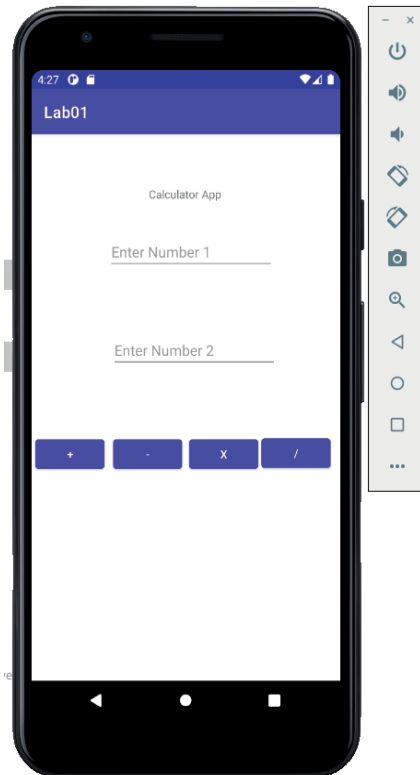


Lab 4

Creating Your First Application

Objectives:

- Create your first Android application
- Build a “Simple Calculator” Application



Create your First Android application

The first Android application will be a simple one because we want you to focus more on the basic issues that you may encounter while creating an Android application. It is important to focus on how to create the first Android application using Android Studio, where the application will be saved, and how it will be run or tested using an Android emulator.

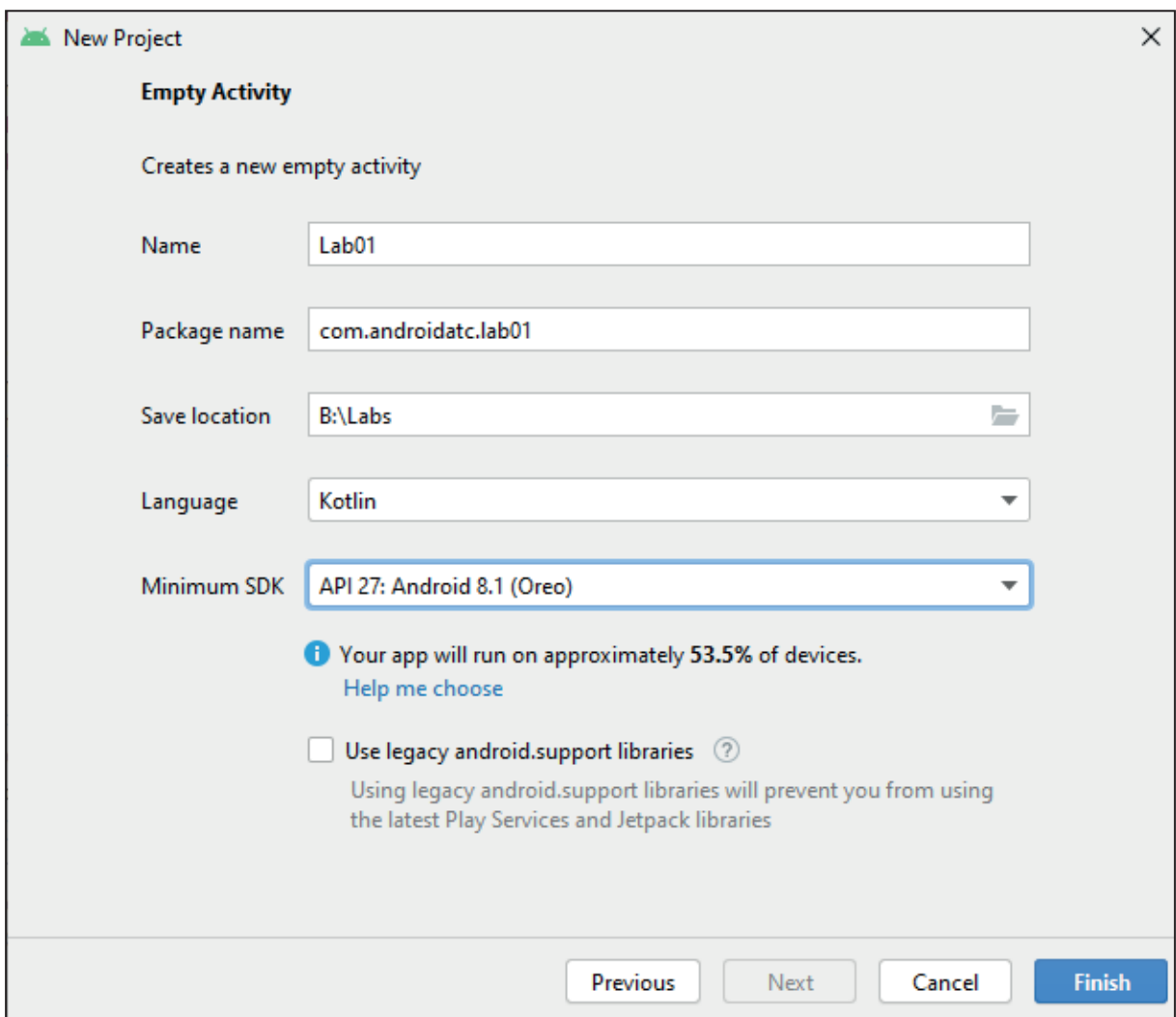
Scenario: In this lab, the first Android application will be creating a simple calculator using Android Studio. The steps are as follows:

