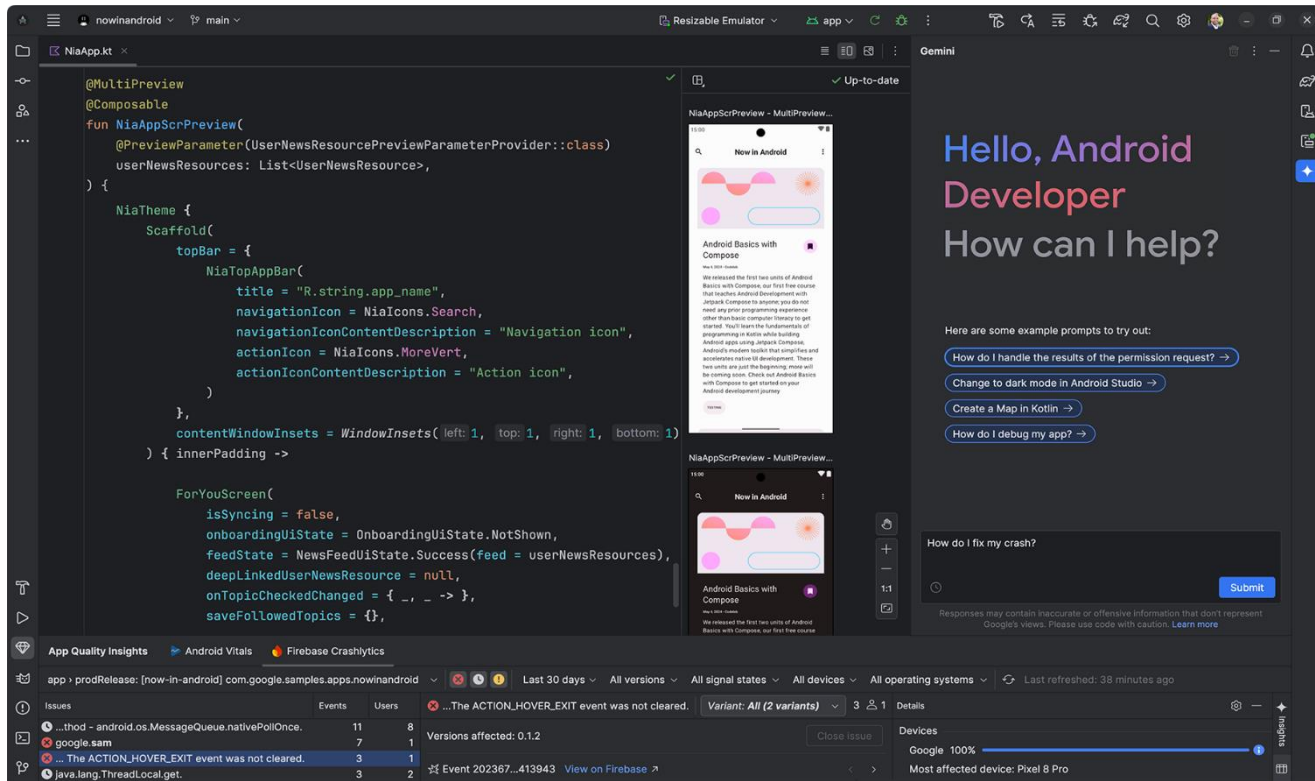
MediaStore

Settings

ContentProvider



Android Studio



<https://developer.android.com/studio/>

Project Structure

A project in Android Studio contains everything that defines your workspace for an app

