

JAVA vs KOTLIN



Maturity



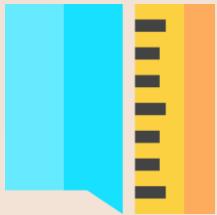
Performance



Versatility



JAVA vs KOTLIN



Conciseness



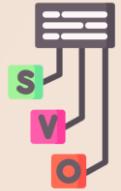
Null Safety



Modern Features

The Complete Android Developer Course

JAVA vs KOTLIN



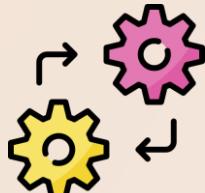
Syntax



Null Safety



Lambda Expressions



Coroutines



Your Choice?



- Project Requirements
- Team Expertise
- Personal Preferences

The Complete Android Developer Course



KOTLIN SYNTAX

In Kotlin, “fun main() { }” is the entry point of a Kotlin Program.

It serves as the starting point of execution when you run a Kotlin App.



Kotlin Variables

In Kotlin, variables are used to store information that you want to use later in your program.

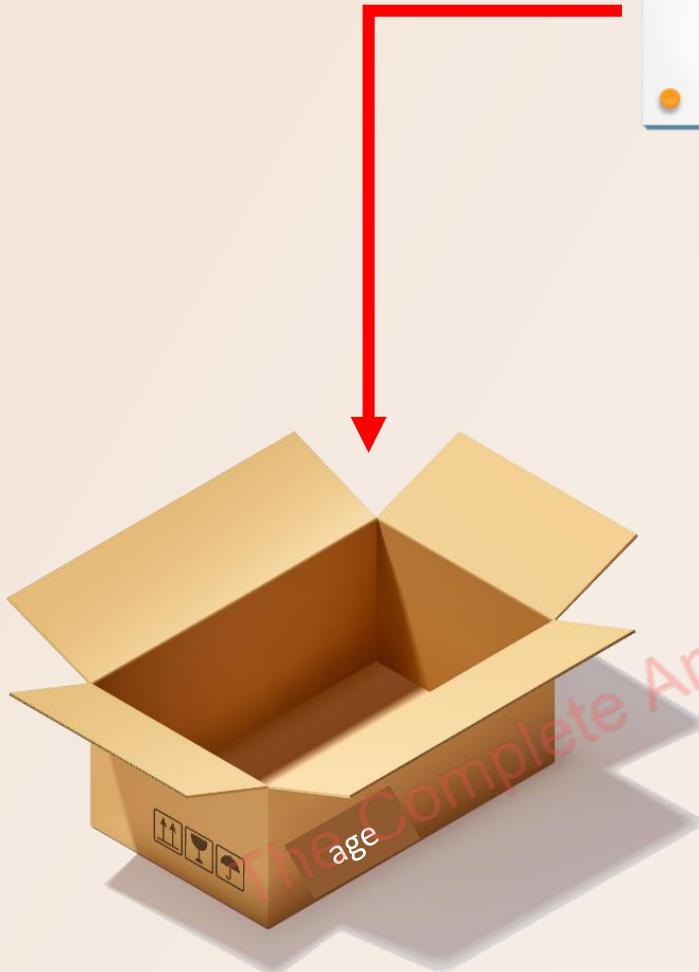
5
King



The Complete Android Developer Course

Kotlin Variables

It's like writing your age on a piece of paper and putting it in the box.



The Complete Android Developer Course

Kotlin Variables

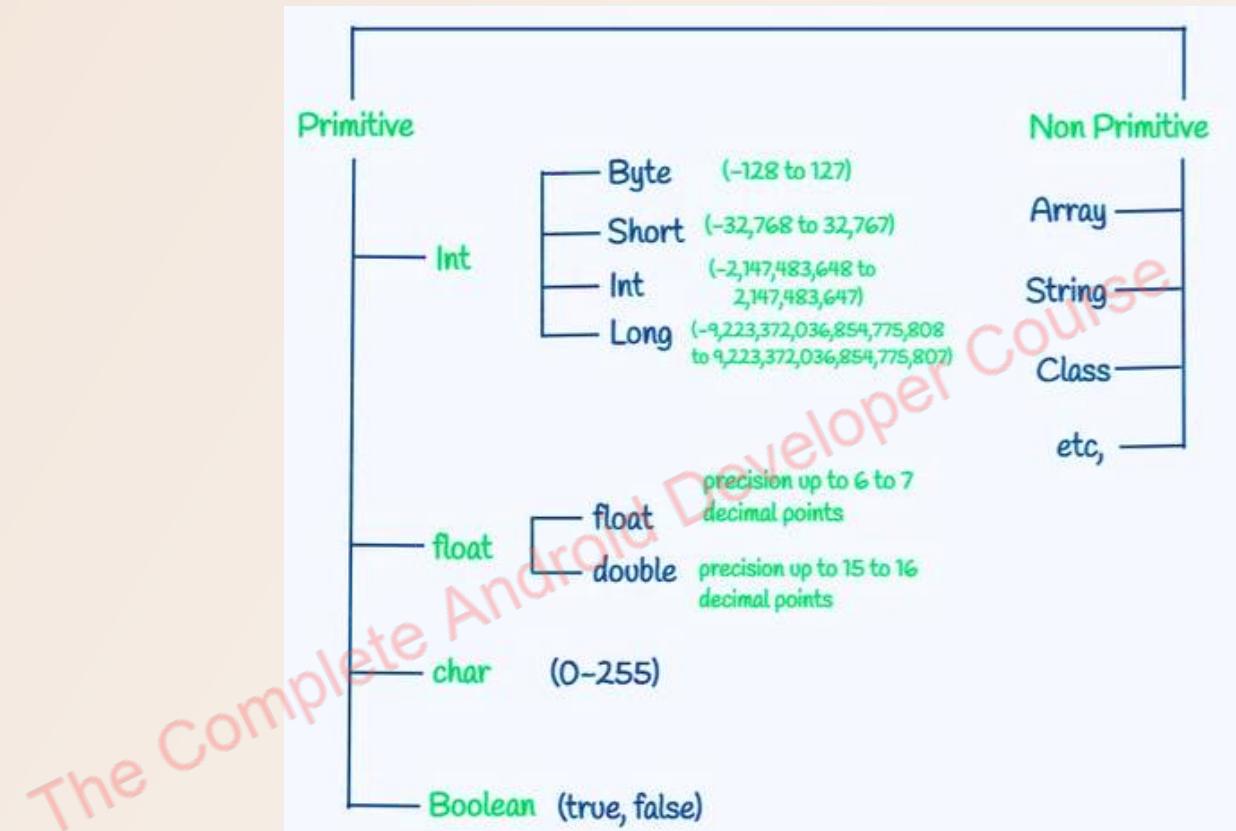


The Complete Android Developer Course

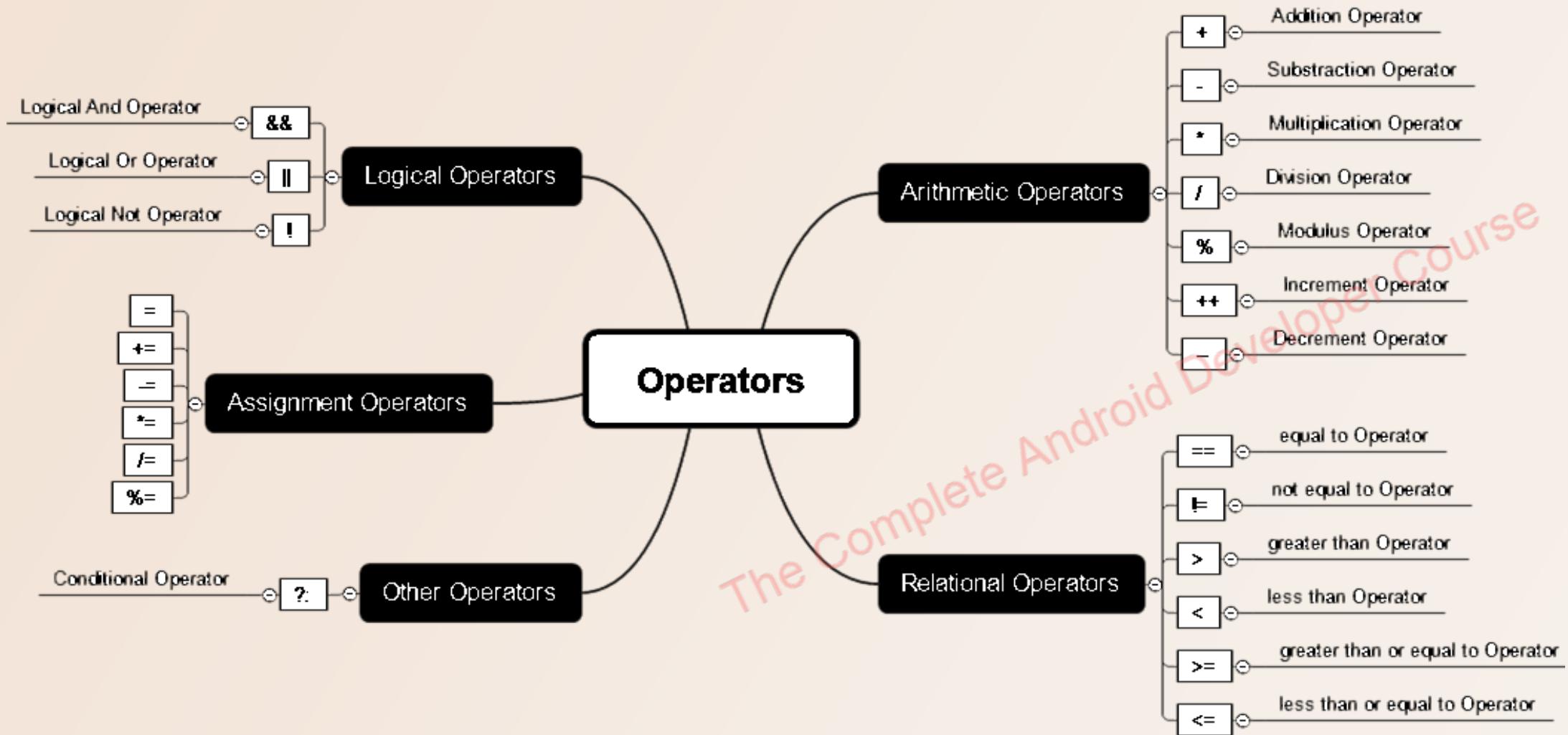
Kotlin Data Types

Data types are used to define the type of data that a variable can hold.

Each data type has specific characteristics and uses.



Kotlin Operators



The Complete Android Developer Course

```
// Arithmetic Operators: + - * / %
val result: Int = 7 % 3      //    7/3 = 2 + (remainder=1)

// Logical Operators: && || !
val result2 = !true

// Assignment Operators: = += -= *= /= %=
var x = 10
x = 15
println("The Value of x: "+x)

var y = 5
y += 3      // y = y + 3
println("The Value of y: "+y)

// Relational Operators: == != > < >= <=
var z = x <= y
println("The Value of z: "+z)
```

File Edit View Navigate Code Refactor Build Run Tools VCS Window Help KotlinBasics - HelloKotlin.kt [KotlinBasics.app.main]

KotlinBasics > app > src > main > java > com > mastercoding > kotlinbasics > HelloKotlin.kt > main

Gradle Device Manager Notifications Running Devices Device Explorer

Version Control Run Profiler Logcat App Quality Insights Build TODO Problems Terminal Services App Inspection Layout Inspector

Gradle build finished in 1 s 525 ms (a minute ago) 23:35 CRLF UTF-8 4 spaces

11:46 AM 11/1/2023

The Complete Android Developer Course

```
// String Declaration
val text1 : String = "Hello My Friends"
val text2 = " Welcome Back!"

// String Concatenation
val text3 = text1 + text2
println("The full text: "+text3)

// String Templates
val name = "Jack"
val age = 30
val info = "My name is $name and I am $age years old"
println(info)

// String Interpolation
val x = 5
val y = 3
val result = "The sum of $x and $y is ${x + y}"
println(result)
```

File Edit View Navigate Code Refactor Build Run Tools VCS Window Help KotlinBasics - HelloKotlin.kt [KotlinBasics.app.main]

KotlinBasics > app > src > main > java > com > mastercoding > kotlinbasics > HelloKotlin.kt

Gradle Device Manager

Resource Manager Project Bookmarks Build Variants Structure

Version Control Run Profiler Logcat App Quality Insights Build TODO Problems Terminal Services App Inspection

Gradle build finished in 1 s 694 ms (a minute ago)

Nexus 6P API 33 44:2 CRLF 12:13 PM 11/1/2023

File Edit View Navigate Code Refactor Build Run Tools VCS Window Help KotlinBasics - HelloKotlin.kt [KotlinBasics.app.main]

KotlinBasics > app > src > main > java > com > mastercoding > kotlinbasics > HelloKotlin.kt

HelloKotlinKt Nexus 6P API 33

Resource Manager

Project

Bookmarks

Build Variants

Structure

Gradle

Device Manager

Notifications

Running Devices

Device Explorer

Version Control

Run

Profiler

Logcat

App Quality Insights

Build

TODO

Problems

Terminal

Services

App Inspection

Layout Inspector

Gradle build finished in 1 s 694 ms (2 minutes ago)

44:2 CRLF UTF-8 4 spaces

12:14 PM 11/1/2023

Show hidden icons

1 5

```
// String Functions and Properties
val text = "Welcome to our course"
val length = text.length
val subText = text.substring(0,7)
println("the length of text is: "+length)
println("the substring "+subText)

// String Comparison
val str1 = "Hello"
val str2 = "Hello"
val comparisonResult = str1.equals(str2)
println("String Comparison Result: "+comparisonResult)

// String Literals
val newLineText = "This is the first line \n This is the second line"
println(newLineText)
```

The Complete Android Developer Course

```
// if statement
val age = 14
if (age > 18){
    println("You are an adult")
}

// if-else statement
val score = 40
if (score >= 60){
    println("Pass")
} else {
    println("Fail")
}
```

File Edit View Navigate Code Refactor Build Run Tools VCS Window Help KotlinBasics - HelloKotlin.kt [KotlinBasics.app.main]

KotlinBasics > app > src > main > java > com > mastercoding > kotlinbasics > HelloKotlin.kt > main

Gradle Device Manager Notifications Running Devices Device Explorer

Version Control Run Profiler Logcat App Quality Insights Build TODO Problems Terminal Services App Inspection Layout Inspector

Gradle build finished in 1 s 210 ms (2 minutes ago)

16:6 CRLF UTF-8 4 spaces

1:29 PM 11/1/2023

The Complete Android Developer Course

```
// when Expression
val day = 5
when(day){
    1 -> println("Monday")
    2 -> println("Tuesday")
    3 -> println("Wednesday")
    else -> println("Unknown day")
}

// Nested Conditionals
val isRaining = true
val isCold = false

if(isRaining){
    if (isCold){
        println("Use Umbrella and Coat")
    }else{
        println("Use only Umbrella")
    }
}else{
    println("No need for an umbrella")
}
```

Font size: 28pt Reset to 13pt in all editors

File Edit View Navigate Code Refactor Build Run Tools VCS Window Help KotlinBasics - HelloKotlin.kt [KotlinBasics.app.main]

KotlinBasics > app > src > main > java > com > mastercoding > kotlinbasics > HelloKotlin.kt > main

Gradle Device Manager

Resource Manager Project Bookmarks Build Variants Structure

Version Control Run Profiler Logcat App Quality Insights Build TODO Problems Terminal Services App Inspection

Layout Inspector

Gradle build finished in 1 s 210 ms (2 minutes ago)

16:6 CRLF UTF-8 4 spaces

1:29 PM 11/1/2023

The Complete Android Developer Course

File Edit View Navigate Code Refactor Build Run Tools VCS Window Help KotlinBasics - HelloKotlin.kt [KotlinBasics.app.main]

KotlinBasics > app > src > main > java > com > mastercoding > kotlinbasics > HelloKotlin.kt > main

HelloKotlin.kt

```
var x = 1
do {
    println("This will be printed at least once..")
    x++
}while (x < 0)

// break: terminates the loop and transfers control to the statement
//         following the loop
// continue: Skips the current iteration and proceeds to the next
//             iteration of the loop

for (i in 1..10){
    if ( i == 5){
        break
    }
    println(i)
}

for (i in 1..10){
    if ((i % 2) == 0){
        continue
    }
    println(i)
}
```

Resource Manager

Project

Bookmarks

Build Variants

Structure

Version Control Run Profiler Logcat App Quality Insights Build TODO Problems Terminal Services App Inspection

Gradle

Device Manager

Notifications

Running Devices

Device Explorer

Layout Inspector

Gradle build finished in 1 s 571 ms (3 minutes ago)

29:1 CRLF UTF-8 4 spaces

2:16 PM 11/1/2023

The Complete Android Developer Course

```
3 ► fun main(){
4     // Array: is a collection of elements of the same data type
5     //
6     // organized in a specific order and accessed by
7     // an index.
8
9     // Array Declaration
10    val osNames = arrayOf("Windows", "Android", "MacOS", "Linux")
11
12    // Accessing Elements
13    val firstElement = osNames[0]
14    println(firstElement)
15
16    // Modifying Elements
17    osNames[1] = "iOS"
```

File Edit View Navigate Code Refactor Build Run Tools VCS Window Help KotlinBasics - HelloKotlin.kt [KotlinBasics.app.main]

KotlinBasics > app > src > main > java > com > mastercoding > kotlinbasics > HelloKotlin.kt > main

HelloKotlinKt Nexus 6P API 33

Gradle Device Manager

Resource Manager

Project

Bookmarks

Build Variants

Structure

Version Control

Run

Profiler

Logcat

App Quality Insights

Build

TODO

Problems

Terminal

Services

App Inspection

Layout Inspector

Gradle build finished in 2 s 166 ms (6 minutes ago)

30:1 CRLF UTF-8 4 spaces

10:55 AM 11/2/2023

```
// Modifying Elements
osNames[1] = "iOS"
println(osNames[1])

// Array Size
val size = osNames.size
println("The size of this array: $size")

// Iterating through an array
for (name in osNames){
    println(name)
}

osNames.forEach { name -> println(name) }
```

The Complete Android Developer Course

File Edit View Navigate Code Refactor Build Run Tools VCS Window Help KotlinBasics - HelloKotlin.kt [KotlinBasics.app.main]

KotlinBasics > app > src > main > java > com > mastercoding > kotlinbasics > HelloKotlin.kt > main

HelloKotlinKt Nexus 6P API 33

Resource Manager

Project

Bookmarks

Build Variants

Structure

Version Control Run Profiler Logcat App Quality Insights Build TODO Problems Terminal Services App Inspection

Gradle

Device Manager

Notifications

Running Devices

Device Explorer

1 HelloKotlin.kt

2

3 ► fun main(){

4 // Range: interval between 2 values

5

6 // Closed Range: startValue..endValue

7 val range = 1..5

8

9 // Half-Open Range: startValue until endValue

10 val range2 = 1 until 5

11

12 // Iterating through a specific range

13 for (i in range2){

14 println(i)

15 }

16

The Complete Android Developer Course

Gradle build finished in 1 s 760 ms (2 minutes ago)

Layout Inspector

14:19 CRLF UTF-8 4 spaces

11:18 AM 11/2/2023

The Complete Android Developer Course

```
3 ► fun main(){
4     // Functions: are blocks of code that encapsulate a specific
5     // task or functionality
6
7     // Function Declaration:
8     fun functionName(parameter1: Type, parameter2: Type): ReturnType{
9         // Function Body
10        // perform some operations
11        // Optionally return a value
12    }
13
14    sayHello(name: "Jack")
15    var result = sumTwoNumbers(x: 5.3, y: 3.7)
16
17    println(result)
18}
19
20 fun sayHello(name: String, age: String = "Not Specified"){
21     println("Hello $name , you're age is $age")
22}
```

File Edit View Navigate Code Refactor Build Run Tools VCS Window Help KotlinBasics - HelloKotlin.kt [KotlinBasics.app.main]

KotlinBasics > app > src > main > java > com > mastercoding > kotlinbasics > HelloKotlin.kt

Resource Manager

Project

Bookmarks

Build Variants

Structure

Gradle

Device Manager

Notifications

Running Devices

Device Explorer

Version Control Run Profiler Logcat App Quality Insights Build TODO Problems Terminal Services App Inspection

Gradle build finished in 1 s 565 ms (2 minutes ago)

Layout Show desktop

34:2 CRLF UTF-8 4 spaces

12:35 PM 11/2/2023

File Edit View Navigate Code Refactor Build Run Tools VCS Window Help KotlinBasics - HelloKotlin.kt [KotlinBasics.app.main]

KotlinBasics > app > src > main > java > com > mastercoding > kotlinbasics > HelloKotlin.kt

HelloKotlinKt Nexus 6P API 33

Resource Manager

Project

Bookmarks

Build Variants

Structure

Gradle

Device Manager

Notifications

Running Devices

Device Explorer

Version Control

Run

Profiler

Logcat

App Quality Insights

Build

TODO

Problems

Terminal

Services

App Inspection

Layout Inspector

Gradle build finished in 1 s 565 ms (2 minutes ago)

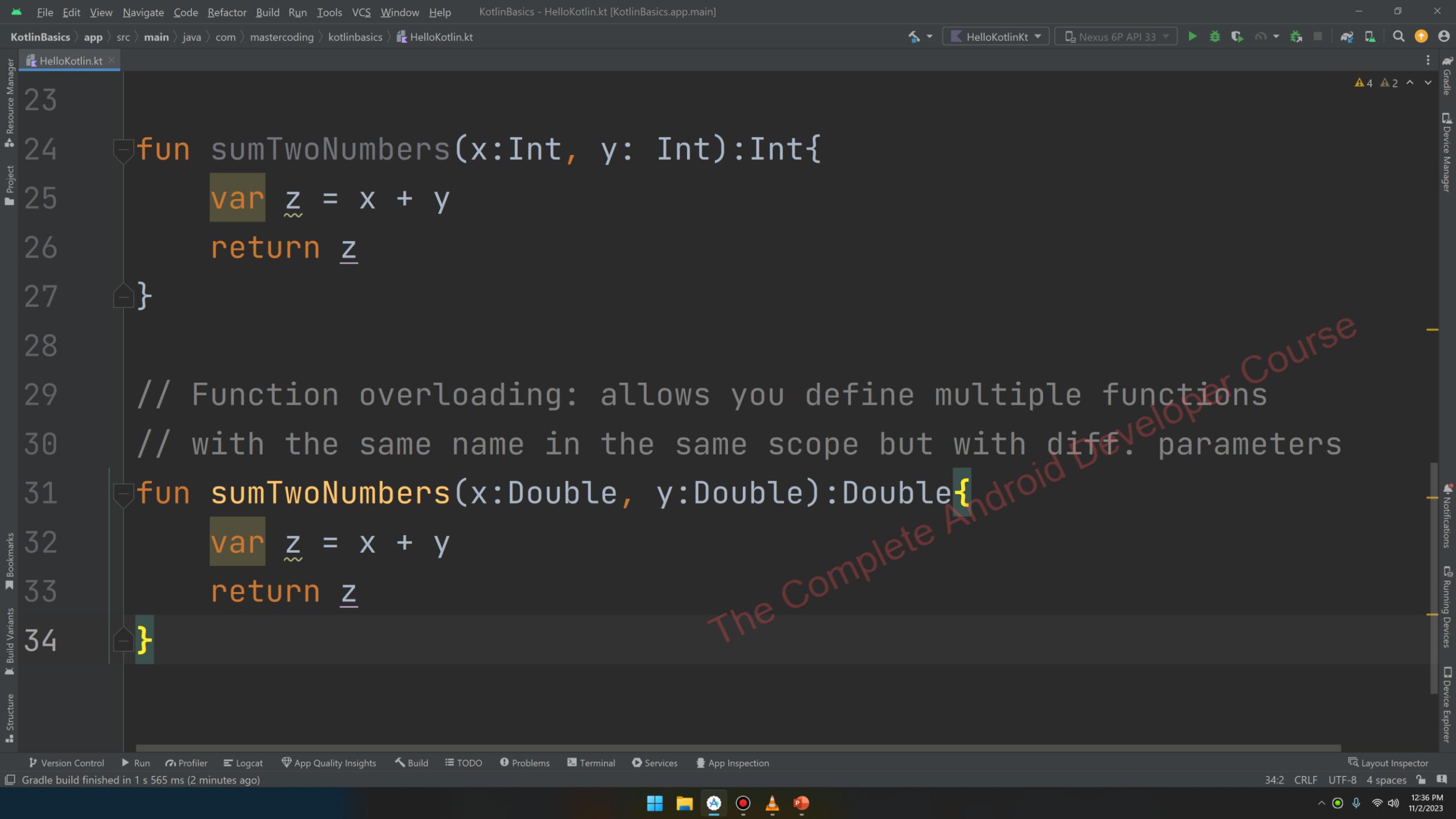
34:2 CRLF UTF-8 4 spaces

12:36 PM 11/2/2023

4 2 ^ v

```
23
24     fun sumTwoNumbers(x:Int, y: Int):Int{
25         var z = x + y
26         return z
27     }
28
29     // Function overloading: allows you define multiple functions
30     // with the same name in the same scope but with diff. parameters
31     fun sumTwoNumbers(x:Double, y:Double):Double{
32         var z = x + y
33         return z
34     }
```

The Complete Android Developer Course



Object-Oriented Programming

You can create different things by combining these bricks in various ways...

In programming, OOP is a bit like playing with LEGO bricks. Instead of bricks, you have objects

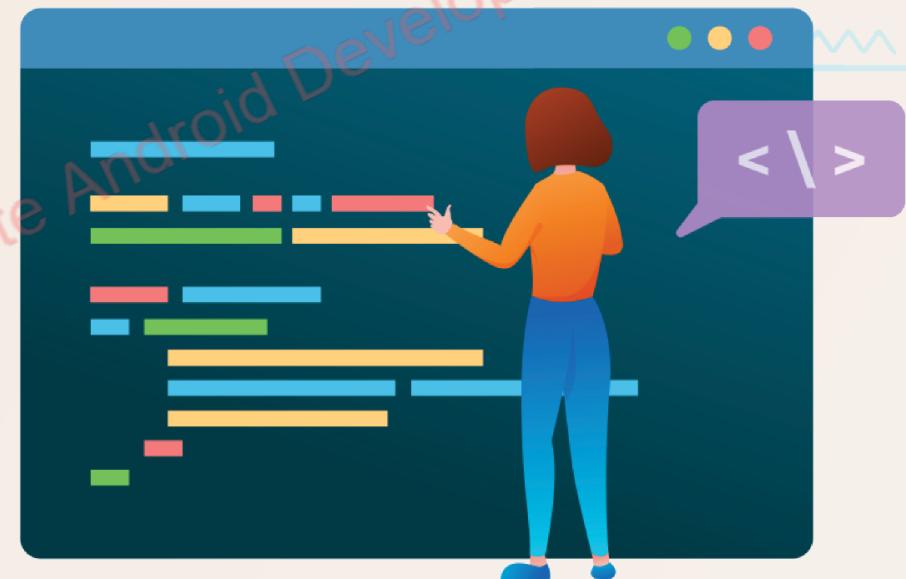


The Complete Android Developer Course

Object-Oriented Programming

OOP stands for Object-Oriented Programming.

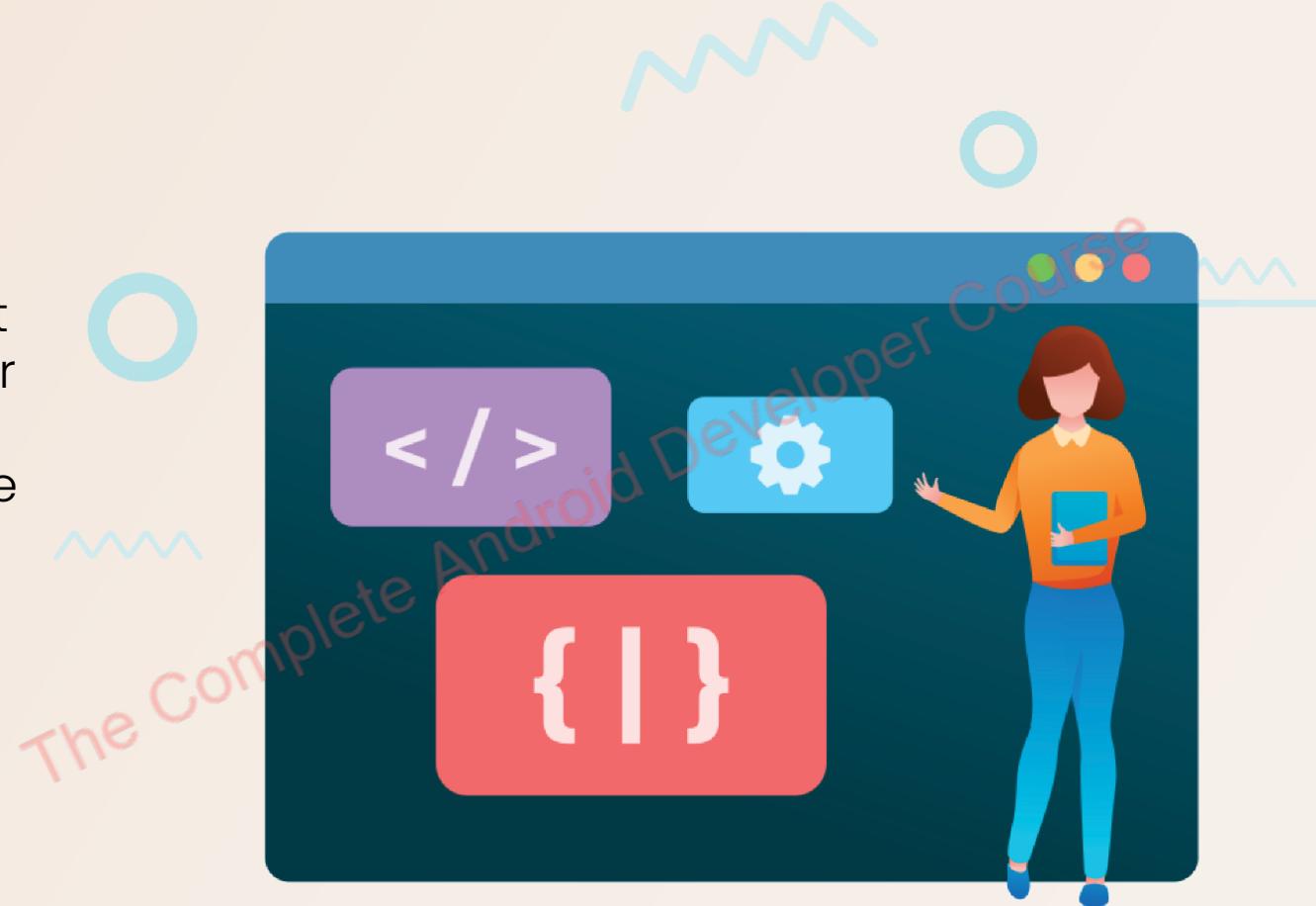
Procedural programming is about writing procedures or methods that perform operations on the data, while object-oriented programming is about creating objects that contain both data and methods.



Object-Oriented Programming

Object-oriented programming has several advantages over procedural programming:

- OOP is faster and easier to execute
- OOP provides a clear structure for the programs
- OOP helps to keep the Kotlin code DRY "Don't Repeat Yourself", and makes the code easier to maintain, modify and debug
- OOP makes it possible to create full reusable applications with less code and shorter development time



OOP Concepts

- Classes
- Objects
- Constructors
- Class Functions
- Inheritance
- Polymorphism
- Encapsulation
- Interfaces
- Abstraction

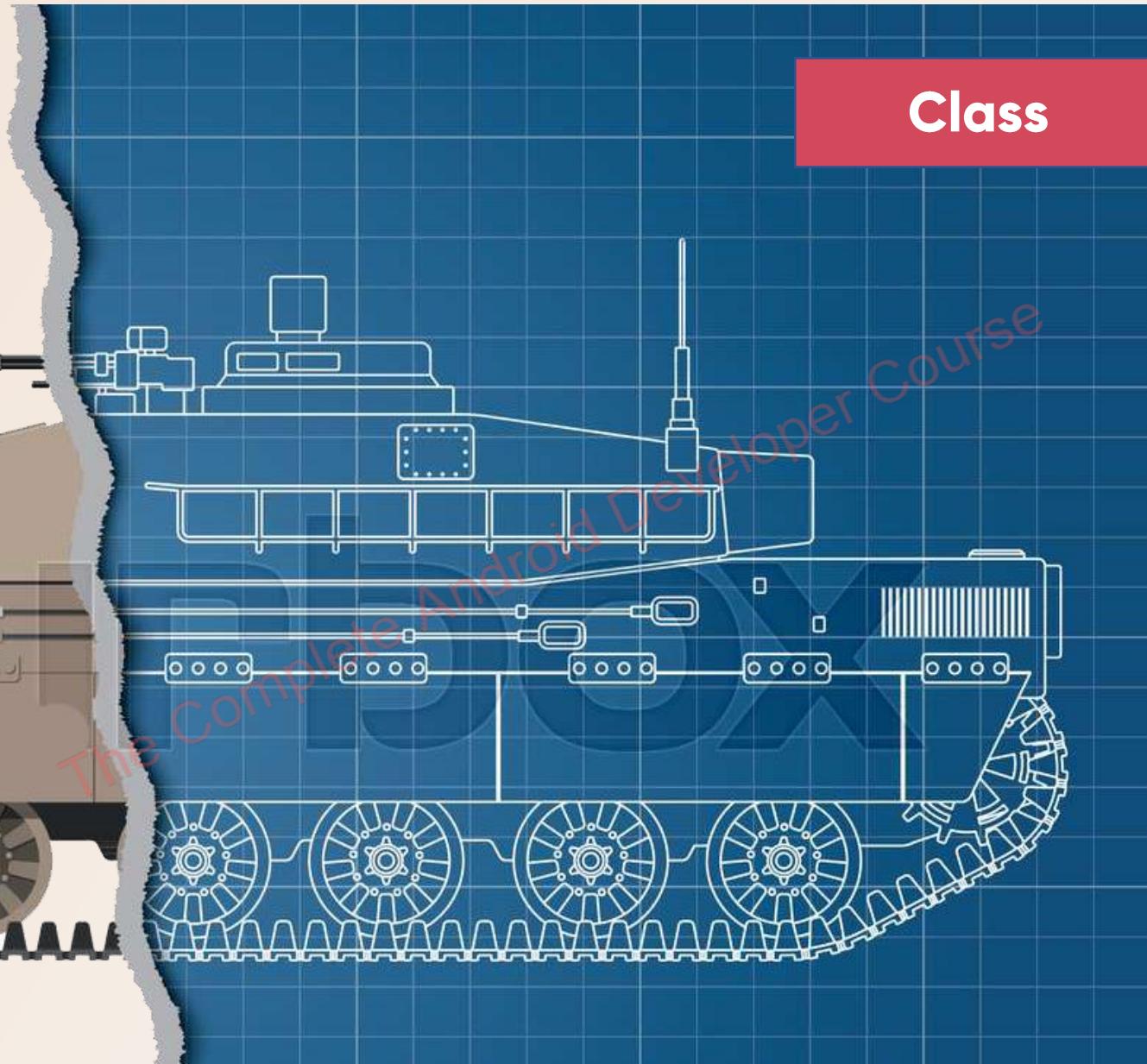


Class

A class is a blueprint that defines the properties and behaviors of objects. It encapsulates data and functions that operate on the data.

Class

Object



The Complete Android Developer Course

```
File Edit View Navigate Code Refactor Build Run Tools VCS Window Help KotlinBasics - HelloKotlin.kt [KotlinBasics.app.main]
KotlinBasics > app > src > main > java > com > mastercoding > kotlinbasics > HelloKotlin.kt Robot Robot(name: String)
HelloKotlinKt Nexus 6P API 33 ...
Gradle Device Manager
Resource Manager
Project
HelloKotlin.kt
3 ► fun main() {
4     // Creating Objects from Robot class
5     val robot1 : Robot = Robot(name: "Buddy")
6     robot1.walk()
7     robot1.speak("Hello My Friends")
8
9     val robot2 = Robot(name: "Max")
10    robot2.walk()
11    robot2.speak("Hello From Robot2")
12
13 }
14 // The primary Constructor: define and initialize properties
15 // for a class
16
17 // class MyClass(parameter1:Type, parameter2:Type){}
18
19 class Robot(val name: String){
20     fun walk(){
21         println("The Robot is Walking Now...")
22     }
23     fun speak(message: String){
24         println("$name says: $message")
25     }
26 }
27

```

Version Control Run Profiler Logcat App Quality Insights Build TODO Problems Terminal Services App Inspection

Gradle build finished in 1 s 518 ms (43 minutes ago)

Layout Inspector

20:12 CRLF UTF-8 4 spaces

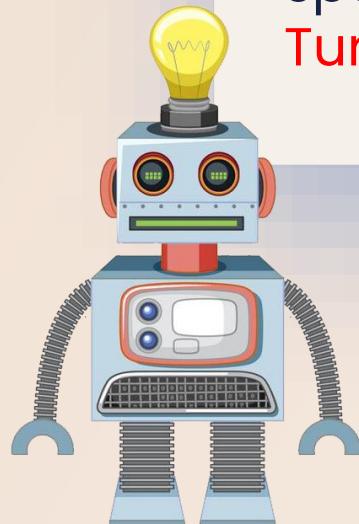
2:25 PM 11/2/2023

Inheritance

The Problem: If you want to add a new feature “ShutDown”, you need to implement the same code for each robot. Moreover, we need to duplicate the code for walk(), speak(), and ShutDown().

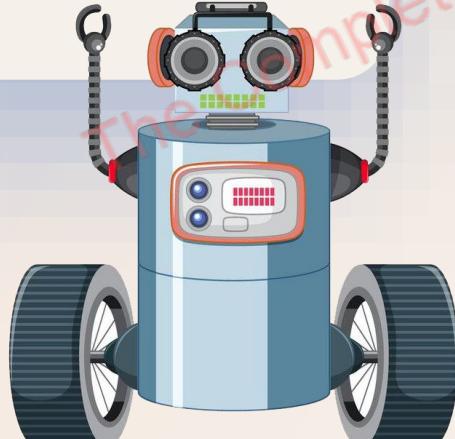
Robot

walk()
speak()
TurnOnLight()



SuperRobot

walk()
speak()
CleanHouse()

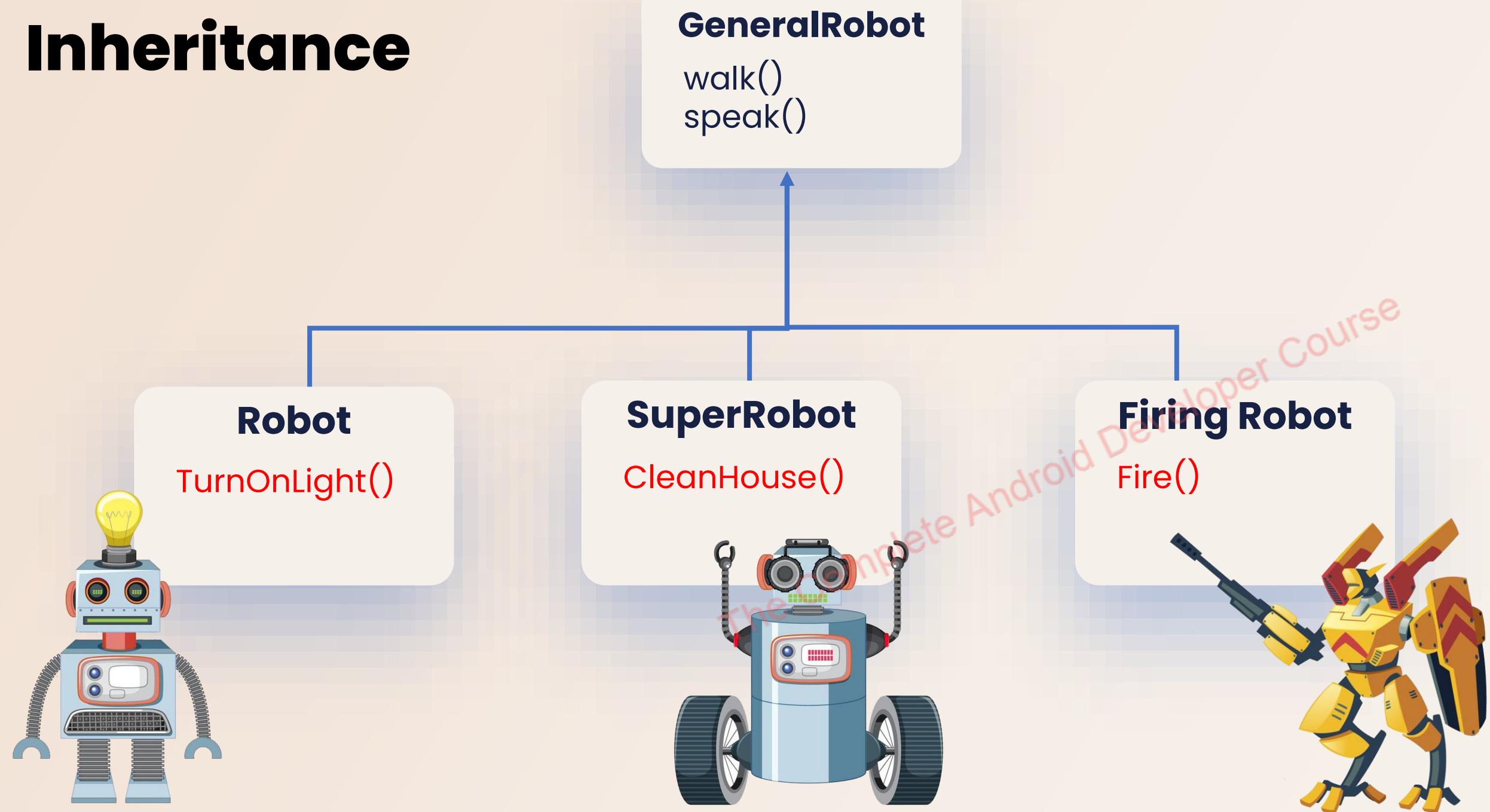


Firing Robot

walk()
speak()
Fire()



Inheritance



The screenshot shows the Android Studio interface with several Kotlin files open in the editor. A large watermark reading "The Complete Android Developer Course" is diagonally across the center.

