



Lab 01

Android Development

2025-2026



Agenda

- What is Android?
- Hello World Example
- Android Project Structure
- Add Two Numbers Example
- Hands-on



What is Android?

- An open-source software platform used primarily to power mobile phones
- Essentially, a unified platform to develop apps that will run on many different phones and devices



What tools will be used in Lab?

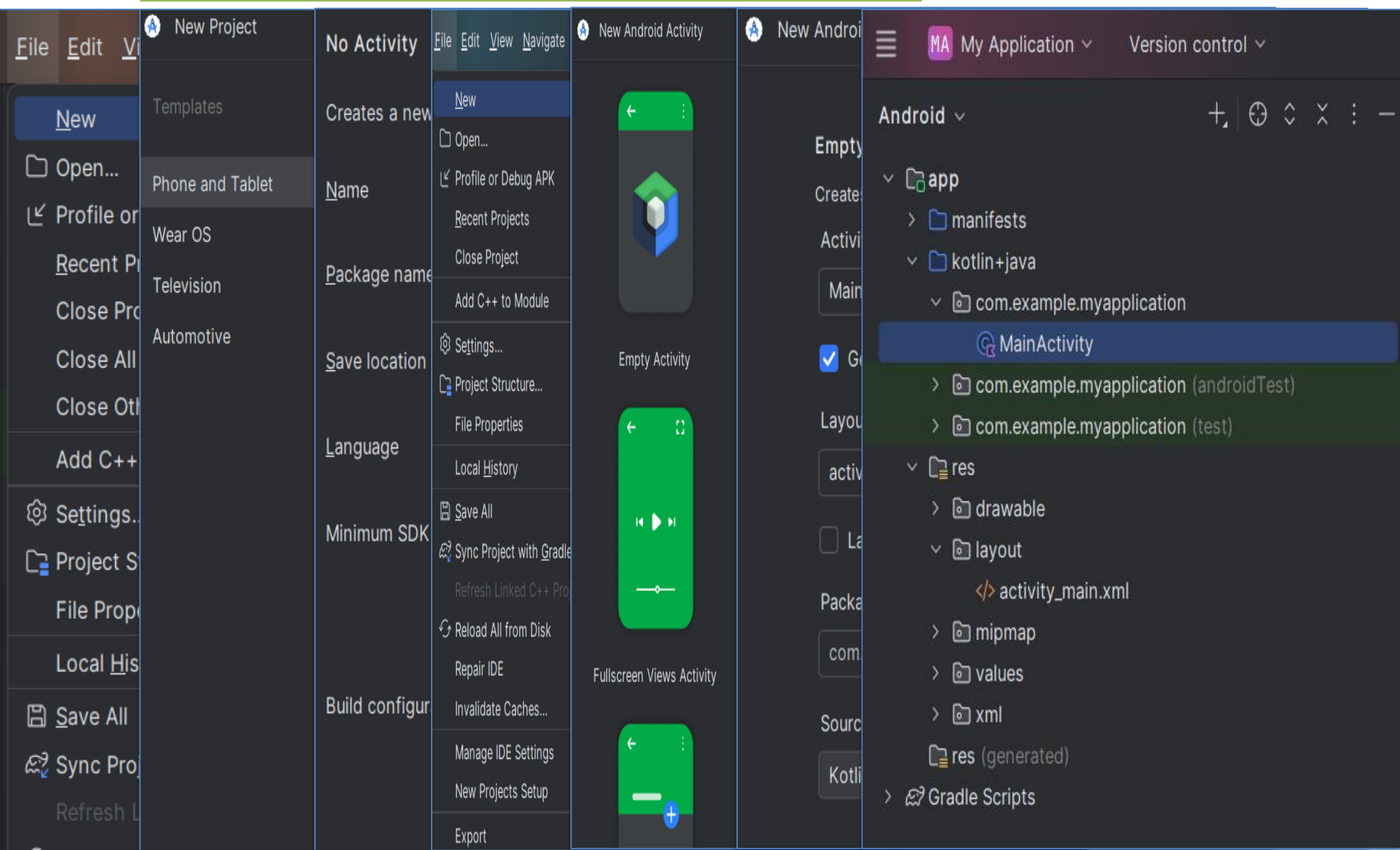
- Android studio
- Code is written in **Kotlin**