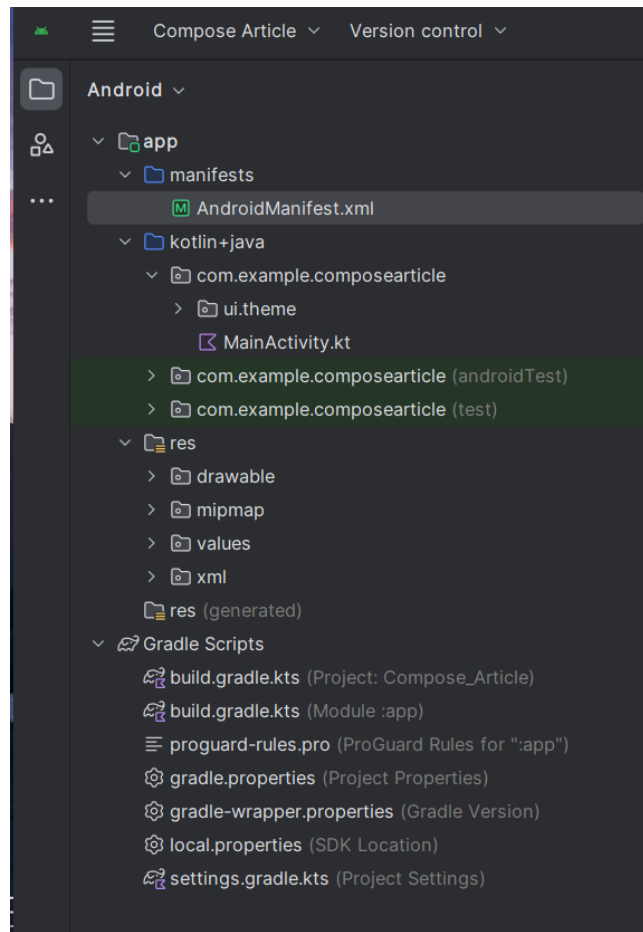
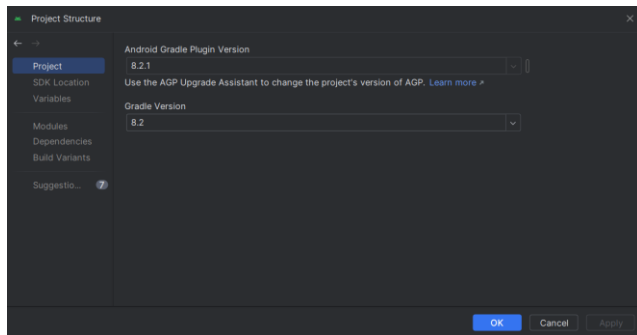
Source code, Assets, manifests

Build configurations

The project files are displayed in Android view (by default).

What is project view?

Project Structure: change various settings for your project



Kotlin vs. Java

KOTLIN is a cross platform, statically types, general purpose programming language with type inference.

KOTLIN is designed to **interoperate fully** with java but type inference allows its syntax to be more concise.

```
class MainActivity : ComponentActivity() {  
    override fun onCreate(savedInstanceState: Bundle?) {  
        super.onCreate(savedInstanceState)  
        setContent {  
            Greeting(name = "Android")  
        }  
    }  
}  
  
@Composable  
fun Greeting(name: String) {  
    Text("Hello $name!")  
}
```

Nullability in the type system helps prevent NullPointerExceptions

Semicolons are optional

Lambdas allow you to pass code to a function as a parameter

Named parameters make code easier to read

String templates simplify concatenation

What is Kotlin?

Statically-typed, modern programming language created by JetBrains

Compiles to Java bytecode

Runs in the Java Virtual Machine (JVM)

Tries to fix Java's shortcomings and issues

Java interoperability; Kotlin and Java classes can be used in the same project

Variable Declaration

Use `val` for a variable whose value never changes.

Can't reassign a value to a variable that was declared using `val`.

Use `var` for a variable whose value can change.

```
var count: Int = 10
count = 15

// not meant to be changed
val languageName: String = "Kotlin"
```

Type Inference

Compiler can infer the type based off the type of the assigned value.

Since the value of "Kotlin" is of type String, the compiler infers that languageName is also a String.

```
val languageName = "Kotlin"  
val upperCaseName = languageName.toUpperCase()  
  
// Fails to compile  
languageName.inc()
```

Null Safety

Kotlin variables can't hold null values by default

For a variable to hold a null value, it must be of a *nullable* type. specify a variable as being nullable by suffixing its type with ?

Handle nullable variables carefully or risk a dreaded NullPointerException

```
// Fails to compile
```

```
val languageName: String = null
```

```
// this will compile
```

```
val languageName: String? = null
```

Conditionals

Several mechanisms for implementing conditional logic

If else statement

```
if (count == 42) {  
    println("I have the answer.")  
} else if (count > 35) {  
    println("The answer is close.")  
} else {  
    println("The answer eludes me.")  
}
```

Conditional Expressions

Each conditional branch returns the result of the expression

```
val answerString: String = if (count == 42) {  
    "I have the answer."  
} else if (count > 35) {  
    "The answer is close."  
} else {  
    "The answer eludes me."  
}  
  
println(answerString)
```

When Expression

Each branch in a when expression represented by a condition

```
val answerString = when {  
    count == 42 -> "I have the answer."  
    count > 35 -> "The answer is close."  
    else -> "The answer eludes me."  
}  
  
println(answerString)
```