Left Editor: HelloKotlin.kt

```
1 package com.mastercoding.kotlinbasics
2
3 fun main() {
4     // Creating Objects from Robot class
5     val ordinaryRobot = OrdinaryRobot(name: "Max")
6     ordinaryRobot.walk()
7
8     val superRobot = SuperRobot(name: "Buddy")
9     superRobot.cleanHouse()
10
11    val firingRobot = FiringRobot(name: "hero")
12    firingRobot.fire()
13
14 }
15
16
```

Right Editor: GeneralRobot.kt

```
1
2
3 open class GeneralRobot (val name: String){
4     fun walk(){
5         println("The Robot is Walking Now...")
6     }
7     fun speak(message: String){
8         println("$name says: $message")
9     }
10 }
```

FiringRobot.kt

```
1
2
3 class FiringRobot(name: String): GeneralRobot(name) {
4     fun fire(){
5         println("Firing ....")
6     }
7 }
```

SuperRobot.kt

```
1
2
3 class SuperRobot (name: String):GeneralRobot(name){
4     fun cleanHouse(){
5         println("The Robot is cleaning the house")
6     }
7 }
```

OrdinaryRobot.kt

```
1
2
3 class OrdinaryRobot (name: String):GeneralRobot(name){
4     fun turnOnLight(){
5         println("The light is turn off")
6     }
7 }
```

Bottom Center: Inheritance

Bottom Navigation Bar:

- Version Control
- Run
- Profiler
- Logcat
- App Quality Insights
- Build
- TODO
- Problems
- Terminal
- Services
- App Inspection

Gradle build finished in 1 s 64 ms (2 minutes ago)

16:1 CRLF UTF-8 4 spaces

Layout Inspector

REC Camera icon

3:09 PM

11/2/2023

The Complete Android Developer Course

A screenshot of the Android Studio IDE interface. The top navigation bar includes File, Edit, View, Navigate, Code, Refactor, Build, Run, Tools, VCS, Window, Help, and the current project name, KotlinBasics - GeneralRobot.kt [KotlinBasics.app.main]. The left sidebar contains Project, Resource Manager, Bookmarks, Build Variants, Structure, and a large red watermark reading 'The Complete Android Developer Course'. The main content area displays five Kotlin code files:

- GeneralRobot.kt**:

```
open class GeneralRobot (){  
    var name:String  
    var modelYear:String  
  
    // init block: used to execute code when an instance of a class is created  
    init {  
        name = ""  
        modelYear = ""  
        println("A new robot is created")  
    }  
  
    // Secondary Constructors:  
    constructor(name: String, modelYear: String): this(){  
        this.name = name  
        this.modelYear = modelYear  
    }  
    constructor(name: String):this(){  
        this.name = name  
        this.modelYear = "Unknown Model Year"  
    }  
  
    fun walk(){  
        println("The Robot is Walking Now...")  
    }  
    fun speak(message: String){  
        println("$name says: $message")  
    }  
}
```
- FiringRobot.kt**:

```
package com.mastercoding.kotlinbasics  
  
class FiringRobot: GeneralRobot {  
    constructor(name: String, modelYear: String):super(name,modelYear)  
    constructor(name: String):super(name)  
  
    fun fire(){  
        println("Firing ....")  
    }  
}
```
- SuperRobot.kt**:

```
package com.mastercoding.kotlinbasics  
  
class SuperRobot (name: String):GeneralRobot(name){  
    fun cleanHouse(){  
        println("The Robot is cleaning the house")  
    }  
}
```
- OrdinaryRobot.kt**:

```
package com.mastercoding.kotlinbasics  
  
class OrdinaryRobot (name: String):GeneralRobot(name){  
    fun turnOnLight(){  
        println("The light is turn off")  
    }  
}
```
- HelloKotlin.kt**:

```
fun main() {  
    // Creating Objects from Robot class  
    val ordinaryRobot = OrdinaryRobot( name: "Max")  
    ordinaryRobot.walk()  
  
    val superRobot = SuperRobot( name: "Buddy")  
    superRobot.cleanHouse()  
  
    val firingRobot = FiringRobot( name: "hero")  
    val firingRobot2 = FiringRobot( name: "Hero2", modelYear: "2024")  
    firingRobot.fire()  
    println(firingRobot2.modelYear)  
}
```

The bottom navigation bar includes Version Control, Run, Profiler, Logcat, App Quality Insights, Build, TODO, Problems, Terminal, Services, and App Inspection. A status bar at the bottom shows 'Gradle build finished in 1 s 556 ms (15 minutes ago)', '24:5 CRLF UTF-8 4 spaces', and system icons.

File Edit View Navigate Code Refactor Build Run Tools VCS Window Help KotlinBasics - GeneralRobot.kt [KotlinBasics.app.main]

KotlinBasics > app > src > main > java > com > mastercoding > kotlinbasics > GeneralRobot

HelloKotlinKt Nexus 6P API 33

Resource Manager

Project

Bookmarks

Build Variants

Structure

GeneralRobot.kt

```
// Getters and Setters
// custom getter
get(){
    println("Getting the model Year, Please Wait...")
    return field
}

// Custom Setter:
set(value){
    println("Changing the model year, Please Wait")

    if (value.equals("2025")){
        println("You can't create future robots")
    }else{
        field = value
    }
}
```

FiringRobot.kt

```
class FiringRobot: GeneralRobot {
    constructor(name:String, modelYear: String)
        :super(name,modelYear)

    constructor(name:String):super(name)

    fun fire(){
        println("Firing ....")
    }
}
```

SuperRobot.kt

```
package com.mastercoding.kotlinbasics

class SuperRobot (name: String):GeneralRobot(name){
    fun cleanHouse(){
        println("The Robot is cleaning the house")
    }
}
```

OrdinaryRobot.kt

```
package com.mastercoding.kotlinbasics

class OrdinaryRobot (name: String):GeneralRobot(name){
    fun turnOnLight(){
        println("The light is turn off")
    }
}
```

The Complete Android Developer Course

Version Control Run Profiler Logcat App Quality Insights Build TODO Problems Terminal Services App Inspection

Gradle build finished in 1 s 543 ms (moments ago)

Layout Inspector

25:1 CRLF UTF-8 4 spaces

12:42 PM 11/3/2023

The Complete Android Developer Course

```
File Edit View Navigate Code Refactor Build Run Tools VCS Window Help KotlinBasics - GeneralRobot.kt [KotlinBasics.app.main]
KotlinBasics > app > src > main > java > com > mastercoding > kotlinbasics > GeneralRobot
HelloKotlinKt Nexus 6P API 33
HelloKotlin.kt GeneralRobot.kt FiringRobot.kt
1 open class GeneralRobot (){
2     internal var serialNumber: String = "12345"
3
4     /* Visibility Modifiers
5      * public (default): visible every where
6      * private: not accessible from other classes
7      * protected: visible with subclasses
8      * internal: visible within the same module
9      */
10
11     var name:String
12     var modelYear:String = ""
13
14     constructor(name:String):super(name)
15     fun fire(){
16         println("Firing ....")
17     }
18
19     fun cleanHouse(){
20         println("The Robot is cleaning the house")
21     }
22
23     fun turnOnLight(){
24         println("The light is turn off")
25     }
26
27 }
```

Visibility Modifiers

Version Control Run Profiler Logcat App Quality Insights Build TODO Problems Terminal Services App Inspection

Gradle build finished in 1 s 543 ms (20 minutes ago)

Layout Inspector

3:5 CRLF UTF-8 4 spaces

1:01 PM 11/3/2023

The Complete Android Developer Course

```
File Edit View Navigate Code Refactor Build Run Tools VCS Window Help KotlinBasics - RobotActions.kt [KotlinBasics.app.main]
```

KotlinBasics > app > src > main > java > com > mastercoding > kotlinbasics > RobotActions

Resource Manager

Project

HelloKotlin.kt GeneralRobot.kt

```
3 abstract class GeneralRobot():RobotActions{  
4  
5     // Abstract Class: cannot be instantiated by their  
6     // own, and are typically meant to be subclassed  
7  
8     // Abstract Functions: don't have an implementation  
9     // in the abstract class but must be implemented  
10    // in subclasses  
11  
12    // Interface: is a blueprint for a set of functions or  
13    // properties that a class must implement
```

RobotActions.kt

```
3 interface RobotActions {  
4     fun start()  
5     fun stop()  
6 }
```

Resource Manager

FiringRobot.kt

```
3 class FiringRobot: GeneralRobot {  
4     constructor(name:String, modelYear: String)  
5         :super(name,modelYear)  
6  
7     constructor(name:String):super(name)  
8  
9     fun fire(){  
10        println("Firing .....")  
11    }  
12  
13    override fun start(){  
14        println("Starting firing robot")  
15    }  
16  
17    override fun stop(){  
18        println("Stopping firing robot")  
19    }
```

SuperRobot.kt

```
3 class SuperRobot (name: String):GeneralRobot(name){  
4     fun cleanHouse(){  
5         println("The Robot is cleaning the house")  
6     }  
7  
8     override fun start(){  
9         TODO(reason: "Not yet implemented")  
10    }  
11  
12    override fun stop(){  
13        TODO(reason: "Not yet implemented")  
14    }
```

OrdinaryRobot.kt

```
3 class OrdinaryRobot (name: String):GeneralRobot(name){  
4     fun turnOnLight(){  
5         println("The light is turn off")  
6     }  
7  
8     override fun start(){  
9         TODO(reason: "Not yet implemented")  
10    }  
11  
12    override fun stop(){  
13        TODO(reason: "Not yet implemented")  
14    }
```

Gradle Device Manager Notifications Running Devices Device Explorer

Version Control Run Profiler Logcat App Quality Insights Build TODO Problems Terminal Services App Inspection

Gradle build finished in 1 s 961 ms (2 minutes ago)

Layout Inspector

3:25 CRLF UTF-8 4 spaces

1:33 PM 11/3/2023

Interfaces and Abstract

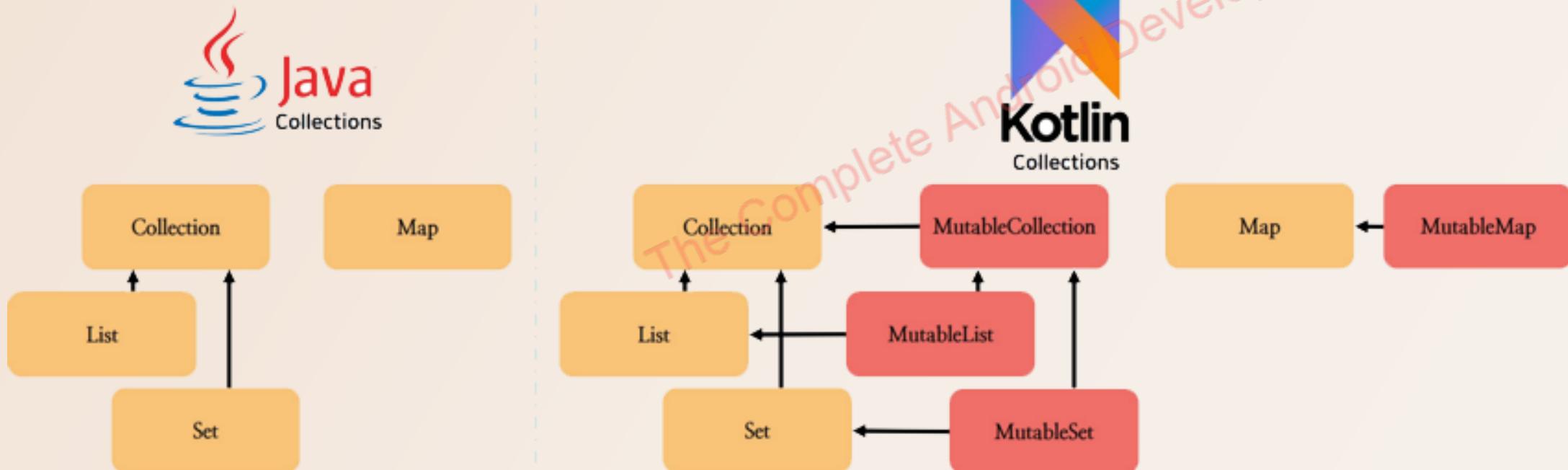
Kotlin Collections

A collection usually contains a number of objects of the same type and these objects in the collection are called elements or items.

Kotlin Standard Library provides a rich set of tools for managing collections.

Types of Collections

- Immutable Collection
- Mutable Collection



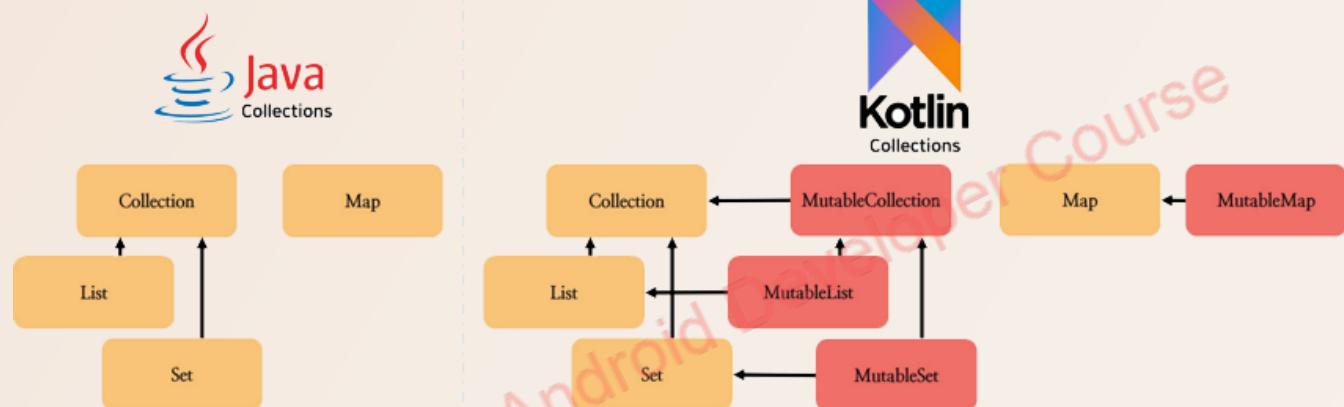
Mutable vs Immutable

Immutable Collection

It means that it supports only read-only functionalities and can not be modified its elements.

Immutable Collections and their corresponding methods are:

- List – listOf() and listOf<T>()
- Set – setOf()
- Map – mapOf()



Mutable Collection

It supports both read and write functionalities. Mutable collections and their corresponding methods are:

- List – mutableListOf(), arrayListOf() and ArrayList
- Set – mutableSetOf(), hashSetOf()
- Map – mutableMapOf(), hashMapOf() and HashMap

File Edit View Navigate Code Refactor Build Run Tools VCS Window Help KotlinBasics - HelloCollections.kt [KotlinBasics.app.main]

KotlinBasics > app > src > main > java > com > mastercoding > kotlinbasics > HelloCollections.kt main

HelloCollectionsKt Nexus 6P API 33

Resource Manager

Project

Bookmarks

Build Variants

Structure

Version Control Run Debug Profiler Logcat App Quality Insights Build TODO Problems Terminal Services App Inspection

Gradle

Device Manager

Notifications

Running Devices

Device Explorer

Layout Inspector

Gradle build finished in 1 s 542 ms (2 minutes ago)

26:25 CRLF UTF-8 4 spaces

12:22 PM 11/5/2023

```
3 ► fun main(){
4     /*
5         List: Ordered collection in which we can access elements
6             by using indices that define a position for each
7             element
8     */
9
10    // Immutable: read-only list
11    val fruits = listOf("Apple", "Banana", "Cherry")
12
13    // Accessing elements of Immutable list
14    for (item in fruits){
15        println(item)
16    }
17
18    // Mutable List: Supports both read and write functionalities
19    val vegetables = mutableListOf("Potato", "Tomato", "Broccoli")
20    val colors = arrayListOf("Red", "Green", "Blue")
```

The Complete Android Developer Course

Mutable & Immutable Lists

The Complete Android Developer Course

```
// File Edit View Navigate Code Refactor Build Run Tools VCS Window Help KotlinBasics - HelloCollections.kt [KotlinBasics.app.main]
KotlinBasics > app > src > main > java > com > mastercoding > kotlinbasics > HelloCollections.kt main
HelloCollections.kt ▾ Nexus 6P API 33 ▾ ... ▾ Gradle ▾ Device Manager
Resource Manager
HelloCollections.kt ×
18 // Mutable List: Supports both read and write functionalities
19 val vegetables = mutableListOf("Potato", "Tomato", "Broccoli")
20 val colors = arrayListOf("Red", "Green", "Blue")
21
22 // Adding Elements
23 vegetables.add("Carrots")
24 colors.add("Purple")
25
26 // Removing Elements
27 vegetables.removeAt(index: 0)
28 colors.removeAt(index: 0)
29
30 // Updating Elements
31 vegetables[1] = "Garlic"
32 colors[1] = "Yellow"
33
34 // Printing all elements of a list
35 for (item in vegetables){
36     println(item)
37 }
38
39 for (item in colors){
40     println(item)
41 }
```

Project

Bookmarks

Build Variants

Structure

Version Control

Run

Debug

Profiler

Logcat

App Quality Insights

Build

TODO

Problems

Terminal

Services

App Inspection

Layout Inspector

Gradle build finished in 1 s 542 ms (3 minutes ago)

26:25 CRLF UTF-8 4 spaces

12:23 PM 11/5/2023

File Edit View Navigate Code Refactor Build Run Tools VCS Window Help KotlinBasics - HelloCollections.kt [KotlinBasics.app.main]

KotlinBasics > app > src > main > java > com > mastercoding > kotlinbasics > HelloCollections.kt main

HelloCollectionsKt Nexus 6P API 33

Resource Manager

Project

Bookmarks

Build Variants

Structure

Version Control Run Debug Profiler Logcat App Quality Insights Build TODO Problems Terminal Services App Inspection

Gradle

Device Manager

Notifications

Running Devices

Device Explorer

Layout Inspector

3 ► fun main(){

4 /*

5 * Set: It is a collection of unordered unique elements

6 * (No Duplicate Elements)

7 */

8

9 // Immutable Sets

10 val colors = setOf("Red", "Green", "Blue")

11

12 // Iterating over set elements

13 for (item in colors){

14 println(item)

15 }

16

17 // Mutable Set: supports both read and write functionalities

18 val fruits = mutableSetOf("Apple", "Banana", "Cherry")

The Complete Android Developer Course

Mutable & Immutable Sets

Gradle build finished in 1 s 785 ms (5 minutes ago)

34:1 CRLF UTF-8 4 spaces

12:58 PM 11/5/2023

The Complete Android Developer Course

```
// Mutable Set: supports both read and write functionalities
val fruits = mutableSetOf("Apple", "Banana", "Cherry")

// Adding elements to a set
fruits.add("melon")
fruits.add("Apple")

// Iterating over set elements
for (item in fruits){
    println(item)
}

// Removing elements from a set
fruits.remove(element: "melon")

// Updating elements in a set
fruits.remove(element: "Banana")
fruits.add("Orange")
```

Font size: 36pt Reset to 13pt

File Edit View Navigate Code Refactor Build Run Tools VCS Window Help KotlinBasics - HelloCollections.kt [KotlinBasics.app.main]

KotlinBasics > app > src > main > java > com > mastercoding > kotlinbasics > HelloCollections.kt main

Gradle

Device Manager

Notifications

Running Devices

Device Explorer

Version Control Run Debug Profiler Logcat App Quality Insights Build TODO Problems Terminal Services App Inspection

Layout Inspector

Gradle build finished in 1 s 785 ms (6 minutes ago)

34:1 CRLF UTF-8 4 spaces

12:59 PM 11/5/2023

File Edit View Navigate Code Refactor Build Run Tools VCS Window Help KotlinBasics - HelloCollections.kt [KotlinBasics.app.main]

KotlinBasics > app > src > main > java > com > mastercoding > kotlinbasics > HelloCollections.kt main

HelloCollectionsKt Nexus 6P API 33

Resource Manager

Project

Bookmarks

Build Variants

Structure

Version Control Run Debug Profiler Logcat App Quality Insights Build TODO Problems Terminal Services App Inspection

Gradle

Device Manager

Notifications

Running Devices

Device Explorer

Layout Inspector

Gradle build finished in 2 s 193 ms (6 minutes ago)

6 * Map: is an object that maps keys to values
7 * Map keys are unique
8 * Map values can be duplicates
9 * */
10 // Immutable Maps
11 val fruits = mapOf("apple" to 5, "banana" to 8, "cherry" to 12)
12
13 // Accessing Map elements
14 val quantityOfBananas = fruits["banana"]
15 println("The quantity of bananas: "+quantityOfBananas)
16
17 // Mutable Map
18 val vegetablesPrice = mutableMapOf("potato" to 1.5,
19 "tomato" to 3.5, "broccoli" to 6.0)
20
21 // Updating the price of tomato
22 vegetablesPrice["tomato"] = 4.0
23
24 // Adding a new element to the map
25 vegetablesPrice.put("garlic" , 6.0)
26
27 // printing all map elements
28 for ((key,value) in vegetablesPrice){
29 println("Vegetable: \$key , Price: \$value")
30

The Complete Android Developer Course

Mutable & Immutable Maps

2:17 PM 11/5/2023

Kotlin Literals

In programming, a literal is a constant value assigned to a variable.

```
val x = 10
```

```
var double1 = 13.8
```

```
val str = "Welcome to Our Course"
```

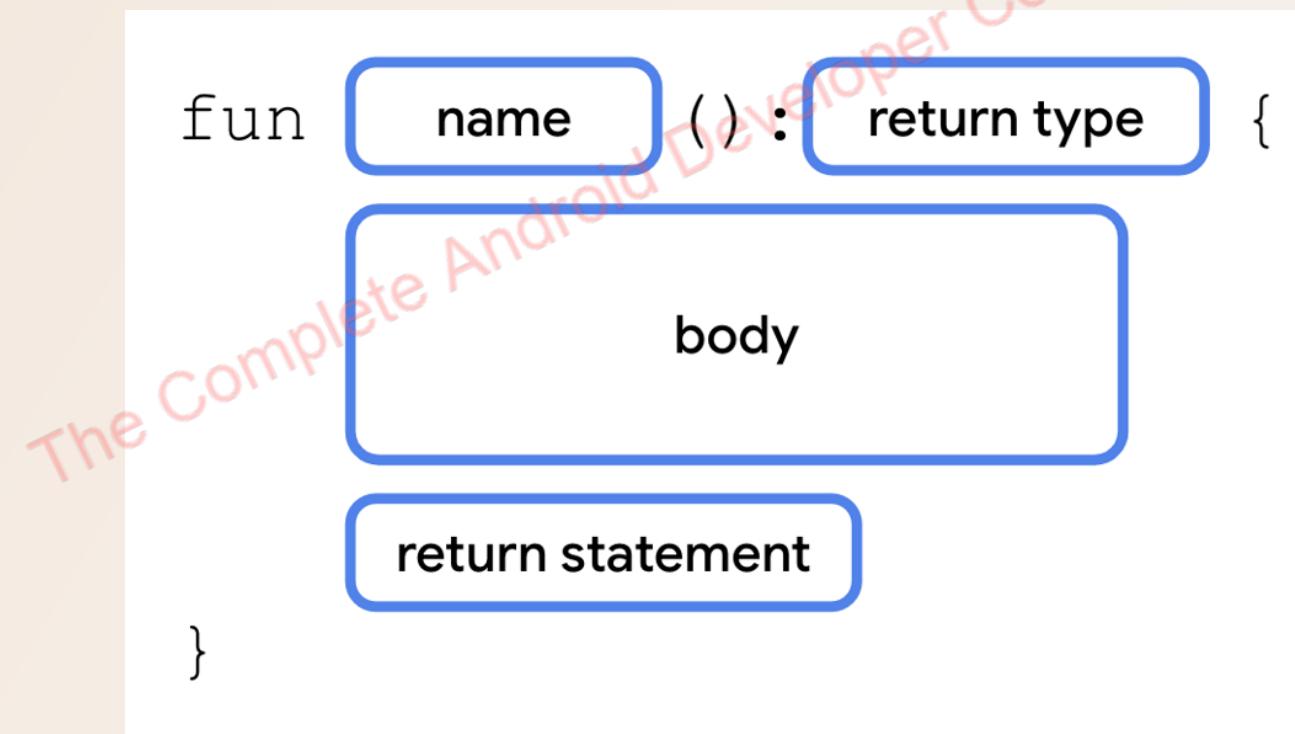
The Complete Android Developer Course

Function Literals

When we assign a function to a variable, it becomes a function literal.

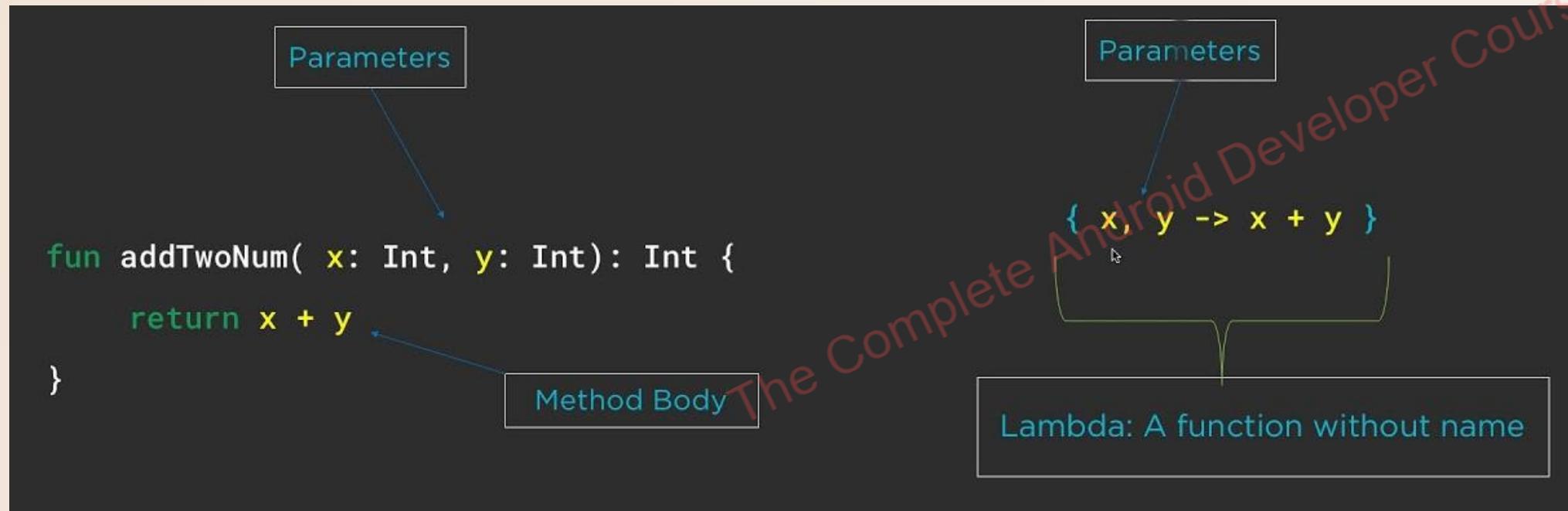
Kotlin provides 2 types of function literals. They are:

- Lambda Expression
- Anonymous Function



Lambda Expression

Lambda expression is a short way to create a function. To define a lambda expression, enclose the method body inside braces.



Lambda Expression

There are two ways to execute the lambda:

- Add parentheses after the variable name
- Call invoke() method

```
val printWelcome = { println("Welcome to our Course") }  
printWelcome()  
printWelcome.invoke()
```

Syntax of a Lambda Expression

```
val lambda: (DataType1, DataType2) → ReturnType = { variable1: DataType1, variable2: DataType2 → methodBody }
```

**Variable
Name**

**Function
Type**

Lambda

```
val add : (Int, Int) -> Int = {a: Int, b: Int -> a + b}
```

Shorter Syntax

```
val lambda: (DataType1, DataType2) → ReturnType = { variable1: DataType1, variable2: DataType2 → methodBody }
```

Variable
Name

Function
Type

Lambda

```
val add : (Int, Int) -> Int = {a: Int, b: Int -> a+ b}
```

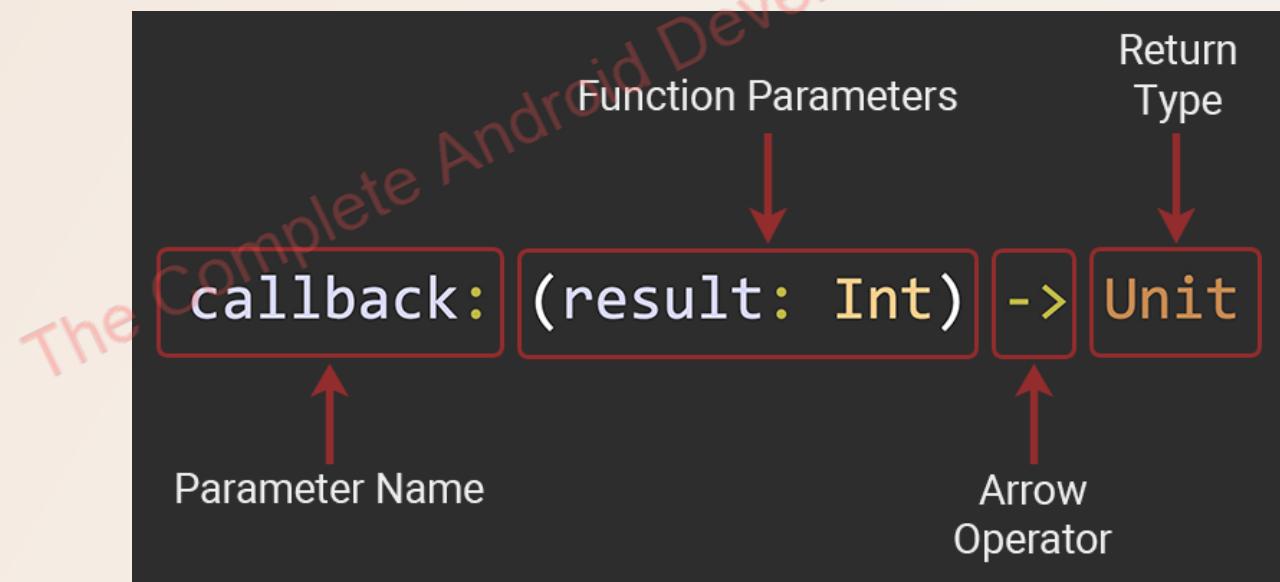
```
val add = {a: Int, b: Int -> a+ b}
```

```
val add: (Int, Int) -> Int = {a, b -> a+b }
```

Lambda Expression Types

There are 4 types depending on the parameters and return type:

- With Parameters and No Return Value
- With Parameters and Return Value
- No Parameters and No Return Value
- No Parameters and Return Value



File Edit View Navigate Code Refactor Build Run Tools VCS Window Help KotlinBasics - HelloCollections.kt [KotlinBasics.app.main]

KotlinBasics > app > src > main > java > com > mastercoding > kotlinbasics > HelloCollections.kt main

HelloCollectionsKt Nexus 6P API 33

Resource Manager

Project

Bookmarks

Build Variants

Structure

Version Control Profiler Logcat App Quality Insights TODO Problems Terminal Services App Inspection Layout Inspector

13 // 1- With Parameters and Return Type
14 val add: (Int, Int) -> Int = {a,b -> a+b}
15 val result = add(5,3)
16 println(result)
17
18 // 2- With Parameters & No Return Value
19 val add2 : (Int, Int) -> Unit = {a,b -> println(a+b) }
20 add2(5,5)
21
22 // 3- No Parameters But with Return Value
23 val add3: () -> String = {"Welcome to our course"}
24 println(add3.invoke())
25
26 // 4- No Parameters & no Return Value
27 val add4: () -> Unit = {println("No params, no return value")}
28 add4.invoke()
29
30 // Direct use of lambda expressions
31 println({a:Int, b:Int -> a * b} (4,5))

The Complete Android Developer Course

Notifications

Running Devices

Device Explorer

Anonymous Function

It is a function without a name.

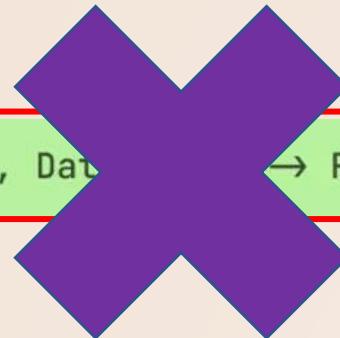
```
val variableName: (DataType1, DataType2) → ReturnType = fun(parameter1, parameter2): ReturnType { methodBody }
```

```
val add: (Int, Int) -> Int = fun(a, b): Int { return a + b }
```

Shorter Syntax Anonymous Function

It is a function without a name.

val variableName: (DataType1, Dat → ReturnType = fun(parameter1, parameter2): ReturnType { methodBody }



val variableName = fun(DataType1, DataType2): ReturnType { methodBody }

File Edit View Navigate Code Refactor Build Run Tools VCS Window Help KotlinBasics - HelloCollections.kt [KotlinBasics.app.main]

KotlinBasics > app > src > main > java > com > mastercoding > kotlinbasics > HelloCollections.kt main

HelloCollectionsKt Nexus 6P API 33

Resource Manager

Project

Bookmarks

Build Variants

Structure

Version Control Run Profiler Logcat App Quality Insights Build TODO Problems Terminal Services App Inspection

Gradle Device Manager

Notifications Running Devices Device Explorer

Anonymous Functions

```
3 > fun main(){
4     /*
5         Anonymous Function: a function without a name
6
7         val anonymousFunction = fun(parameters): ReturnType{
8             Function Body
9             Return Statement if needed
10            }
11        */
12
13    val numbers = listOf(1,2,3,4,5)
14
15    // An anonymous Function that squares a number
16    val squareAnonymousFunction = fun(x:Int): Int{
17        return x * x
18    }
19
20    val squaredNumbers = numbers.map(squareAnonymousFunction)
21    println(squaredNumbers) // [1 , 4, 9, 16, 25]
```

The Complete Android Developer Course

Gradle build finished in 2 s 540 ms (4 minutes ago)

Layout Inspector

14:1 CRLF UTF-8 4 spaces

12:38 PM 11/6/2023

File Edit View Navigate Code Refactor Build Run Tools VCS Window Help KotlinBasics - HelloCollections.kt [KotlinBasics.app.main]

KotlinBasics > app > src > main > java > com > mastercoding > kotlinbasics > HelloCollections.kt main

HelloCollectionsKt Nexus 6P API 33

Resource Manager

Project

Structure

Build Variants

Bookmarks

Gradle

Device Manager

Anonymous Functions

The Complete Android Developer Course

```
// Types of Anonymous Functions

// With Parameters and Return Value
val multiply = fun (a:Int, b:Int): Int{return a*b}
println(multiply.invoke(6,6))

// With Parameters and No Return Value
val multiply2 = fun(a:Int, b:Int){ println(""+a*b) }
multiply2(4,5)

// No Parameters and Return Value
val msg = fun():String{return "Welcome to our course!"}
println(msg())

// No Parameters and No Return Value
val msg2 = fun() : Unit{ println("Welcome Again!") }
msg2()
```

Version Control Run Profiler Logcat App Quality Insights Build TODO Problems Terminal Services App Inspection

Layout Inspector

Gradle build finished in 2 s 540 ms (5 minutes ago)

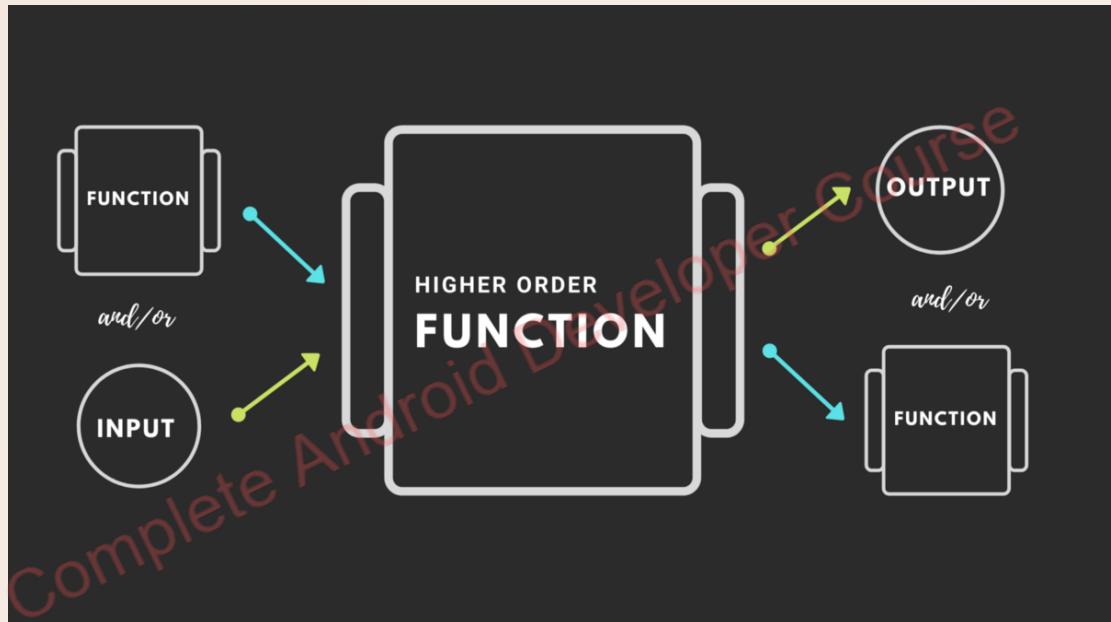
14:1 CRLF UTF-8 4 spaces

12:39 PM 11/6/2023

Higher Order Function

It is a function that takes a function as a parameter or returns a function or both.

In general, lambda expressions are passed as an argument to a higher-order function or returned from it. We can also use anonymous functions for the same.