Hello World Example

■ Creating a New Project

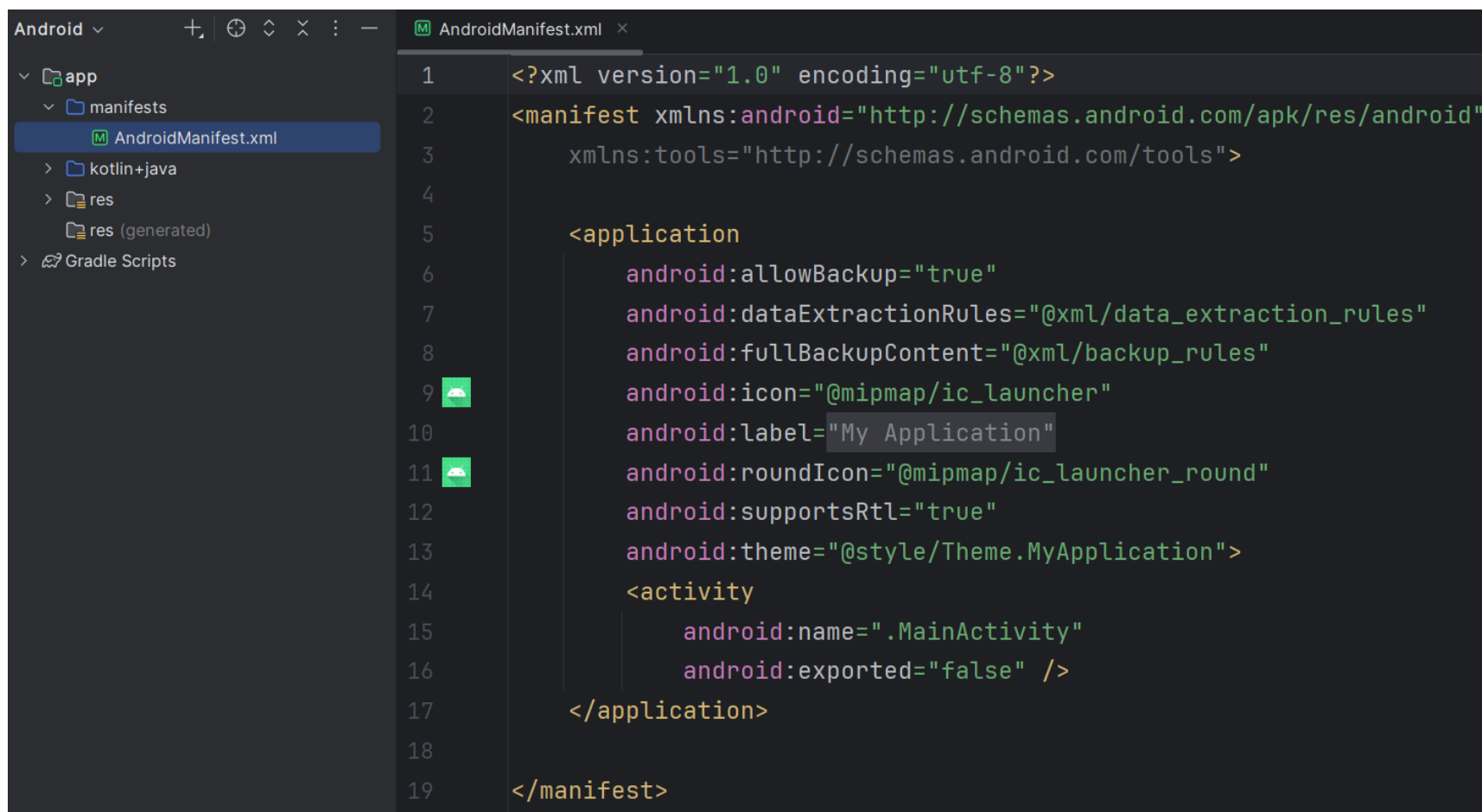
1. In Android studio :choose File→New→New Project
2. Next, You need to fill out your project properties.
 - “My Application”: is the plain English name of your application.
 - Type ‘HelloWorld’ as a project name
 - A project name should be one word. (“CamelCase” naming convention is preferred)
 - “The package name”: for example ‘android.section’
3. Next, you need to choose the *build target*.
 - *The build target tells the build tools* which version of the Android platform you are building for.
4. Specify An activity that is going to be represented by a Kotlin class.
 - Its name should adhere to **kotlin class** naming conventions: start with an upper-case letter and use **CamelCase** to separate words.
 - So, type ‘HelloWorld’ for your activity name or leave it as ‘ HelloWorldActivity’.
5. Finally, click on the Finish button, and Android studio will create your project.





Manifest File

- This file explains what the app consists of, its main building blocks, required permissions, and more.



```
1  <?xml version="1.0" encoding="utf-8"?>
2  <manifest xmlns:android="http://schemas.android.com/apk/res/android"
3          xmlns:tools="http://schemas.android.com/tools">
4
5      <application
6          android:allowBackup="true"
7          android:dataExtractionRules="@xml/data_extraction_rules"
8          android:fullBackupContent="@xml/backup_rules"
9          android:icon="@mipmap/ic_launcher"
10         android:label="My Application"
11         android:roundIcon="@mipmap/ic_launcher_round"
12         android:supportsRtl="true"
13         android:theme="@style/Theme.MyApplication">
14          <activity
15              android:name=".MainActivity"
16              android:exported="false" />
17      </application>
18
19  </manifest>
```



Strings File

- This is another XML file that contains all the text that your application uses.
 - For example, the names of buttons, labels, default text, and similar types of strings go into this file.