Functions

Use the fun keyword followed by the function name

```
fun generateAnswerString(): String {  
    val answerString = if (count == 42) {  
        "I have the answer."  
    } else {  
        "The answer eludes me"  
    }  
  
    return answerString  
}
```

Anonymous Functions

Can contain any number of expressions

Anonymous function that takes a `String` as input and returns the length of the input `String` as output of type `Int`

```
val stringLengthFunc: (String) -> Int = { input ->
    input.length
}

val stringLength: Int = stringLengthFunc("Android")
```


Classes & Properties

Classes represent state using properties. A [property](#) is a class-level variable that can include a getter, a setter, and a backing field.

```
class Car {  
    val wheels = listOf<Wheel>()  
}  
  
val car = Car() // construct a Car  
val wheels = car.wheels // retrieve the wheels value from the Car  
  
class Car(val wheels: List<Wheel>) // custom constructor
```

Classes & Encapsulation

The doorLock property is kept private from anything outside of the Car class. To unlock the car, you must call the unlockDoor() function passing in a valid key

```
class Car(val wheels: List<Wheel>) {  
  
    private val doorLock: DoorLock = ...  
  
    val gallonsOfFuelInTank: Int = 15  
        private set // custom getter and setter, setter is private  
  
    fun unlockDoor(key: Key): Boolean {  
        // Return true if key is valid for door lock, false otherwise  
    }  
}
```

Feature Recap

Null safety (thus drastically limiting the number of NPEs)

If, try-catch, when expressions

var and val keywords

Extension functions, inline functions, multi-value return functions

Named function parameters

Functional programming constructs (e.g. higher order functions/lambdas)

Semi-colons are optional

Coding Conventions

Kotlin <https://developer.android.com/kotlin/style-guide>

Improve the readability of software

Highlight potential bugs or avoid mistakes

Reduce training and learning curve

Kotlin Naming Conventions

Naming

If a source file contains only a single top-level class, the file name should reflect the case-sensitive name plus the `.kt` extension. Otherwise, if a source file contains multiple top-level declarations, choose a name that describes the contents of the file, apply PascalCase, and append the `.kt` extension.

```
// MyClass.kt
class MyClass { }
```



```
// Bar.kt
class Bar { }
fun Runnable.toBar(): Bar = // ...
```



```
// Map.kt
fun <T, O> Set<T>.map(func: (T) -> O): List<O> = // ...
fun <T, O> List<T>.map(func: (T) -> O): List<O> = // ...
```



Kotlin Coding Conventions

Braces

Braces are not required for `when` branches and `if` statement bodies which have no `else if/else` branches and which fit on a single line.

```
if (string.isEmpty()) return

when (value) {
    0 -> return
    // ...
}
```

Braces are otherwise required for any `if`, `for`, `when` branch, `do`, and `while` statements, even when the body is empty or contains only a single statement.

```
if (string.isEmpty())
    return // WRONG!

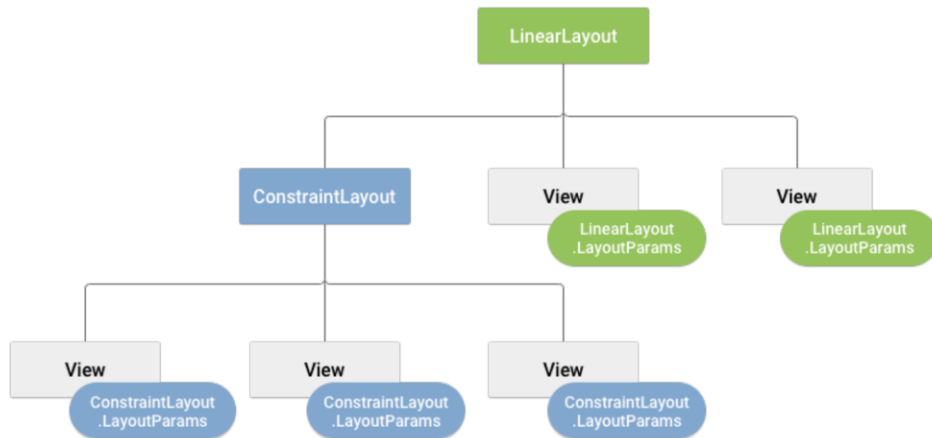
if (string.isEmpty()) {
    return // Okay
}
```