File Edit View Navigate Code Refactor Build Run Tools VCS Window Help KotlinBasics - HelloCollections.kt [KotlinBasics.app.main]

KotlinBasics > app > src > main > java > com > mastercoding > kotlinbasics > HelloCollections.kt main

HelloCollectionsKt Nexus 6P API 33

Resource Manager

Project

Bookmarks

Build Variants

Structure

Higher Order Functions

The Complete Android Developer Course

```
3 ► ┌ fun main(){
4   /*
5     Higher-Order Functions: accept one or more functions as parameters
6     can return a function as its result
7   */
8
9   val addResult = operateOnNumbers(a: 5, b: 3){x, y -> x+y}
10  val multiplyResult = operateOnNumbers(a: 5, b: 3){x, y -> x*y}
11
12  println("Addition Result: $addResult")
13  println("Multiplication Result: $multiplyResult")
14
15 }
16 // Higher-Order Function
17 fun operateOnNumbers(a:Int, b:Int, operation:(Int, Int)-> Int):Int{
18   return operation(a, b)
19 }
```

Version Control Run Profiler Logcat App Quality Insights Build TODO Problems Terminal Services App Inspection

Gradle build finished in 2 s 335 ms (3 minutes ago)

Layout Inspector

8:1 CRLF UTF-8 4 spaces

12:51 PM 11/6/2023

The Complete Android Developer Course

"it" Keyword

```
5     "it" keyword: implicit name for a single parameter in a lambda expression
6     or anonymous function when that lambda or fun takes only one parameter
7 */
8
9 val numbers = listOf(1,2,3,4,5)
10
11 // Using Lambda Expressions to square every number
12 val squaredNumbs = numbers.map { x:Int -> x*x }
13 println(squaredNumbs)
14
15 // Using an anonymous function to square every number
16 val squaredNumbsAnonymous = fun(x:Int):Int{
17     return x * x
18 }
19 println(numbers.map(squaredNumbsAnonymous))
20
21 // Using "it" keyword
22 val squaredNumbers2 = numbers.map{it * it}
23 println(squaredNumbers2)
24
25 // Getting the even numbers from the list
26 val evenNumbers = numbers.filter { it %2 ==0 }
27 println(evenNumbers)
```

File Edit View Navigate Code Refactor Build Run Tools VCS Window Help KotlinBasics - HelloCollections.kt [KotlinBasics.app.main]

KotlinBasics > app > src > main > java > com > mastercoding > kotlinbasics > HelloCollections.kt main

Resource Manager

Project

Bookmarks

Build Variants

Structure

Version Control Run Profiler Logcat App Quality Insights Build TODO Problems Terminal Services App Inspection

Gradle

Device Manager

Notifications

Running Devices

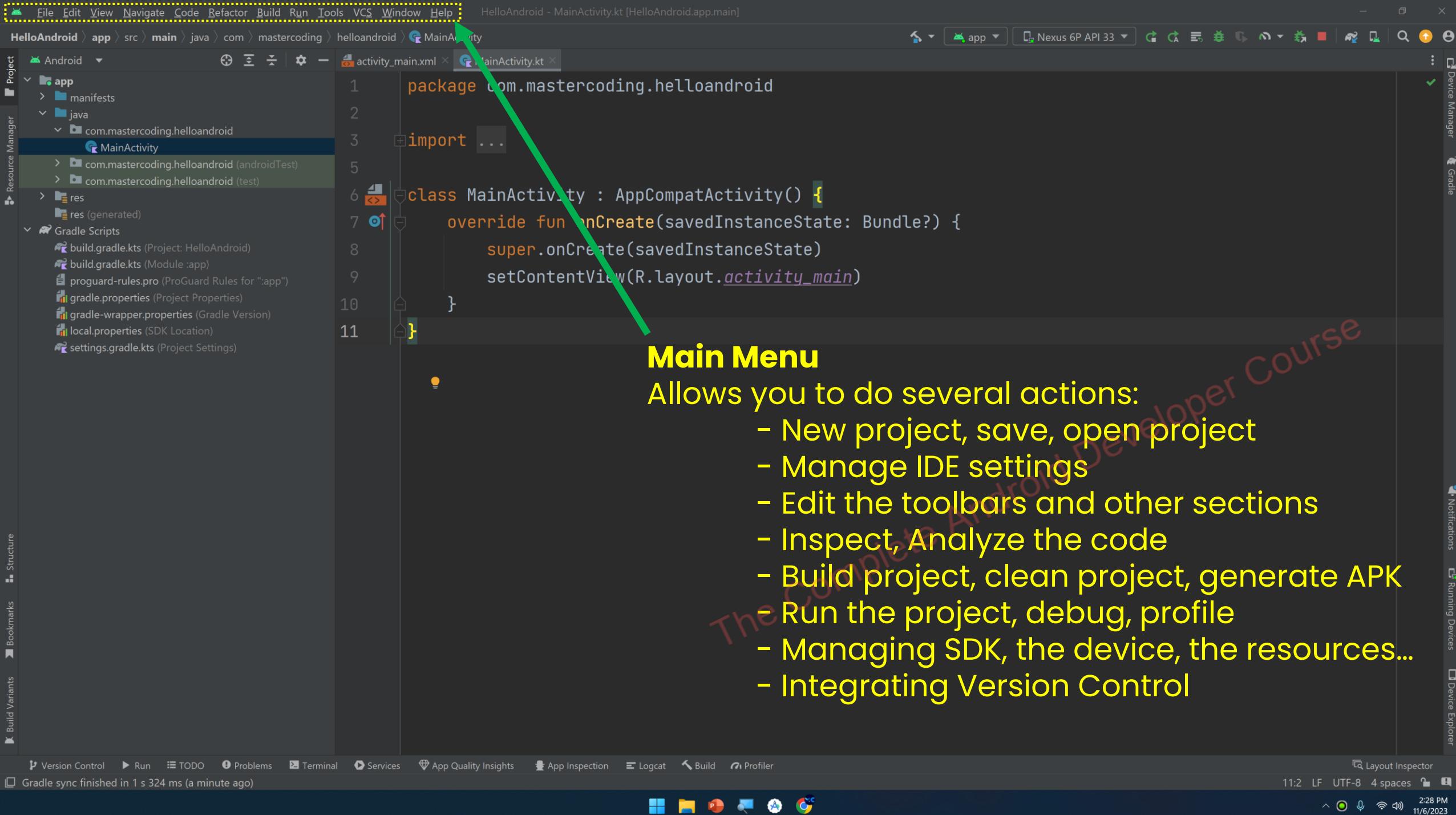
Device Explorer

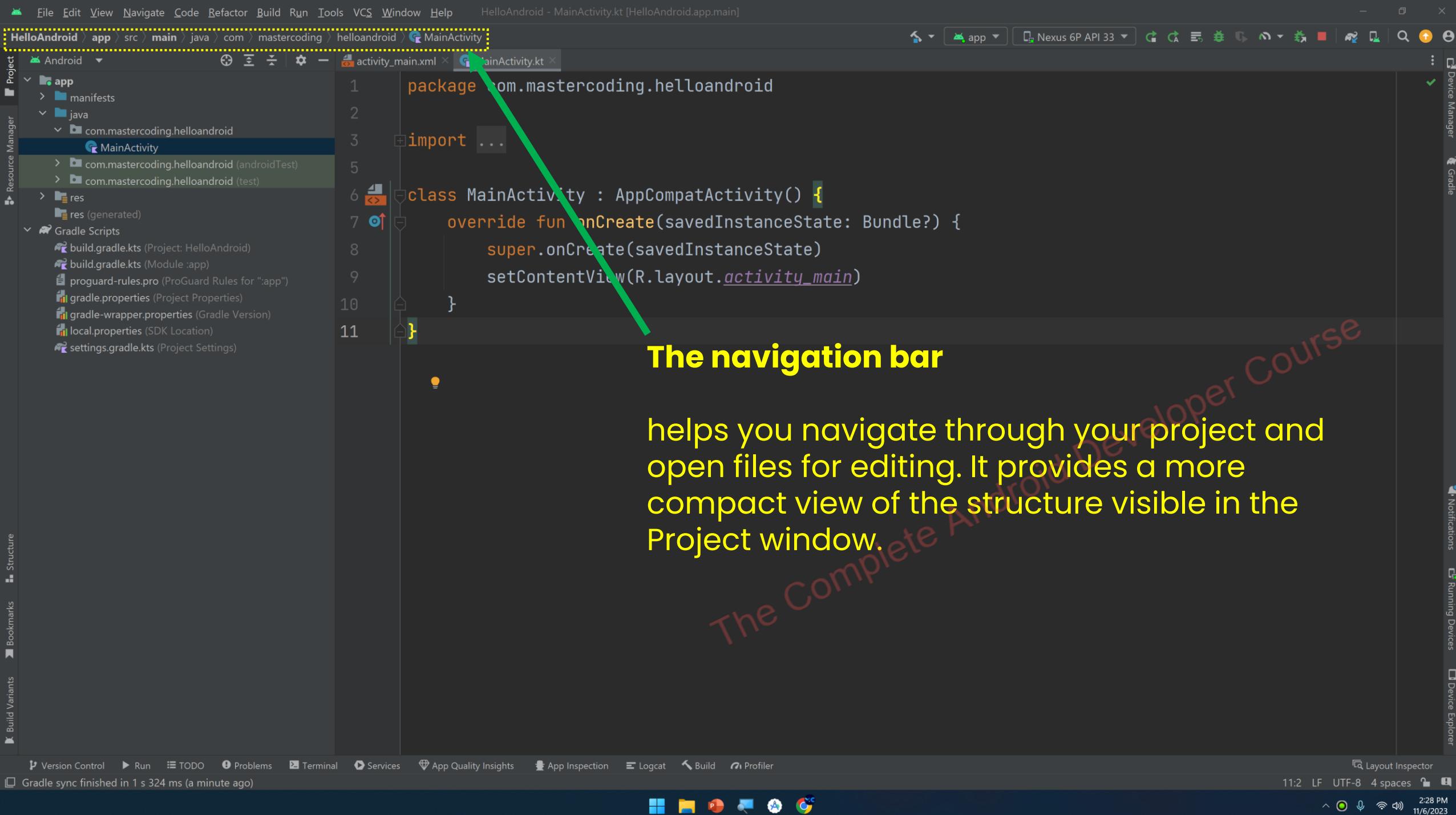
Layout Inspector

Gradle build finished in 826 ms (2 minutes ago)

28:5 CRU Bandicam 8 4 spaces

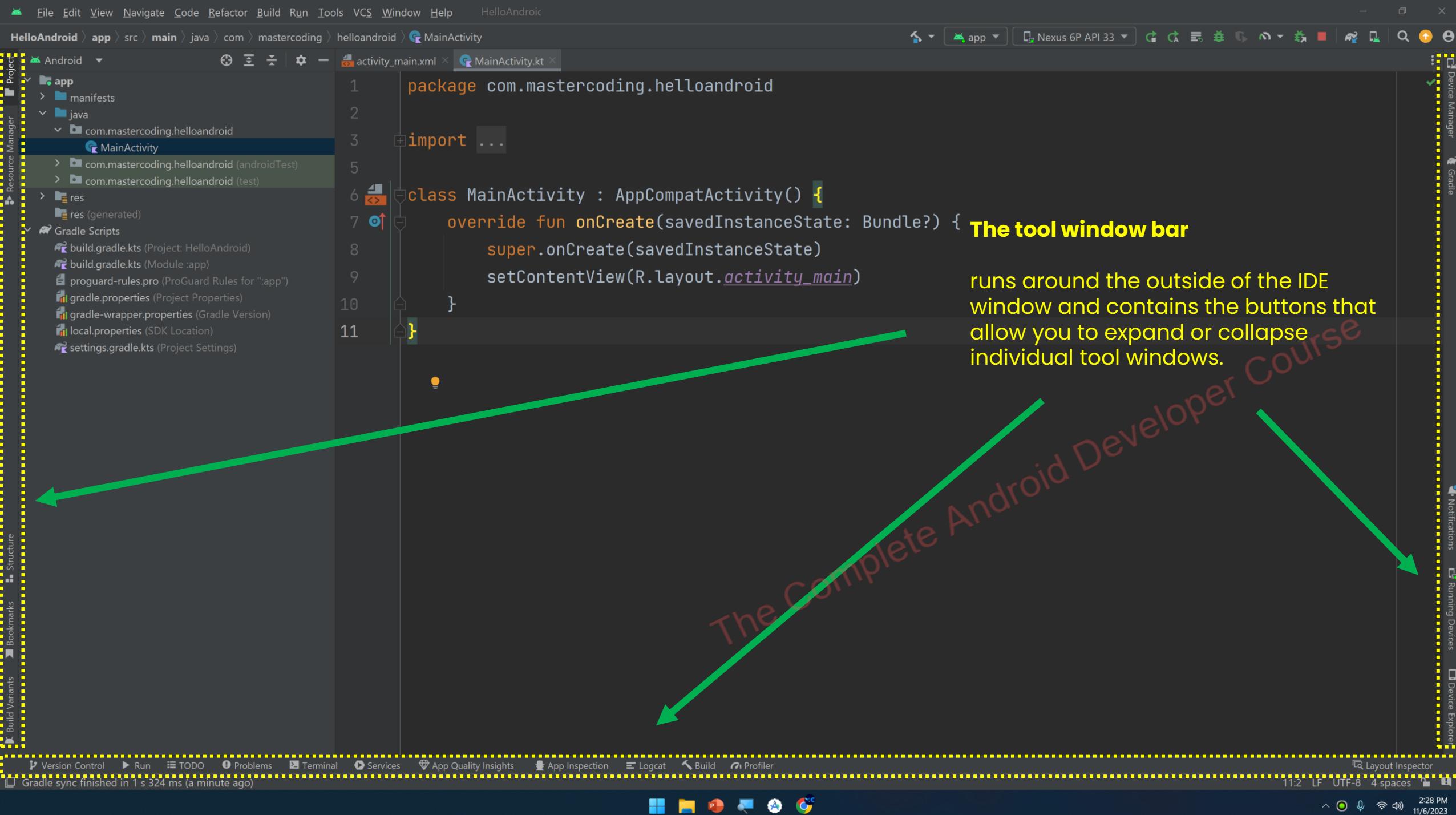
1:23 PM 11/6/2023





The navigation bar

helps you navigate through your project and open files for editing. It provides a more compact view of the structure visible in the Project window.



File Edit View Navigate Code Refactor Build Run Tools VCS Window Help HelloWorld

HelloWorld > app > src > main > java > com > mastercoding > helloworld > MainActivity

activity_main.xml MainActivity.java

Android app manifest Java (generated) res (generated) Gradle Scripts

```
1 package com.mastercoding.helloworld;
2
3 import ...
4
5
6
7 public class MainActivity extends AppCompatActivity {
8
9     @Override
10    protected void onCreate(Bundle savedInstanceState) {
11        super.onCreate(savedInstanceState);
12        setContentView(R.layout.activity_main);
13    }
14
15 }
```

The status bar displays the status of your project and the IDE itself, as well as any warnings or messages.

Version Control TODO Problems Terminal Logcat Profiler App Inspection Event Log Layout Inspector

* daemon started successfully (13 minutes ago)

15:1 LF UTF-8 4 spaces

8:21 AM 9/17/2022

The Complete Android Development Course

File Edit View Navigate Code Refactor Build Run Tools VCS Window Help HelloWorld

HelloWorld > app > src > main > java > com > mastercoding > helloworld > MainActivity

activity_main.xml MainActivity.java

```
1 package com.mastercoding.helloworld;
2
3 import ...
4
5
6
7 public class MainActivity extends AppCompatActivity {
8
9     @Override
10    protected void onCreate(Bundle savedInstanceState) {
11        super.onCreate(savedInstanceState);
12        setContentView(R.layout.activity_main);
13    }
14
15 }
```

Project Resource Manager

Gradle Device Manager

Version Control TODO Problems Terminal Logcat Profiler App Inspection Event Log Layout Inspector

* daemon started successfully (13 minutes ago)

15:1 LF UTF-8 4 spaces 8:21 AM 9/17/2022

Searching for specific action

At any time, you can search across your source code, databases, actions, elements of the user interface, and so on, by double-pressing the Shift key, or clicking the magnifying glass in the upper right-hand corner of the Android Studio window.

This can be very useful if, for example, you are trying to locate a particular IDE action that you have forgotten how to trigger.

The screenshot shows the Android Studio interface with the following details:

- File Bar:** File, Edit, View, Navigate, Code, Refactor, Build, Run, Tools, VCS, Window, Help, HelloWorld.
- Project Bar:** HelloWorld > app > src > main > java > com > mastercoding > helloworld > MainActivity
- Toolbars:** Includes icons for Version Control, TODO, Problems, Terminal, Logcat, Profiler, App Inspection, Event Log, Layout Inspector, and others.
- Code Editor:** Displays the `MainActivity.java` file content:

```
1 package com.mastercoding.helloworld;
2
3 import ...
4
5 public class MainActivity extends AppCompatActivity {
6
7     @Override
8     protected void onCreate(Bundle savedInstanceState) {
9         super.onCreate(savedInstanceState);
10        setContentView(R.layout.activity_main);
11    }
12
13 }
14
15 }
```
- Run Tab:** Shows "app" and "Nexus 6 API 33".
- Bottom Status Bar:** Shows "daemon started successfully (13 minutes ago)", "15:1 LF UTF-8 4 spaces", and system icons.
- Right Sidebar:** Contains sections for Gradle, Device Manager, Device File Explorer, and Emulator.
- Watermark:** A large watermark reading "The Complete Android Developer Course" is diagonally across the center of the screen.

Android Studio Editor

Appears when a Java, Kotlin, XML or other based file is selected for editing.

The screenshot shows the Android Studio interface with the following details:

- File Bar:** File, Edit, View, Navigate, Code, Refactor, Build, Run, Tools, VCS, Window, Help, HelloWorld.
- Project Bar:** HelloWorld > app > src > main > java > com > mastercoding > helloworld > MainActivity
- Toolbars:** Includes icons for Device Manager, Gradle, and other development tools.
- Code Editor:** Displays the Java code for `MainActivity.java`. The code is as follows:

```
package com.mastercoding.helloworld;
import ...;

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
}
```
- Document Tabs:** A blue arrow points from the text "Document Tabs" to the tabs for `activity_main.xml` and `MainActivity.java` at the top of the code editor.
- Bottom Navigation:** Includes tabs for Version Control, TODO, Problems, Terminal, Logcat, Profiler, App Inspection, Event Log, Layout Inspector, and a status bar showing the time (8:21 AM), date (9/17/2022), and battery level.
- Side Panels:** Resource Manager, Favorites, Structure, and Build Variants.

Document Tabs

Appears when a Java, Kotlin, XML or other based file is selected for editing.

The Complete Android Developer Course

The image shows the Android Studio interface with the code editor open. The gutter area on the left side of the editor contains yellow square icons with numbers from 1 to 15, each accompanied by a small icon. A large red arrow points from the text "The Complete Android Developer Course" to one of these icons. The code editor shows the MainActivity.java file with the following code:

```
package com.mastercoding.helloworld;
import ...
public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
}
```

The Editor Gutter Area

Used by the editor to display informational icons and controls

File Edit View Navigate Code Refactor Build Run Tools VCS Window Help HelloWorld

HelloWorld > app > src > main > java > com > mastercoding > helloworld > MainActivity

Android app Nexus 6 API 33

Project Resource Manager Favorites Structure Build Variants

Version Control TODO Problems Terminal Logcat Profiler App Inspection Event Log Layout Inspector

* daemon started successfully (13 minutes ago)

8:21 AM 9/17/2022

The screenshot shows the Android Studio interface with the following details:

- File Bar:** File, Edit, View, Navigate, Code, Refactor, Build, Run, Tools, VCS, Window, Help, HelloWorld.
- Project Bar:** HelloWorld > app > src > main > java > com > mastercoding > helloworld > MainActivity
- Toolbars:** Includes icons for Version Control, TODO, Problems, Terminal, Logcat, Profiler, App Inspection, Event Log, Layout Inspector, and a Daemon status message.
- Code Editor:** The file `MainActivity.java` is open. A blue arrow points from the title bar to the cursor position in the code editor, which is currently at line 10, column 16, indicating the current position relative to the code structure.
- Code Content:**

```
1 package com.mastercoding.helloworld;
2
3 import ...
4
5
6 public class MainActivity extends AppCompatActivity {
7
8     @Override
9     protected void onCreate(Bundle savedInstanceState) {
10        super.onCreate(savedInstanceState);
11        setContentView(R.layout.activity_main);
12    }
13
14 }
15
```
- Right Panel:** Shows the `Code Structure Location` feature, which displays the current position of the cursor as it relates to the overall structure of the code.
- Bottom Status Bar:** Shows the time (8:21 AM), date (9/17/2022), and system status icons.

Code Structure Location

Displays the current position of the cursor as it relates to the overall structure of the code

The Complete Android Developer Course

* daemon started successfully (13 minutes ago)

15:1 LF UTF-8 4 spaces

8:21 AM 9/17/2022

The screenshot shows the Android Studio interface with the following details:

- Top Bar:** File, Edit, View, Navigate, Code, Refactor, Build, Run, Tools, VCS, Window, Help, HelloWorld.
- Title Bar:** HelloWorld > app > src > main > java > com > mastercoding > helloworld > MainActivity
- Toolbar:** Includes icons for Run, Stop, Build, Clean, Sync, Find, Replace, and others.
- Project Navigators:** Project (left), Resource Manager, Favorites, Structure, Build Variants.
- Editor Area:** Displays the code for `MainActivity.java`.

```
1 package com.mastercoding.helloworld;
2
3 import ...
4
5 public class MainActivity extends AppCompatActivity {
6
7     @Override
8     protected void onCreate(Bundle savedInstanceState) {
9         super.onCreate(savedInstanceState);
10        setContentView(R.layout.activity_main);
11    }
12
13 }
14
15
```
- Bottom Bar:** Version Control, TODO, Problems, Terminal, Logcat, Profiler, App Inspection, Event Log, Layout Inspector.
- Bottom Status Bar:** daemon started successfully (13 minutes ago), 15:1, LF, UTF-8, 4 spaces, 8:21 AM, 9/17/2022.

The Editor Area

The main area where the code is displayed, entered and edited by the user.

The screenshot shows the Android Studio interface with the following details:

- Top Bar:** File, Edit, View, Navigate, Code, Refactor, Build, Run, Tools, VCS, Window, Help, HelloWorld.
- Project Bar:** HelloWorld > app > src > main > java > com > mastercoding > helloworld > MainActivity
- Editor:** activity_main.xml (Preview tab) and MainActivity.java. The Java code is as follows:

```
1 package com.mastercoding.helloworld;
2
3 import ...
4
5 public class MainActivity extends AppCompatActivity {
6
7     @Override
8     protected void onCreate(Bundle savedInstanceState) {
9         super.onCreate(savedInstanceState);
10        setContentView(R.layout.activity_main);
11    }
12
13 }
14
15 }
```
- Sidebar:** Shows validation markers (green checkmark, red circle with exclamation, yellow question mark) along the right edge of the code editor.
- Status Bar:** The status bar at the bottom right shows "The Complete Android Developer Course" watermark. It also includes icons for battery, signal, and time (8:21 AM, 9/17/2022).
- Bottom Bar:** Version Control, TODO, Problems, Terminal, Logcat, Profiler, App Inspection, Event Log, Layout Inspector.
- Bottom Left:** Daemon status: * daemon started successfully (13 minutes ago).

A large blue arrow points from the sidebar area towards the status bar area.

The Validation and Marker Sidebar

When you are typing code, the editor is analyzing the code to check for warnings and syntax errors.

The Complete Android Developer Course

This screenshot shows the Android Studio interface with the Layout Editor open. The project navigation bar at the top includes File, Edit, View, Navigate, Code, Refactor, Build, Run, Tools, VCS, Window, Help, and HelloWorld. Below the bar are tabs for activity_main.xml and MainActivity.java. The main area features a toolbar with icons for file operations, a palette of UI components, and two preview panes showing the layout on a Nexus 6 API 33 device. The left pane displays the text "Hello World!" and the right pane shows a solid teal background. A component tree on the left lists a ConstraintLayout containing a single TextView. The bottom navigation bar includes Version Control, TODO, Problems, Terminal, Logcat, Profiler, App Inspection, Event Log, and Layout Inspector. The bottom status bar shows daemon startup information and system status.

Layout Editor

With Android Studio Layout Editor, we can build layouts by dragging components onto the screen instead of writing the layout XML by hand.

Don't worry; you can still switch view and edit XML with Text editor if you want

File Edit View Navigate Code Refactor Build Run Tools VCS Window Help HelloWorld

activity_main.xml MainActivity.java

app src main res layout activity_main.xml

Android Project Resource Manager

app manifests java com.mastercoding.helloworld MainActivity com.mastercoding.helloworld (androidTest) com.mastercoding.helloworld (test) java (generated) res res (generated) Gradle Scripts

Palette

Common Text Buttons Widgets Layouts Containers Helpers Google Legacy

Component Tree

ConstraintLayout Ab TextView "Hello World!"

activity_main.xml Pixel 33 Default (en-us)

Code Split Design

Gradle Layout Validation Device Manager

Version Control TODO Problems Terminal Logcat Profiler App Inspection Event Log Layout Inspector

* daemon started successfully (3 minutes ago)

20:1 LF UTF-8 4 spaces

3:20 PM 9/17/2022

The Complete Android Developer Course

This screenshot shows the Android Studio interface with the Layout Editor open. The project navigation bar at the top includes File, Edit, View, Navigate, Code, Refactor, Build, Run, Tools, VCS, Window, Help, and HelloWorld. The main window displays the layout file 'activity_main.xml' and the Java file 'MainActivity.java'. The Layout Editor's toolbar includes icons for Undo, Redo, Copy, Paste, Delete, and various selection tools. The palette on the left lists components like TextView, Button, ImageView, RecyclerView, ScrollView, and Switch under categories such as Common, Text, Buttons, Widgets, Layouts, Containers, Helpers, Google, and Legacy. The preview pane shows a white screen with a single TextView containing the text "Hello World!". Below the preview are two smaller panes showing the device screen in landscape and portrait orientations. The bottom navigation bar contains links for Version Control, TODO, Problems, Terminal, Logcat, Profiler, App Inspection, Event Log, and Layout Inspector. The status bar at the bottom right shows the time as 3:20 PM, the date as 9/17/2022, and system information like battery level and signal strength.

File Edit View Navigate Code Refactor Build Run Tools VCS Window Help HelloWorld

Android Studio Project Resource Manager Favorites Build Variants

activity_main.xml MainActivity.java

app manifests java com.mastercoding.helloworld (androidTest) com.mastercoding.helloworld (test) java (generated) res res (generated) Gradle Scripts

With Android Studio Layout Editor, you can:

- Add components to the layout using drag and drop
- Add and change all the attributes (properties) for components
- View and Edit constraints for layouts
- Prepare a responsive design with a preview
- Edit XML code for layout

Common Text Buttons Widgets Layouts Containers Helpers Google Legacy

Ab TextView

- Button
- ImageView
- RecyclerView
- FragmentConta...
- ScrollView
- Switch

Component Tree

- ConstraintLayout
- Ab TextView "Hello World!"

Hello World!

Event Log Layout Inspector

Version Control TODO Problems Terminal Logcat Profiler App Inspection

* daemon started successfully (3 minutes ago)

20:1 LF UTF-8 4 spaces

3:20 PM 9/17/2022

The Complete Android Developer Course

File Edit View Navigate Code Refactor Build Run Tools VCS Window Help HelloWorld

HelloWorld > app > src > main > res > layout > activity_main.xml

activity_main.xml MainActivity.java

Android Project Resource Manager

app manifests java com.mastercoding.helloworld MainActivity com.mastercoding.helloworld (androidTest) com.mastercoding.helloworld (test) java (generated) res res (generated) Gradle Scripts

Palette

Provides a list of components and layouts that you can drag into your layout in the editor.

Here you find Buttons, TextViews, Spinners, ImageViews etc.

Common Text Buttons Widgets Layouts Containers Helpers Google Legacy

Component Tree ConstraintLayout Ab TextView "Hello World!"

activity_main.xml Pixel 33 HelloWorld Default (en-us)

Code Split Design Attributes Layout Validation Device Manager

Event Log Layout Inspector

Version Control TODO Problems Terminal Logcat Profiler App Inspection

* daemon started successfully (3 minutes ago)

20:1 LF UTF-8 4 spaces

3:20 PM 9/17/2022

The Complete Android Developer Course

File Edit View Navigate Code Refactor Build Run Tools VCS Window Help HelloWorld

HelloWorld > app > src > main > res > layout > activity_main.xml

Android app Nexus 6 API 33

Project Resource Manager

activity_main.xml MainActivity.java

Palette

Common Text Buttons Widgets Layouts Containers Helpers Google Legacy

activity_main.xml Pixel 33 HelloWorld Default (en-us)

Component Tree

ConstraintLayout

Ab TextView "Hello World!"

Event Log Layout Inspector

Version Control TODO Problems Terminal Logcat Profiler App Inspection

* daemon started successfully (3 minutes ago)

20:1 LF UTF-8 4 spaces

3:20 PM 9/17/2022

The Complete Android Developer Course

File Edit View Navigate Code Refactor Build Run Tools VCS Window Help HelloWorld

Project

Resource Manager

Toolbar

Provides buttons to configure your layout design in the editor and helps you change some layout attributes.

In addition, here you can change the size of the display to check if constraints work as expected.

Here you can also find alignments and connection tools.

Code Split Design Attributes Layout Validation Device Manager

Version Control TODO Problems Terminal Logcat Profiler App Inspection Event Log Layout Inspector

* daemon started successfully (3 minutes ago)

20:1 LF UTF-8 4 spaces

3:20 PM 9/17/2022

The Complete Android Developer Course

This screenshot shows the Android Studio interface with the following components visible:

- Top Bar:** File, Edit, View, Navigate, Code, Refactor, Build, Run, Tools, VCS, Window, Help, HelloWorld.
- Toolbar:** Includes icons for Run, Stop, Refresh, and various developer tools.
- Project Navigators:** Project, Resource Manager, Structure, Favorites, Build Variants.
- Central Area:**
 - Activity Layout Editor:** Displays a ConstraintLayout containing a single TextView with the text "Hello World!". A blue arrow points from the text "The Complete Android Developer Course" towards the layout editor.
 - Code Editor:** Shows the file `MainActivity.java`.
 - Palette:** Shows categories like Common, Text, Buttons, Widgets, Layouts, Containers, Helpers, Google, and Legacy, with `Ab TextView` selected.
 - Component Tree:** Shows the hierarchy: ConstraintLayout > Ab TextView "Hello World!".
- Right Side:** Includes tabs for Code, Split, and Design, with Design selected. Other tabs include Layout Validation and Device Manager.
- Bottom Bar:** Version Control, TODO, Problems, Terminal, Logcat, Profiler, App Inspection, Event Log, Layout Inspector.
- Status Bar:** Shows daemon started successfully (3 minutes ago), 20:1 LF, UTF-8, 4 spaces, and system icons.

The Complete Android Developer Course

File Edit View Navigate Code Refactor Build Run Tools VCS Window Help HelloWorld

HelloWorld > app > src > main > res > layout > activity_main.xml

Android Project Resource Manager

activity_main.xml MainActivity.java

Palette

Common Text Buttons Widgets Layouts Containers Helpers Google Legacy

activity_main.xml Pixel 33 Default (en-us)

Attributes

Attributes are properties for components in your layout.

Most of the time, attributes will describe visual aspects of components, but sometimes they are used to set layout behavior as well and set parameters for tools.

Component Tree

ConstraintLayout Ab TextView "Hello World!"

Event Log Layout Inspector

Version Control TODO Problems Terminal Logcat Profiler App Inspection

* daemon started successfully (3 minutes ago)

20:1 LF UTF-8 4 spaces

3:20 PM 9/17/2022

The Complete Android Developer Course

File Edit View Navigate Code Refactor Build Run Tools VCS Window Help HelloWorld

HelloWorld > app > src > main > res > layout > activity_main.xml

Android app Nexus 6 API 33

MainActivity.java

Project Resource Manager

Manifest

Every app project must have an `AndroidManifest.xml` file (with precisely that name) at the root of the project source set.

The manifest file describes essential information about your app to the Android build tools, the Android operating system, and Google Play.

Component Tree

ConstraintLayout

Ab TextView "Hello World!"

Code Split Design

Attributes

Layout Validation Device Manager

Version Control TODO Problems Terminal Logcat Profiler App Inspection Event Log Layout Inspector

* daemon started successfully (3 minutes ago)

20:1 LF UTF-8 4 spaces

3:20 PM 9/17/2022

The Complete Android Developer Course

This screenshot shows the Android Studio interface for a project named "HelloWorld". The top navigation bar includes File, Edit, View, Navigate, Code, Refactor, Build, Run, Tools, VCS, Window, and Help. The title bar shows the project name "HelloWorld". The left sidebar has sections for Project, Resource Manager, Java, Favorites, Structure, and Build Variants. The main area displays the "activity_main.xml" layout file, which contains a single TextView with the text "Hello World!". The Java code for MainActivity.java is also visible, showing the standard boilerplate. The bottom status bar shows the terminal output: "daemon started successfully (3 minutes ago)". The bottom right corner shows the system clock as 3:20 PM on 9/17/2022.

File Edit View Navigate Code Refactor Build Run Tools VCS Window Help HelloWorld

HelloWorld > app > src > main > res > layout > activity_main.xml

Android app Nexus 6 API 33

Project

Resource Manager

Java

Contains the Java source code files, separated by package names, including JUnit test code.

Component Tree

ConstraintLayout

Ab TextView "Hello World!"

activity_main.xml

MainActivity.java

Palette

Common

Text

Buttons

Widgets

Layouts

Containers

Helpers

Google

Legacy

Pixel 33 Default (en-us)

Code Split Design

Attributes

Layout Validation

Device Manager

Event Log Layout Inspector

Version Control TODO Problems Terminal Logcat Profiler App Inspection

* daemon started successfully (3 minutes ago)

20:1 LF UTF-8 4 spaces

3:20 PM 9/17/2022

The Complete Android Developer Course

The `res/values` folder is used to store the values for the resources that are used in many Android projects to include features of color, styles, dimensions etc.

File Edit View Navigate Code Refactor Build Run Tools VCS Window Help HelloWorld

HelloWorld > app > src > main > res > layout > activity_main.xml

Android app Nexus 6 API 33

Project Resource Manager

activity_main.xml MainActivity.java

Palette

Common Ab TextView

- Text
- Buttons
- Widgets
- Layouts
- Containers

Helpers Google Legacy

Component Tree

- ConstraintLayout
- Ab TextView "Hello World!"

Pixel 33 Default (en-us)

Code Split Design

Gradle Layout Validation Device Manager

Version Control TODO Problems Terminal Logcat Profiler App Inspection Event Log Layout Inspector

* daemon started successfully (3 minutes ago) 20:1 LF UTF-8 4 spaces

3:20 PM 9/17/2022

The Complete Android Developer Course

File Edit View Navigate Code Refactor Build Run Tools VCS Window Help HelloWorld

Project

Android app manifest Java com.mastercoding.helloworld MainActivity com.mastercoding.helloworld (androidTest) com.mastercoding.helloworld (test) java (generated) res drawable ic_launcher_background.xml ic_launcher_foreground.xml (v24) layout activity_main.xml mipmap ic_launcher (6) ic_launcher_round (6) values colors.xml strings.xml themes (2) xml backup_rules.xml data_extraction_rules.xml res (generated) Gradle Scripts

activity_main.xml MainActivity.java

Palette

Common Ab TextView
Text Button
Buttons ImageView
Widgets RecyclerView
Layouts FragmentConta...
Containers ScrollView
Helpers Switch
Google Legacy

Component Tree

ConstraintLayout
Ab TextView "Hello World!"

Hello World!

androidx.constraintlayout.widget.ConstraintLayout

Code Split Design

Gradle

Resource Manager

Structure

Favorites

Build Variants

Version Control

TODO

Problems

Terminal

Logcat

Profiler

App Inspection

Event Log

Layout Inspector

daemon started successfully (58 minutes ago)

22:39 LF UTF-8 4 spaces

4:14 PM 9/17/2022

The Complete Android Developer Course

File Edit View Navigate Code Refactor Build Run Tools VCS Window Help HelloWorld

HelloWorld

Android app Nexus 6 API 33

activity_main.xml MainActivity.java

Palette

Common Text Buttons Widgets Layouts Containers Helpers Google Legacy

activity_main.xml 0dp 33 Default (en-us)

Component Tree

ConstraintLayout Ab TextView "Hello World!"

layout

Normally we store every xml layout file inside the res/layout folder.

androidx.constraintlayout.widget.ConstraintLayout

Version Control TODO Problems Terminal Logcat Profiler App Inspection Event Log Layout Inspector

* daemon started successfully (58 minutes ago) 22:39 LF UTF-8 4 spaces

4:14 PM 9/17/2022

The Complete Android Developer Course

File Edit View Navigate Code Refactor Build Run Tools VCS Window Help HelloWorld

Project

Android app manifests java com.mastercoding.helloworld MainActivity com.mastercoding.helloworld (androidTest) com.mastercoding.helloworld (test) java (generated) res drawable ic_launcher_background.xml ic_launcher_foreground.xml (v24) layout activity_main.xml mipmap ic_launcher (6) ic_launcher_round (6) values colors.xml strings.xml themes (2) xml backup_rules.xml data_extraction_rules.xml res (generated) Gradle Scripts

activity_main.xml MainActivity.java

Palette

Common Text Buttons Widgets Layouts Containers Helpers Google Legacy

activity_main.xml Pixel 33 Hello World Default (en-us)

Component Tree

ConstraintLayout Ab TextView "Hello World!"

androidx.constraintlayout.widget.ConstraintLayout

Version Control TODO Problems Terminal Logcat Profiler App Inspection Event Log Layout Inspector

* daemon started successfully (58 minutes ago)

4:14 PM 9/17/2022

The Complete Android Developer Course

This screenshot shows the Android Studio interface for a project named "HelloWorld".

Project Structure:

- app: Contains manifests, java, res (drawable, layout, mipmap, values, xml), and generated files.
- java: Contains com.mastercoding.helloworld.MainActivity, com.mastercoding.helloworld (androidTest), and com.mastercoding.helloworld (test).
- res: Contains drawable (ic_launcher_background.xml, ic_launcher_foreground.xml), layout (activity_main.xml), mipmap (ic_launcher, ic_launcher_round), values (colors.xml, strings.xml), themes, and xml (backup_rules.xml, data_extraction_rules.xml).

Code Editor:

MainActivity.java:

```
package com.mastercoding.helloworld;
import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
}
```