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="app_name">My Application</string>
</resources>
```




Layout File

- The layout file specifies the layout of your screen.

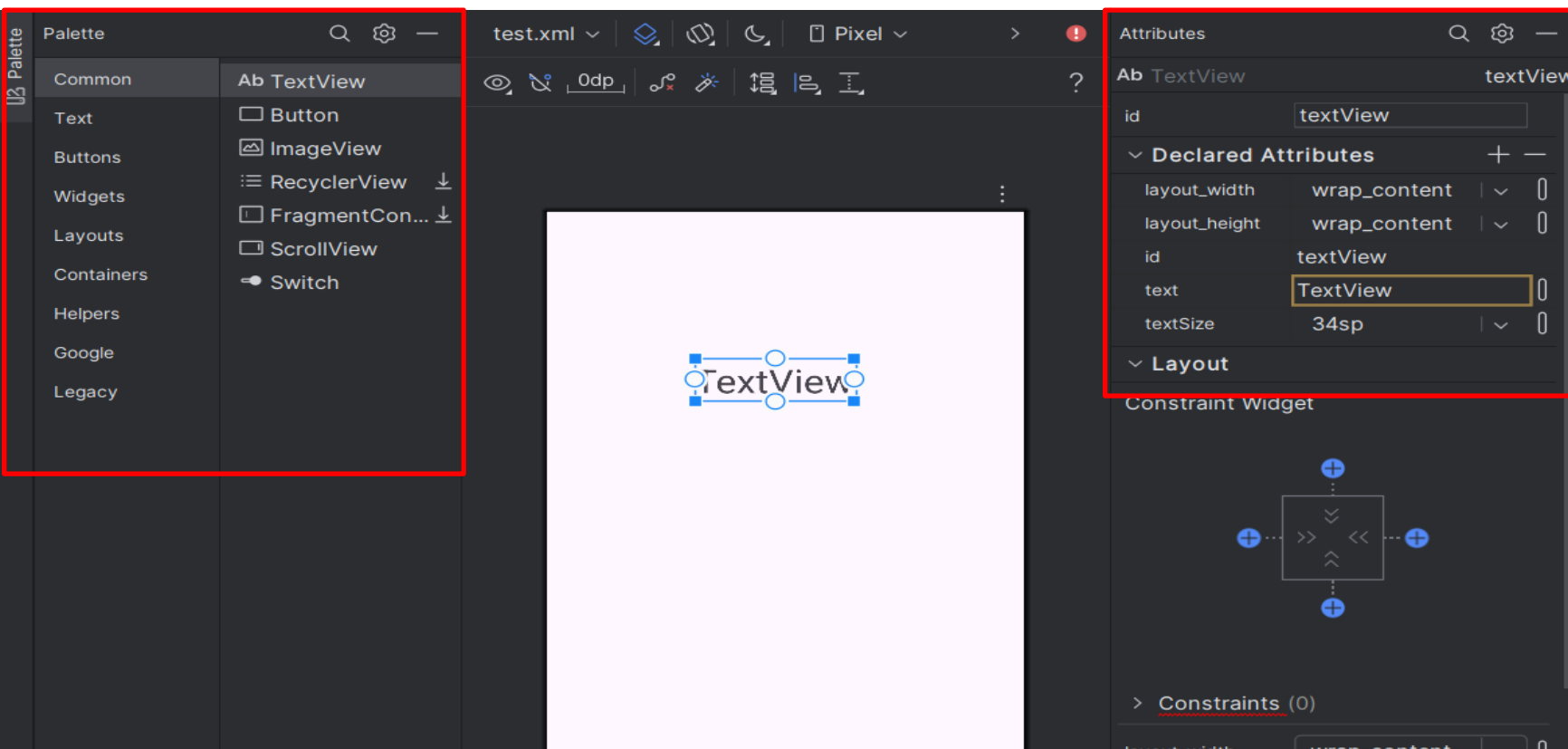
The screenshot shows the Android Studio IDE. On the left, the 'Project' view displays the file structure of an Android app, with the 'layout' folder and 'activity_main.xml' file selected. The main editor area shows the XML code for 'activity_main.xml'. The code defines a 'ConstraintLayout' with a 'TextView' inside. The 'TextView' is configured with 'wrap_content' dimensions and the text 'Hello World!'. It is also constrained to the parent layout on all four sides.

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <androidx.constraintlayout.widget.ConstraintLayout
3     xmlns:android="http://schemas.android.com/apk/res/android"
4     xmlns:app="http://schemas.android.com/apk/res-auto"
5     xmlns:tools="http://schemas.android.com/tools"
6     android:id="@+id/main"
7     android:layout_width="match_parent"
8     android:layout_height="match_parent"
9     tools:context=".MainActivity">
10
11     <TextView
12         android:layout_width="wrap_content"
13         android:layout_height="wrap_content"
14         android:text="Hello World!"
15         app:layout_constraintBottom_toBottomOf="parent"
16         app:layout_constraintEnd_toEndOf="parent"
17         app:layout_constraintStart_toStartOf="parent"
18         app:layout_constraintTop_toTopOf="parent" />
19
20 </androidx.constraintlayout.widget.ConstraintLayout>
21
```



Adding and Customizing Layout Elements

- Add elements to your layout by dragging them from the palette into the design view.
- Adjust their attributes such as size, ID, and text in the Attributes panel.