Traditional View Hierarchy

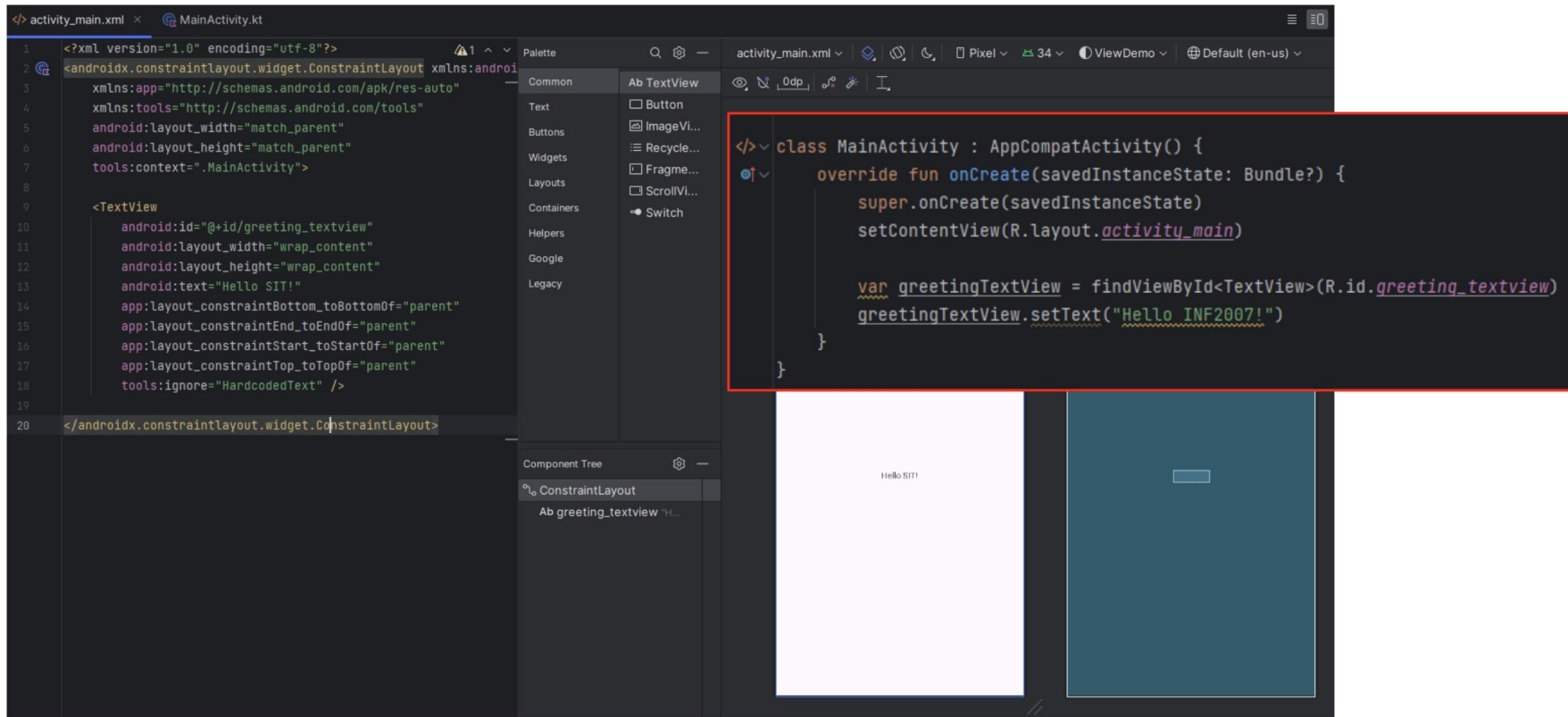
Represented as a tree of UI components within a particular layout.