activity_main.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <TextView
        android:id="@+id/textView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello World!"/>

```

Design View:

The design view shows a white ConstraintLayout containing a single TextView with the text "Hello World!". The background of the activity is a dark teal color. The "color" section of the palette is highlighted.

Color Palette:

colors.xml: The colors.xml is an XML file which is used to store the colors for the resources. An Android project contains 3 essential colors namely:

- colorPrimary
- colorPrimaryDark
- colorAccent

Bottom Navigation:

- Version Control, TODO, Problems, Terminal, Logcat, Profiler, App Inspection
- Event Log, Layout Inspector
- daemon started successfully (58 minutes ago)
- 22:39 LF UTF-8 4 spaces
- 4:14 PM 9/17/2022

The Complete Android Developer Course

File Edit View Navigate Code Refactor Build Run Tools VCS Window Help HelloWorld

Project

HelloWorld

Android app Nexus 6 API 33

activity_main.xml MainActivity.java

Palette

Common Text Buttons Widgets Layouts Containers Helpers Google Legacy

activity_main.xml Pixel 33 HelloWorld Default (en-us)

Code Split Design

Attributes

Layout Validation Device Manager

Component Tree

strings

One of the most important as well as widely used values file is the strings.xml due to its applicability in the Android project.

Basic function of the strings.xml is to define the strings in one file so that it is easy to use same string in different positions in the android project plus it makes the project looks less messy.

androidx.constraintlayout.widget.ConstraintLayout

Version Control TODO Problems Terminal Logcat Profiler App Inspection Event Log Layout Inspector

* daemon started successfully (58 minutes ago) 22:39 LF UTF-8 4 spaces

4:14 PM 9/17/2022

The Complete Android Developer Course

File Edit View Navigate Code Refactor Build Run Tools VCS Window Help HelloWorld

Project

Resource Manager

Themes

Styles and themes on Android allow you to separate the details of your app design from the UI structure and behavior, similar to stylesheets in web design.

A style is a collection of attributes that specify the appearance for a single View. A style can specify attributes such as font color, font size, background color, and much more.

A theme is a collection of attributes that's applied to an entire app, activity, or view hierarchy—not just an individual view. When you apply a theme, every view in the app or activity applies each of the theme's attributes that it supports.

Themes can also apply styles to non-view elements, such as the status bar and window background.

Version Control TODO Problems Terminal Logcat Profiler App Inspection

Event Log Layout Inspector

daemon started successfully (58 minutes ago)

4:14 PM 9/17/2022

The Complete Android Developer Course

File Edit View Navigate Code Refactor Build Run Tools VCS Window Help HelloWorld

HelloWorld

Android app Nexus 6 API 33

activity_main.xml MainActivity.java

Palette

Common Text Buttons Widgets Layouts Containers Helpers Google Legacy

0dp

Pixel 33 Default (en-us)

Code Split Design

Project

Resource Manager

Gradle

Layout Validation

Device Manager

Component Tree

Gradle

ConstraintLayout

Ab TextView "Hello World!"

The Android build system compiles app resources and source code, and packages them into APKs or Android App Bundles that you can test, deploy, sign, and distribute.

Android Studio uses Gradle, an advanced build toolkit, to automate and manage the build process, while allowing you to define flexible custom build configurations.

Each build configuration can define its own set of code and resources, while reusing the parts common to all versions of your app.

The Android plugin for Gradle works with the build toolkit to provide processes and configurable settings that are specific to building and testing Android applications.

Version Control TODO Problems Terminal Logcat Profiler App Inspection

Event Log Layout Inspector

* daemon started successfully (58 minutes ago)

22:39 LF UTF-8 4 spaces

4:14 PM 1 9/17/2022

The Views & ViewGroups

ViewGroup

A ViewGroup is a special subclass of the View class that serves as a container for multiple Views.

There are several useful widgets that extend the ViewGroup Class.

Most are actually subclasses of another ViewGroup known as a Layout.

Views

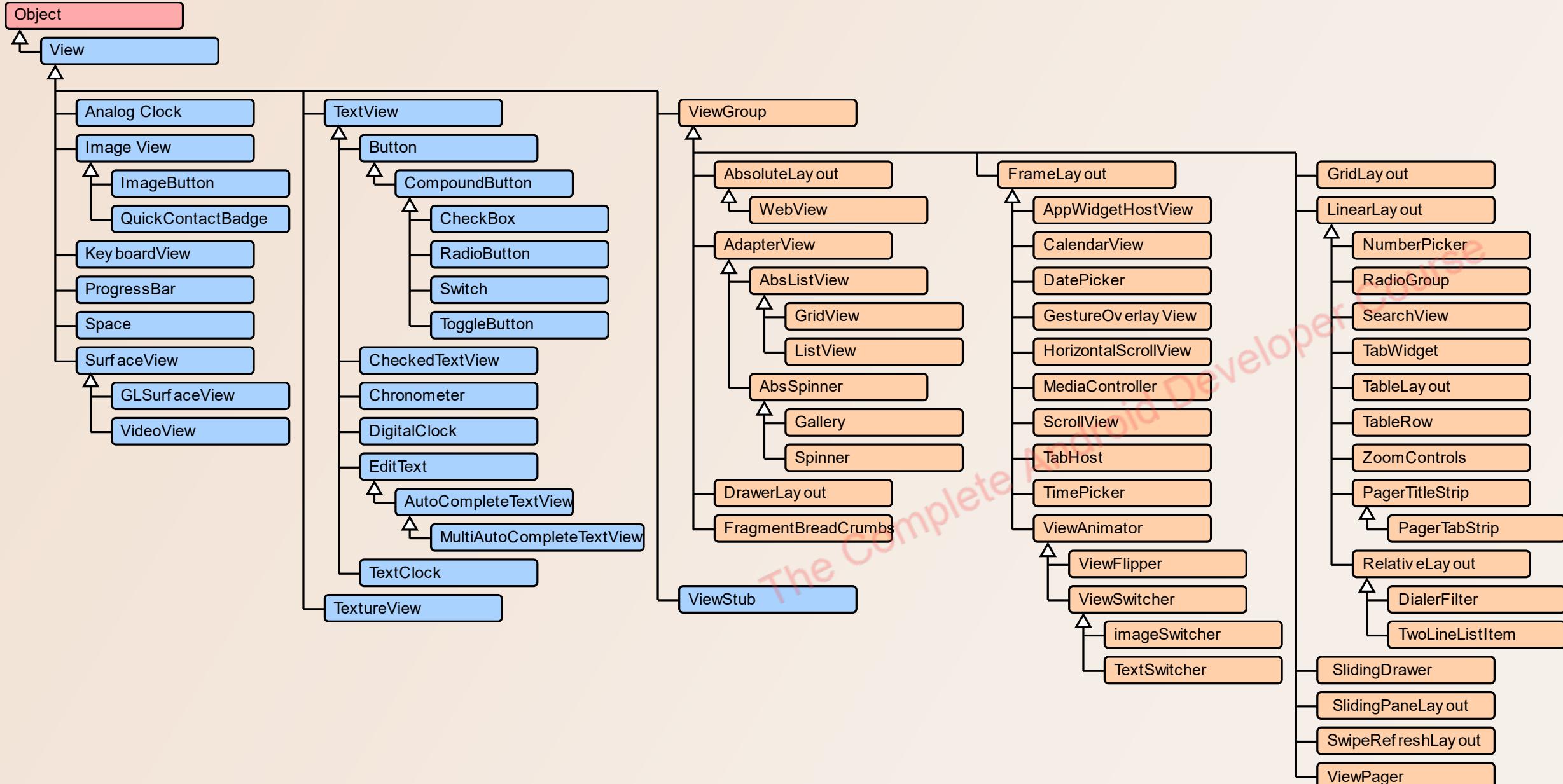
A View is a simple building block of a user interface.

It is a small rectangular box that can be TextView, EditText, or even a button.

It occupies the area on the screen in a rectangular area and is responsible for drawing and event handling.

The Complete Android Developer Course

The View Class



App Creation Steps

- 1- The Required Dependencies and Libraries
- 2- Adding Permissions
- 3- Designing the Layout
- 4- Adding the Functionalities and Logic
- 5- Running and Testing

The Complete Android Developer Course

Constraint Layout

It is a View Group which solves problems like nesting and performance.

This not only helps developers to build more complex and large UIs, but also it comes with a flat hierarchy.

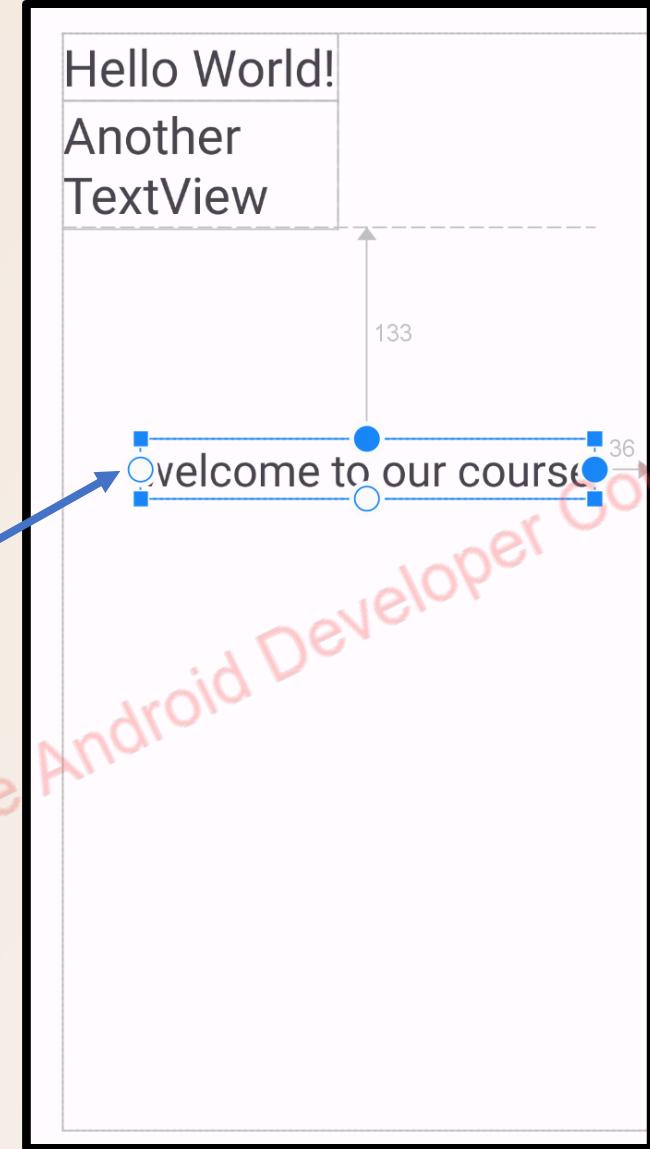
ConstraintLayout works based on the relationships between views and those are called as **Constraints**. The essential block of ConstraintLayout is developing constraints.



Constraint Layout

Views and widgets such as Buttons, Texts etc. are added inside the **<ConstraintLayout>** tag, which then becomes the parent to all the nested elements.

Anchor Point



Constraint Layout

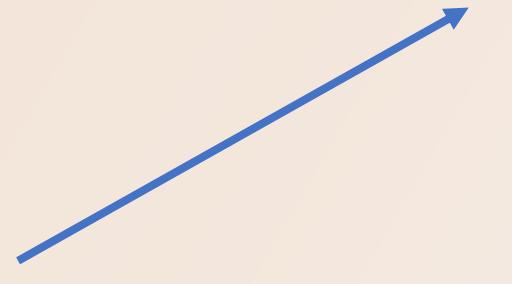
Constraints' Attributes in XML:

app:layout_constraint<**SOURCE_ANCHOR**>_to<**TARGET_ANCHOR**>of="**<TARGET_ID>**"

start
end
top
bottom

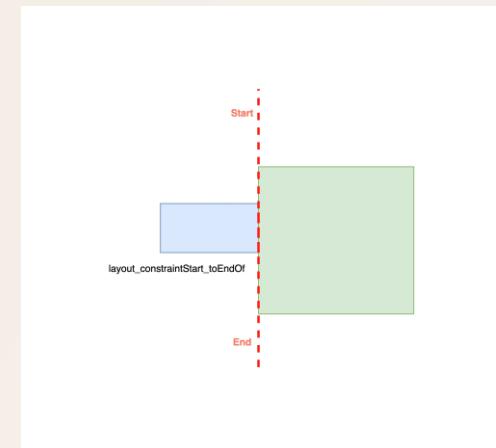
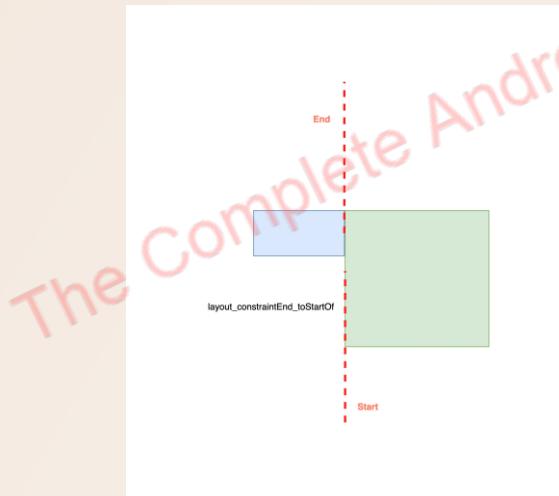
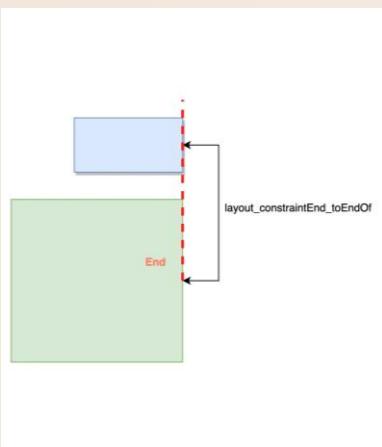
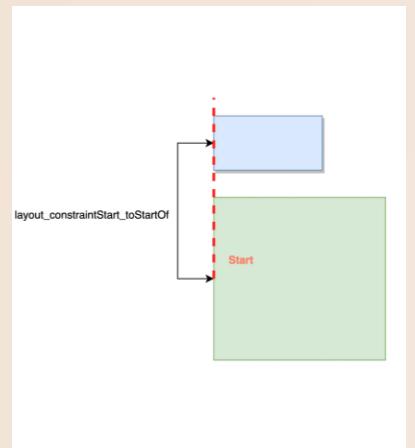
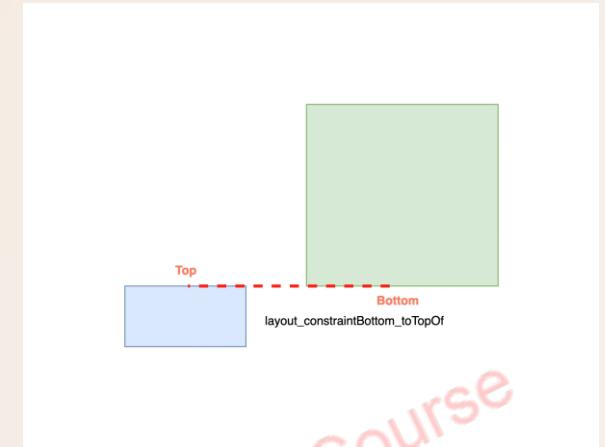
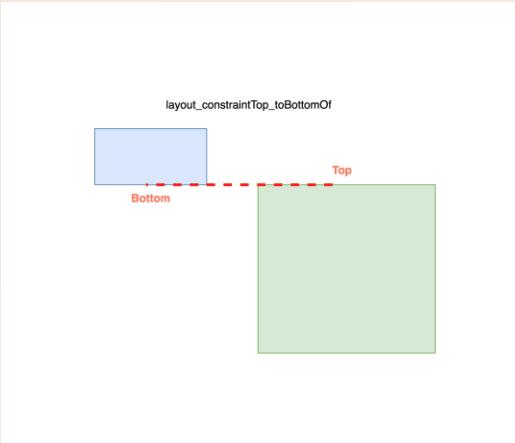
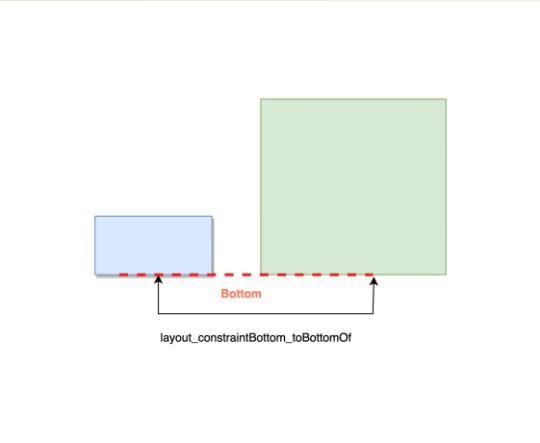
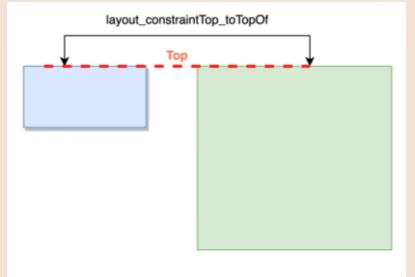
start
end
top
bottom

@+id/anyViewID
parent



The Complete Android Developer Course

Constraint Layout



The Complete Android Developer Course

View Size

You can use the corner handles to resize a view, but this hard codes the size so the view will not resize for different content or screen sizes. To select a different sizing mode, click a view and open the Attributes window on the right side of the editor.

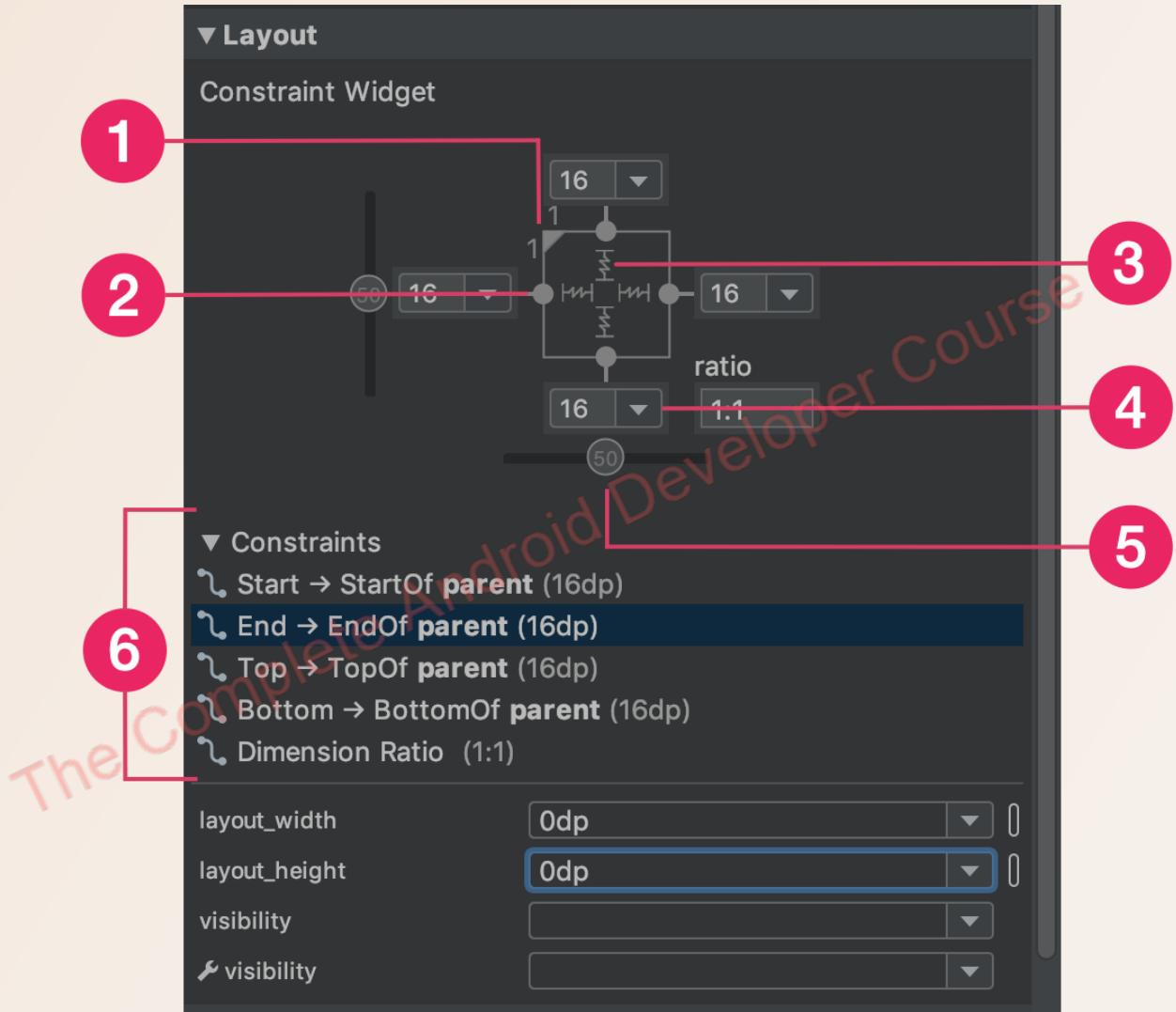
When selecting a view, the Attributes window includes controls for **1** size ratio, **2** deleting constraints, **3** height/width mode, **4** margins, and **5** constraint bias. You can also highlight individual constraints in the Layout Editor by clicking on them in the **6** constraint list.

You can change the way the height and width are calculated by clicking the symbols indicated with callout **3**

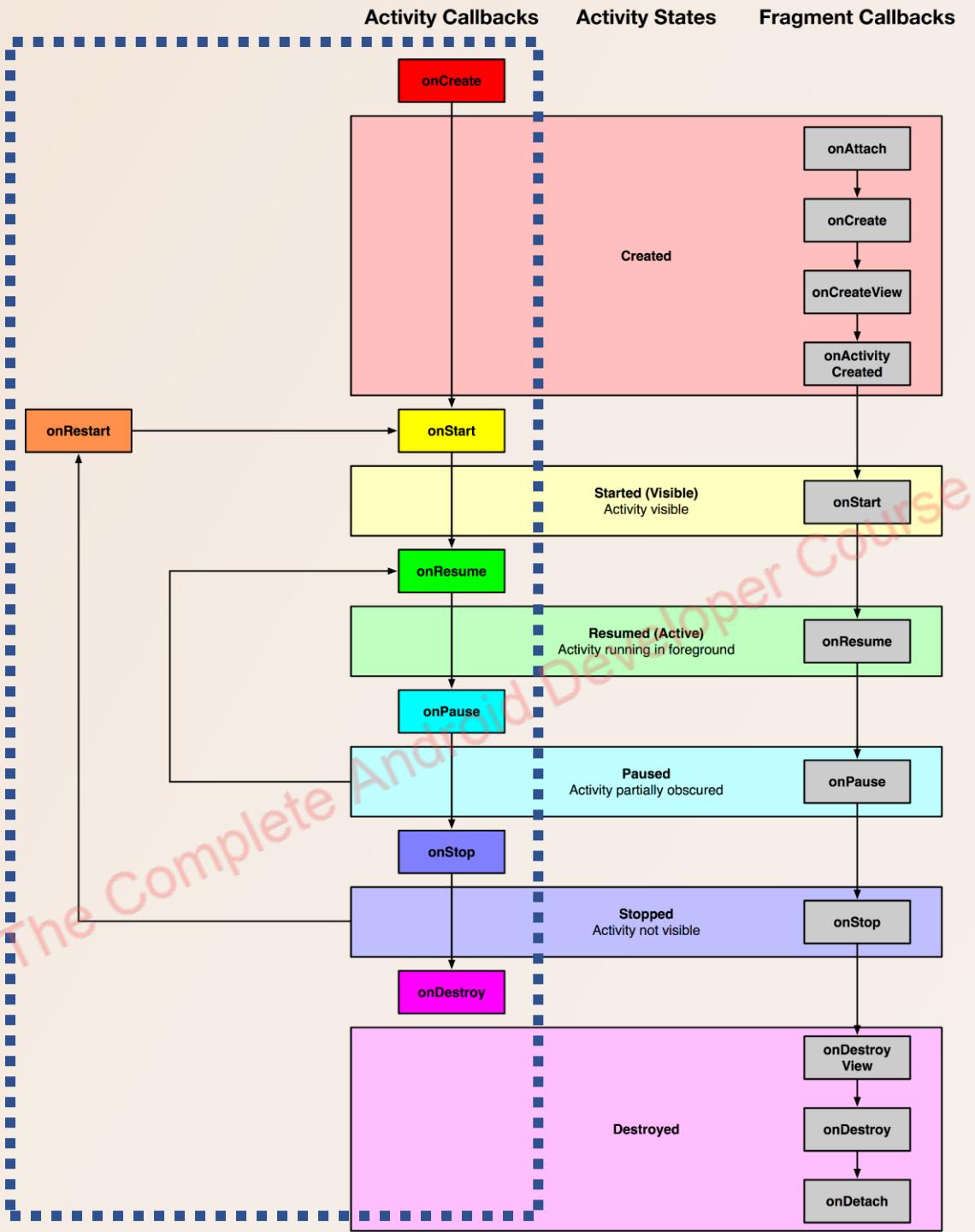
 **Fixed:** You specify a specific dimension in the text box below or by resizing the view in the editor.

 **Wrap Content:** The view expands only as much as needed to fit its contents

 **Match Constraints:** The view expands as much as possible to meet the constraints on each side (after accounting for the view's margins). However, you can modify that behavior with the following attributes and values



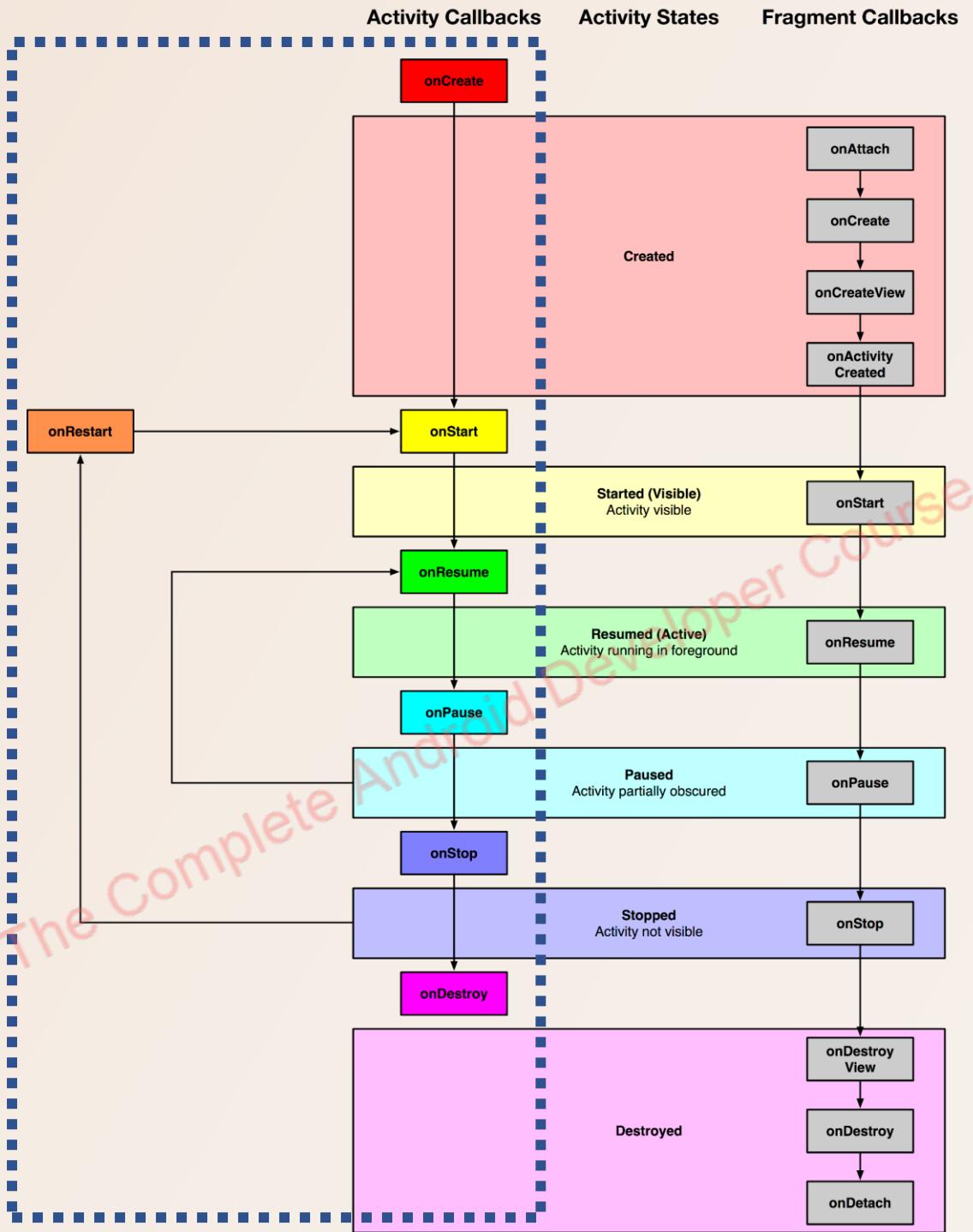
Activity Life Cycle



onCreate()

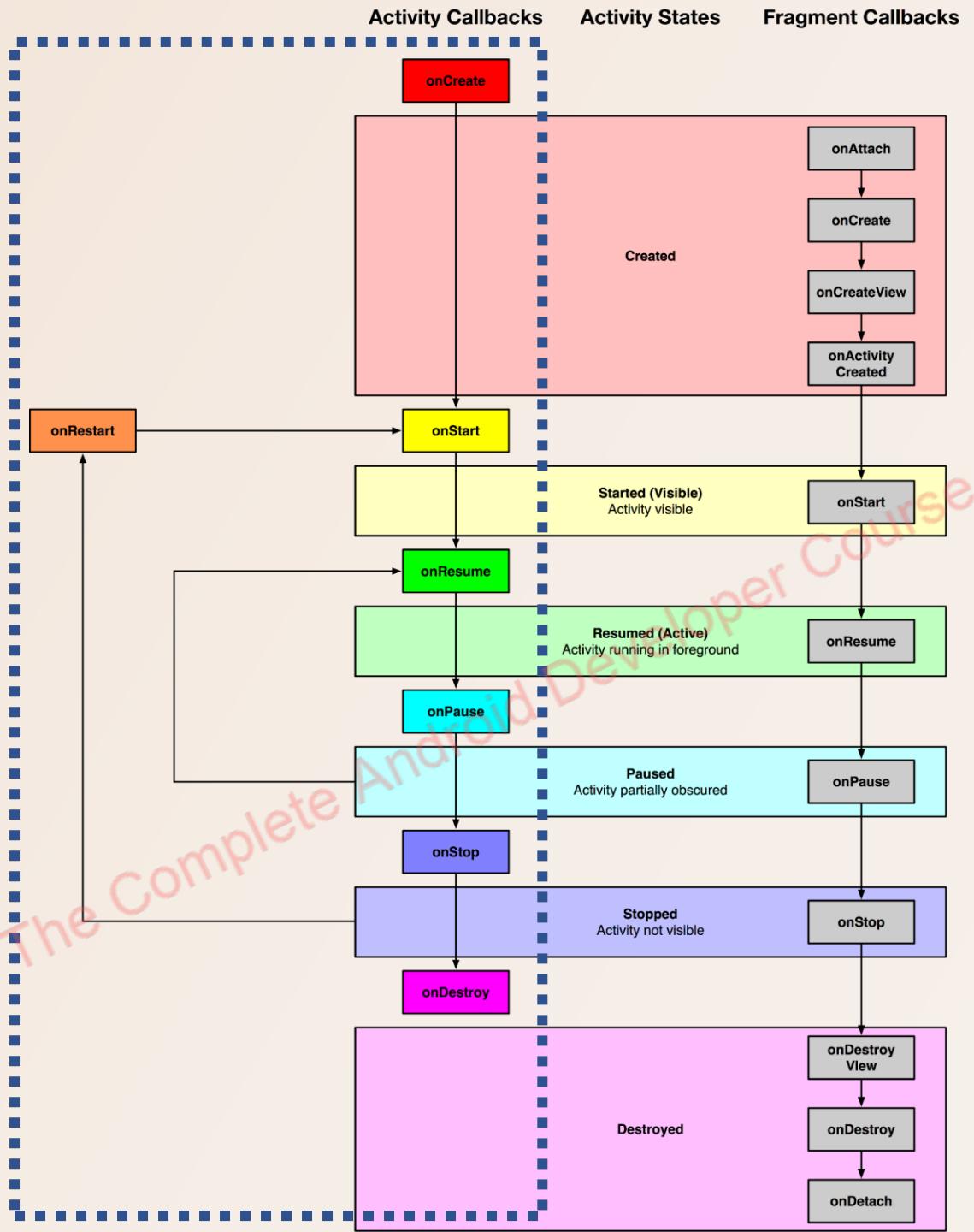
The method that is called when the activity is first created and the ideal location for most initialization tasks to be performed.

The method is passed an argument in the form of a *Bundle* object that may contain dynamic state information (typically relating to the state of the user interface) from a prior invocation of the activity.



onRestart()

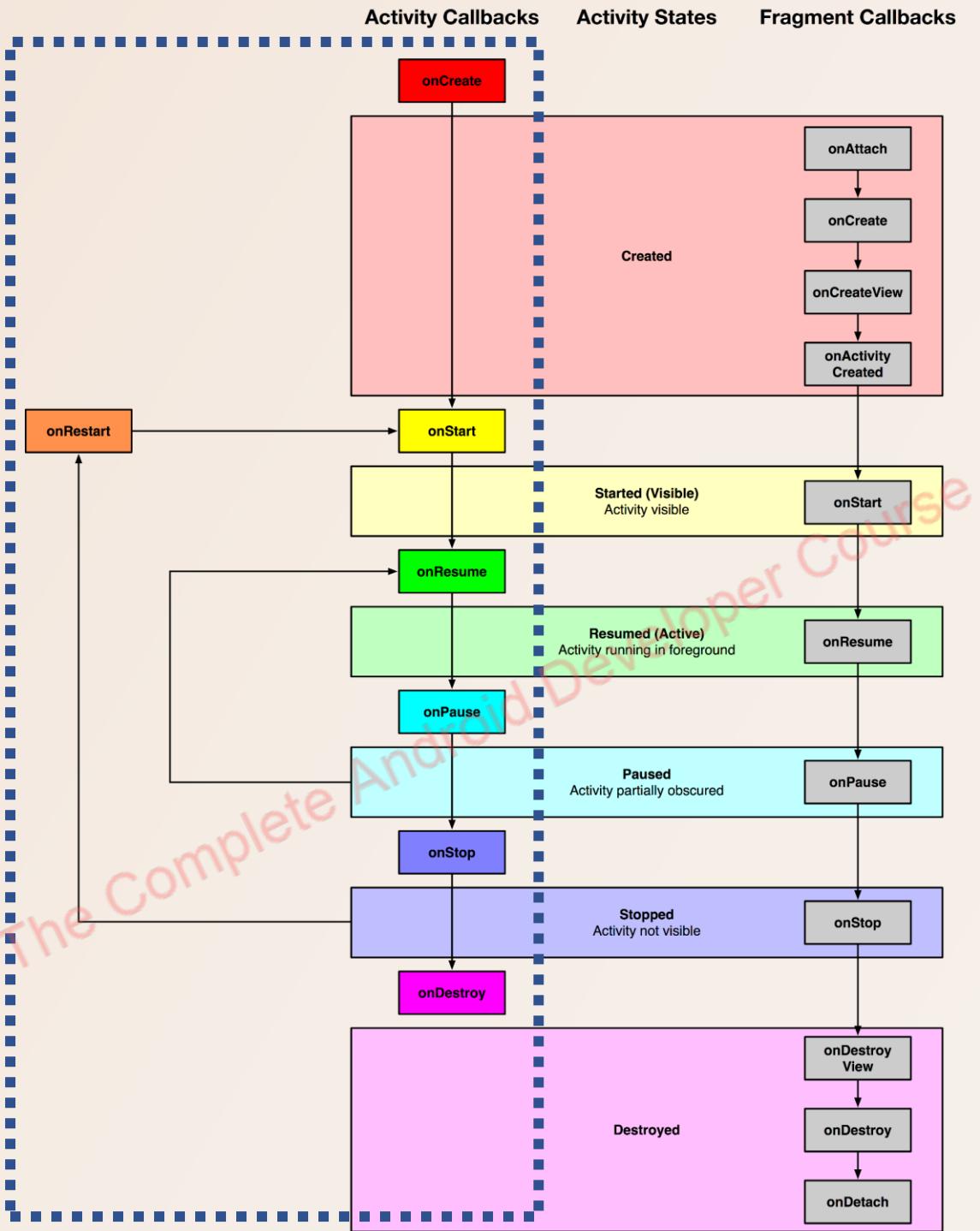
onRestart() – Called when the activity is about to restart after having previously been stopped by the runtime system.



onStart()

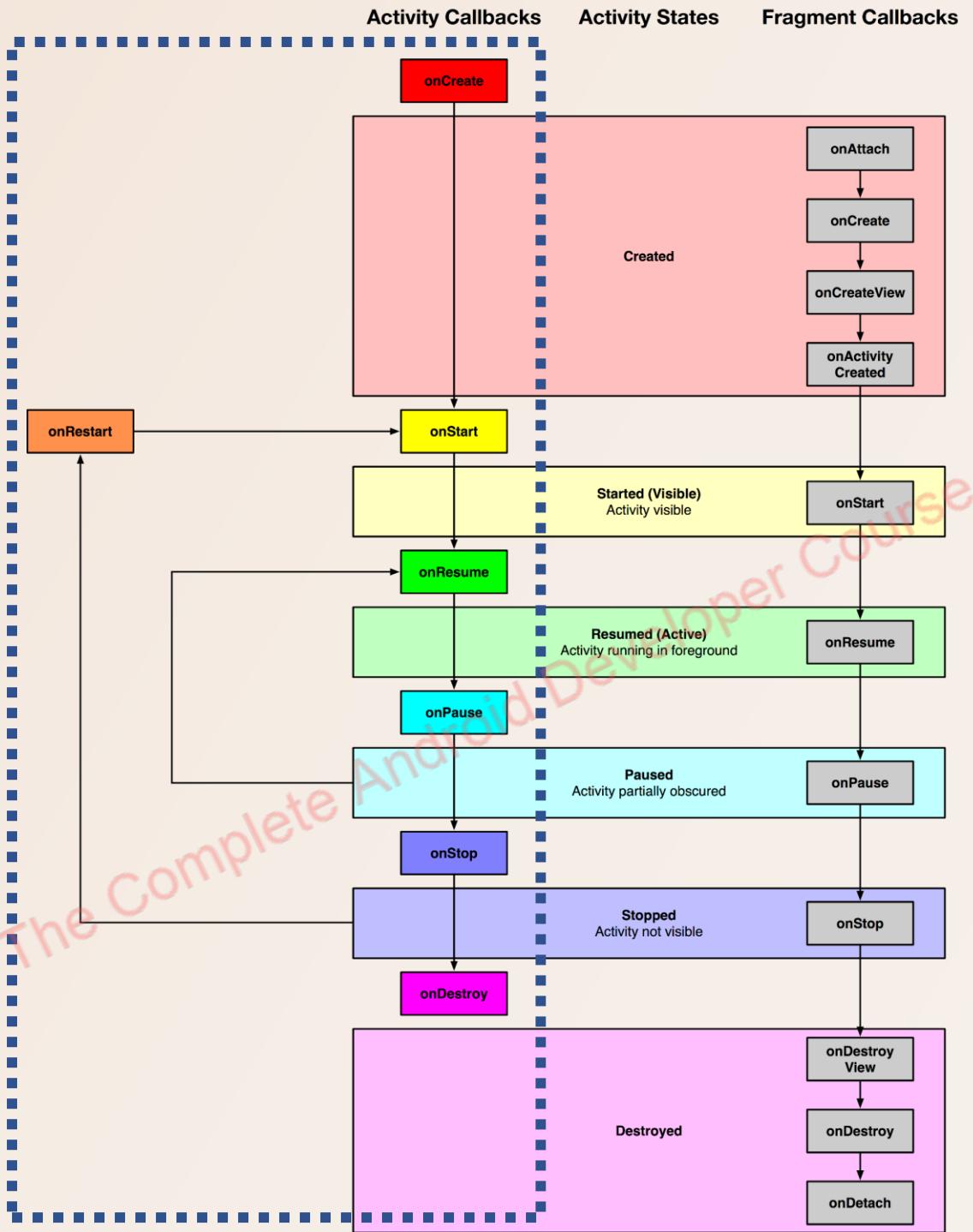
Always called immediately after the call to the onCreate() or onRestart() methods, this method indicates to the activity that it is about to become visible to the user.

This call will be followed by a call to onResume() if the activity moves to the top of the activity stack, or onStop() in the event that it is pushed down the stack by another activity.



onResume()

Indicates that the activity is now at the top of the activity stack and is the activity with which the user is currently interacting.



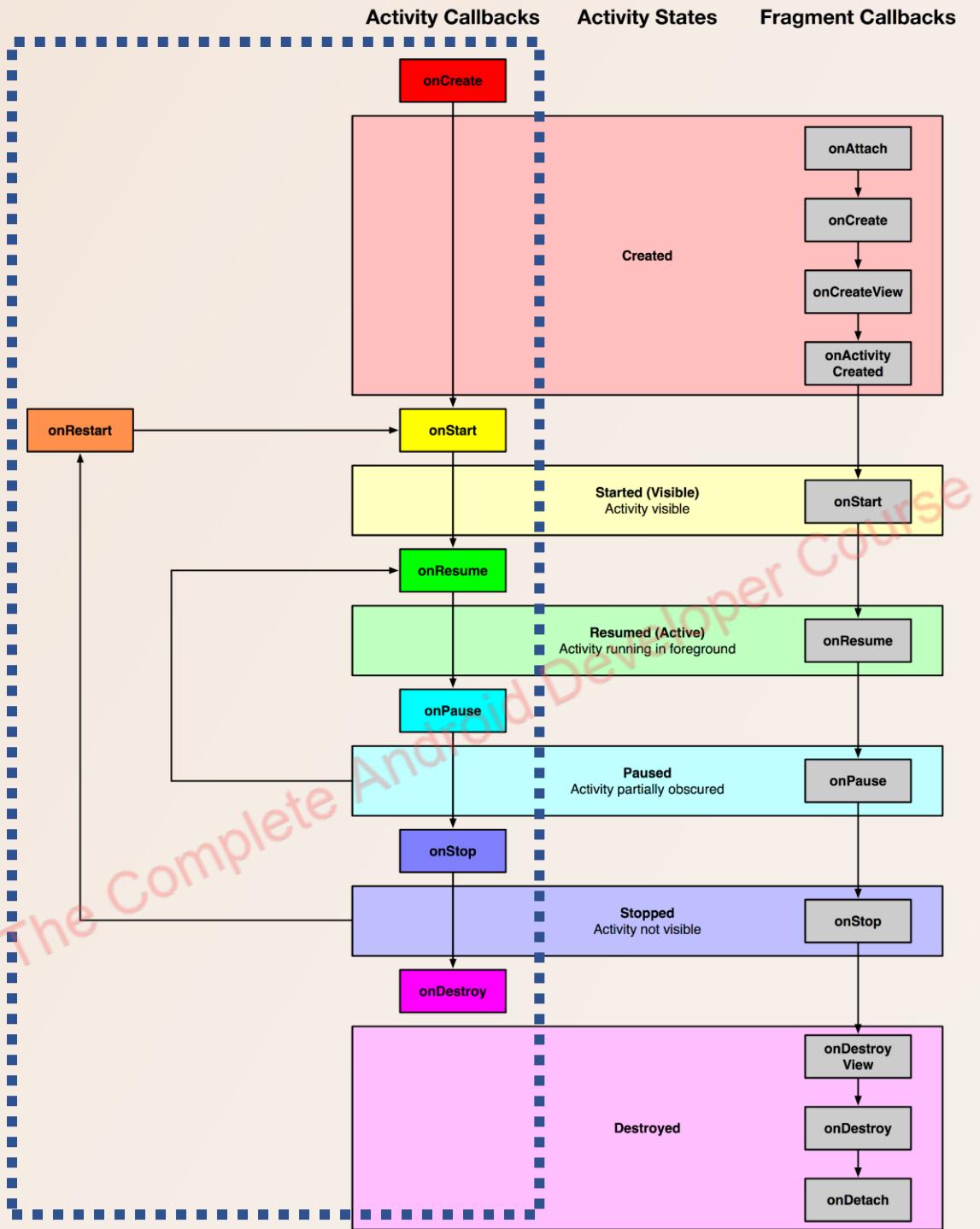
onPause()

Indicates that a previous activity is about to become the foreground activity. This call will be followed by a call to either the onResume() or onStop() method depending on whether the activity moves back to the foreground or becomes invisible to the user.

Steps may be taken within this method to store persistent state information not yet saved by the app.

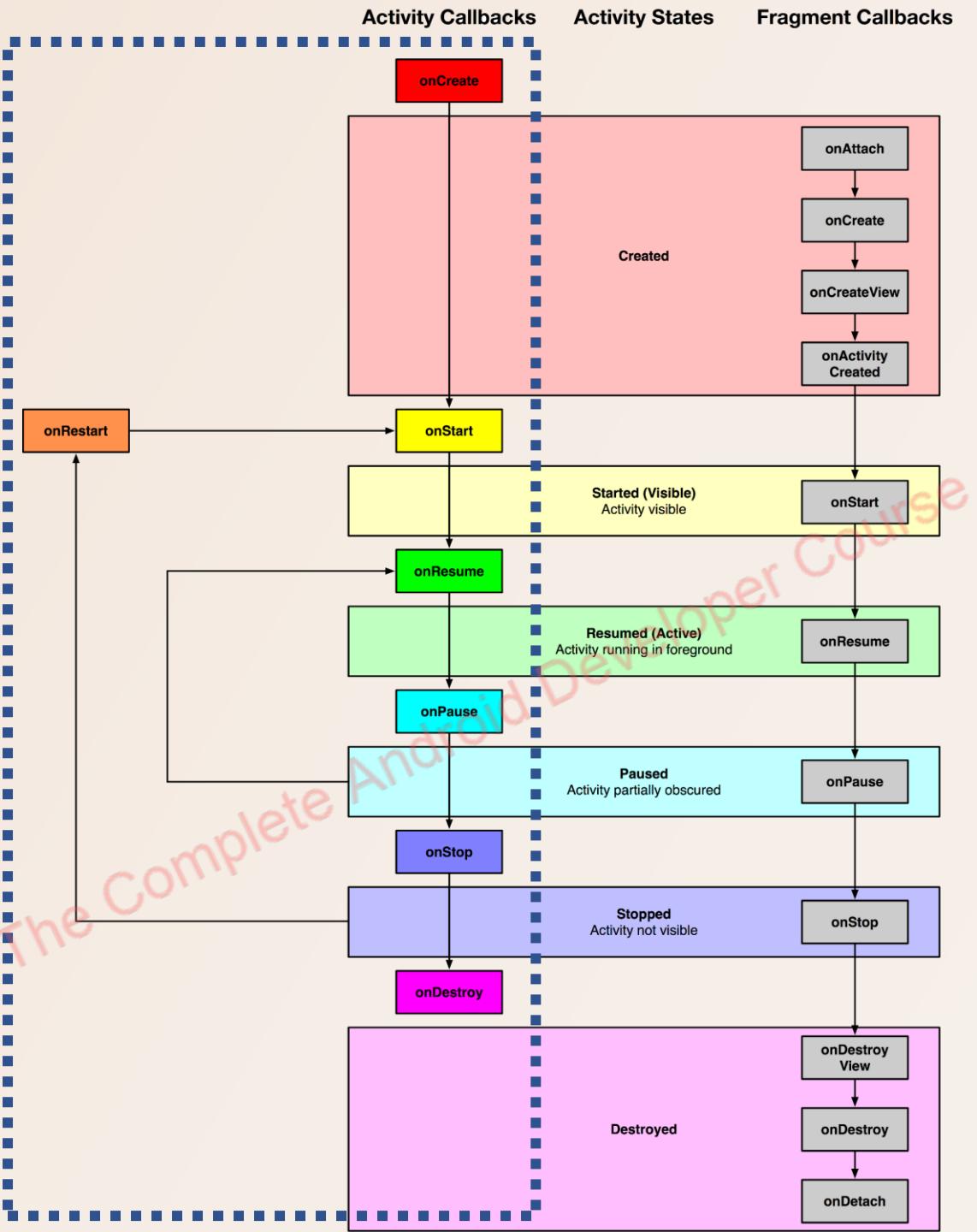
To avoid delays in switching between activities, time consuming operations such as storing data to a database or performing network operations should be avoided within this method.

This method should also ensure that any CPU intensive tasks such as animation are stopped.



onStop()

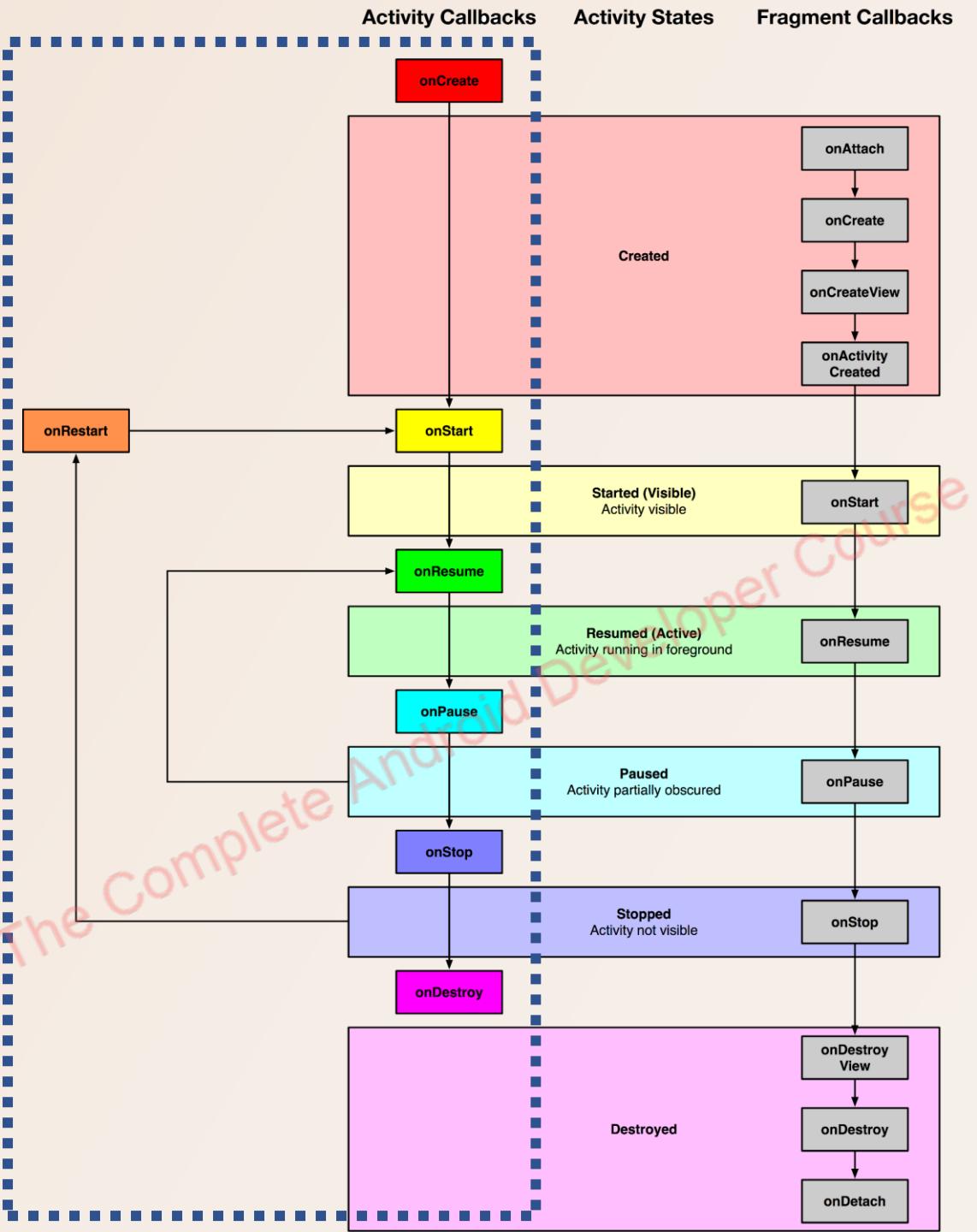
The activity is now no longer visible to the user. The two possible scenarios that may follow this call are a call to `onRestart()` in the event that the activity moves to the foreground again, or `onDestroy()` if the activity is being terminated.



onDestroy()

The activity is about to be destroyed, either voluntarily because the activity has completed its tasks and has called the `finish()` method or because the runtime is terminating it either to release memory or due to a configuration change (such as the orientation of the device changing).

It is important to note that a call will not always be made to `onDestroy()` when an activity is terminated.



The Complete Android Developer Course

```
File Edit View Navigate Code Refactor Build Run Tools VCS Window Help LifeCycleApp - MainActivity.kt [LifeCycleApp.app.main]
LifeCycleApp > app > src > main > java > com > mastercoding > lifecyclean > MainActivity > onCreate
activity_main.xml > MainActivity.kt
class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        Toast.makeText(context, "onCreate() is called",
                      Toast.LENGTH_SHORT
                ).show()
    }

    override fun onStart() {
        super.onStart()
        // Perform tasks when the activity starts

        Toast.makeText(context, "onStart() is called",
                      Toast.LENGTH_SHORT
                ).show()
    }
}
```

Activity Life Cycle Methods

The Complete Android Developer Course

```
override fun onResume() {
    super.onResume()
    // Handle activities when the app is in the foreground
    Toast.makeText(context: this,
        text: "onResume() is called",
        Toast.LENGTH_SHORT
    ).show()
}

override fun onPause() {
    super.onPause()
    // pause or release resources when the activity loses focus