Adding and Customizing Layout Elements

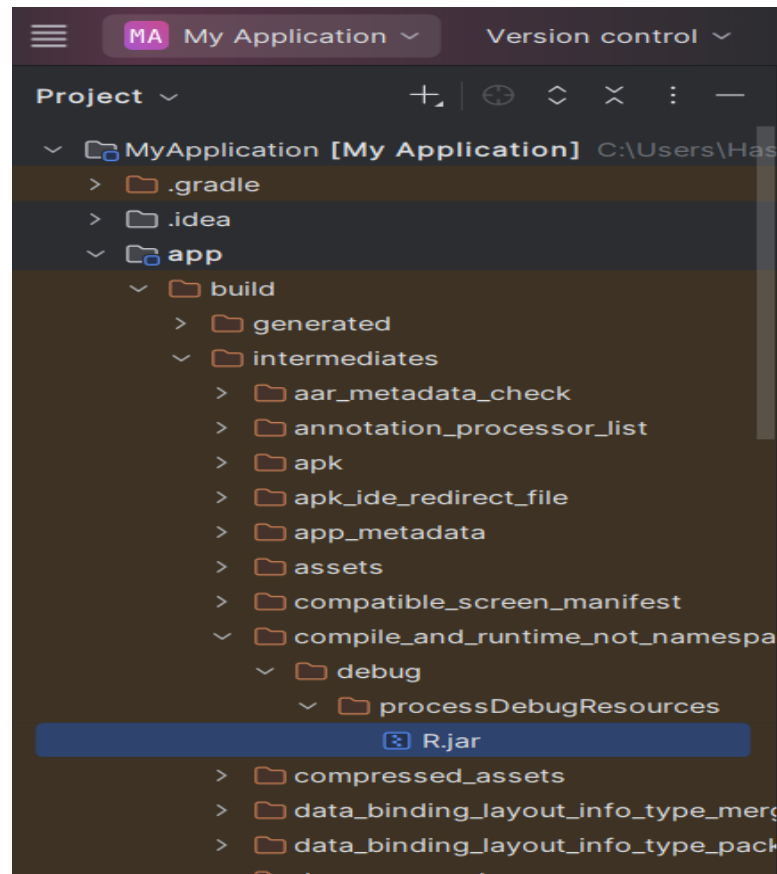
- Alternatively, you can define them directly in the layout XML code.

```
</> activity_main.xml ×
1  <?xml version="1.0" encoding="utf-8"?>
2  <androidx.constraintlayout.widget.ConstraintLayout
3      xmlns:android="http://schemas.android.com/apk/res/android"
4      xmlns:tools="http://schemas.android.com/tools"
5      xmlns:app="http://schemas.android.com/apk/res-auto"
6      android:id="@+id/main"
7      android:layout_width="match_parent"
8      android:layout_height="match_parent"
9      tools:context=".MainActivity">
10     <TextView
11         android:layout_width="wrap_content"
12         android:layout_height="wrap_content"
13         android:text="Hello World!"
14         app:layout_constraintBottom_toBottomOf="parent"
15         app:layout_constraintStart_toStartOf="parent"
16         app:layout_constraintEnd_toEndOf="parent"
17         app:layout_constraintTop_toTopOf="parent" />
18
19     </androidx.constraintlayout.widget.ConstraintLayout>
20
```



R File

- It is an automatically generated file, and as such, you never modify it.
- It is recreated every time you change anything.