Updates through methods like `findViewById()`, `setText()`, `addChild()`

Manual manipulation increases the chance of errors and maintenance complexity



Android View Class



Jetpack Compose

Modern toolkit for building native Android UI

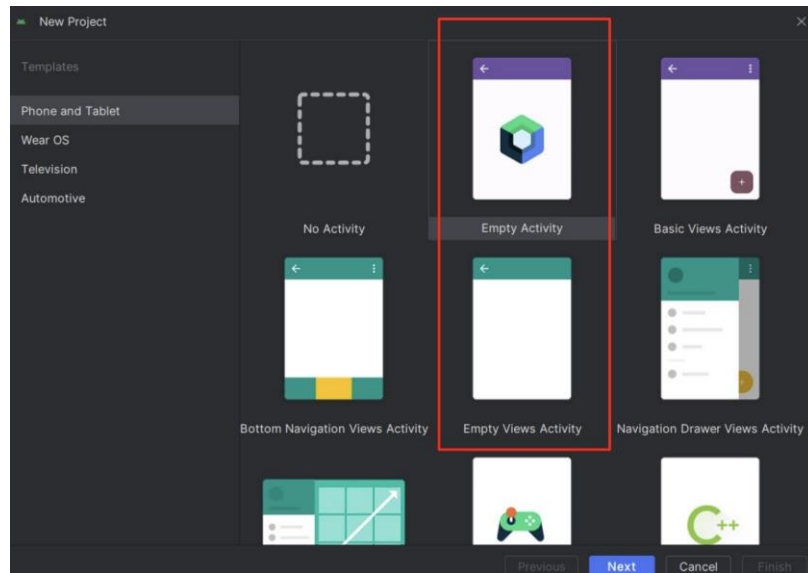
Simplifies UI development with

- Less boilerplate code and intuitive Kotlin APIs
- Powerful state management
- Easier to create dynamic layouts

Build a simple UI component with declarative functions.

Don't need to edit any XML layouts with any Layout Editor.

Note: Jetpack Compose requires minimum SDK of at least API level 21 and above



Composable Functions

Jetpack Compose is built around composable functions.

- Free you from the process of UI's construction
- It uses a Kotlin compiler plugin (DSL) to transform these composable functions into the app's UI components.

@Composable

- An annotation to tell that this is a composable function
- Note: Composable function can only be called by other composable functions

@Preview the UI in Design window

Composable Functions

Flexibility of Composable Functions

- Use if statements for conditional UI elements
- Incorporate loops for dynamic content
- Call helper functions within composable functions

UI Component

Composable Functions

```
Text("Hello World")
```

```
Button(onClick = { /* Do something */ }) { Text("Click Me")}
```

```
Icon(Icons.Default.Home, contentDescription = "Home")
```

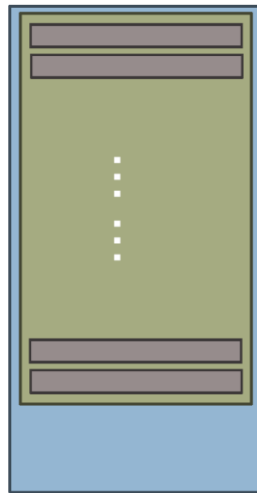
```
Image(painterResource(id = R.drawable.sample), contentDescription = "Sample Image")
```

More: <https://developer.android.com/develop/ui/compose/components> (Implemented Material Design)

Layouts

```
Row { Text("Item 1") Text("Item 2") }  
Column { Text("Item 1") Text("Item 2") }  
Box { Text("Bottom") Text("Top") }
```

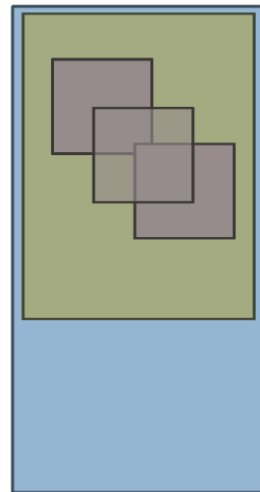
Column



Row



Box

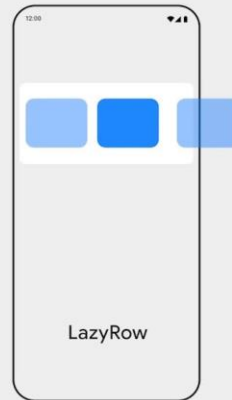


Lazy Lists: Efficient rendering for long lists.

```
e.g. LazyColumn { items(messages)  
  { message -> MessageRow(message) } }
```



LazyColumn



LazyRow



Lazy grids

Modifiers

Change:

- Composable's size
- Layout
- Appearance

Add high-level interactions, e.g., making an element clickable

List of Compose modifiers

- <https://developer.android.com/jetpack/compose/modifiers-list?authuser=1>
- Common modifiers: padding, size, fillMaxSize, fillMaxHeight, fillMaxWidth
- `Text("Styled Text", modifier = Modifier.padding(16.dp).background(Color.Blue))`

State Management

remember: stores a value in memory across recompositions.

MutableState: reactive state that triggers UI updates. Composable is the observer.

Example:

```
var count by remember { mutableStateOf(0) }
```

```
Button(onClick = { count++ }) {
```

```
    Text("Count: $count")
```

```
}
```

Summary

Increasing popularity and growth of mobile devices

Mobile device features evolved from single use cases such as calls to multiple features and general-purpose computing

Android is the popular platform and architecture for mobile devices

Android Platform Architecture

Android Studio

Kotlin

Jetpack Compose