    Toast.makeText(context: this,
        text: "onPause() is called",
        Toast.LENGTH_SHORT
    ).show()
}

override fun onStop() {
    super.onStop()
    // release resources that should be closed when activity is not active
}
```

Activity Life Cycle Methods

The Complete Android Developer Course

```
override fun onRestart() {
    super.onRestart()
    // Prepare for a restart
    Toast.makeText(context: this,
        text: "onRestart() is called",
        Toast.LENGTH_SHORT
    ).show()
}

override fun onDestroy() {
    super.onDestroy()

    // perform final clean up
}
```

Font size: 31pt Reset to 13pt

Activity Life Cycle Methods

The Complete Android Developer Course

This screenshot shows the Android Studio interface with four tabs open:

- MainActivity.kt**: Contains code for starting an explicit intent to the second activity.
- SecondActivity.kt**: Contains code for receiving data from the first activity via an intent extra.
- activity_main.xml**: The layout for the first activity, featuring a text view and two buttons.
- activity_second.xml**: The layout for the second activity, featuring a text view and two buttons.

The **MainActivity.kt** code includes:

```
import android.os.Bundle  
import androidx.appcompat.app.AppCompatActivity  
import kotlinx.android.synthetic.main.activity_main.*  
  
class MainActivity : AppCompatActivity() {  
  
    lateinit var myButton: Button  
    lateinit var openWeb: Button  
  
    override fun onCreate(savedInstanceState: Bundle?) {  
        super.onCreate(savedInstanceState)  
        setContentView(R.layout.activity_main)  
  
        // Intents: used to navigate from one component to another  
  
        // Types of Intents:  
        // 1- Explicit: they explicitly specify the target component  
        // 2- Implicit  
  
        myButton = findViewById(R.id.goToNextBtn)  
        myButton.setOnClickListener {  
            // Create an Explicit Intent  
            val explicitIntent = Intent(packageContext, SecondActivity::class.java)  
  
            explicitIntent.putExtra("myname", "Jack")  
            startActivity(explicitIntent)  
  
            openWeb = findViewById(R.id.openWebBtn)  
            openWeb.setOnClickListener {  
                // Create implicit intent with the action  
                // VIEW and a URL  
                val implicitIntent = Intent(Intent.ACTION_VIEW,  
                    Uri.parse("https://www.google.com"))  
  
                startActivity(implicitIntent)  
            }  
        }  
    }  
}
```

The **SecondActivity.kt** code includes:

```
import android.os.Bundle  
import androidx.appcompat.app.AppCompatActivity  
import kotlinx.android.synthetic.main.activity_second.*  
  
class SecondActivity : AppCompatActivity() {  
  
    lateinit var titleTextView: TextView  
  
    override fun onCreate(savedInstanceState: Bundle?) {  
        super.onCreate(savedInstanceState)  
        setContentView(R.layout.activity_second)  
  
        titleTextView = findViewById(R.id.myTextView)  
  
        // Receive data from the Intent  
        val receivedData = intent.getStringExtra("myname")  
  
        // using the received data as needed  
        titleTextView.text = "Welcome $receivedData to second activity"  
    }  
}
```

The **activity_main.xml** layout includes:

```
<?xml version="1.0" encoding="utf-8"?>  
<androidx.constraintlayout.widget.ConstraintLayout  
    xmlns:app="http://schemas.android.com/apk/res-auto"  
    xmlns:tools="http://schemas.android.com/tools"  
    android:layout_width="match_parent"  
    android:layout_height="match_parent"  
    tools:context=".MainActivity">  
  
    <TextView  
        android:id="@+id/textView"  
        android:layout_width="wrap_content"  
        android:layout_height="wrap_content"  
        android:text="Hello World!"  
        app:layout_constraintBottom_toBottomOf="parent"  
        app:layout_constraintEnd_toEndOf="parent"  
        app:layout_constraintStart_toStartOf="parent"  
        app:layout_constraintTop_toTopOf="parent" />  
  
    <Button  
        android:id="@+id/goToNextBtn"  
        android:layout_width="wrap_content"  
        android:layout_height="wrap_content"  
        android:layout_marginTop="164dp"  
        android:layout_marginBottom="163dp"  
        android:text="Go to Second Activity"  
        app:layout_constraintBottom_toBottomOf="@+id/textView"  
        app:layout_constraintEnd_toEndOf="parent"  
        app:layout_constraintStart_toStartOf="parent"  
        app:layout_constraintTop_toTopOf="parent" />  
  
    <Button  
        android:id="@+id/openWebBtn"  
        android:layout_width="wrap_content"  
        android:layout_height="wrap_content"  
        android:layout_marginTop="8dp"  
        android:text="Go to Google"  
        app:layout_constraintEnd_toEndOf="parent"  
        app:layout_constraintStart_toStartOf="parent"  
        app:layout_constraintTop_toBottomOf="@+id/textView" />  
/</androidx.constraintlayout.widget.ConstraintLayout>
```

The **activity_second.xml** layout includes:

```
<?xml version="1.0" encoding="utf-8"?>  
<androidx.constraintlayout.widget.ConstraintLayout  
    xmlns:android="http://schemas.android.com/apk/res/android"  
    xmlns:app="http://schemas.android.com/apk/res-auto"  
    xmlns:tools="http://schemas.android.com/tools"  
    android:layout_width="match_parent"  
    android:layout_height="match_parent"  
    tools:context=".SecondActivity">  
  
    <TextView  
        android:id="@+id/myTextView"  
        android:layout_width="wrap_content"  
        android:layout_height="wrap_content"  
        android:text="Welcome to Second Activity"  
        android:textSize="32sp"  
        app:layout_constraintStart_toStartOf="parent"  
        app:layout_constraintTop_toTopOf="parent" />  
/</androidx.constraintlayout.widget.ConstraintLayout>
```

Bottom navigation bar:

- Version Control
- Run
- Profiler
- Logcat
- App Quality Insights
- Build
- TOD
- Layout Inspector

Bottom status bar:

- Install successfully finished in 1 s 496 ms. (5 minutes ago)
- 20:18 LF UTF-8 4 spaces
- 6:31 PM 11/10/2023

File Edit View Navigate Code Refactor Build Run Tools VCS Window Help MoreWidgetsApp - MainActivity.kt [MoreWidgetsApp.app.main]

MoreWidgetsApp > app > src > main > java > com > mastercoding > morewidgetsapp > MainActivity > displayCheckBox

activity_main.xml MainActivity.kt

8 6 1

```
48
49     /** Check Box: allows users to toggle between 2 states: checked & unchecked */
50     fun displayCheckBox() {
51         // Initializing CheckBox
52         var checkBox = findViewById<CheckBox>(R.id.checkBox)
53
54         // Handle checkbox state changes
55         checkBox.setOnCheckedChangeListener { _, isChecked ->
56             // bottomView: represents the "CompoundButton" that has
57             // had checked state changed. It refers to the checkbox
58
59             // isChecked: a boolean indicating whether the button is
60             // checked (true) or unchecked (false)
61
62             // _ : underscore is used when you don't need a parameter from
63             // a lambda expression (as a placeholder)
64
65             if (isChecked) {
66                 // checkbox is checked
67                 Toast.makeText(
68                     context: this, text: "You Checked Tomato",
69                     Toast.LENGTH_SHORT
70                 ).show()
71             } else {
72                 // checkbox is unchecked
73             }
74         }
75     }
76 }
```

The Complete Android Developer Course

Notifications

Running Devices

Device Explorer

Version Control Run TODO Problems Terminal Services App Quality Insights App Inspector

Install successfully finished in 1 s 963 ms. (3 minutes ago)

Layout Inspector

62:75 LF UTF-8 4 spaces

3:14 PM 11/13/2023

Checkbox

File Edit View Navigate Code Refactor Build Run Tools VCS Window Help MoreWidgetsApp - MainActivity.kt [MoreWidgetsApp.app.main]

MoreWidgetsApp > app > src > main > java > com > mastercoding > morewidgetsapp > MainActivity > displayRadioGroup

activity_main.xml > MainActivity.kt

```
77     /** Radio Button: allows users to select a single option from a group of options */
78     fun displayRadioGroup(){
79         // Initializing and Declaring Radio buttons and group
80         val radioGroup = findViewById<RadioGroup>(R.id.radioGroup)
81
82         // Detecting the selected radio button
83         radioGroup.setOnCheckedChangeListener { radioGroup, i ->
84             // radioGroup: represents the "RadioGroup" itself
85             // i: represents the resource ID of Checked RadioButton
86
87             when(i){
88                 R.id.cheeseRadioButton -> {
89                     Toast.makeText(
90                         context: this,
91                         text: "You selected Cheese", Toast.LENGTH_SHORT
92                     ).show()
93                 }
94
95                 R.id.spiceRadioButton -> {
96                     Toast.makeText(
97                         context: this,
98                         text: "You selected Spice", Toast.LENGTH_SHORT
99                     ).show()
100                }
101
102                 R.id.onionRadioButton -> {
103                     Toast.makeText(
104                         context: this,
105                         text: "You selected Onion", Toast.LENGTH_SHORT
106                     ).show()
107                }
108            }
109        }
110    }
111
112    companion object {
113        const val TAG = "RadioGroup Example"
114    }
115}
```

The Complete Android Developer Course

Version Control Run TODO Problems Terminal Services App Quality Insights App

Install successfully finished in 1 s 963 ms. (4 minutes ago)

Layout Inspector

81:1 LF UTF-8 4 spaces

3:15 PM 11/13/2023

RadioGroup & RadioButtons

The Complete Android Developer Course

```
File Edit View Navigate Code Refactor Build Run Tools VCS Window Help MoreWidgetsApp - MainActivity.kt [MoreWidgetsApp.app.main]
MoreWidgetsApp > app > src > main > java > com > mastercoding > morewidgetsapp > MainActivity > displaySpinner
activity_main.xml > MainActivity.kt
Project Resource Manager Device Manager
114     /** Spinner: provides a drop-down menu with a list of items where users can select an item */
115     fun displaySpinner(){
116         // Initialise the spinner : (The View)
117         val spinner:Spinner = findViewById(R.id.mySpinner)
118
119         // (Data source) for the spinner: Array
120         val operatingSystems = arrayOf(
121             "Windows","iOS","Android","Linux")
122
123         // Adapter: a bridge between the data and the UI Components
124         // ArrayAdapter: used to bind an array of data to a spinner
125         // converts each item in the array to a view in a spinner
126         val adapter = ArrayAdapter(
127             context,
128             android.R.layout.simple_spinner_item,
129             operatingSystems
130         )
131
132         // set the adapter to a view
133         spinner.adapter = adapter
134
135         // Handling the item selection in a spinner
136         spinner.onItemSelectedListener = object :
137             AdapterView.OnItemSelectedListener{
138                 // object: serves as the listener for item selection events
139                 override fun onItemSelected(p0: AdapterView<*>?, p1: View?, p2: Int, p3: Long) {
140                     // p0 = parentView
141                     // p1 = selected.itemView
142                     // p2 = position
143                     // p3 = id
144
145                     // Handle the selected item
146                     //val selectedItem = p0?.getItemAtPosition(p2).toString()
147                     val selectedItem2 = operatingSystems[p2]
148                     Toast.makeText(applicationContext,
149                         text: "you selected $selectedItem2", Toast.LENGTH_SHORT).show()
150
151                 }
152             }
```

Version Control Run TODO Problems Terminal Services App Quality Insights App Layout Inspector

Install successfully finished in 1 s 963 ms. (5 minutes ago)

131:1 LF UTF-8 4 spaces

3:16 PM 11/13/2023

File Edit View Navigate Code Refactor Build Run Tools VCS Window Help MoreWidgetsApp - MainActivity.kt [MoreWidgetsApp.app.main]

MoreWidgetsApp > app > src > main > java > com > mastercoding > morewidgetsapp > MainActivity > displayTimePicker

activity_main.xml MainActivity.kt

```
164     /** Time Picker: allows users to select a time of day */
165     fun displayTimePicker(){
166         val timePicker:TimePicker = findViewById(R.id.timePicker)
167
168         // Handle the changes in time
169         timePicker.setOnTimeChangedListener { timePicker, hourOfDay, minute ->
170
171             // timePicker = view : represents the TimePicker itself
172             // i = hourOfDay : selected hour in 24-hour format
173             // i2 = minute : selected minute
174
175             // Formatting the selected time : hh:mm
176             val selectedTime = String.format("%02d:%02d",hourOfDay,minute)
177             Toast.makeText(applicationContext,
178                 text: "Your time: $selectedTime",
179                 Toast.LENGTH_SHORT
180             ).show()
181
182     }
183
184
185 }
```

The Complete Android Developer Course

Version Control Run TODO Problems Terminal Services App Quality Insights App Inspector

Install successfully finished in 1 s 963 ms. (6 minutes ago)

Layout Inspector

175:56 LF UTF-8 4 spaces

3:17 PM 11/13/2023

File Edit View Navigate Code Refactor Build Run Tools VCS Window Help MoreWidgetsApp - MainActivity.kt [MoreWidgetsApp.app.main]

MoreWidgetsApp > app > src > main > java > com > mastercoding > morewidgetsapp > MainActivity > displayDatePicker

activity_main.xml MainActivity.kt

186 */* Date Picker: allows users to select a date */*
187 @RequiresApi(Build.VERSION_CODES.O)
188 fun displayDatePicker(){

189 val datePicker = findViewById<DatePicker>(R.id.datePicker)

190 // Handling Date Changes
191 datePicker.setOnDateChangedListener { datePicker, i, i2, i3 ->
192 // datePicker = view : the datePicker itself
193 // i = year
194 // i2 = monthOfYear
195 // i3 = dayOfMonth
196 Toast.makeText(
197 applicationContext,
198 text: "Year: \$i , Month: \${i2+1} , Day: \$i3",
199 Toast.LENGTH_SHORT
200).show()
201 }
202 }
203 }
204 }
205 }

The Complete Android Developer Course

Version Control Run TODO Problems Terminal Services App Quality Insights Logcat Layout Inspector

Install successfully finished in 1 s 963 ms. (6 minutes ago)

Date Picker

204:6 LF UTF-8 4 spaces

3:18 PM 11/13/2023

File Edit View Navigate Code Refactor Build Run Tools VCS Window Help MoreWidgetsApp - MainActivity.kt [MoreWidgetsApp.app.main]

MoreWidgetsApp > app > src > main > java > com > mastercoding > morewidgetsapp > MainActivity

activity_main.xml MainActivity.kt

Pixel 6 Pro API 33

Device Manager Gradle

Project Resource Manager Bookmarks Build Variants

204 } 205 206 207 208 */** ProgressBar: indicates the progress of an ongoing operation */* 209 fun displayProgressBar(){ 210 211 val progressBar: ProgressBar = findViewById(R.id.myProgressBar) 212 213 // Setting the progress for determinate ProgressBar 214 val progressvalue = 60 215 progressBar.progress = progressvalue 216 }

The Complete Android Developer Course

Notifications Running Devices Device Explorer

Version Control Run TODO Problems Terminal Services App Quality Insights Logcat Layout Inspector

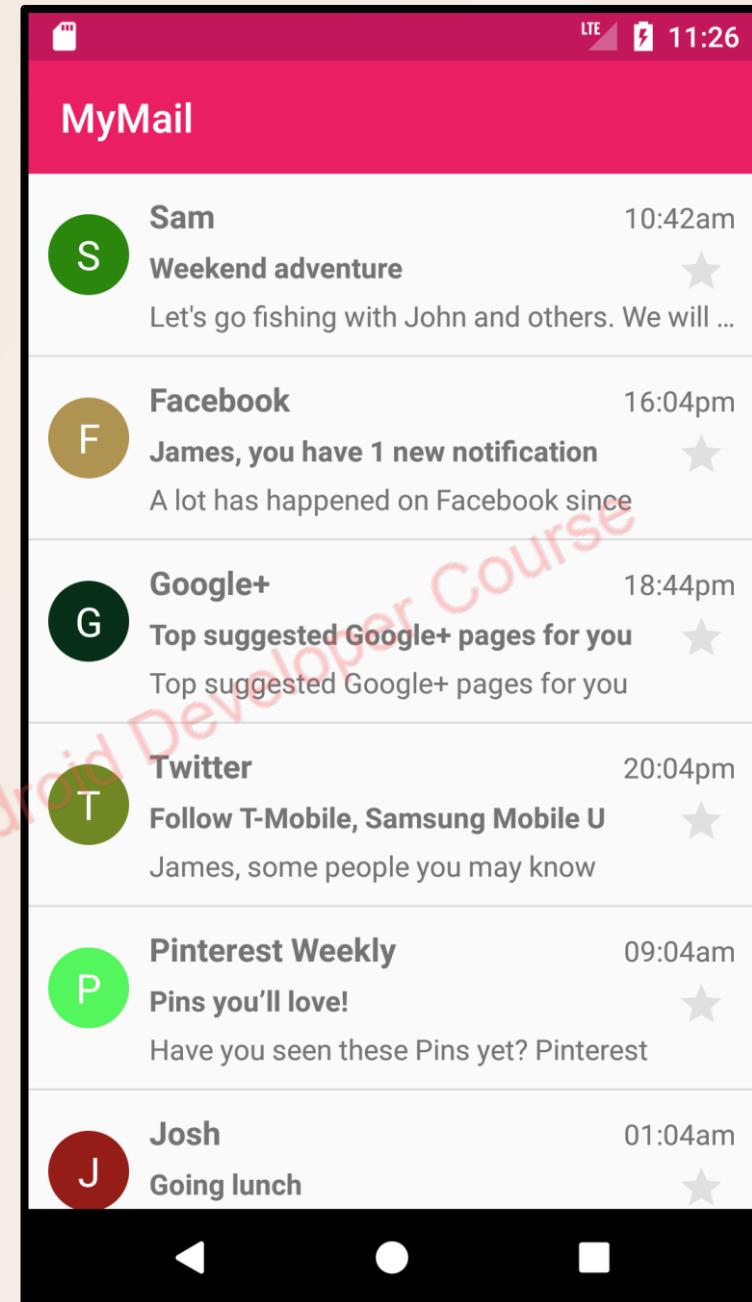
Install successfully finished in 1 s 963 ms. (7 minutes ago)

ProgressBar

RecyclerView

RecyclerView makes it easy to efficiently display large sets of data.

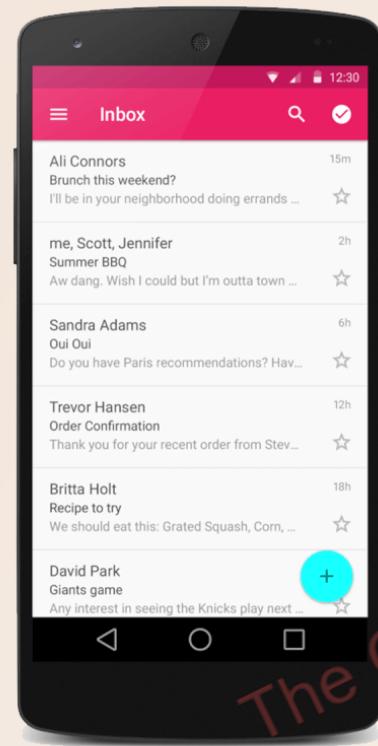
You supply the data and define how each item looks, and the RecyclerView library dynamically creates the elements when they're needed.



RecyclerView

As the name implies, RecyclerView recycles those individual elements. When an item scrolls off the screen, RecyclerView doesn't destroy its view.

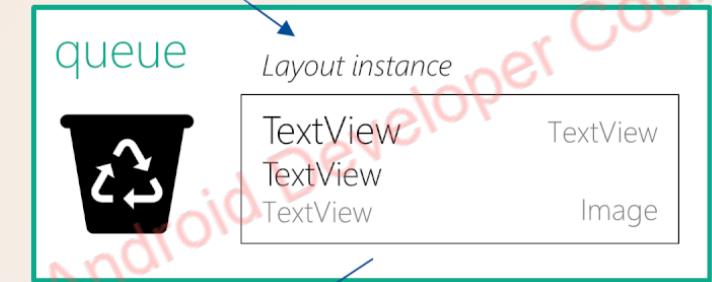
Instead, RecyclerView reuses the view for new items that have scrolled onscreen.



1. Layout instance scrolls out of view



2. Placed in queue



Layout instance

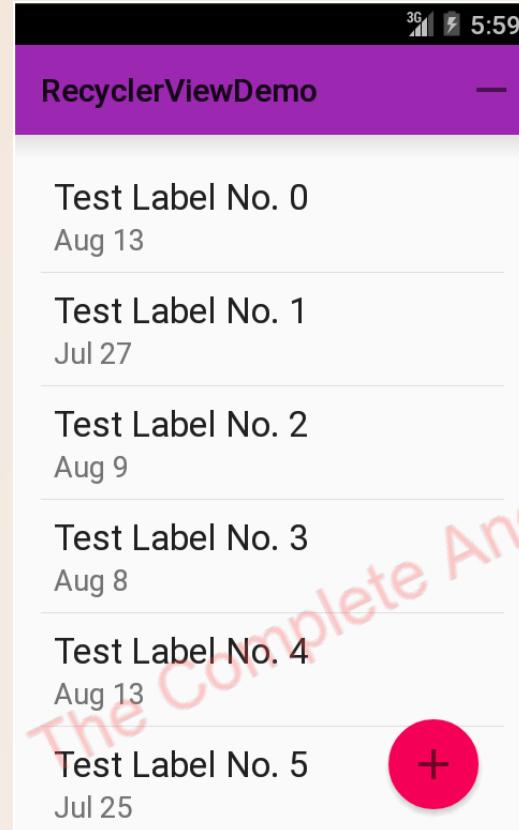


3. Filled with new content & scrolls in again

RecyclerView

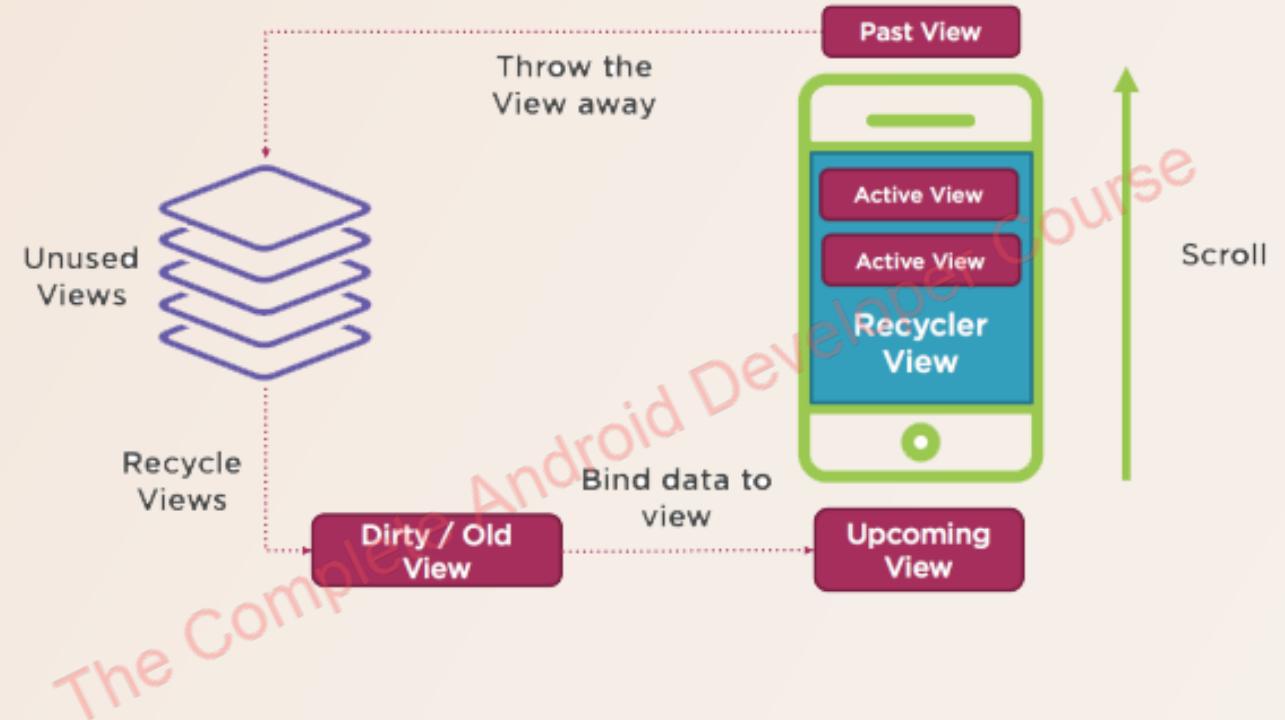
As the name implies, RecyclerView recycles those individual elements. When an item scrolls off the screen, RecyclerView doesn't destroy its view.

Instead, RecyclerView reuses the view for new items that have scrolled onscreen.



RecyclerView

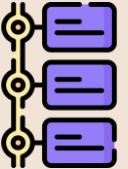
RecyclerView improves performance and your app's responsiveness, and it reduces power consumption.



Creating RecyclerView



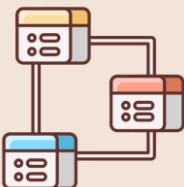
1- The Item Layout: an xml layout representing the layout of a single item in the recyclerview.



2- RecyclerView: creating a recyclerview in the activity and initializing it.



3- Model Class: representing the data class that acts as a structure for holding the information for every item of recyclerview.



4- Adapter Class: Holding all methods dealing with recyclerview implementation.
(creating, binding and determining the count of items).

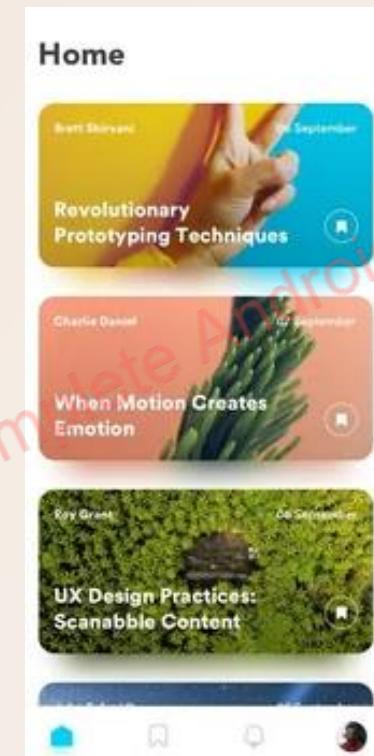
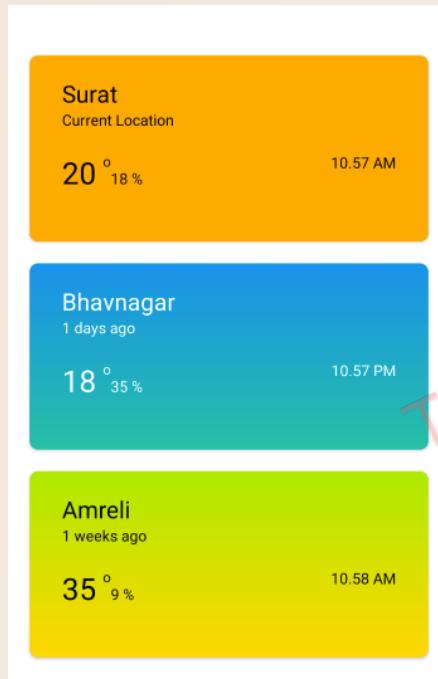


5- View Holder: Holds the references to the views within each item's layout.
Optimizing view lookups.

CardView

Used to display any sort of data by providing a rounded corner layout along with a specific elevation.

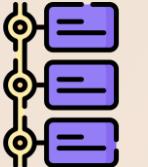
CardView can be used for creating items in listview or inside Recycler View.



Creating CardView App



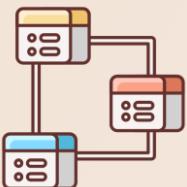
1- The CardView Item Layout: an xml layout representing the layout of a single item in the.recyclerview.



2- RecyclerView: creating a recyclerview in the activity and initializing it.



3- Model Class: representing the data class that acts as a structure for holding the information for every item of recyclerview.

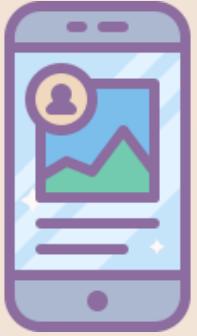


4- Adapter Class: Holding all methods dealing with recyclerview implementation. (creating, binding and determining the count of items).



5- View Holder: Holds the references to the views within each item's layout. Optimizing view lookups.

Android Components



Activity

Represents a single screen with a user interface



Services

Runs in the background to perform long-running operations without a user interface



Broadcast Receivers

Listen for system-wide announcements or intents



Content Providers

Manages access to a structured set of data

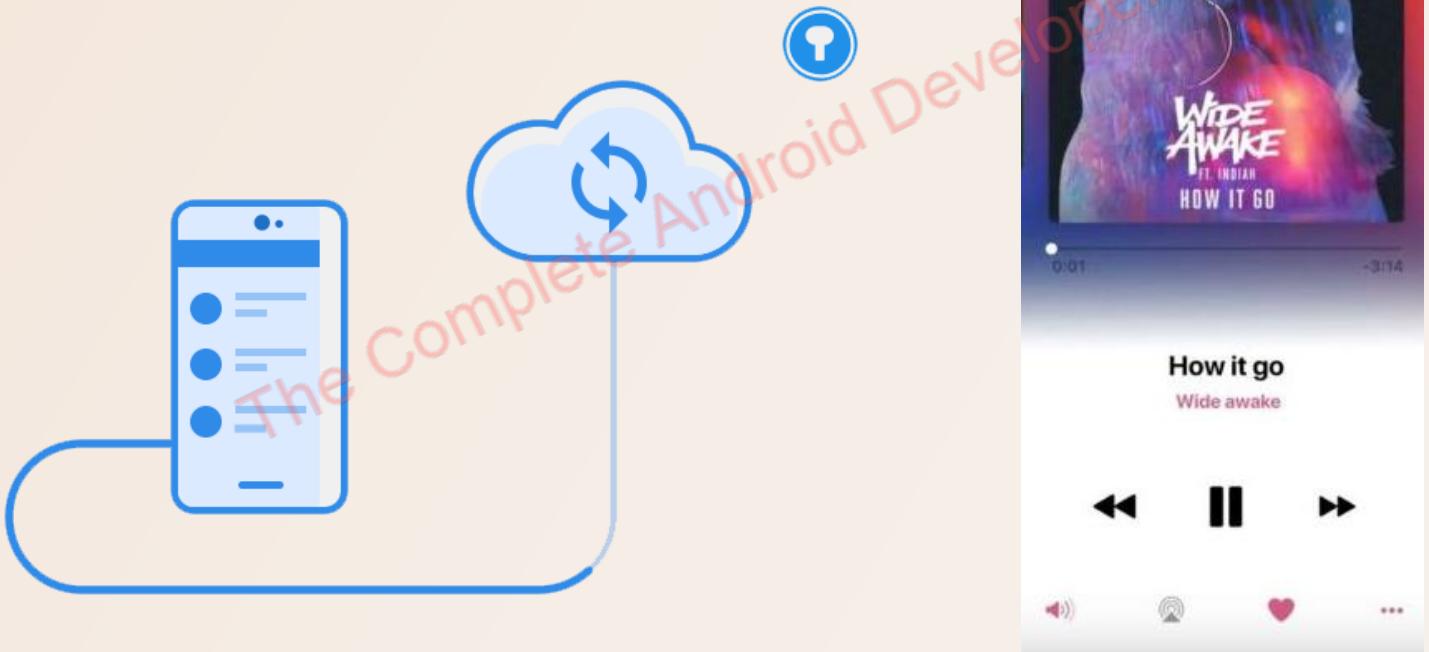
Services



Runs in the background to perform long-running operations without a user interface.

Types of Services:

- 1- Foreground
- 2- Background
- 3- Bound Services

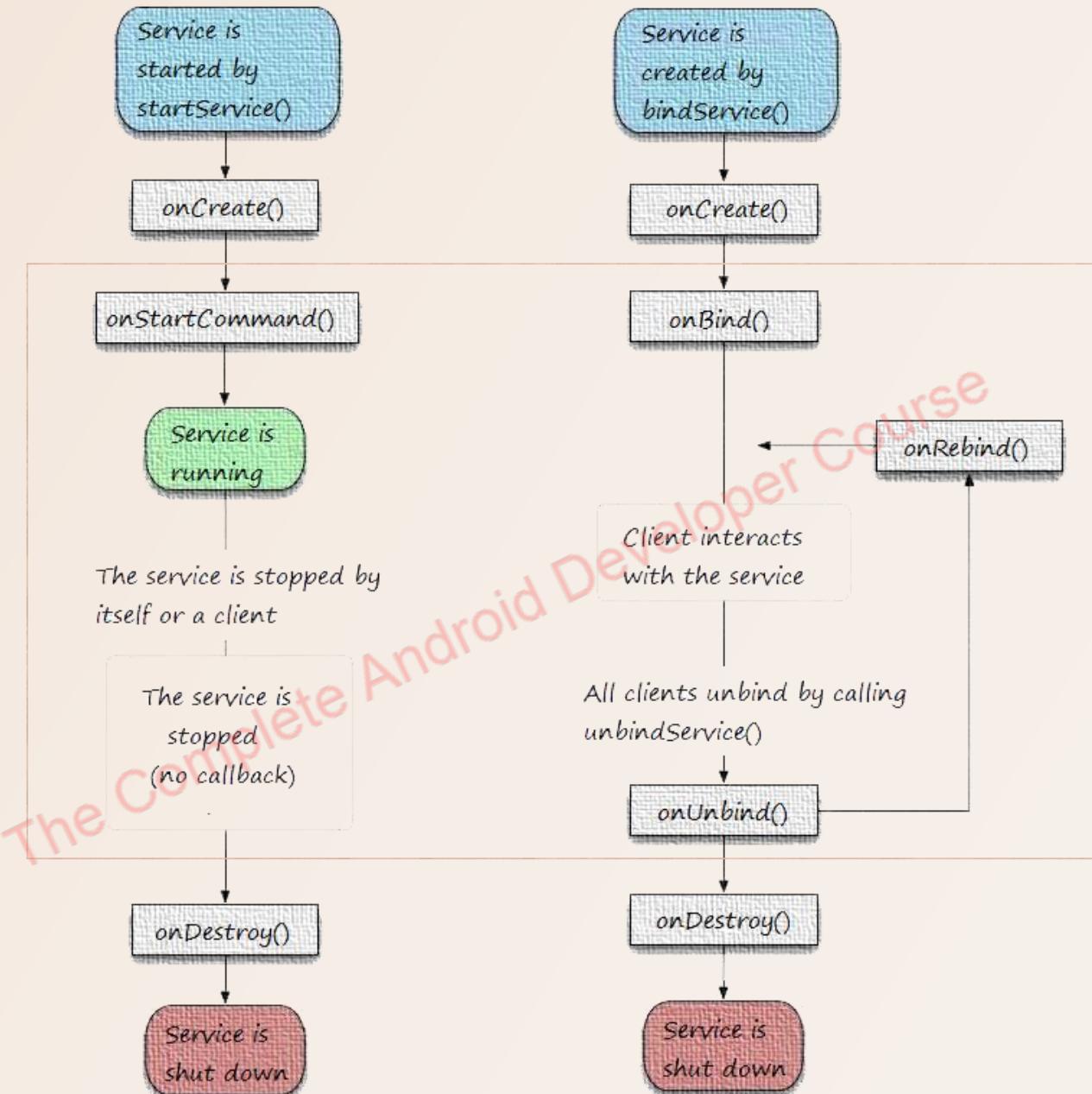


Services Life Cycle



Un Bounded Service

Bounded Service



Broadcast Receivers



Listens for system-wide broadcast events or intents and allows your application to respond to those events.

When any of these events occur, it brings the application into action by either creating a status bar notification or performing a task.

Unlike activities, android BroadcastReceiver doesn't contain any user interface.

The Complete Android Developer Course

Fragments

Fragments in Android are a bit like these LEGO bricks for building your app

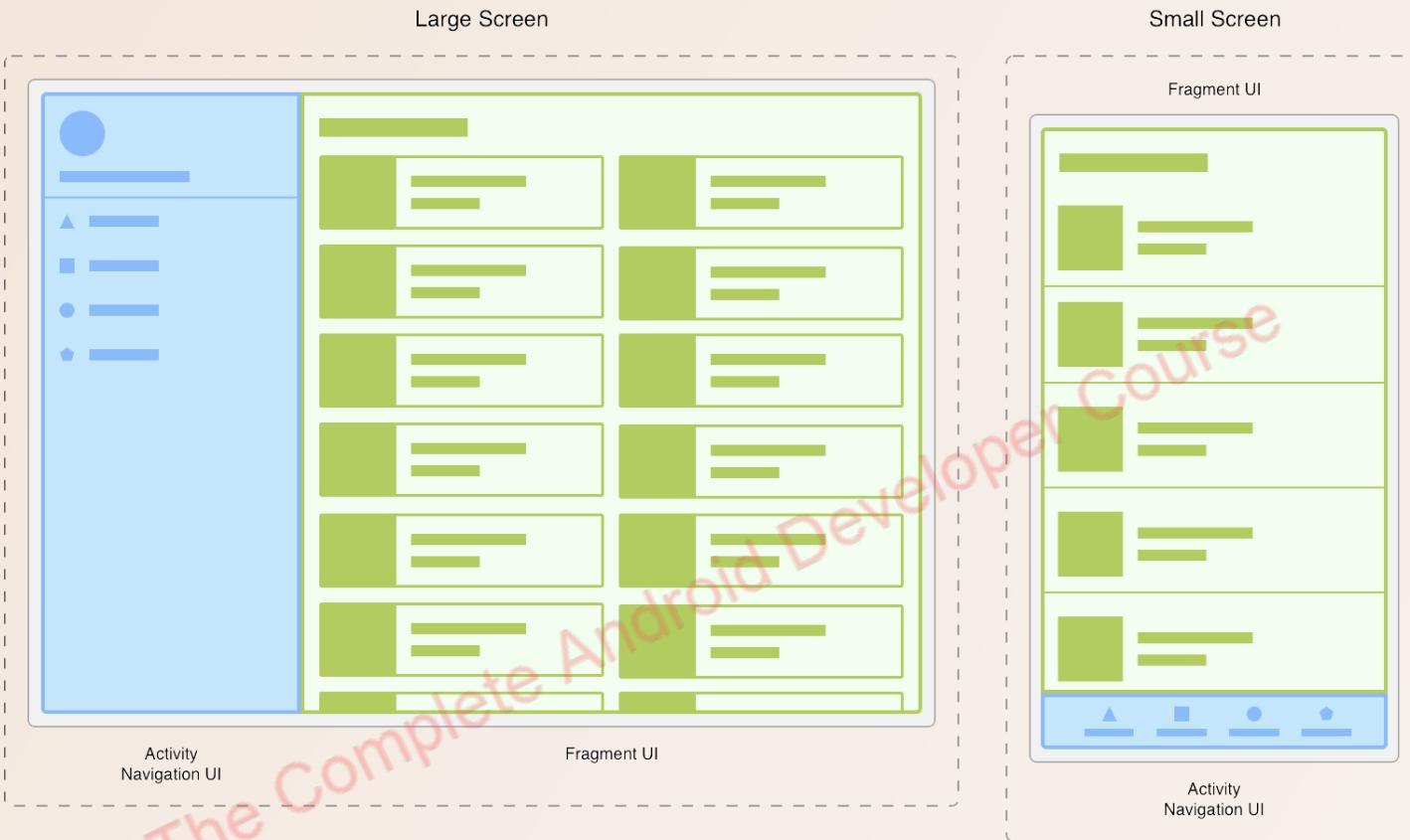


Fragments

A Fragment represents a reusable portion of your app's UI.

A fragment defines and manages its own layout, has its own lifecycle, and can handle its own input events.

Fragments can't live on their own. They must be hosted by an activity or another fragment.

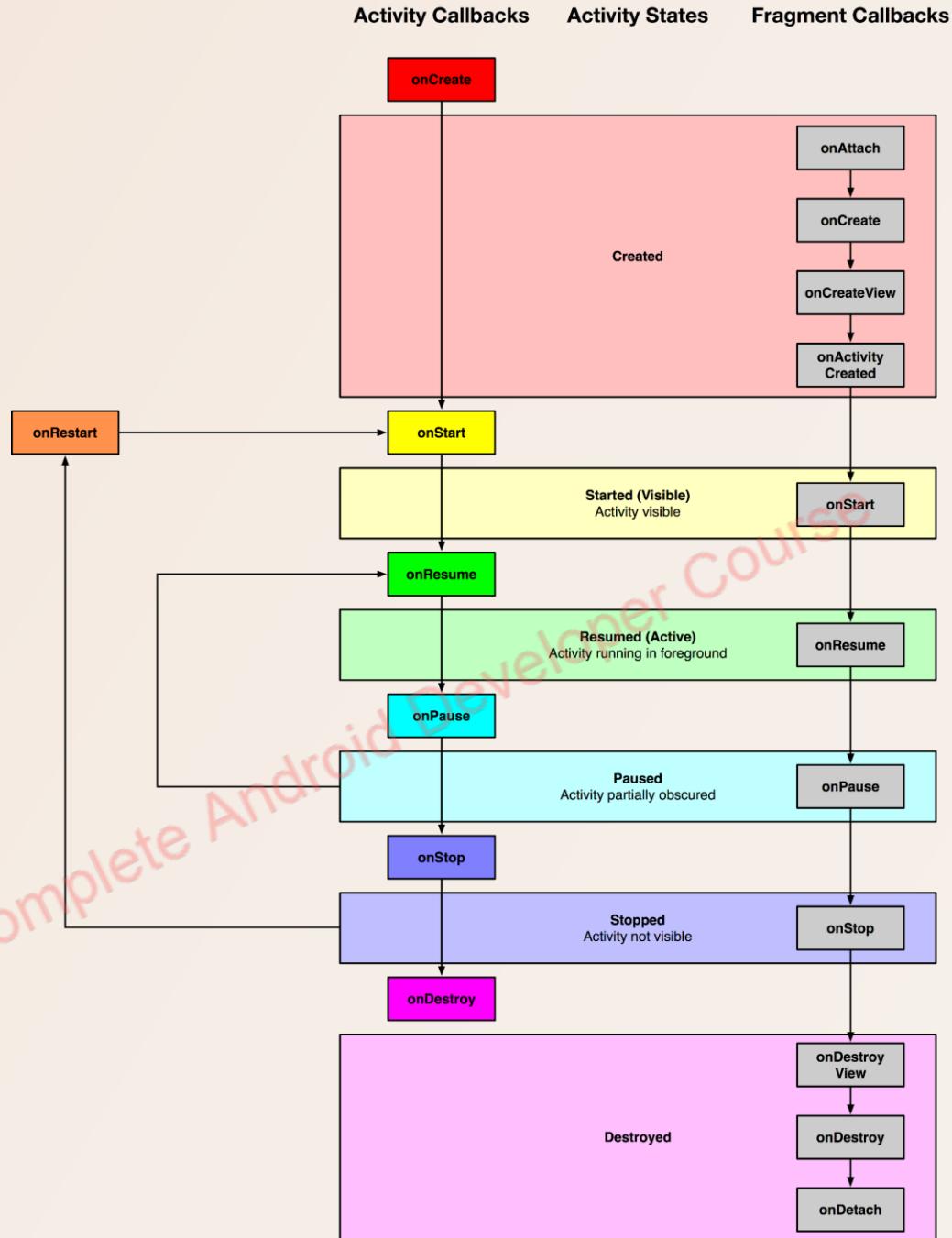


Fragment's Life Cycle

Each Fragment instance has its own lifecycle. When a user navigates and interacts with your app, your fragments transition through various states in their lifecycle as they are added, removed, and enter or exit the screen.

The Fragment class includes callback methods that correspond to each of the changes in a fragment's lifecycle.

These include `onCreate()`, `onStart()`, `onResume()`, `onPause()`, `onStop()`, and `onDestroy()`.

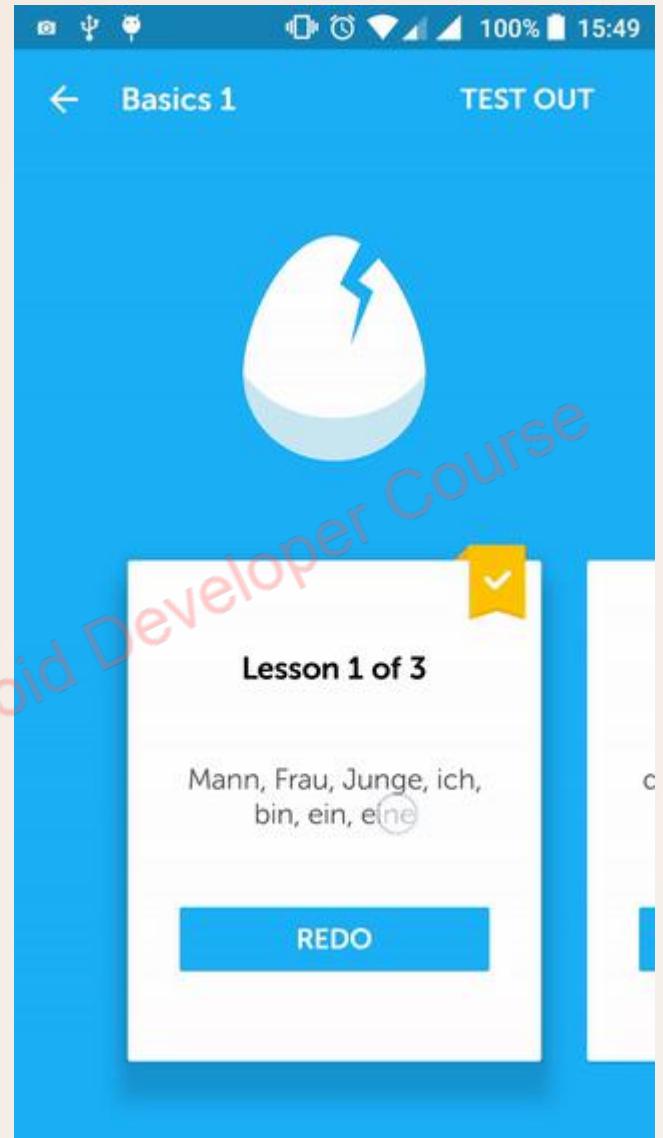


View Pager

ViewPager is a layout manager that allows the user to flip left and right through pages of data.

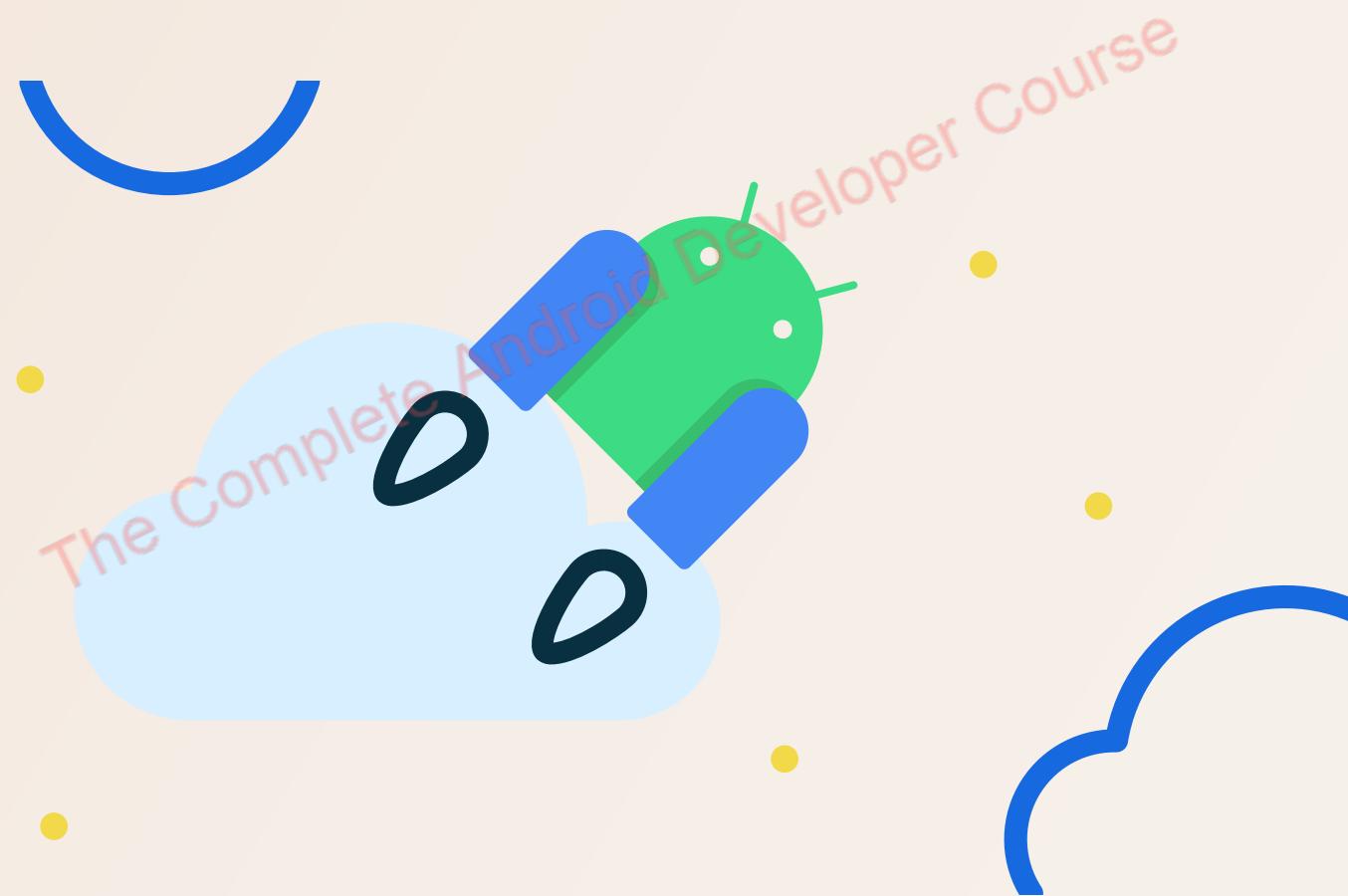
It is mostly found in apps like Youtube, Snapchat where the user shifts right – left to switch to a screen. Instead of using activities **fragments** are used.

It is also used to guide the user through the app when the user launches the app for the first time.



Android Jetpack

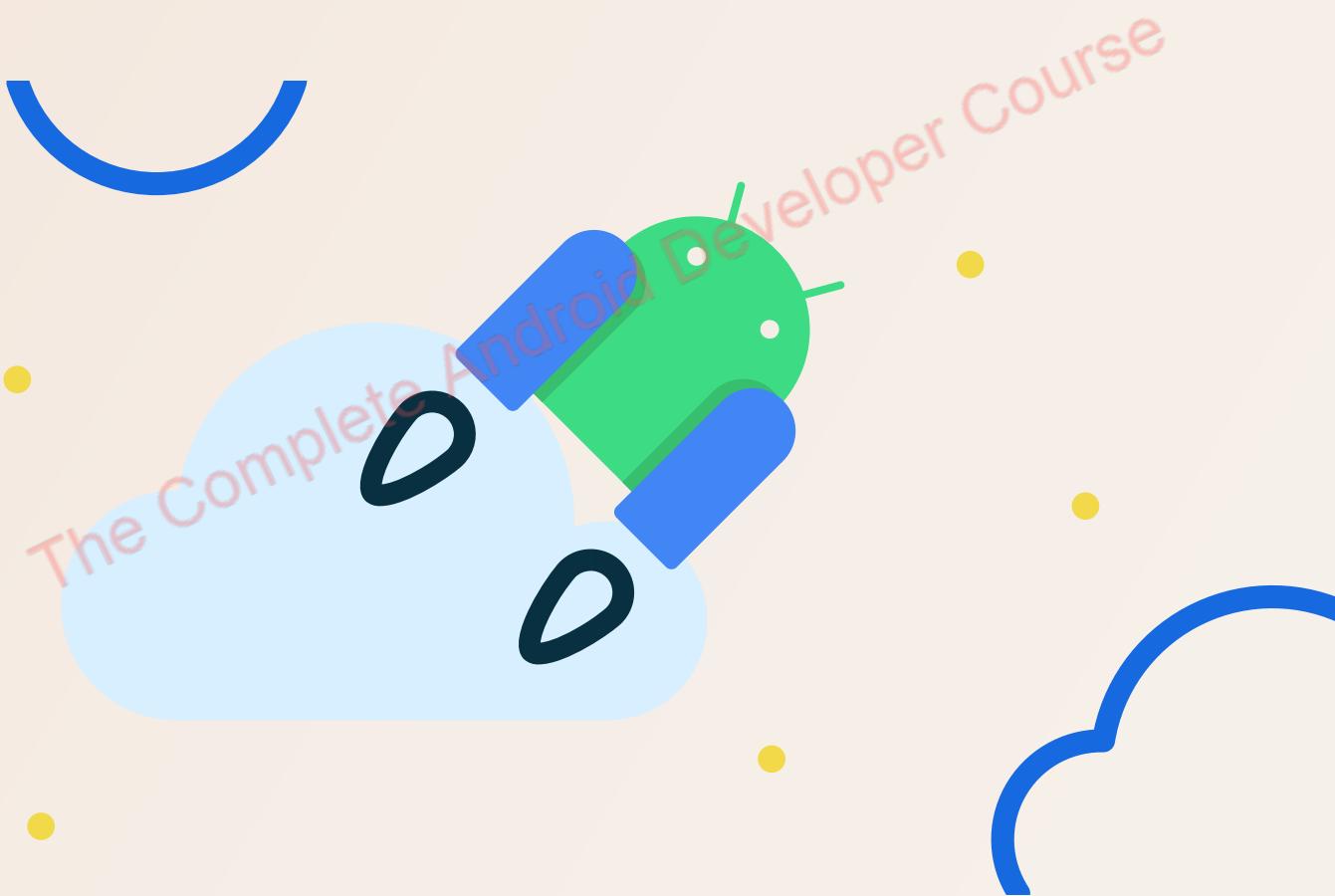
Jetpack is a suite of libraries to help developers follow best practices, reduce boilerplate code, and write code that works consistently across Android versions and devices so that developers can focus on the code they care about.



The Complete Android Developer Course

Why Android Jetpack?

Android Jetpack helps solve major problems such as managing activity life cycles, configuration changes, and preventing memory leaks.



Jetpack Components

Android Jetpack components bring together the existing support library and architecture components and define them into four categories.



Architecture



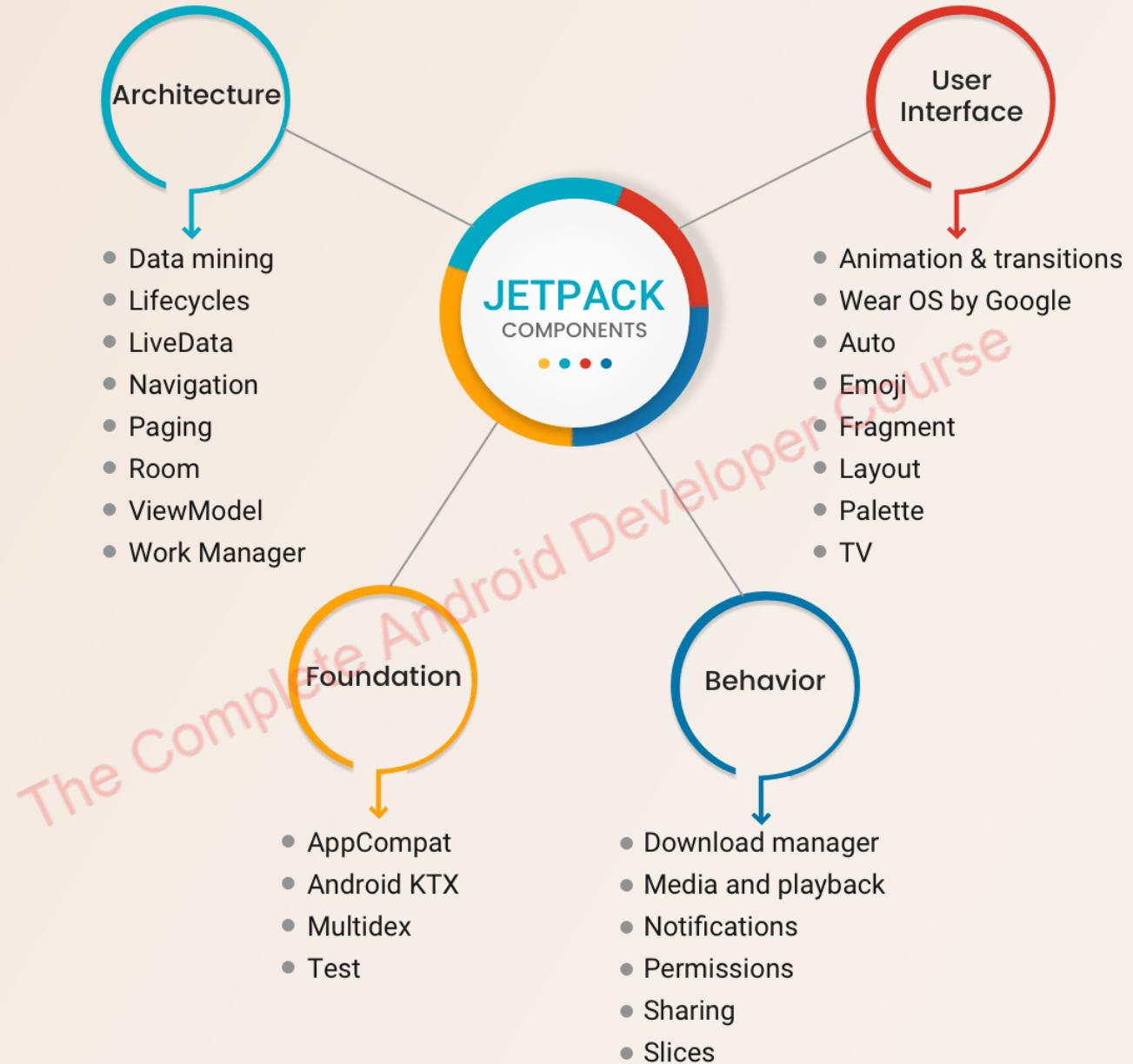
User Interface



Foundation



Behavior



findViewById()

Finds the first descendant view with the given ID, the view itself if the ID matches getId(), or null if the ID is invalid (< 0) or there is no matching view in the hierarchy.

Every time we use findViewById() to get reference to a view, Android system must go through the view hierarchy and find it at runtime. But what about large apps with hundreds of views and layouts????

Modern mobiles nowadays, have a refresh rate 60 HZ per 1 second (1000 milliseconds), so Android system will create our layout with all its views every 16.667 milliseconds.

What about Mobiles with 90 Hz, 120 Hz, 144 Hz ????

The diagram illustrates the process of finding a view by ID. It starts with an XML layout snippet at the top left, which defines a button with ID @+id btn_submit. A purple arrow points from this XML to a Java code block at the bottom right. The Java code uses findViewById to get a reference to the button, and then sets an onClickListener. Two callout boxes with arrows point to specific parts of the code: one points to the findViewById line with the text "Get the reference to the View", and another points to the onClickListener line with the text "Use the reference to access its properties".

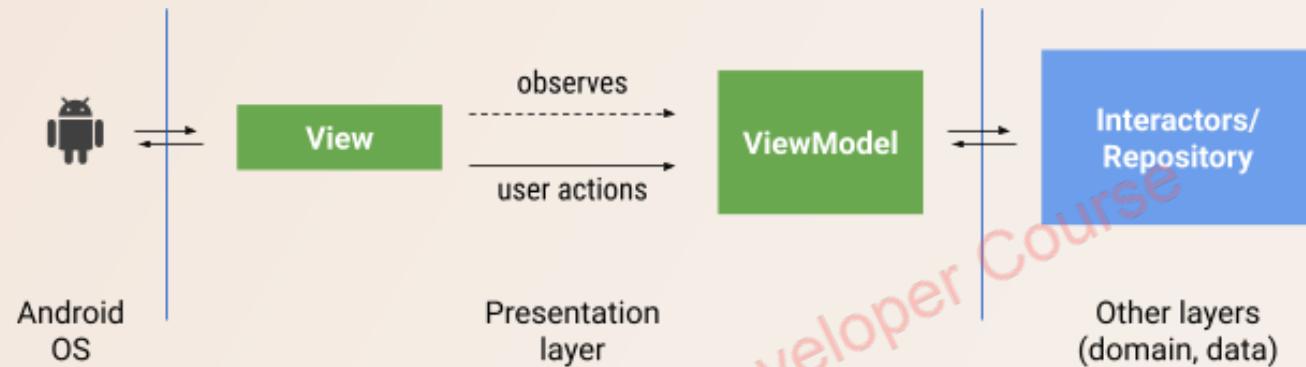
```
<Button  
    android:id="@+id/btn_submit"  
    android:text="Submit"  
    android:textAllCaps="false"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content" />  
  