Kotlin Source Code

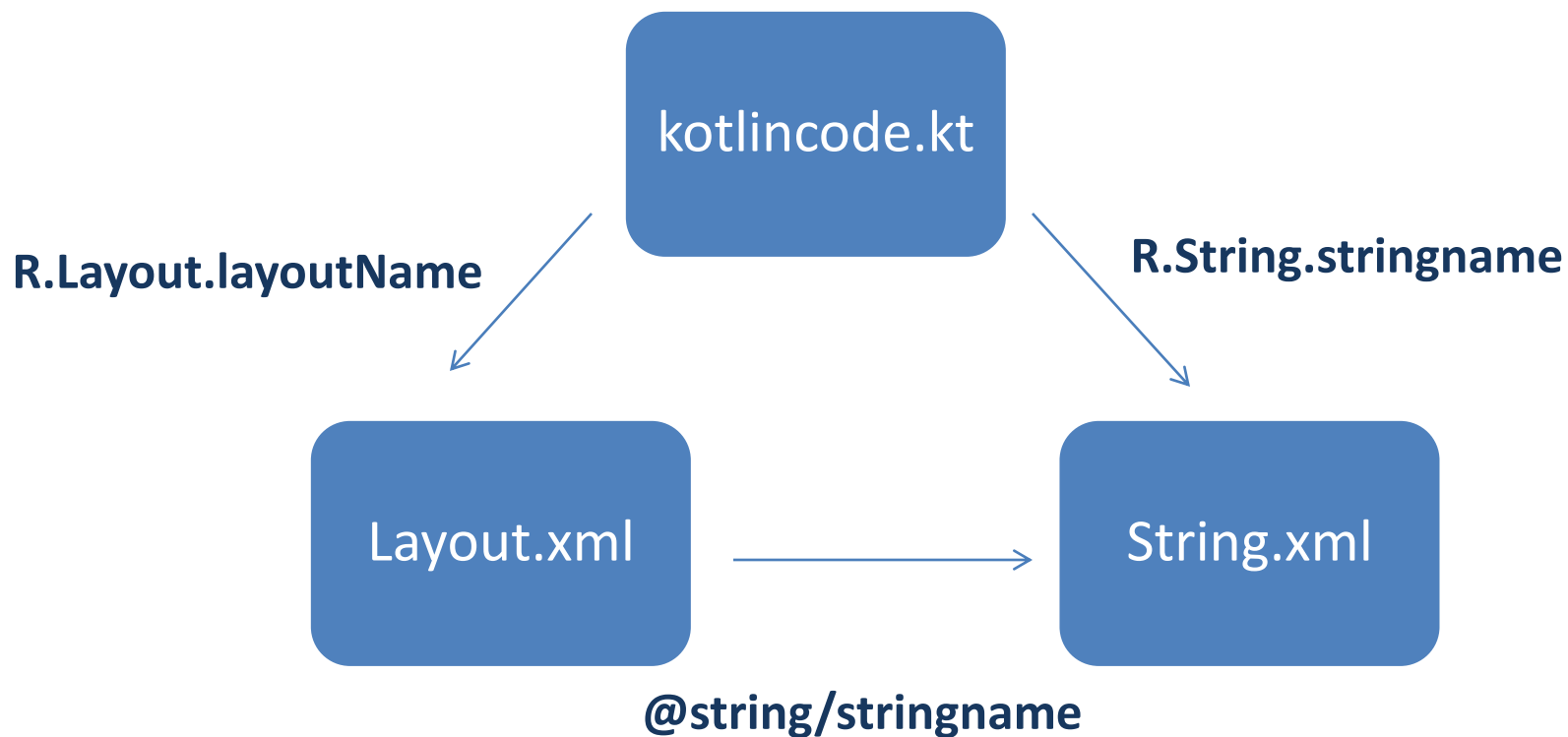
- The Kotlin code is what drives everything.
- This is the code that ultimately gets converted and runs your application

```
package com.example.myapplication  
  
...  
  
class MainActivity : AppCompatActivity() {  
    override fun onCreate(savedInstanceState: Bundle?) {  
        super.onCreate(savedInstanceState)  
        enableEdgeToEdge()  
        setContentView(R.layout.activity_main)  
    }  
}
```

In the Hello World template created by the wizard, the `onCreate` method is overridden to call `setContentView`, which lays out the user interface by inflating a layout resource



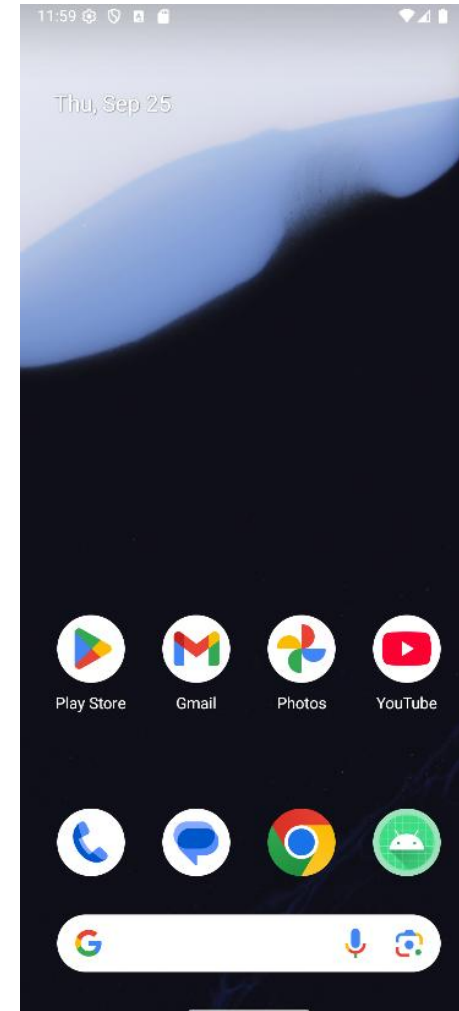
How To access data in files ?





Running Hello World Example

- Running the project
 - First, We will create new Android Virtual Device (AVD)
 - In Android Studio, Tools → Android → Device Manager
 - Specify Device then target then Name
 - Finally, Finish
 - Then right click on the project → run MainActivity





Running Hello World Example Cont'

The screenshot shows the Android Studio interface. The 'Add Device' button is highlighted with a red box. The 'Configure virtual device' dialog is open, showing the 'Pixel 5' device and a list of system images. The 'Running devices' panel on the right shows the 'Pixel 5' device running.

Device Manager

- Form Factor
 - ☒ Phone
 - ☐ Tablet
 - ☐ Wear OS
 - ☐ Desktop
 - ☐ TV
 - ☐ Automotive
- ☐ Show obsolete device profiles

Configure virtual device

Device: Pixel 5

Select system image

API

- API 36.1 "Baklava"; Android 13
- API 36.0 "Baklava"; Android 13
- API 35 "VanillaCreme"; Android 13
- API 34 "UpsideDownCake"; Android 13
- API 33 "Tiramisu"; Android 12
- API 32 "Sv2"; Android 12
- API 31 "S"; Android 12
- API 30 "R"; Android 11
- Show All

Running devices

- Pixel 5
- Medium Phone API 36.0
- Select Multiple Devices...
- Pair Devices Using Wi-Fi
- Troubleshoot Device Connections

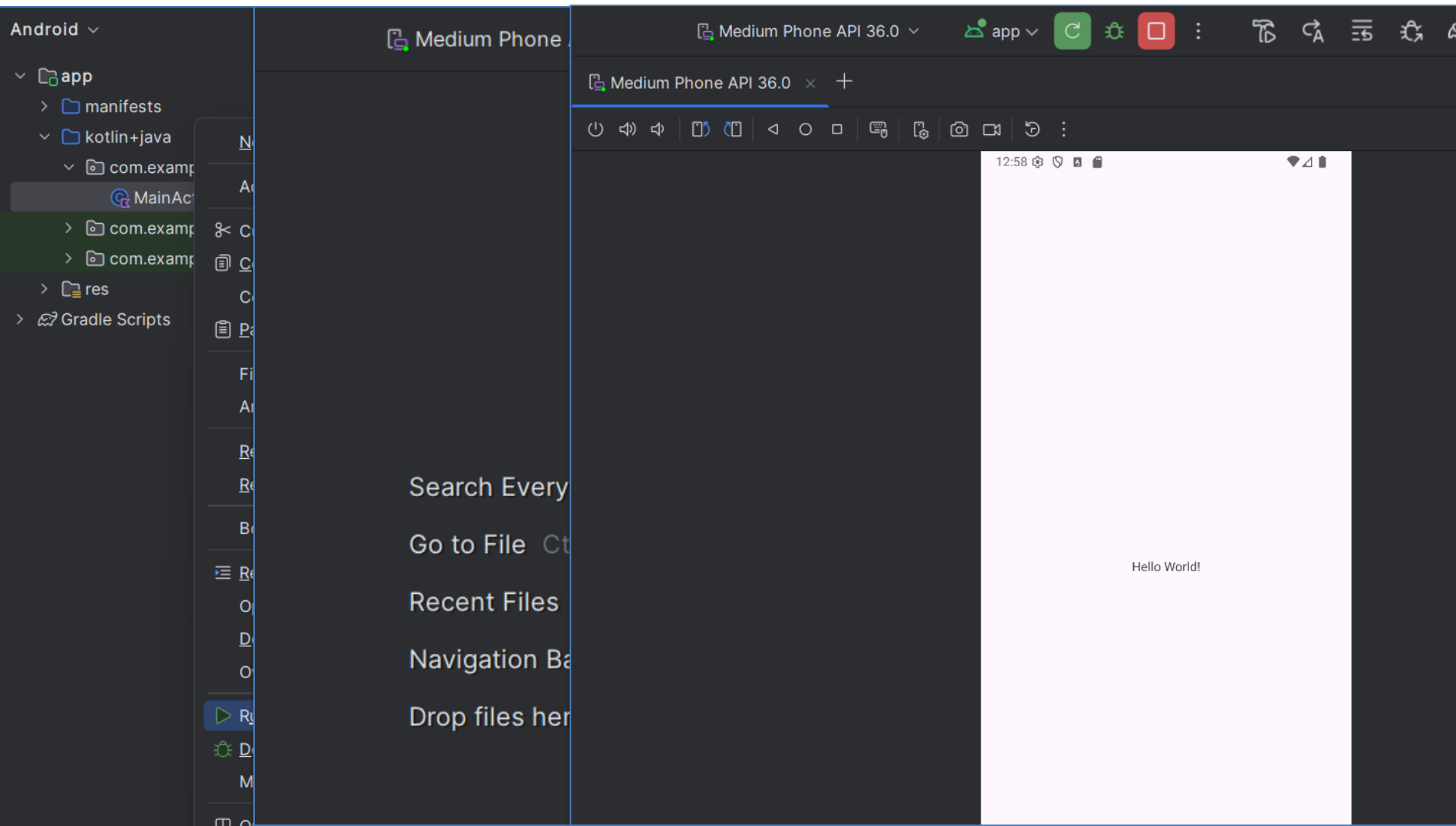
```
package com.example.hello_world

import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.view.View
import android.widget.Button

class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        enableEdgeToEdge()
        setContentView(R.layout.activity_main)
        ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main)) { v, insets }
    }
}
```