var btn_submit = findViewById(R.id.btn_submit) as Button  
  
btn_submit.setOnClickListener {
```

Data Binding

The Data Binding Library is a support library that allows you to bind UI components in your layouts to data sources in your app using a declarative format rather than programmatically.

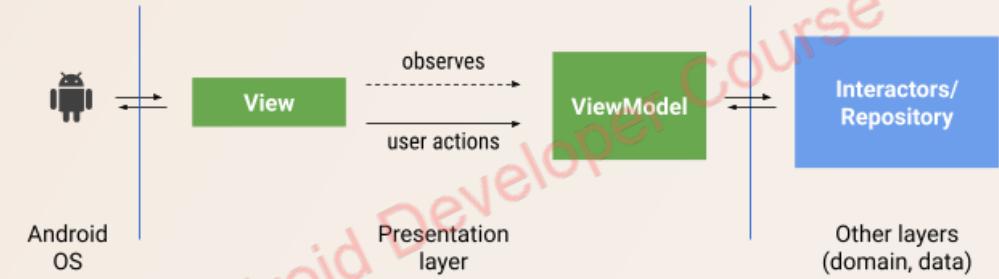
Data binding is the process of integrating views in an XML layout with data objects. The Data Binding Library is responsible for generating the classes required for this procedure.

Using Data binding, a binding object will be created that contains a reference to each view of a layout, so the android system will not go and search the views by their ids again and again....



Why Data Binding?

- You can reduce findViewById calls and enhance your app's performance
- Helps get rid of memory leaks or nullPointerExceptions
- Uses declarative layout, which is more adaptable
- Supercharges developer productivity by writing error-free, shorter, simpler-to-understand, and more maintainable code
- Data and Views are separated from each other
- The compiler verifies types during compile time and displays errors if and when you attempt to assign the incorrect type to a variable, thanks to type-safety



Using Data Binding

The name of the layout we wrapped with
`<layout></layout>` in `activity_main`.

Using that, android data binding library will create a
binding object with the name of: `ActivityMainBinding`

activity_main

ActivityMain Binding

The Complete Android Developer Course



Quadratic Equations

Quadratic equations are algebraic equations that have the form $ax^2+bx+c=0$

Considering this form, the discriminant of a quadratic equation is the value b^2-4ac . This value goes inside the square root of the general quadratic formula and determines the type of solutions that we will have.

$$ax^2 + bx + c = 0$$

$$D = b^2 - 4ac$$

$$ax^2 + bx + c = 0$$

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

- When $D > 0$, the quadratic equation has two real roots.
- When $D < 0$, the quadratic equation has no real roots.
- When $D = 0$, the quadratic equation has a repeated root.

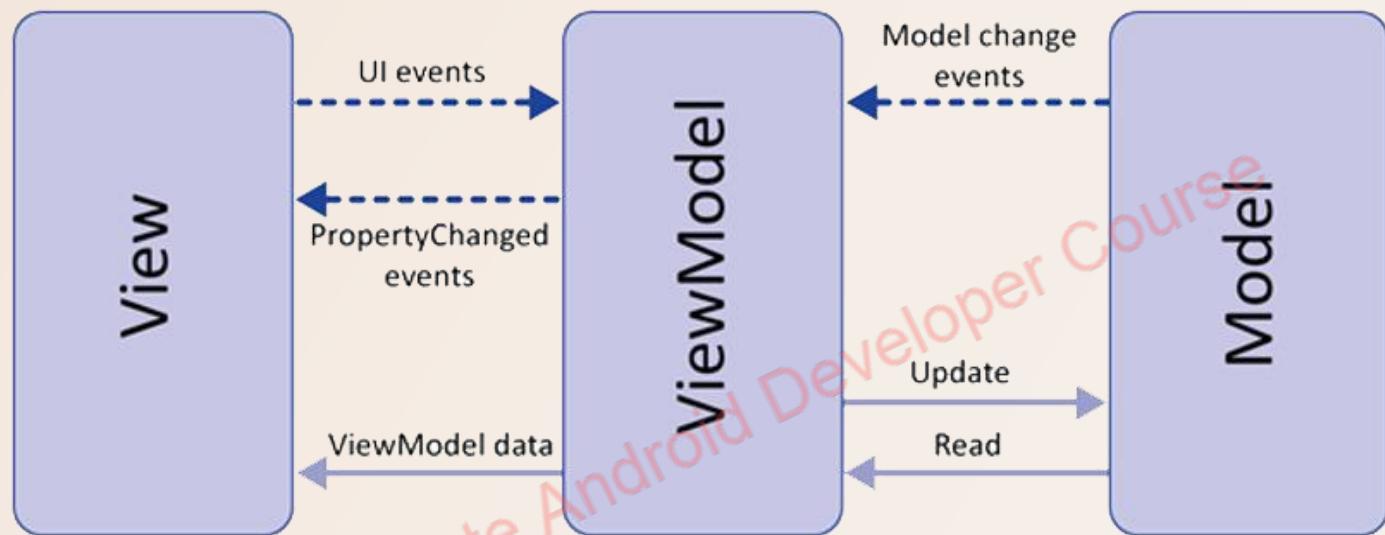
We can understand this better if we remember that the general quadratic formula, which allows us to solve any quadratic equation, is the following:

ViewModel

ViewModel is a class that is responsible for preparing and managing the data for an Activity or a Fragment. It also handles the communication of the Activity / Fragment with the rest of the application.

A ViewModel is always created in association with a scope (a fragment or an activity) and will be retained as long as the scope is alive. E.g. if it is an Activity, until it is finished.

In other words, this means that a ViewModel will not be destroyed if its owner is destroyed for a configuration change (e.g. rotation).

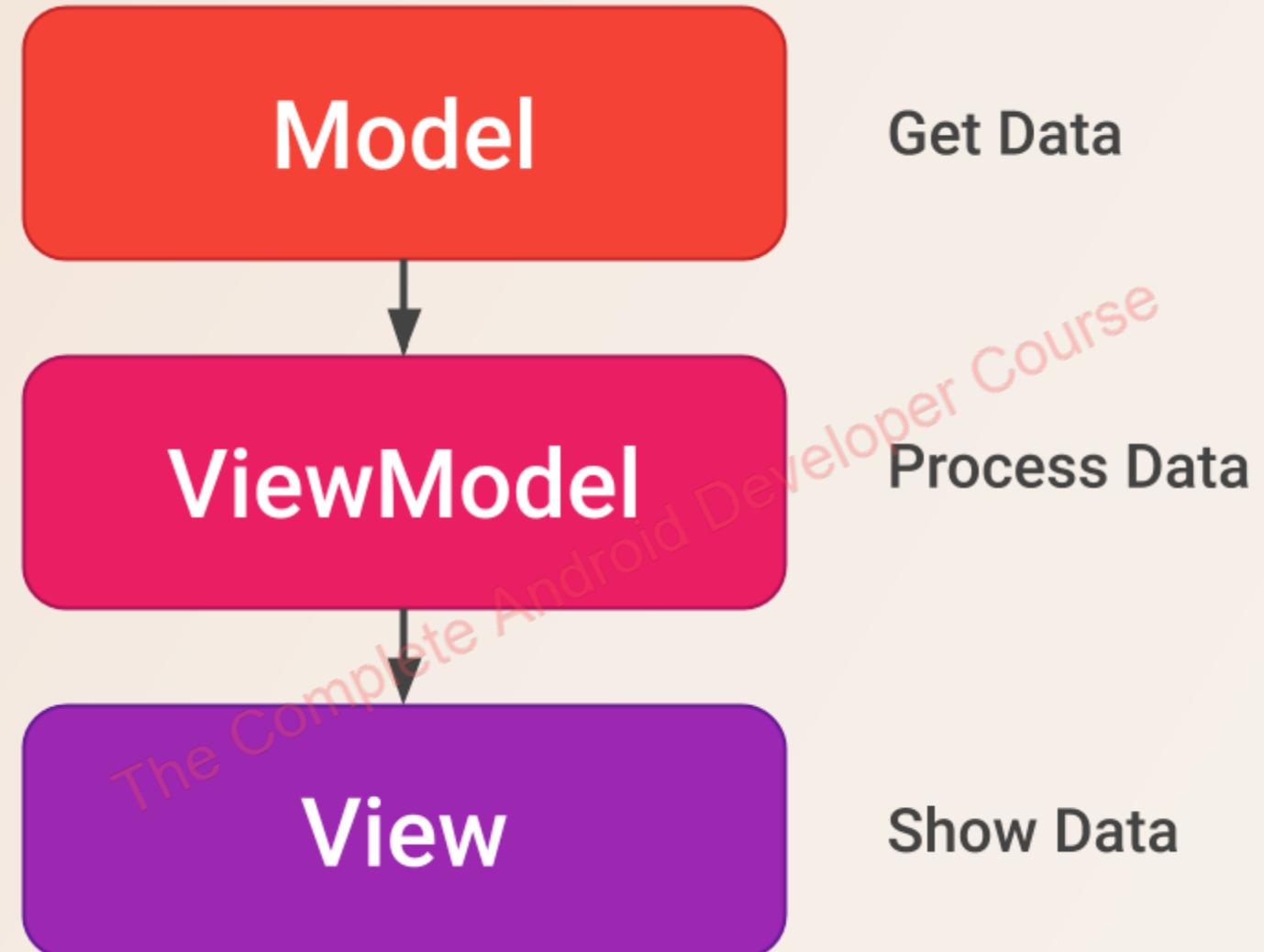


View Model

The ViewModel class is a business logic or screen level state holder. It exposes state to the UI and encapsulates related business logic.

Its principal advantage is that it caches state and persists it through configuration changes.

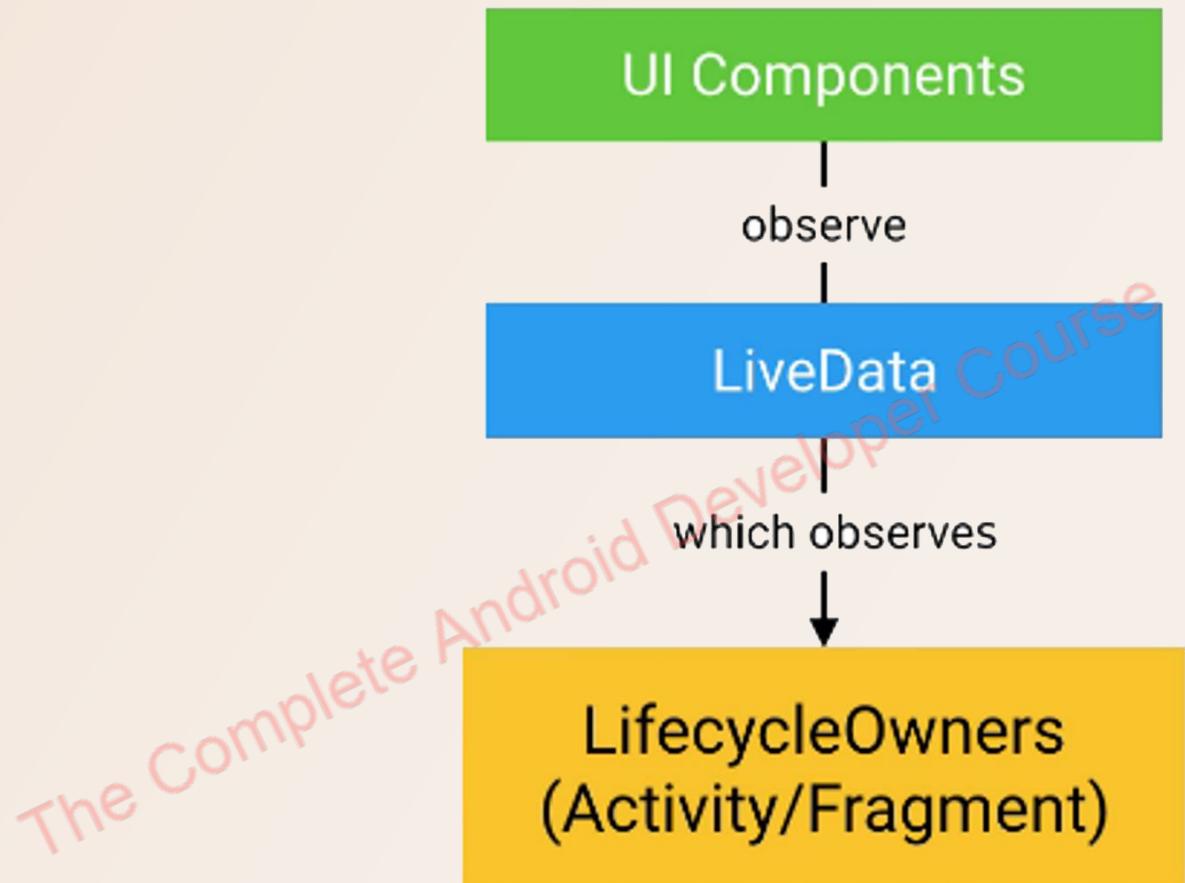
This means that your UI doesn't have to fetch data again when navigating between activities, or following configuration changes, such as when rotating the screen.



LiveData

LiveData is an observable data holder class. Unlike a regular observable, LiveData is lifecycle-aware, meaning it respects the lifecycle of other app components, such as activities, fragments, or services.

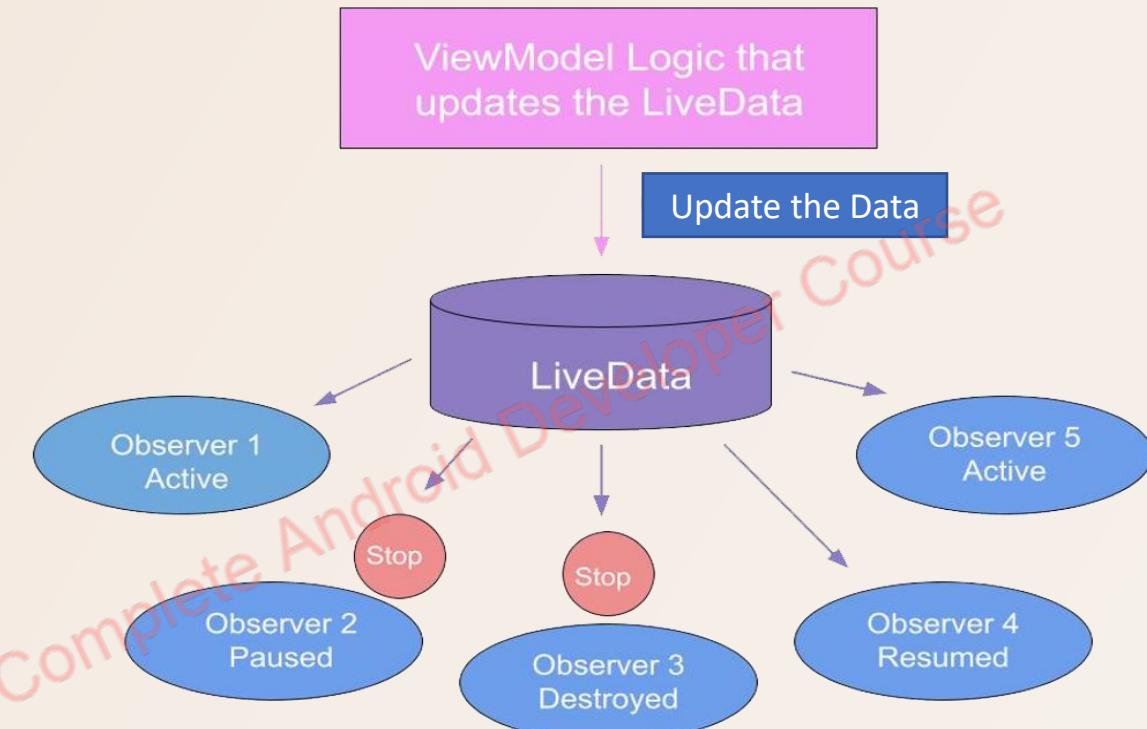
This awareness ensures LiveData only updates app component observers that are in an active lifecycle state.



LiveData

If the lifecycle status is STARTED or RESUMED, LiveData considers an observer to be in an **active state**. Only active observers are notified of lifecycle updates by LiveData.

- It eliminates memory leaks caused by the multiple callbacks that send results to the UI thread, ensuring that the UI is always up to date.
- It de-couples tight integration between data, mediator, and the UI to avoid crashed activities.
- UI components continuously monitor relevant data, and LiveData manages all these tasks automatically as the relevant lifecycle status changes.
- If the activity or fragment is recreated due to configuration changes such as device rotation (Portrait, Landscape), it immediately receives the latest information from LiveData.
- Extended Live Data wraps system services so that they can be shared within the app.

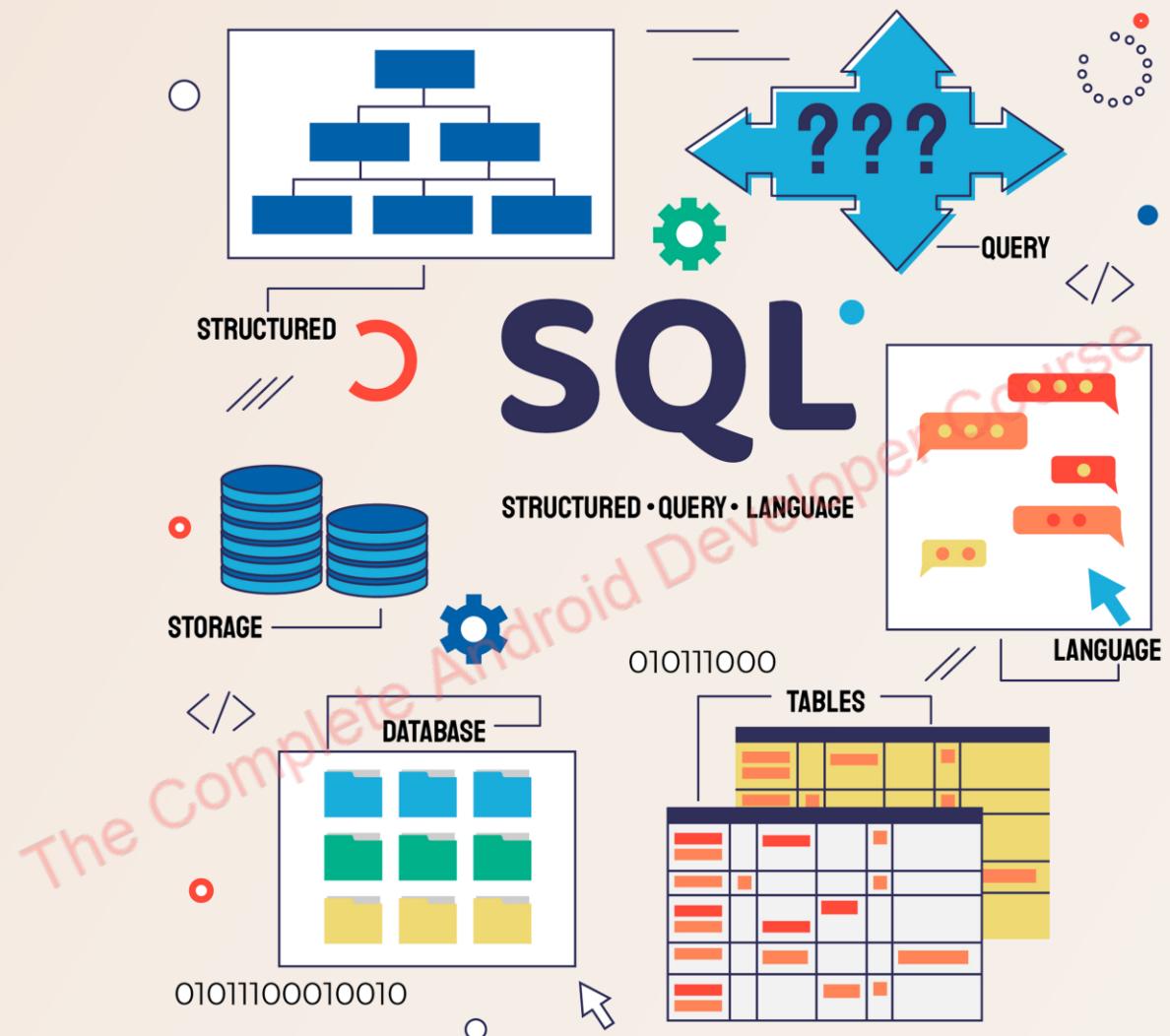


SQLite Database

SQLite Database is an open-source database provided in Android which is used to store data inside the user's device in the form of a Text file.

We can perform so many operations on this data such as adding new data, updating, reading, and deleting this data.

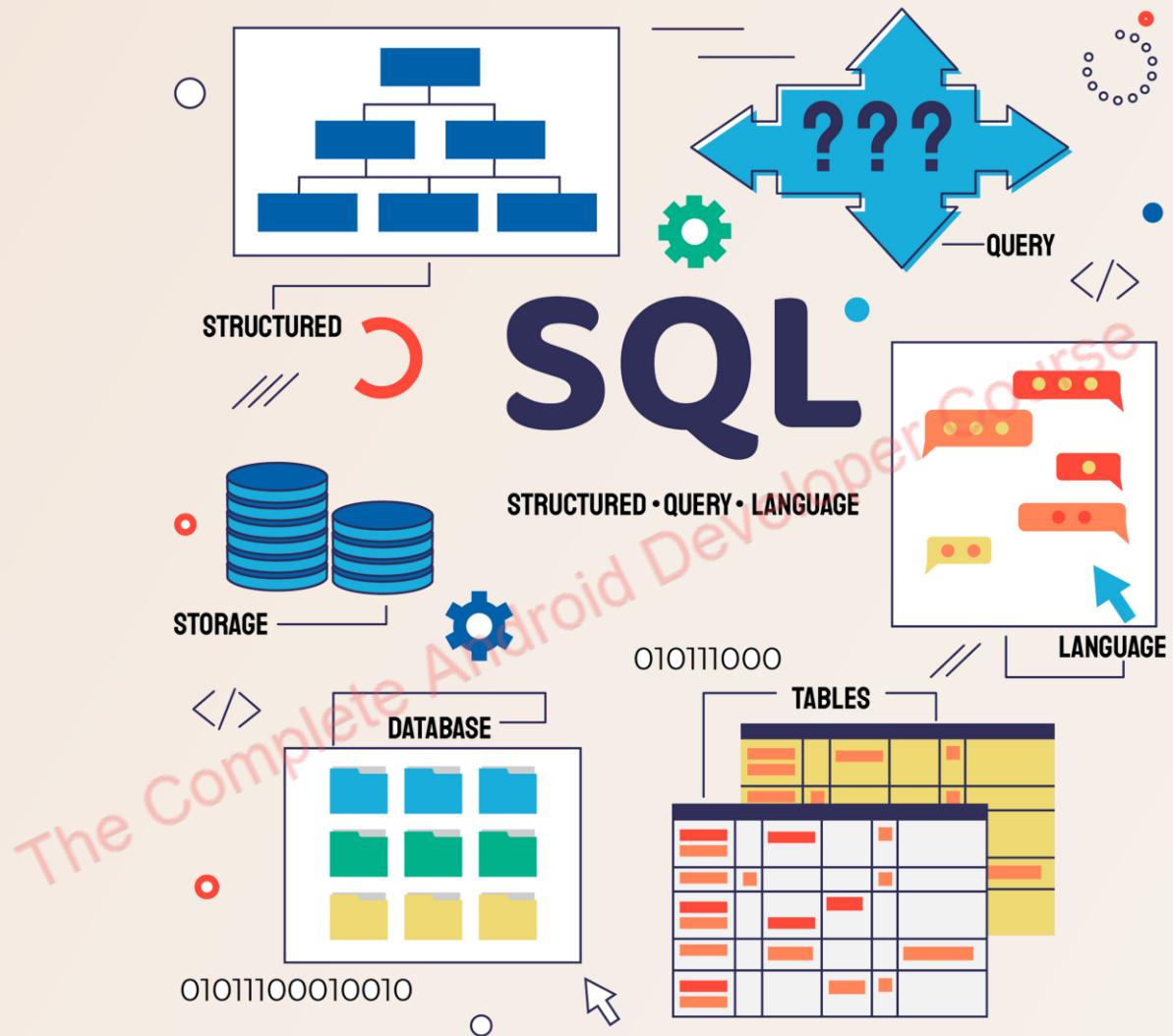
SQLite is an offline database that is locally stored in the user's device and we do not have to create any connection to connect to this database.



ROOM Database

Room is an ORM, Object Relational Mapping library. In other words, Room will map our database objects to Java objects.

Room provides an abstraction layer over SQLite to allow fluent database access while harnessing the full power of SQLite.



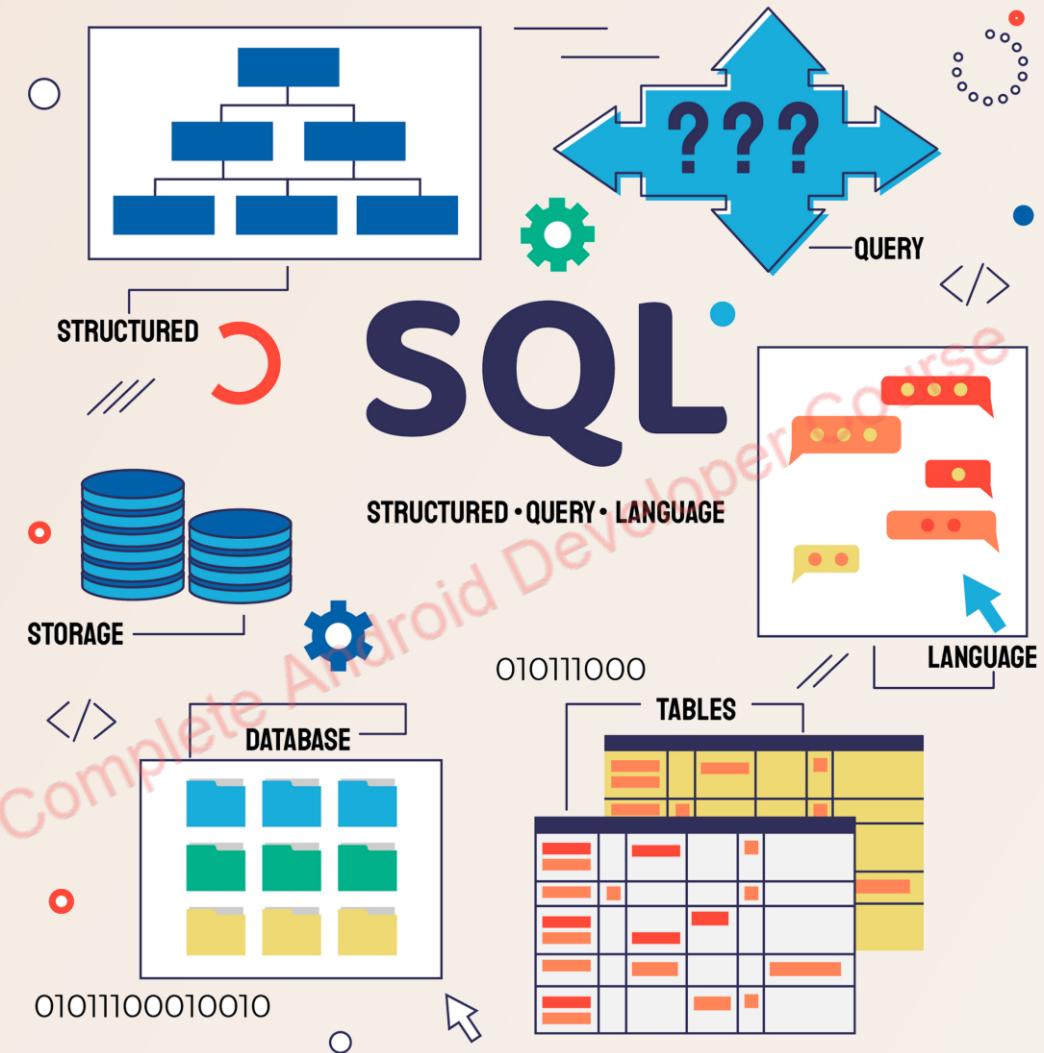
SQLite vs ROOM Database

In the case of SQLite, There is no compile-time verification of raw SQLite queries. But in Room, there is SQL validation at compile time.