Running Hello World Example Cont'





Add Two Numbers Example

Adding Two Numbers

Number 1:

Enter first number

Number 2:

Enter second number

Add

Result will appear here

Adding Two Numbers

Number 1:

5

Number 2:

3

Add

Result: 8



Add Two Numbers Example: What will be changed !!!

- Main.xml file → Layout
- Main.kt file → Code
- String.xml file → Texts used in GUI

Note: R file is automatically changed



Main Layout

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.appcompat.widget.LinearLayoutCompat
    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:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity"
    android:orientation="vertical"
    android:gravity="center"
    android:padding="24dp">

    <TextView
        android:id="@+id/headerText"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Adding Two Numbers"
        android:textSize="22sp"
        android:textStyle="bold"
        android:layout_marginBottom="24dp"/>
```



Main Layout – Number Inputs

```
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Number 1:"
    android:textSize="16sp"
    android:layout_marginBottom="4dp"/>
```

```
<EditText
    android:id="@+id/txt_firstNum"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:ems="10"
    android:hint="Enter first number"
    android:inputType="number"
    android:layout_marginBottom="16dp"
    android:gravity="center"/>
```

```
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Number 2:"
    android:textSize="16sp"
    android:layout_marginBottom="4dp"/>
```

```
<EditText
    android:id="@+id/txt_secondNum"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:ems="10"
    android:hint="Enter second number"
    android:inputType="number"
    android:layout_marginBottom="16dp"
    android:gravity="center"/>
```



Main Layout – Action Button & Output Text

```
<Button
    android:id="@+id/btn_add"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Add"
    android:layout_marginBottom="20dp"/>

<TextView
    android:id="@+id/txt_sum"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Result will appear here"
    android:textSize="18sp"
    android:textStyle="bold"
    android:gravity="center"/>

</androidx.appcompat.widget.LinearLayoutCompat>
```



Addition Class

```
class Addition {  
    private var firstNum: Int = 0  
    private var secondNum: Int = 0  
  
    fun setFirstNum(num: Int) {  
        firstNum = num  
    }  
  
    fun setSecondNum(num: Int) {  
        secondNum = num  
    }  
  
    fun getFirstNum(): Int {  
        return firstNum  
    }  
  
    fun getSecondNum(): Int {  
        return secondNum  
    }  
  
    fun add(): Int {  
        return firstNum + secondNum  
    }  
}
```



Add Two Numbers Example: How to add action on the button?

- First we need to get button id so we can add action listener on it.

findViewById<Button>(R.id.btn_add)

```
val addBtn = findViewById<Button>(R.id.btn_add);
```

- Then add actionListener to the button on the MainActivity.kt in the onCreate Method

```
addBtn.setOnClickListener {  
  
}
```




Add Two Numbers Example: implementation

```
val addBtn = findViewById<Button>(R.id.btn_add);
val firstNum = findViewById<EditText>(R.id.txt_firstNum);
val secondNum = findViewById<EditText>(R.id.txt_secondNum);
val resultText = findViewById<TextView>(R.id.txt_sum)

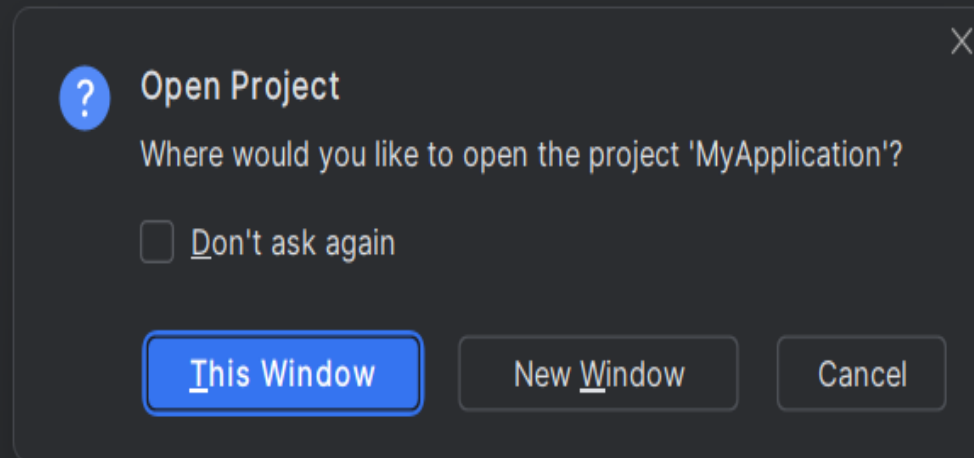
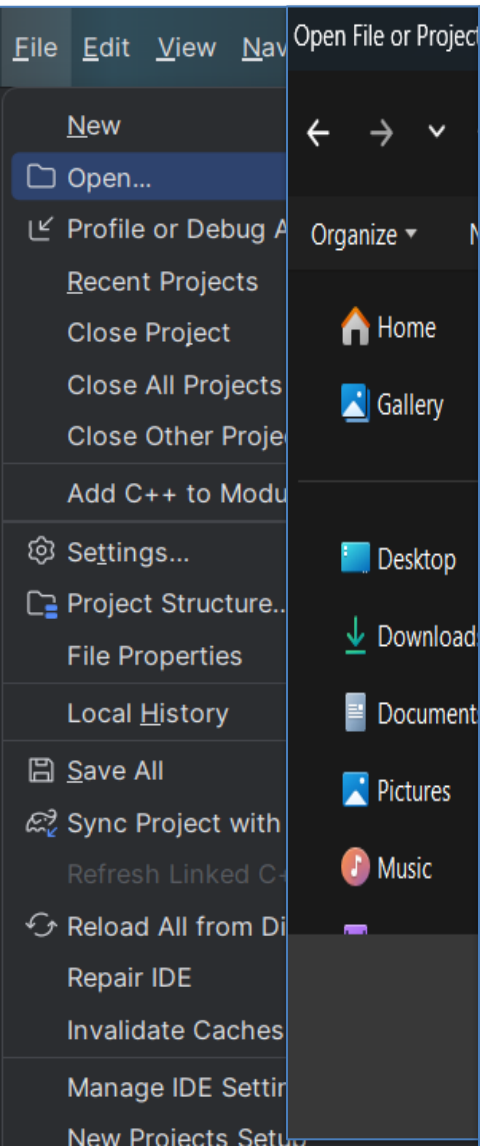
addBtn.setOnClickListener {

    val addition = Addition()
    val fNum = (firstNum.text.toString().trim()).toIntOrNull();
    val sNum = (secondNum.text.toString().trim()).toIntOrNull();

    if (fNum == null || sNum == null) {
        Toast.makeText(context: this, text: "Invalid input. Enter valid integers",
            Toast.LENGTH_SHORT).show()
    } else {
        addition.setFirstNum(fNum)
        addition.setSecondNum(sNum)
        val sum = addition.add()
        resultText.text = "Result: $sum"
    }
}
```



Import a project to the Android Studio





Hands-on

Tax Calculator App

Enter amount

Tax: 10%

Calculate Tax

Result will appear here

Tax Calculator App

100|

Tax: 10%

Calculate Tax

Tax: 10.0
Total: 110.0



Hands-on Hint

- Drag and drop the SeekBar widget into your activity layout.
- Set its max value and progress in the Attributes panel.

The screenshot shows the Android Studio IDE with the 'activity_main.xml' file open. The 'Palette' on the left has the 'SeekBar' widget highlighted under the 'Widgets' category. The central preview window shows a 'Tax Calculator App' layout with a text input field labeled 'Enter amount', a 'SeekBar' widget showing 'Tax: 10%', a 'Calculate Tax' button, and a placeholder text 'Result will appear here'. The 'Attributes' panel on the right shows the configuration for the 'SeekBar' widget with the following attributes:

Declared Attributes	
layout_width	match_parent
layout_height	wrap_content
id	taxSeekBar
max	50
progress	10

The 'Layout' and 'Common Attributes' sections are also visible, with 'max' and 'progress' values highlighted in red in the 'Common Attributes' section.



Hands-on Hint

- Get a reference to the SeekBar from the layout using findViewById.

```
val taxSeekBar = findViewById<SeekBar>(R.id.taxSeekBar)
```

- Attach an OnSeekBarChangeListener to the SeekBar so you can respond to progress changes.
- Override onStartTrackingTouch and onStopTrackingTouch as required by the interface, but keep them empty since they're not needed here.

```
taxSeekBar.setOnSeekBarChangeListener(object : SeekBar.OnSeekBarChangeListener {  
    override fun onProgressChanged(seekBar: SeekBar?, progress: Int, fromUser: Boolean) {  
    }  
    override fun onStartTrackingTouch(seekBar: SeekBar?) {}  
    override fun onStopTrackingTouch(seekBar: SeekBar?) {}  
})
```



Questions