You need to use lots of boilerplate code to convert between SQL queries and Java data objects. But, Room maps our database objects to Java Object without boilerplate code.

As your schema changes, you need to update the affected SQL queries manually. Room solves this problem.

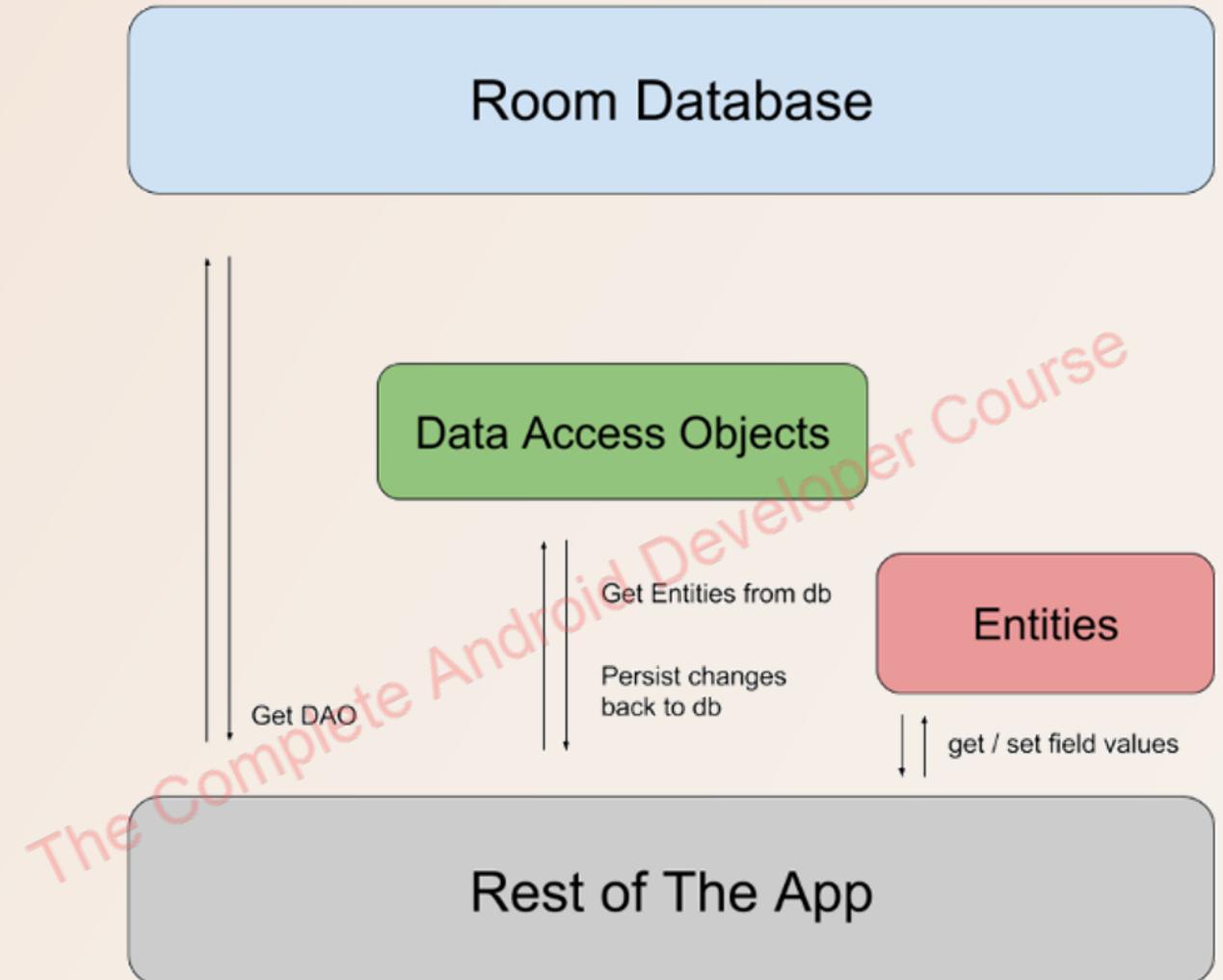
Room is built to work with LiveData and RxJava for data observation, while SQLite does not.



ROOM Database

Room has three main components of Room DB :

- Entity
- Dao
- Database

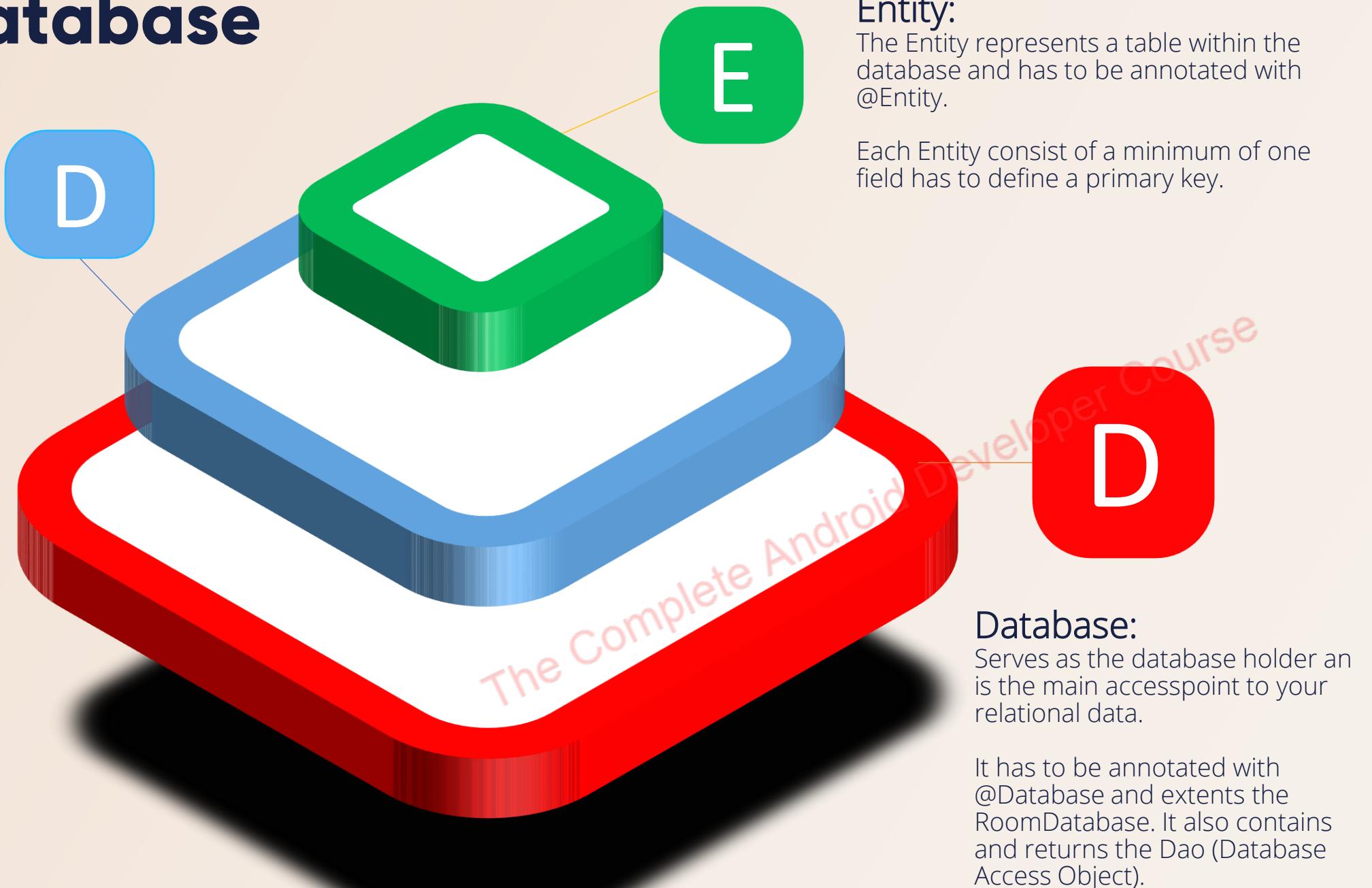


ROOM Database

DAO (Database Access Object):

In Room you use data access objects to access and manage your data. The DAO is the main component of Room and includes methods that offer access to your app's database. It has to be annotated with @Dao.

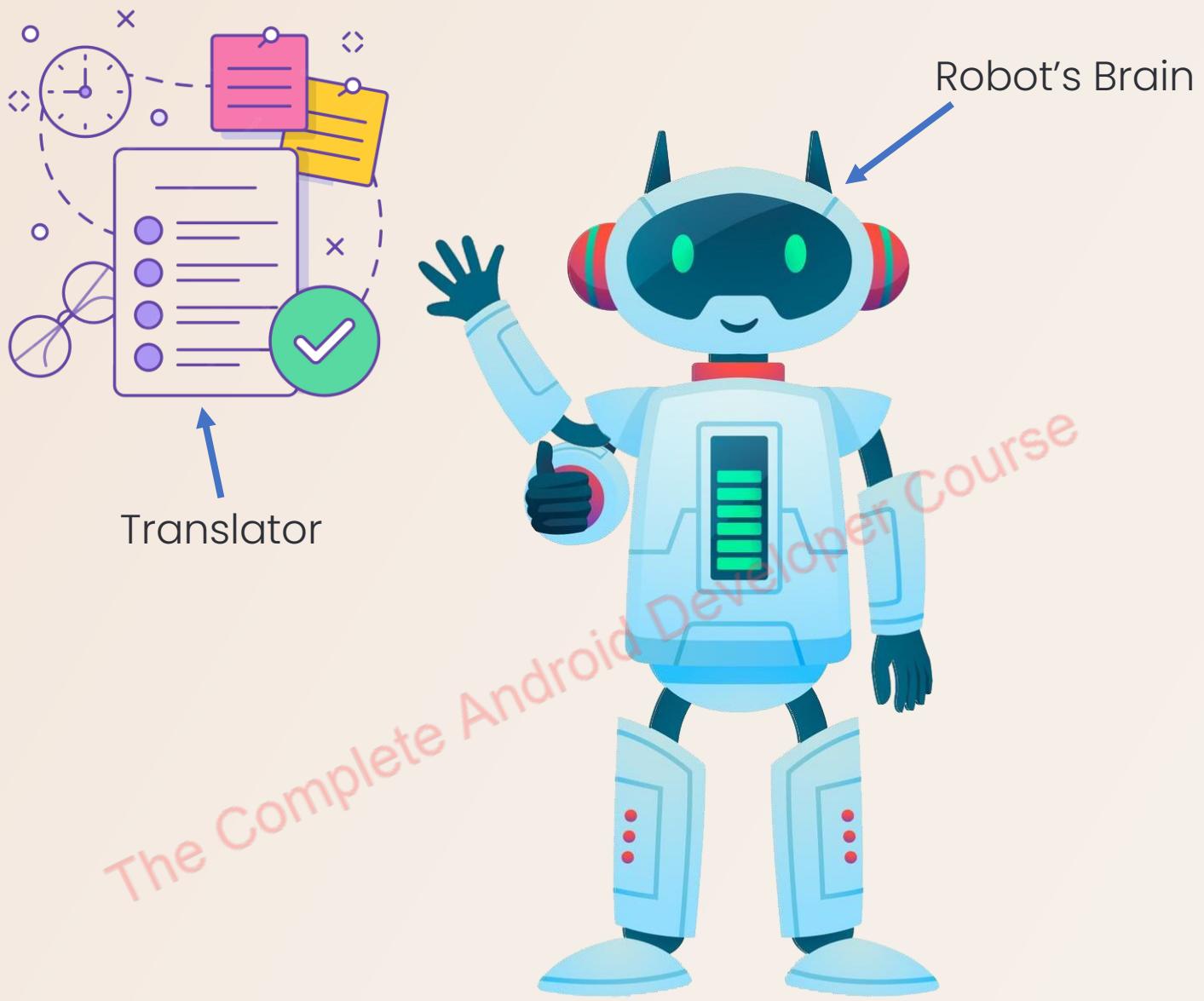
DAOs are used instead of query builders and let you separate different components of your database e.g. current data and statistics, which allows you to easily test your database.



What's MVVM?

Here's how MVVM works with our robot:

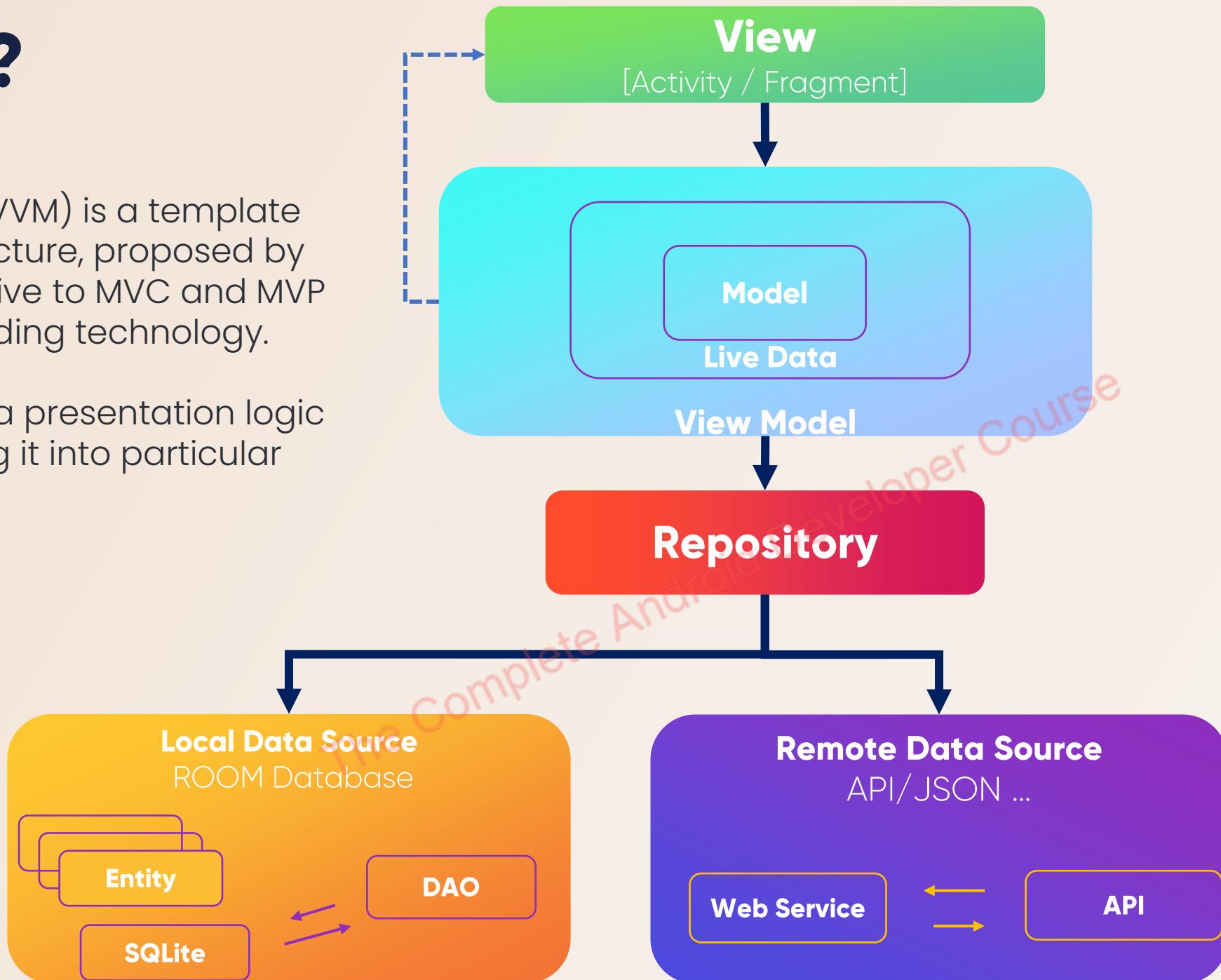
- Model (Robot's Brain)
- View (Remote Control)
- View Model (Translator)



What's MVVM?

Model-View-ViewModel (ie MVVM) is a template of a client application architecture, proposed by John Gossman as an alternative to MVC and MVP patterns when using Data Binding technology.

Its concept is to separate data presentation logic from business logic by moving it into particular class for a clear distinction.



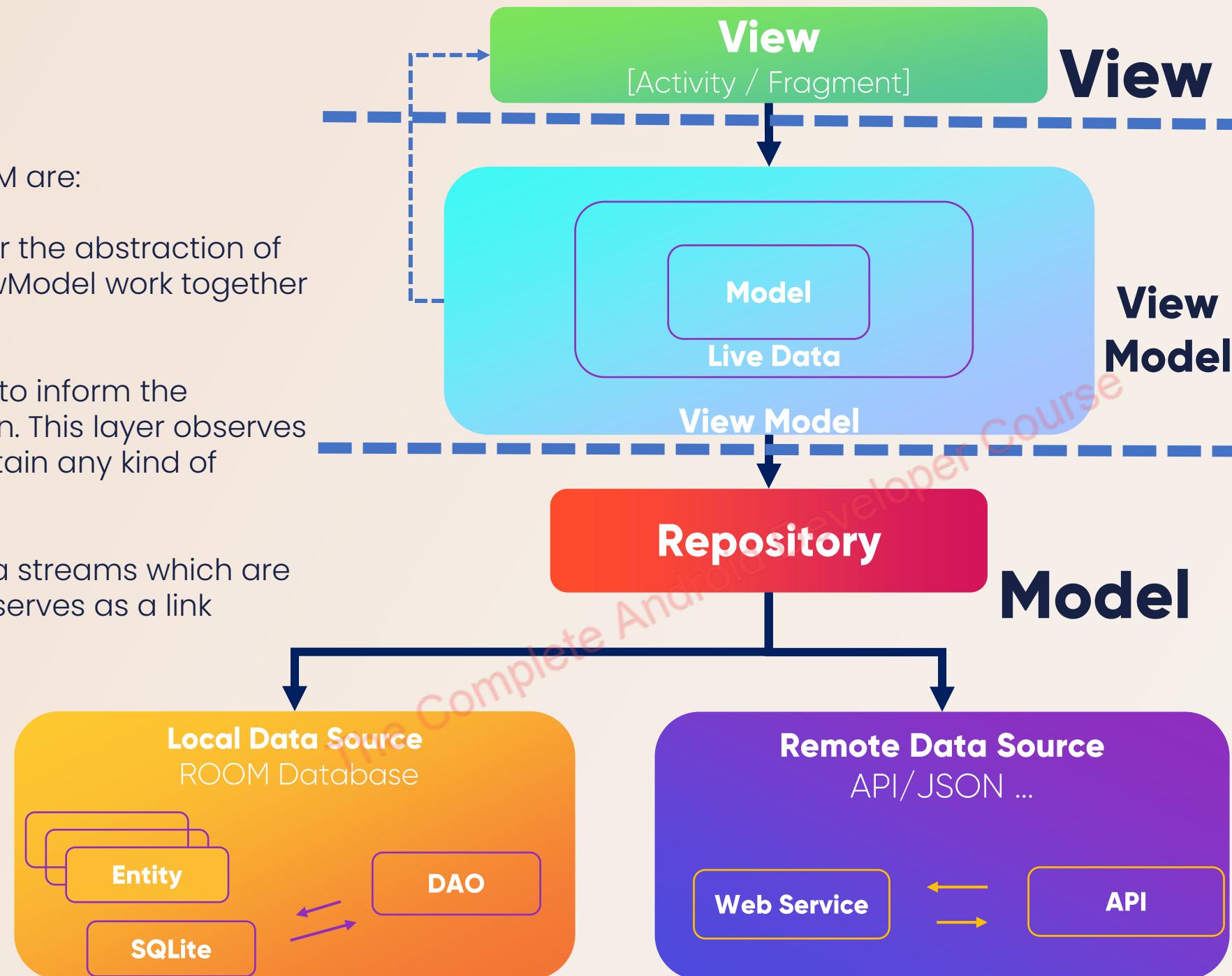
MVVM Layers

The separate code layers of MVVM are:

Model: This layer is responsible for the abstraction of the data sources. Model and ViewModel work together to get and save the data.

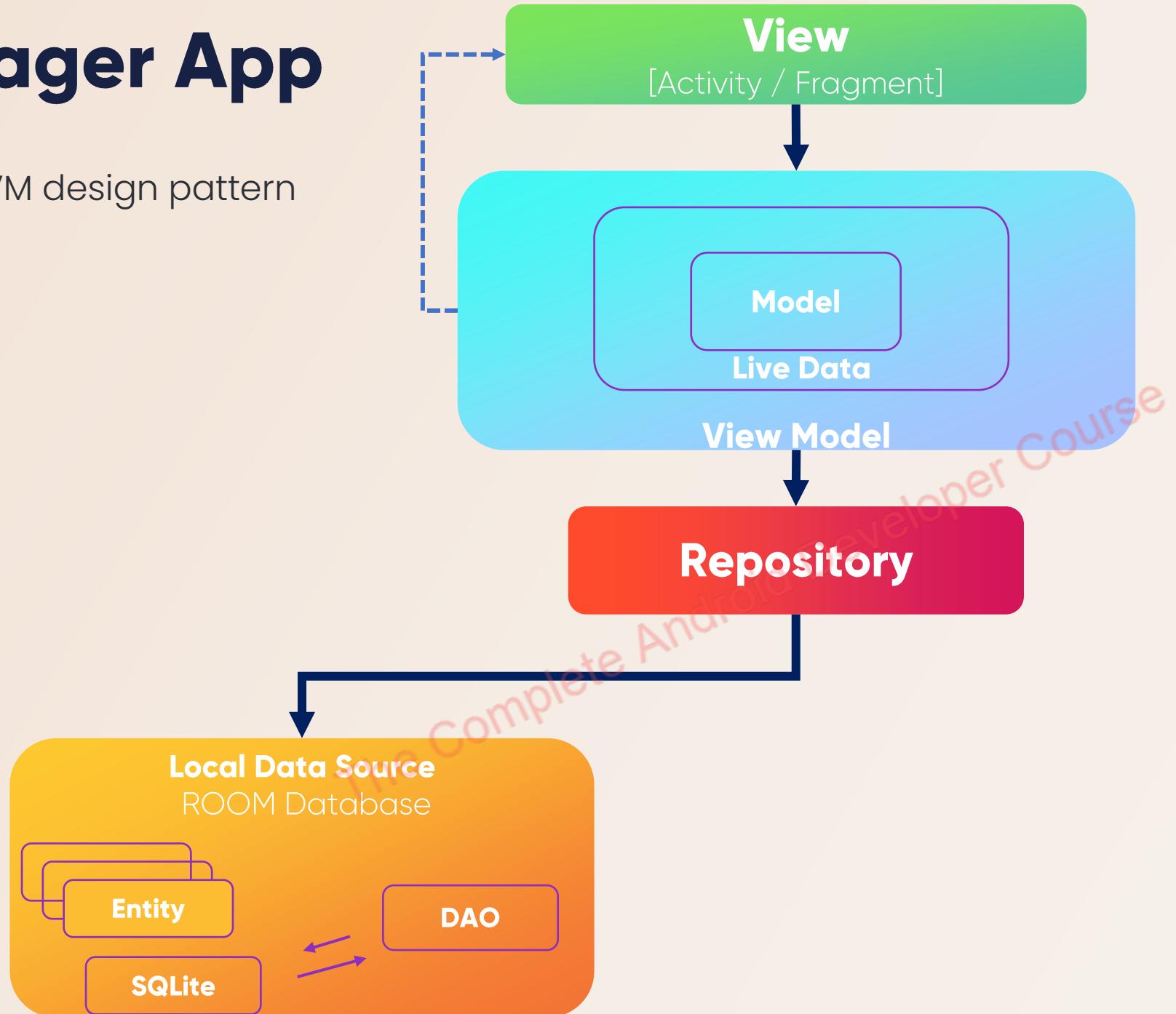
View: The purpose of this layer is to inform the ViewModel about the user's action. This layer observes the ViewModel and does not contain any kind of application logic.

ViewModel: It exposes those data streams which are relevant to the View. Moreover, it serves as a link between the Model and the View.



Contacts Manager App

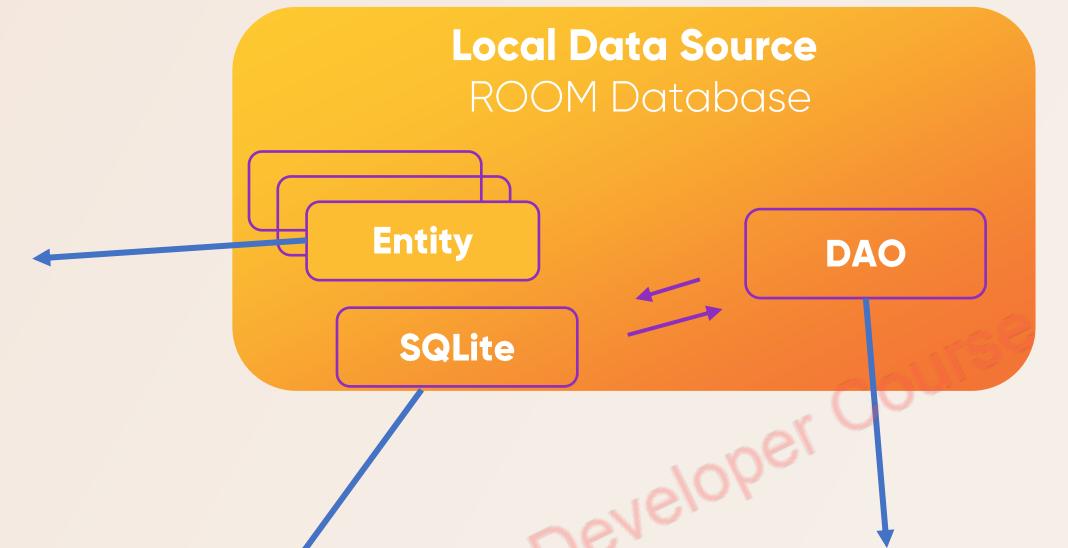
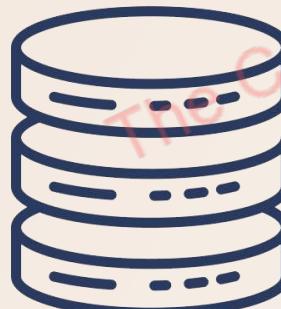
In this app, we'll follow the MVVM design pattern with ROOM database.



Contacts Manager App – [ROOM DB]

contact_id	contact_name	contact_email
1	John	john@gmail.com
2	Sara	sara@gmail.com
3	Ali	ali@gmail.com

Entity: contacts_table



DAO

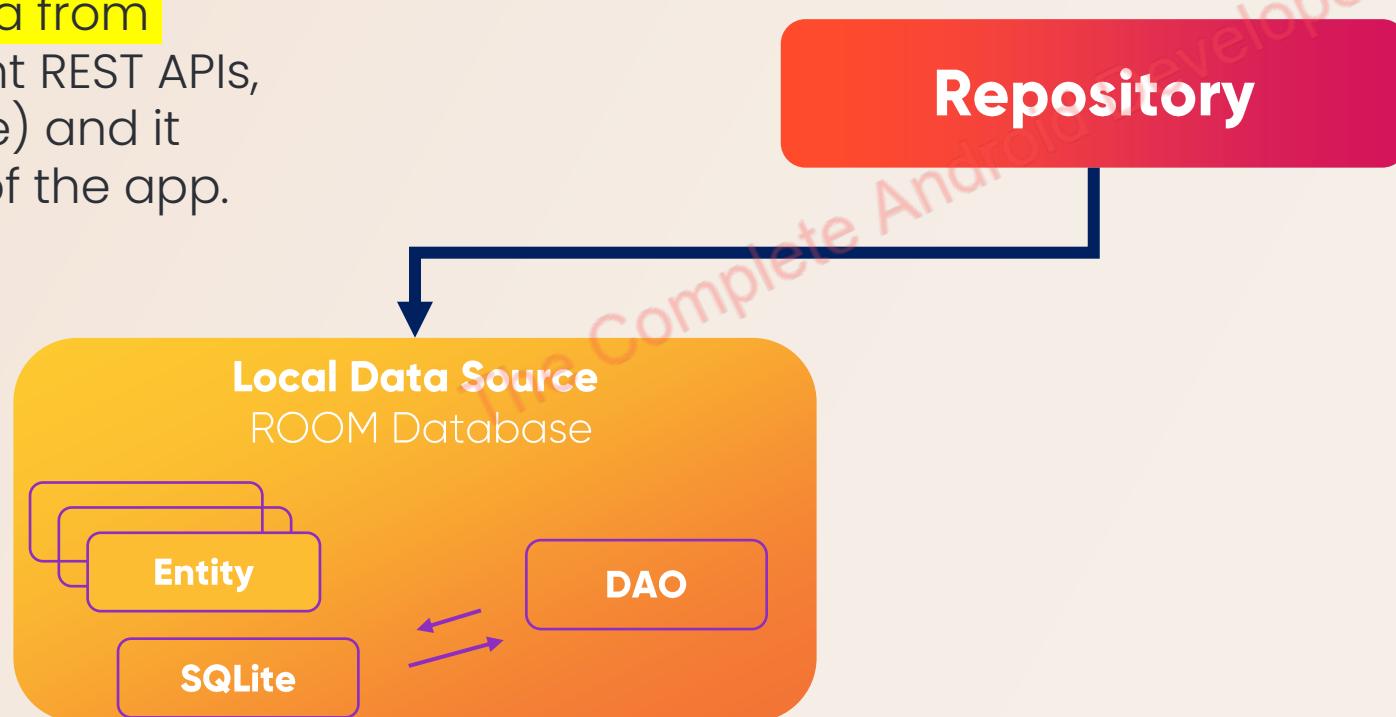
- Insert
- Delete
- Update
- Select
- Other queries

Contacts Manager App – [Repository]

Repository is a class that is mainly used to manage multiple sources of data.

The repository class isolates the data sources from the rest of the app and provides a clean API for data access to the rest of the app.

the Repository can gather data from different data sources (different REST APIs, cache, local database storage) and it provides this data to the rest of the app.



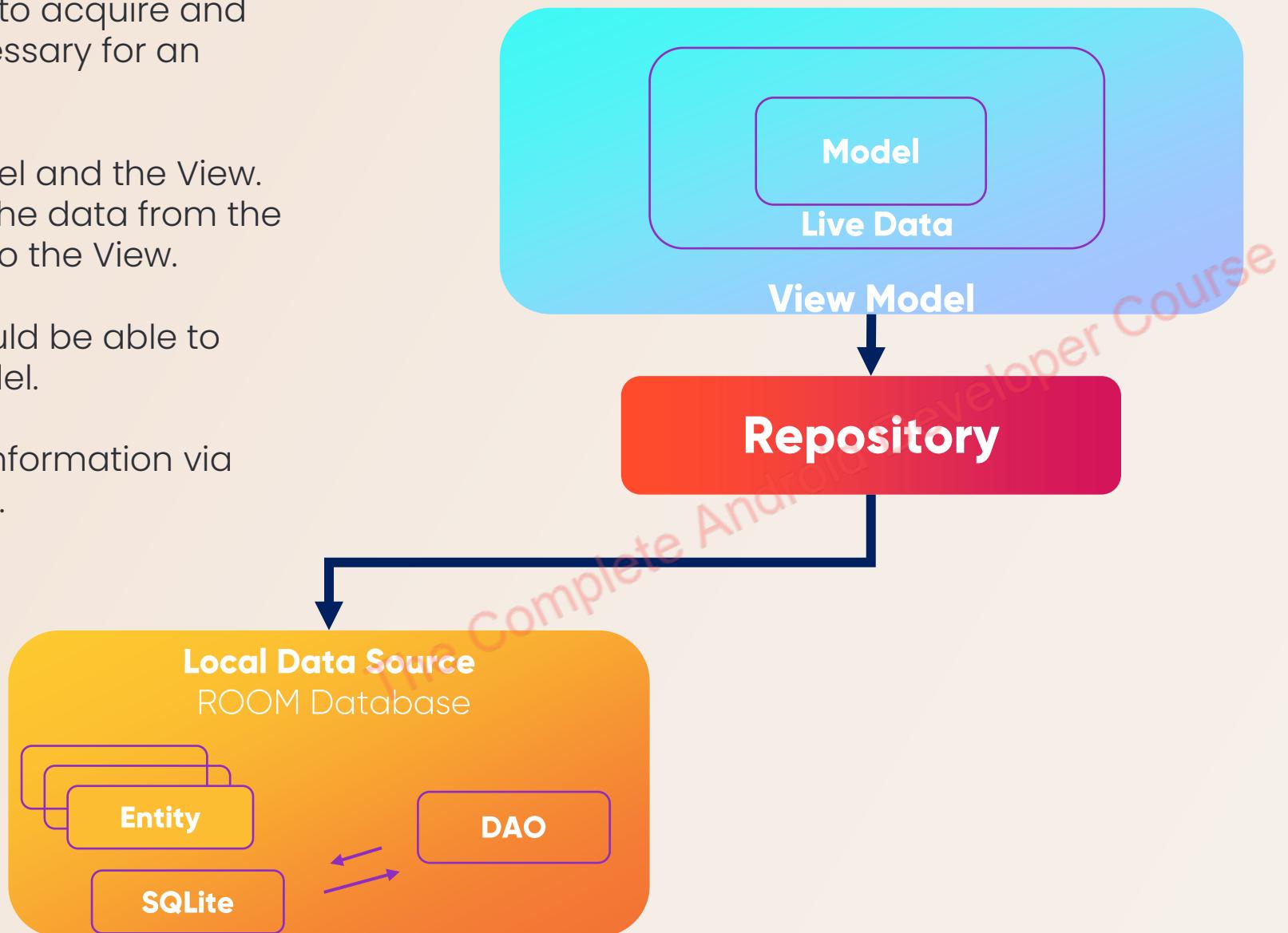
Contacts Manager App – [View Model]

The purpose of the ViewModel is to acquire and keep the information that is necessary for an Activity or a Fragment.

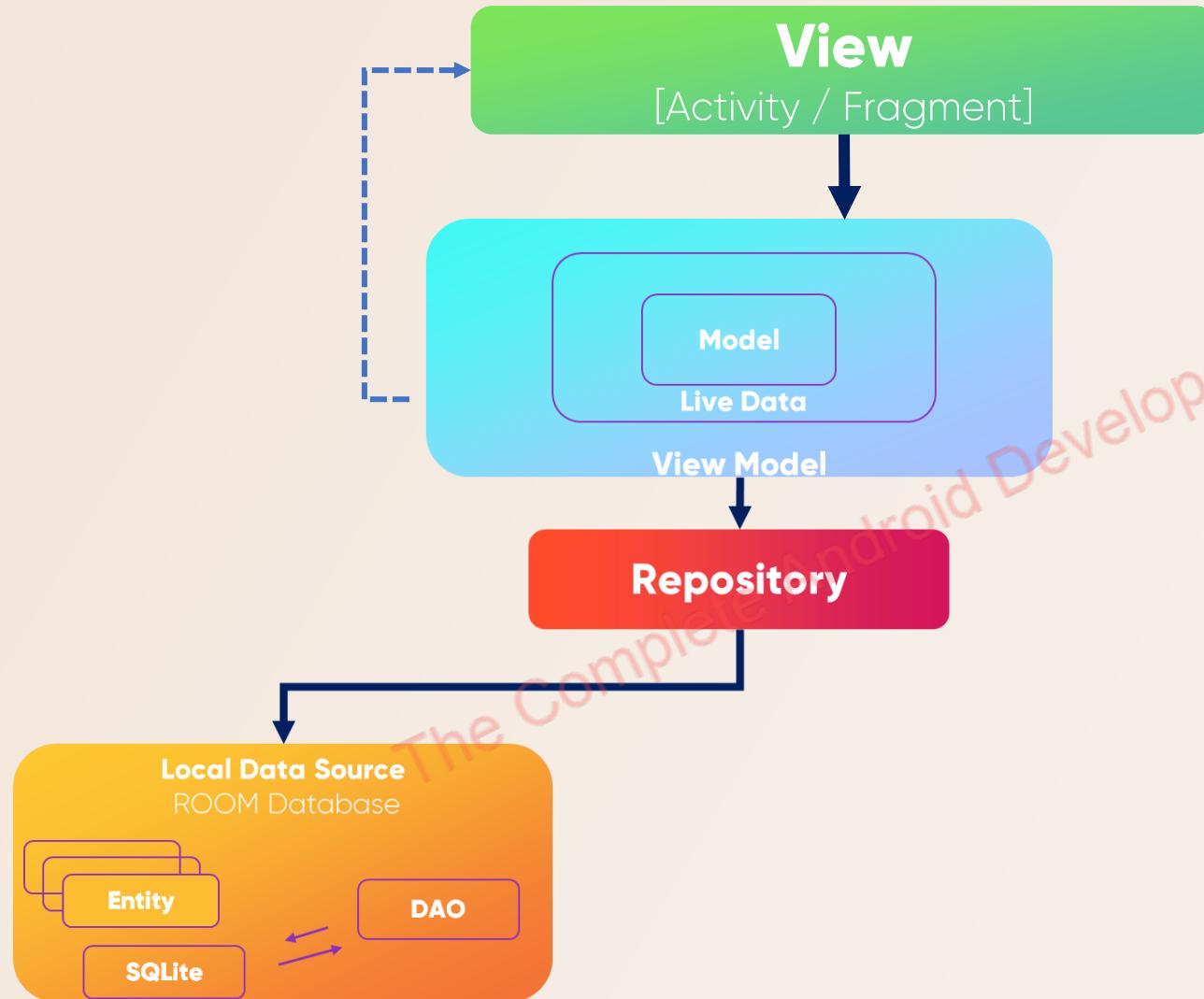
It acts as a link between the Model and the View. It's responsible for transforming the data from the Model. It provides data streams to the View.

The Activity or the Fragment should be able to observe changes in the ViewModel.

ViewModels usually expose this information via LiveData or Android Data Binding.



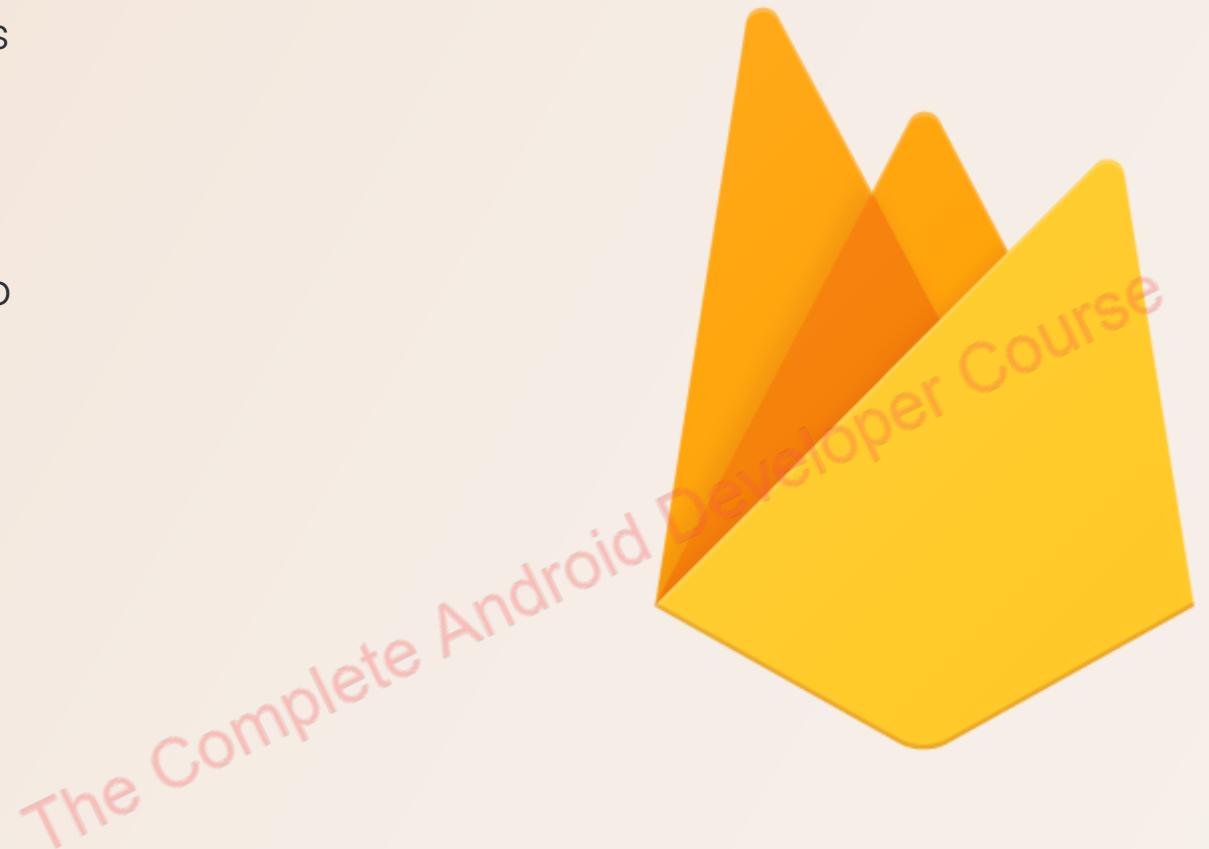
Contacts Manager App – [View]



FIREBASE

Firebase is a powerful and comprehensive platform developed by Google that offers a wide range of tools and services for building and managing mobile and web applications.

Firebase for Android is a popular choice among developers because it simplifies many aspects of app development, including authentication, real-time database management, cloud storage, and more.



FIREBASE SERVICES



Build better apps



Cloud Firestore
Store and sync app data at global scale



Firebase ML BETA
Machine learning for mobile developers



Cloud Functions
Run mobile backend code without managing servers



Authentication
Authenticate users simply and securely



Hosting
Deliver web app assets with speed and security



Cloud Storage
Store and serve files at Google scale



Realtime Database
Store and sync app data in milliseconds

Improve app quality



Crashlytics
Prioritize and fix issues with powerful, realtime crash reporting



Performance Monitoring
Gain insight into your app's performance



Test Lab
Test your app on devices hosted by Google



App Distribution BETA
Distribute pre-release versions of your app to your trusted testers

Grow your business



In-App Messaging BETA
Engage active app users with contextual messages



Google Analytics
Get free and unlimited app analytics



Predictions
Smart user segmentation based on predicted behavior



A/B Testing BETA
Optimize your app experience through experimentation



Cloud Messaging
Send targeted messages and notifications



Remote Config
Modify your app without deploying a new version



Dynamic Links
Drive growth by using deep links with attribution

The Complete Android Developer Course

FIRESTORE

In Cloud Firestore, the unit of storage is the document. A document is a lightweight record that contains fields, which map to values. Each document is identified by a name.

A document representing a user might look like this:

User:

```
first : "jack"  
last : "reacher"  
born : 1992
```



FIRESTORE

Documents live in collections, which are simply containers for documents. For example, you could have a "Users" collection to contain your various users, each represented by a document:

Users:

user1

```
first : "jack"  
last : "reacher"  
born : 1992
```

user2

```
first : "john"  
last : "travolta"  
born : 1982
```



WORK MANAGER

WorkManager is the recommended solution for persistent work. Work is persistent when it remains scheduled through app restarts and system reboots.

Because most background processing is best accomplished through persistent work, WorkManager is the primary recommended API for background processing.

Work manager simplifies and manages background tasks in android apps. It allows you to schedule and run tasks in the background, even when the app is not currently in the foreground



WORK MANAGER

Work manager offers several advantages, such as:

- handling tasks efficiently
- considering network connectivity
- considering device state
- supporting periodic scheduling



WORK MANAGER

Worker

A worker is a class that performs a background task.

Work Request:

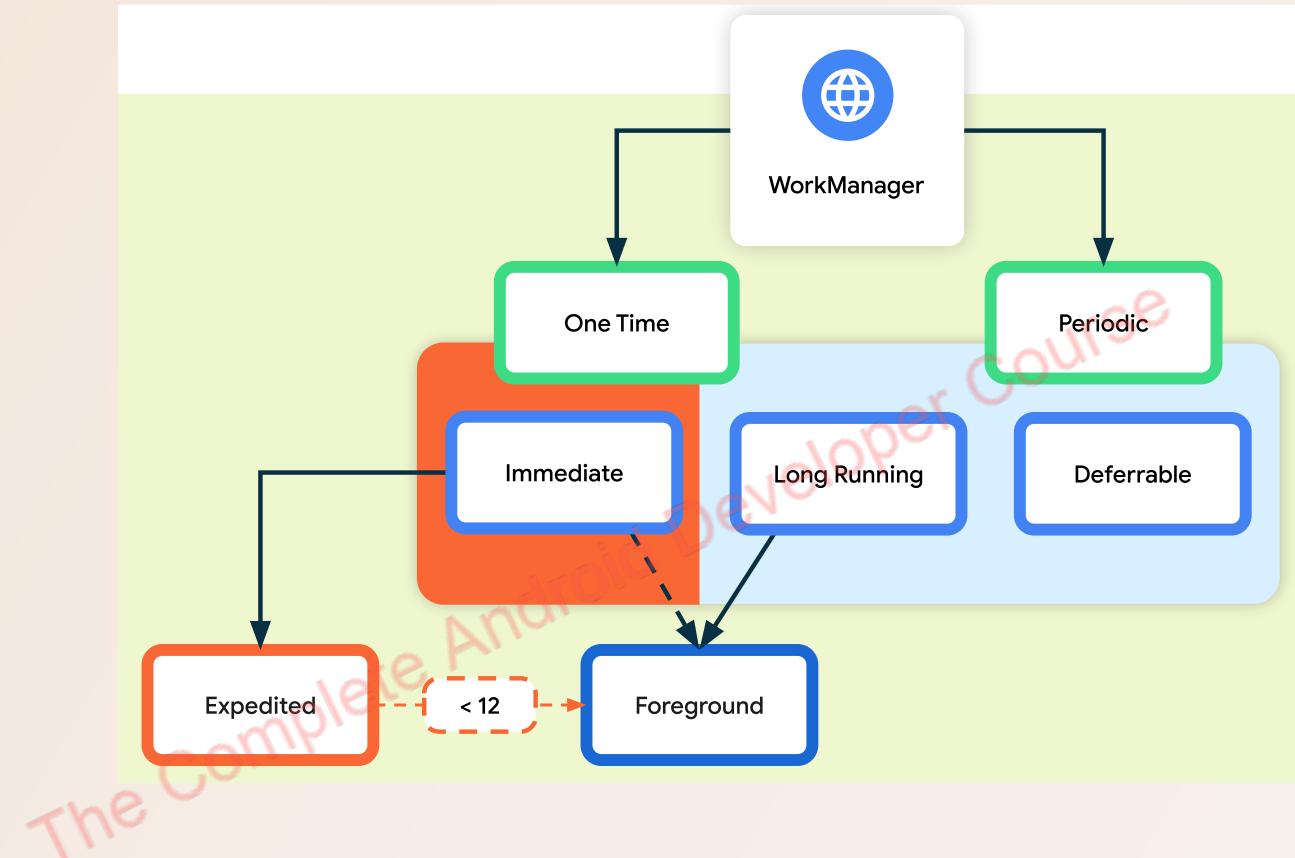
Defines the task to be executed and its constraints.

Types of WorkRequest:

- OneTimeWorkRequest
- PeriodicWorkRequest

Work Manager:

The central component of managing background tasks. It is responsible for scheduling and executing WorkRequest.



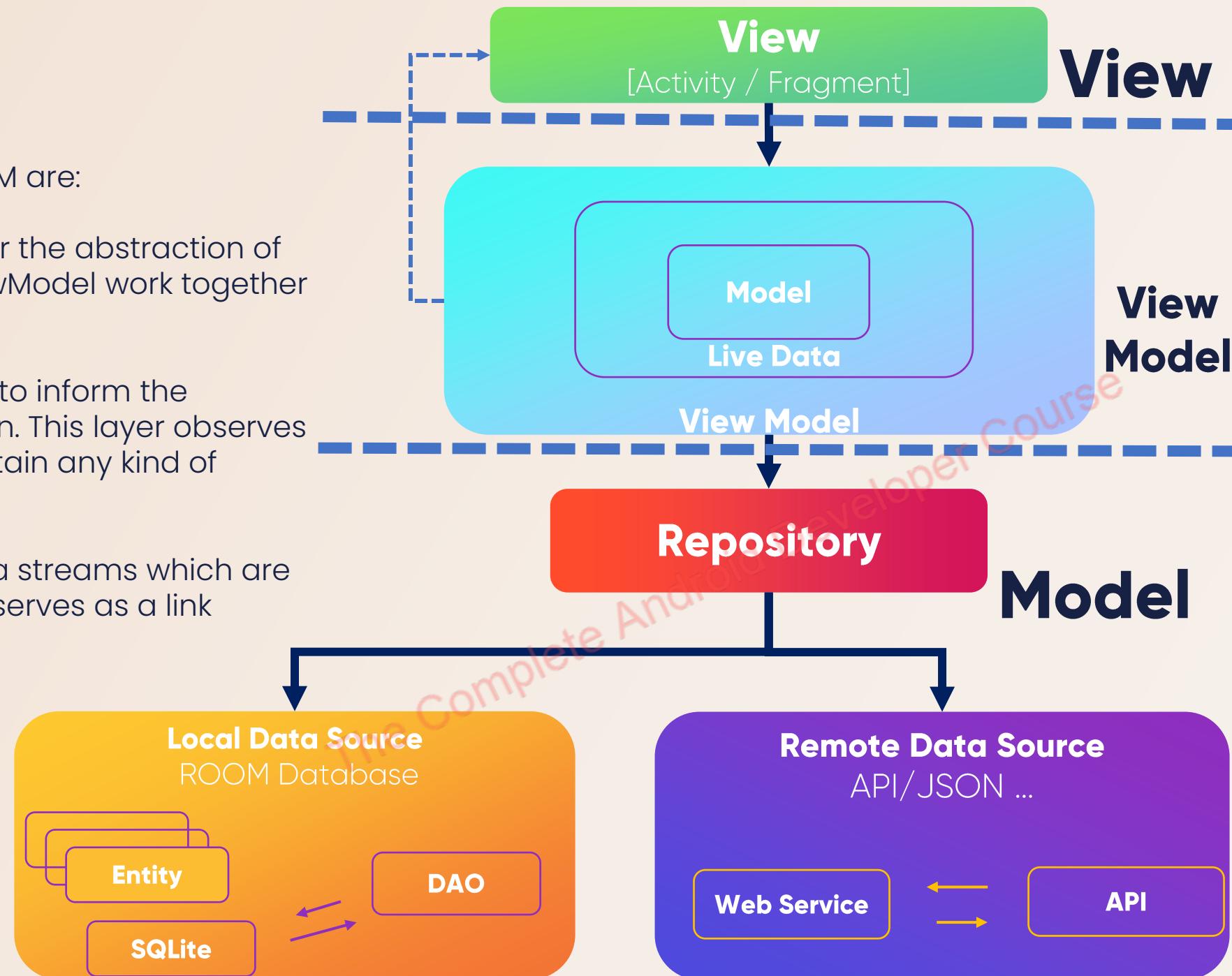
MVVM Layers

The separate code layers of MVVM are:

Model: This layer is responsible for the abstraction of the data sources. Model and ViewModel work together to get and save the data.

View: The purpose of this layer is to inform the ViewModel about the user's action. This layer observes the ViewModel and does not contain any kind of application logic.

ViewModel: It exposes those data streams which are relevant to the View. Moreover, it serves as a link between the Model and the View.



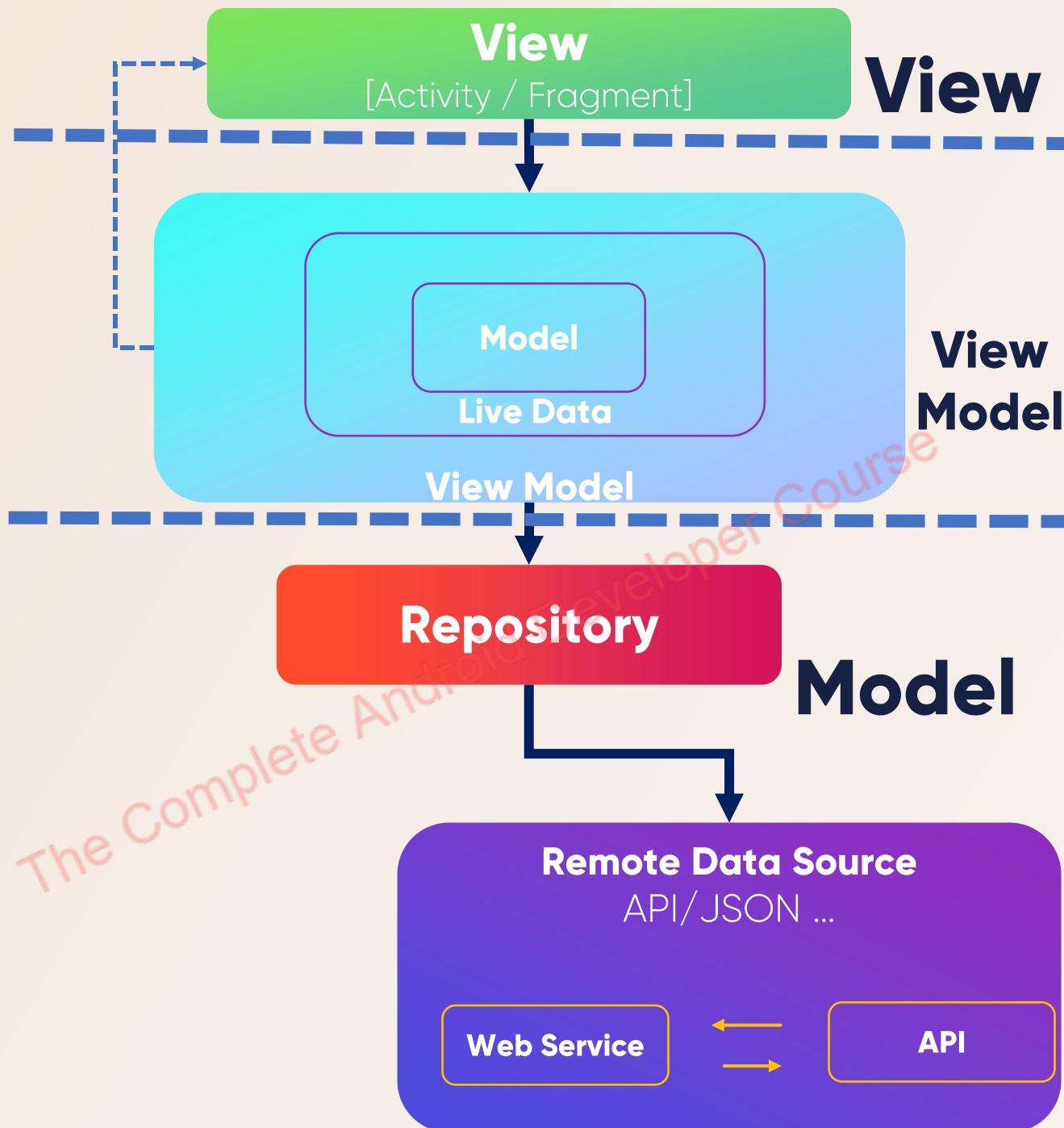
MVVM Layers

The separate code layers of MVVM are:

Model: This layer is responsible for the abstraction of the data sources. Model and ViewModel work together to get and save the data.

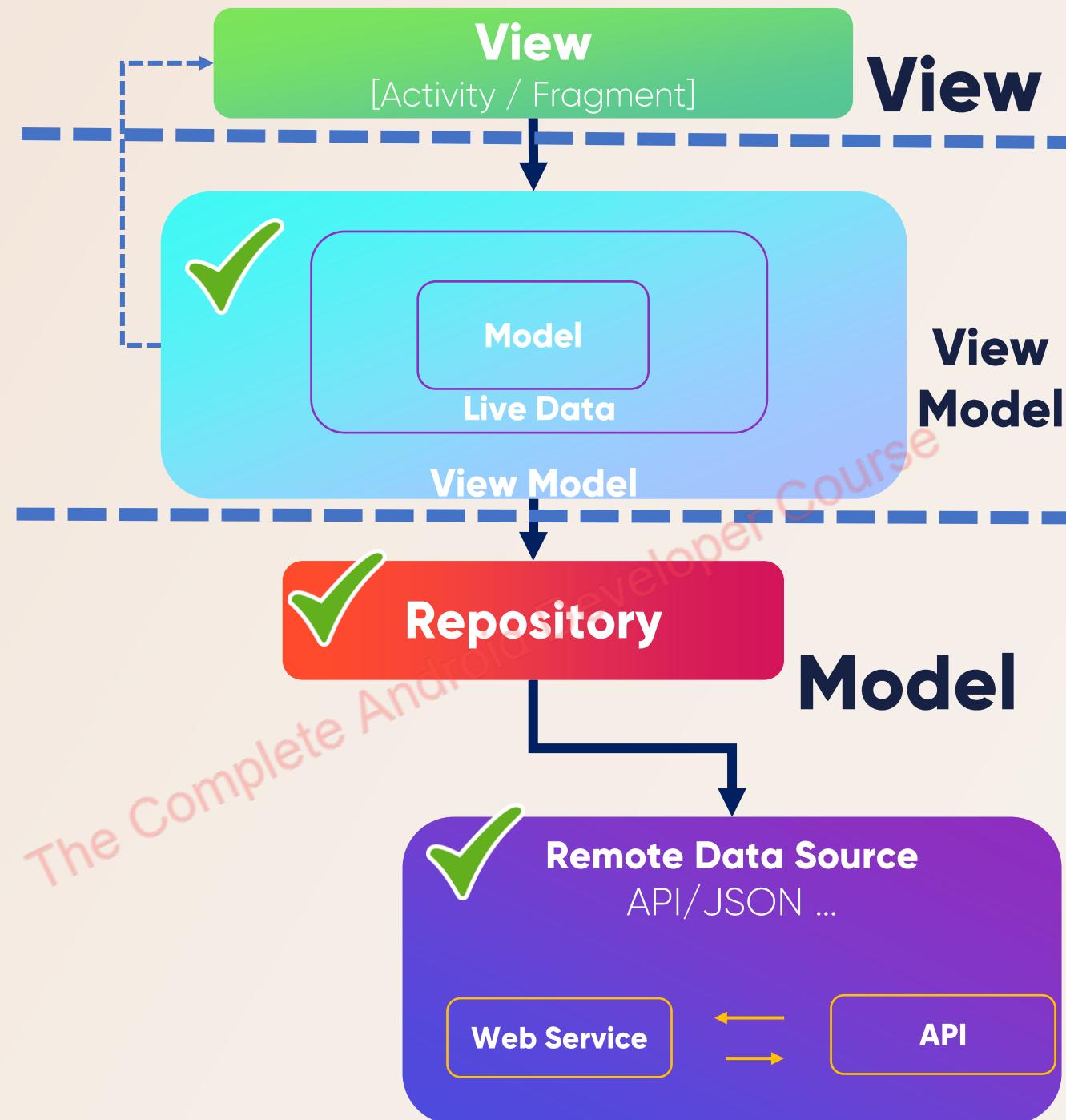
View: The purpose of this layer is to inform the ViewModel about the user's action. This layer observes the ViewModel and does not contain any kind of application logic.

ViewModel: It exposes those data streams which are relevant to the View. Moreover, it serves as a link between the Model and the View.

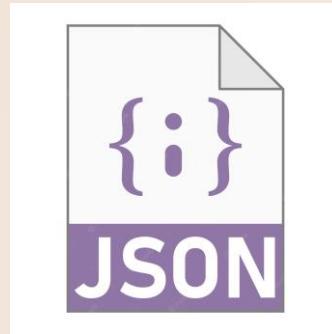


The Quiz App

- Create our local database (MYSQL)
- Create our own API
- Fetch the JSON response from database
- Using Retrofit to create Java Objects
- Display questions in Android App
- Follow MVVM Architecture



Our Own API



```
"results": [  
    {  
        "popularity": 167.729,  
        "vote_count": 3823,  
        "video": false,  
        "poster_path": "/xBHvZcjRiWyobQ9kxBh06B2dtRI.jpg",  
        "id": 419704,  
        "adult": false,  
        "backdrop_path": "/5BwqwxMEjeFtdknRV792Svo0K1v.jpg",  
        "original_language": "en",  
        "original_title": "Ad Astra",  
        "genre_ids": [  
            18,  
            878  
        ],  
        "title": "Ad Astra",  
        "vote_average": 6.1,  
        "overview": "The near future, a time when both hope  
        "release_date": "2019-09-17"  
    },  
    ...  
]
```



Plain Old Java Object

The Complete Android Developer Course



App UI

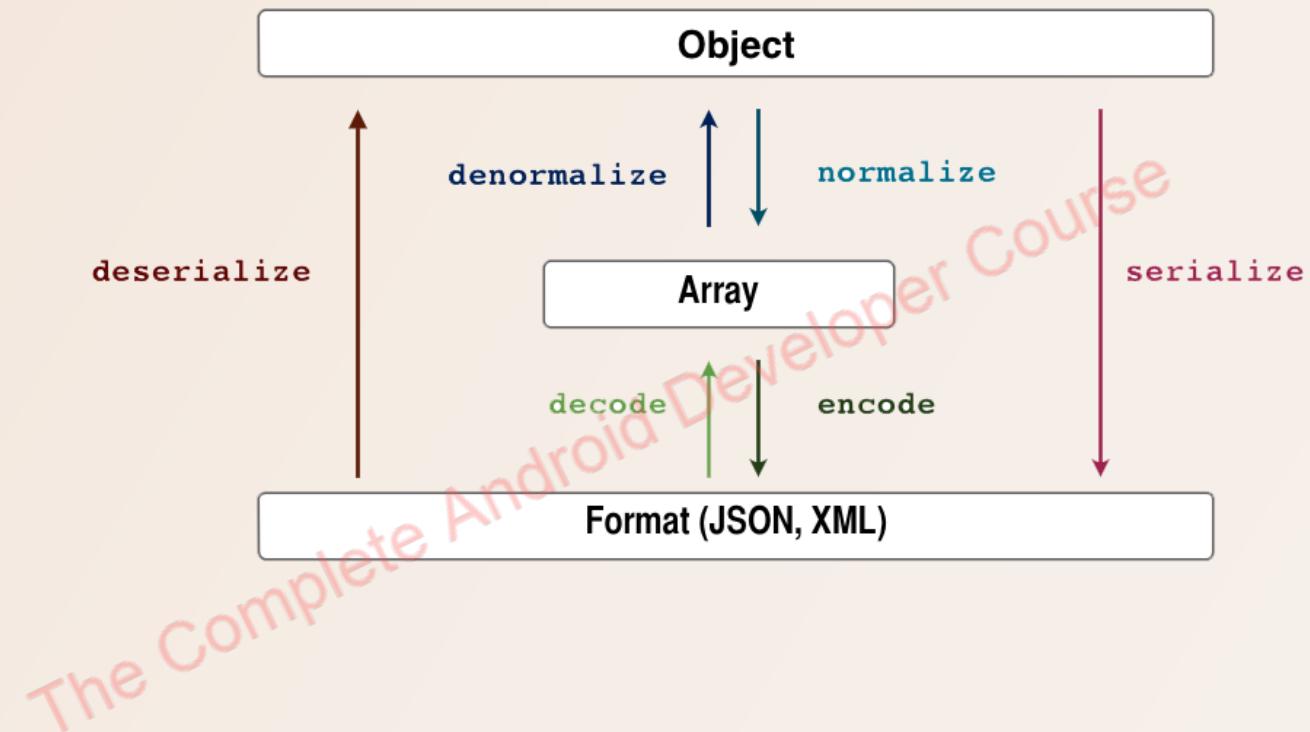
GSON CONVERTER

A Converter which uses Gson for serialization to and from JSON.

Used to convert Java objects into their JSON representation. It can also be used to convert a JSON string to an equivalent Java object.

For this purpose, Gson provides several built in serializers and deserializers. A serializer allows to convert a Json string to corresponding Java type. A deserializers allows to convert from Java to a JSON representation.

A default Gson instance will be created or one can be configured and passed to the GsonConverterFactory to further control the serialization.



USING RETROFIT



Data Classes: POJO

For fetching response we need create POJO class that automatically parse the JSON data using Gson in background. We just need to create this POJO class.

For creating POJO class first method is defining each and every variable by ourself and other best and fast method is using <http://www.jsonschema2pojo.org/> platform.

To serialize JSON we require a converter.



API/Service Interface

We need to create an Interface to define our different methods that will be used for network transactions.



Retrofit Instance

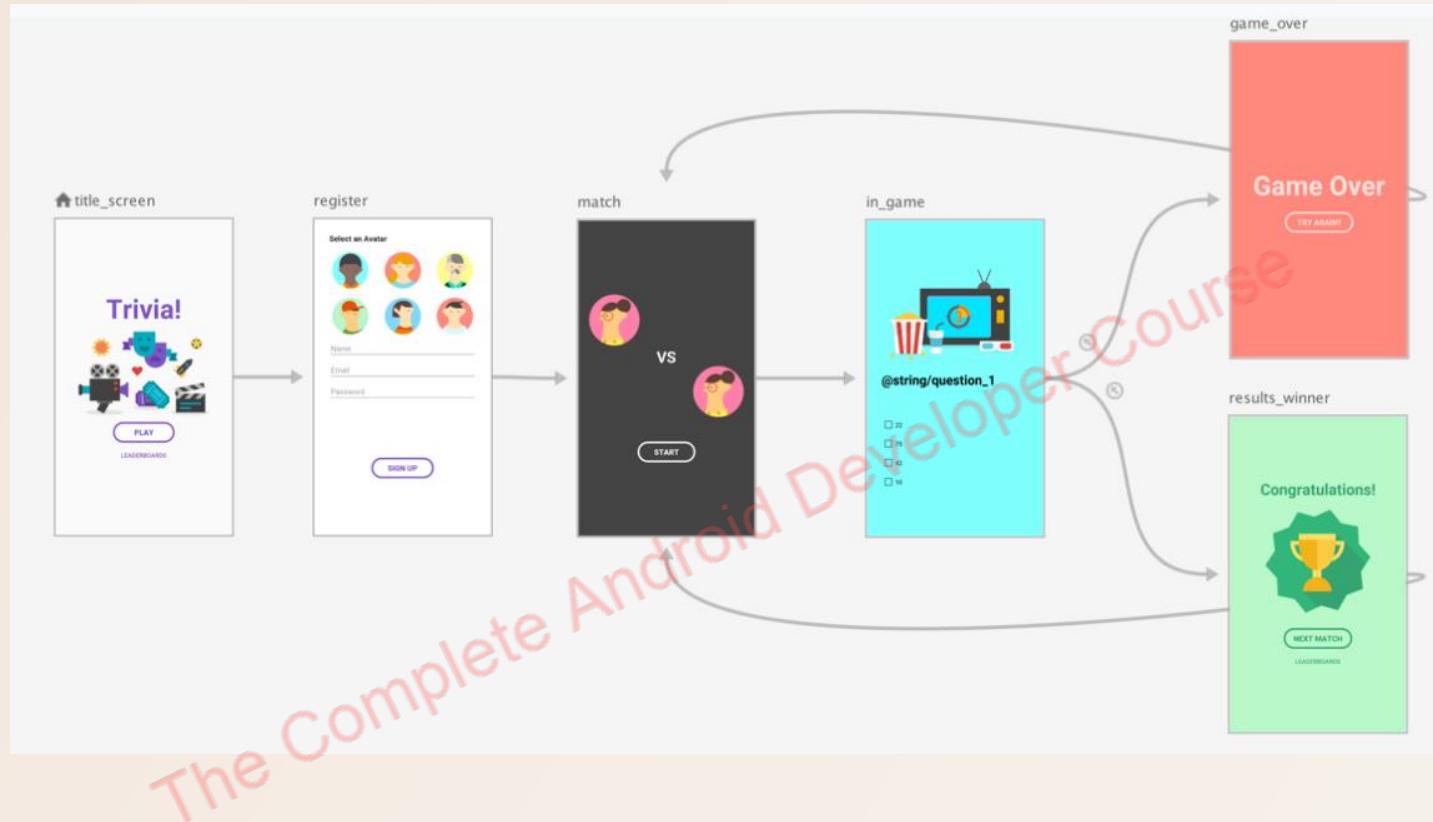
since we are using Retrofit for Network calls, let's create a class that provides us the `instance` of the Retrofit.

The Complete Android Developer Course

NAVIGATION

The Navigation Architecture Component simplifies implementing navigation, while also helping you visualize your app's navigation flow. The library provides several benefits, including:

- Automatic handling of fragment transactions
- Correctly handling up and back by default
- Default behaviors for animations and transitions
- Deep linking as a first-class operation
- Implementing navigation UI patterns (like navigation drawers and bottom nav) with little additional work
- Type safety when passing information while navigating
- Android Studio tooling for visualizing and editing the navigation flow of an app



NAVIGATION PARTS

The Navigation Component consists of three key parts, working together in harmony. They are:

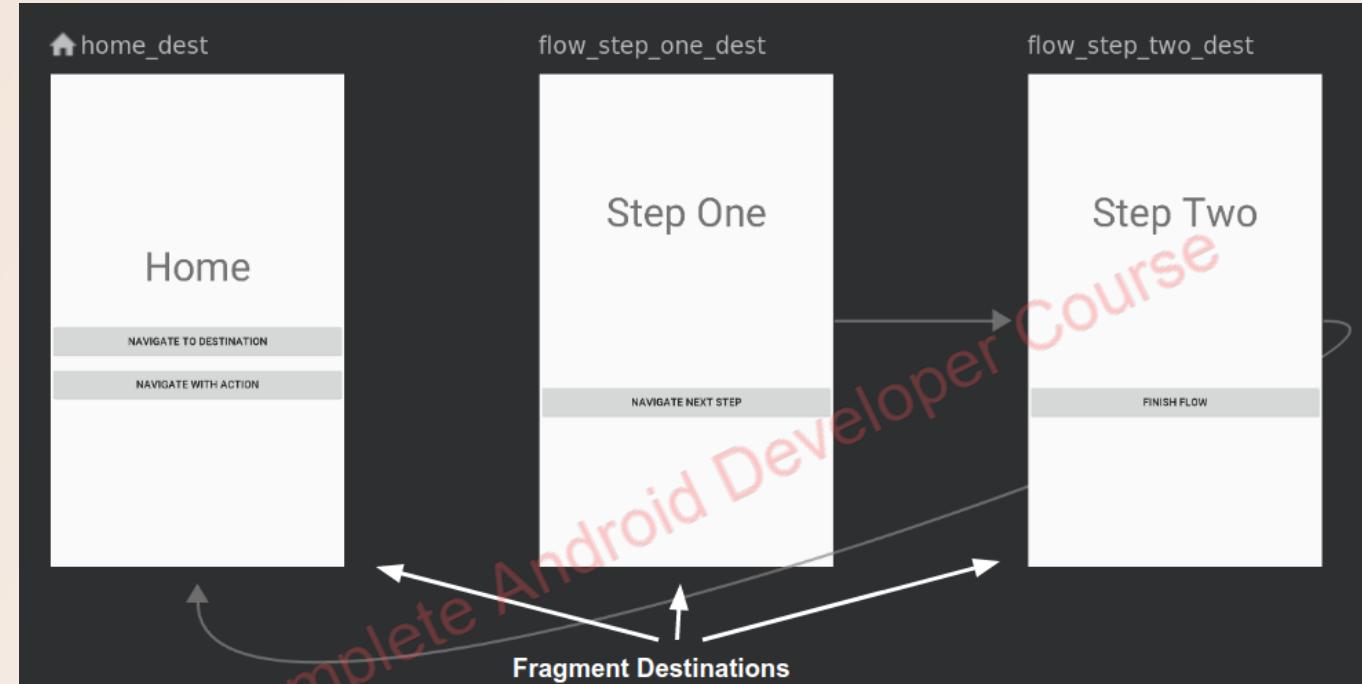
- **Navigation Graph (New XML resource)** – This is a resource that contains all navigation-related information in one centralized location. This includes all the places in your app, known as destinations, and possible paths a user could take through your app.
- **NavController (Layout XML view)** – This is a special widget you add to your layout. It displays different destinations from your Navigation Graph.
- **NavController (Kotlin/Java object)** – This is an object that keeps track of the current position within the navigation graph. It orchestrates swapping destination content in the NavHostFragment as you move through a navigation graph.



NAVIGATION GRAPH

A navigation graph is a new resource type that defines all the possible paths a user can take through an app. It shows visually all the destinations that can be reached from a given destination.

Android Studio displays the graph in its Navigation Editor.



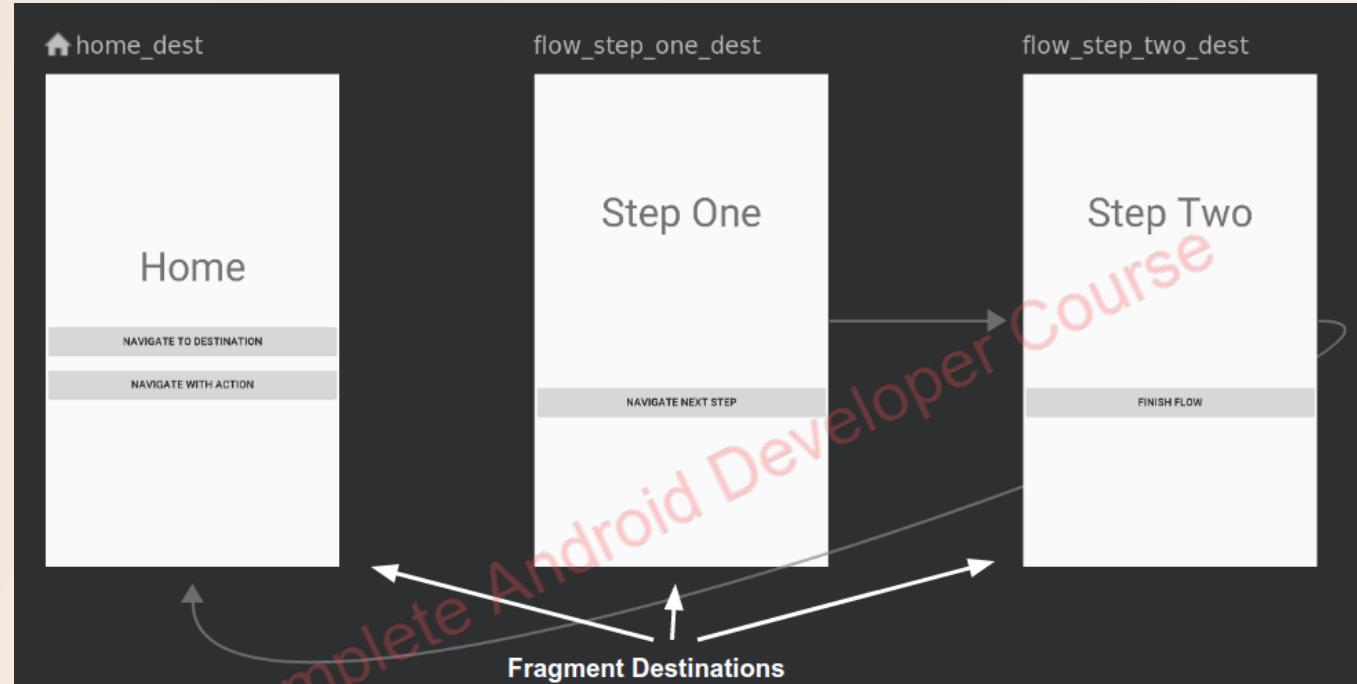
NAV HOST FRAGMENT

NavHostFragment provides an area within your layout for self-contained navigation to occur.

NavHostFragment is intended to be used as the content area within a layout resource defining your app's chrome around it.

Each NavHostFragment has a NavController that defines valid navigation within the navigation host.

This includes the NavGraph as well as navigation state such as current location and back stack that will be saved and restored along with the NavHostFragment itself.



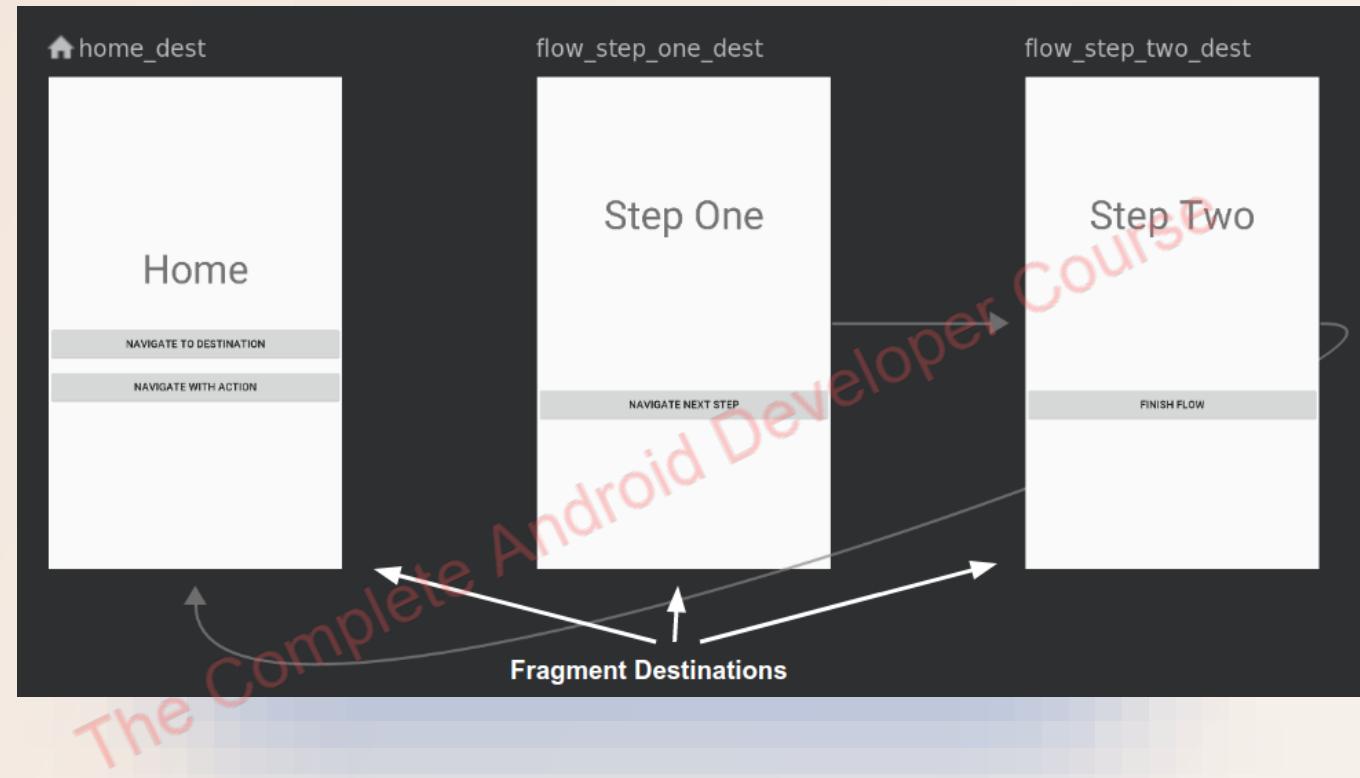
NAV CONTROLLER

Navigating to a destination is done using a NavController, an object that manages app navigation within a NavHost.

Each NavHost has its own corresponding NavController. You can retrieve a NavController by using one of the following methods:

- Fragment.findNavController()
- View.findNavController()
- Activity.findNavController(viewId: Int)

Actions: The routes user can take between your app's destinations. Which are represented by the arrow sign in the Design view.



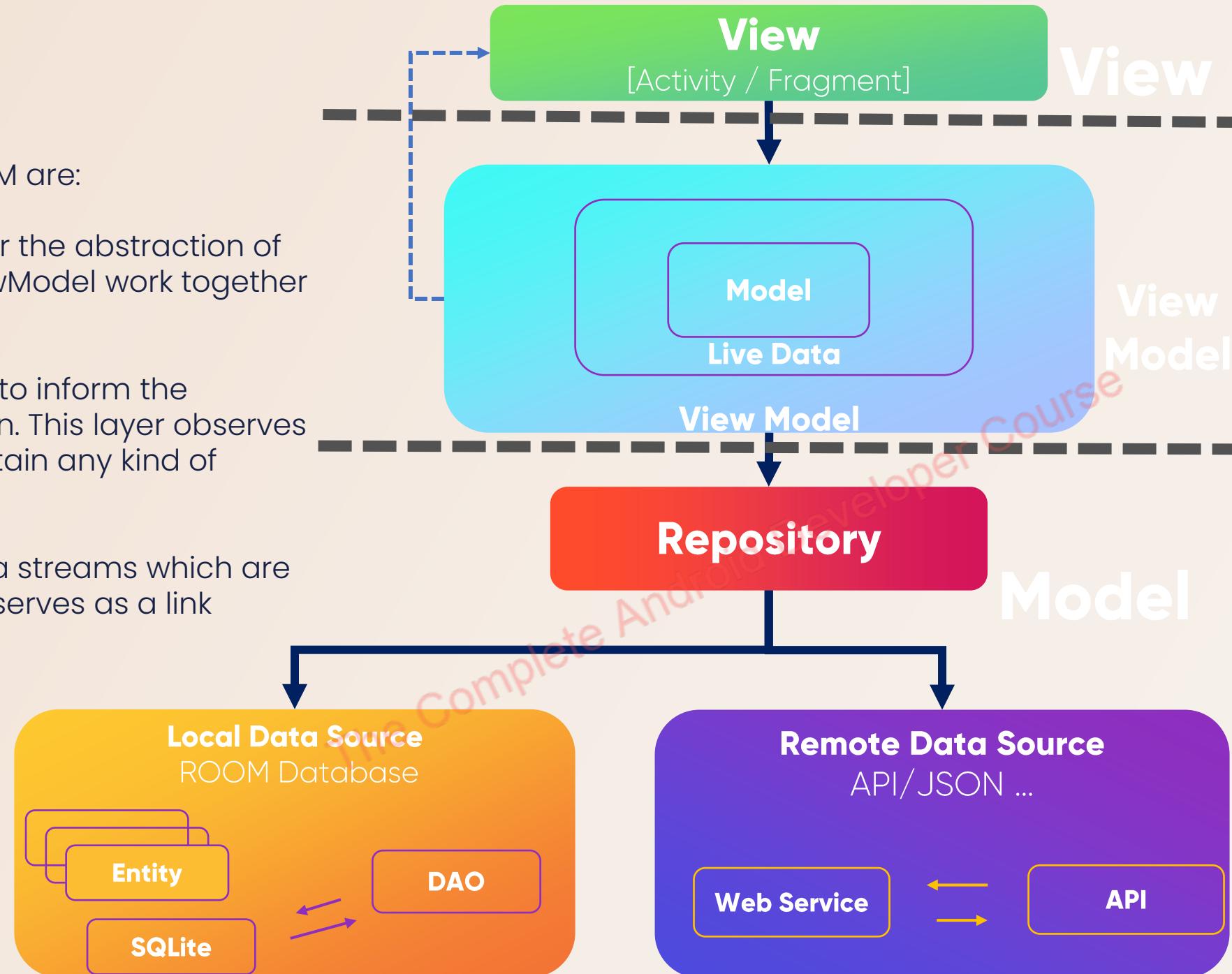
MVVM Layers

The separate code layers of MVVM are:

Model: This layer is responsible for the abstraction of the data sources. Model and ViewModel work together to get and save the data.

View: The purpose of this layer is to inform the ViewModel about the user's action. This layer observes the ViewModel and does not contain any kind of application logic.

ViewModel: It exposes those data streams which are relevant to the View. Moreover, it serves as a link between the Model and the View.



MVVM Layers

The separate code layers of MVVM are:

Model: This layer is responsible for the abstraction of the data sources. Model and ViewModel work together to get and save the data.

View: The purpose of this layer is to inform the ViewModel about the user's action. This layer observes the ViewModel and does not contain any kind of application logic.

ViewModel: It exposes those data streams which are relevant to the View. Moreover, it serves as a link between the Model and the View.