1- Open Android Studio:

2- Click **File** → **New** → **New Project**.

3- Select **Empty Activity**, and click **Next**.

4- As illustrated in the figure below, type: **Lab01** for the project name, create a folder for your labs on a partition otherwise the OS partition, select the minimum SDK: **Android 8.1**, and then click **Finish**.



New Project

Empty Activity

Creates a new empty activity

Name: Lab01

Package name: com.androidatc.lab01

Save location: B:\Labs

Language: Kotlin

Minimum SDK: API 27: Android 8.1 (Oreo)

i Your app will run on approximately 53.5% of devices.
[Help me choose](#)

☐ Use legacy android.support libraries **?**
Using legacy android.support libraries will prevent you from using the latest Play Services and Jetpack libraries

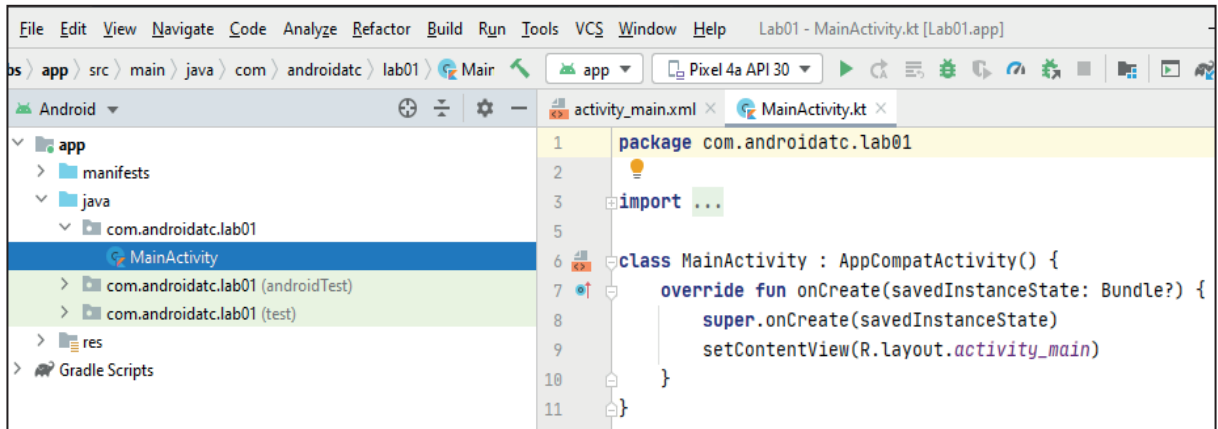
Previous Next Cancel Finish



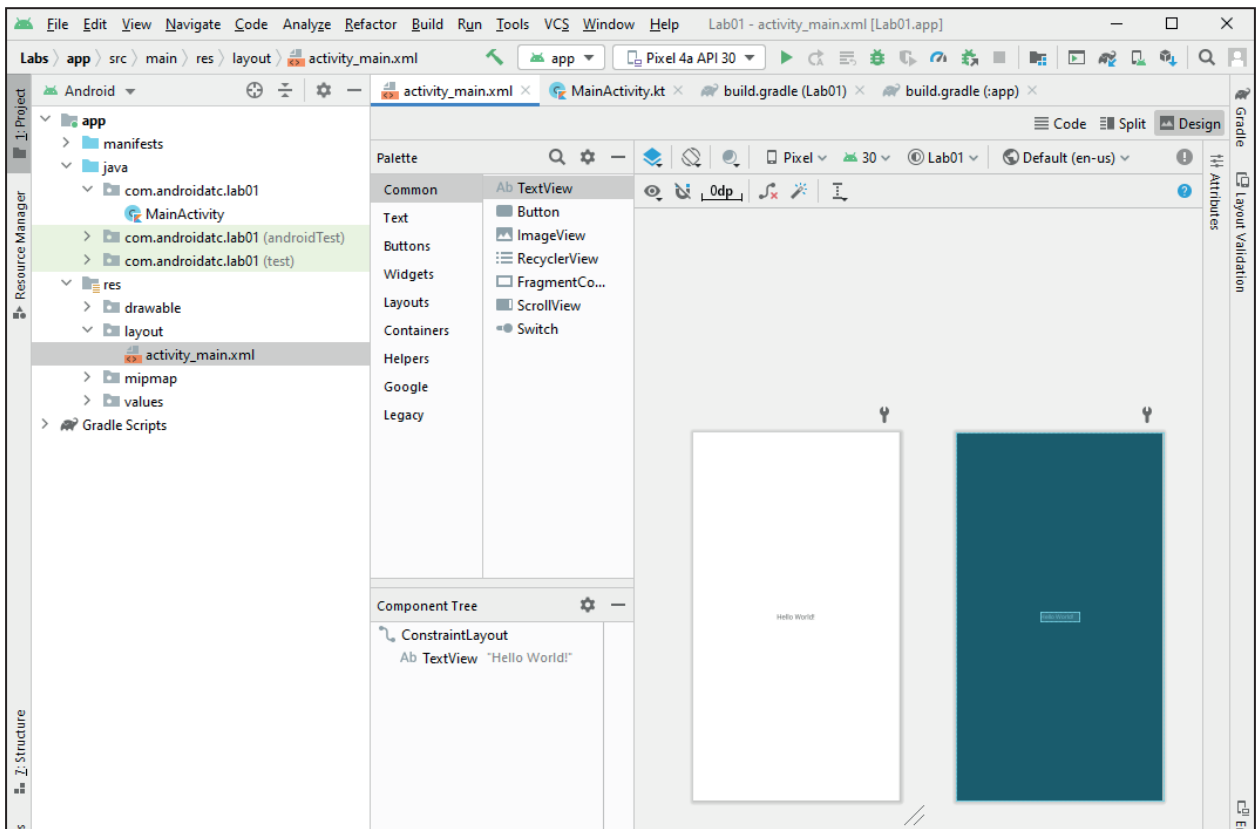
The package name uniquely identifies the app both on the device and in Google Play store. This means that once you have published an app with a specific package name, you can never change it. Changing the name of your app will cause your app to be treated as a brand new app, and existing users of your app will not see the newly packaged app as an update.

Wait a few seconds while Android Studio creates your project, files, and folders.

5- Now, your Android application is ready, and you may begin developing it.



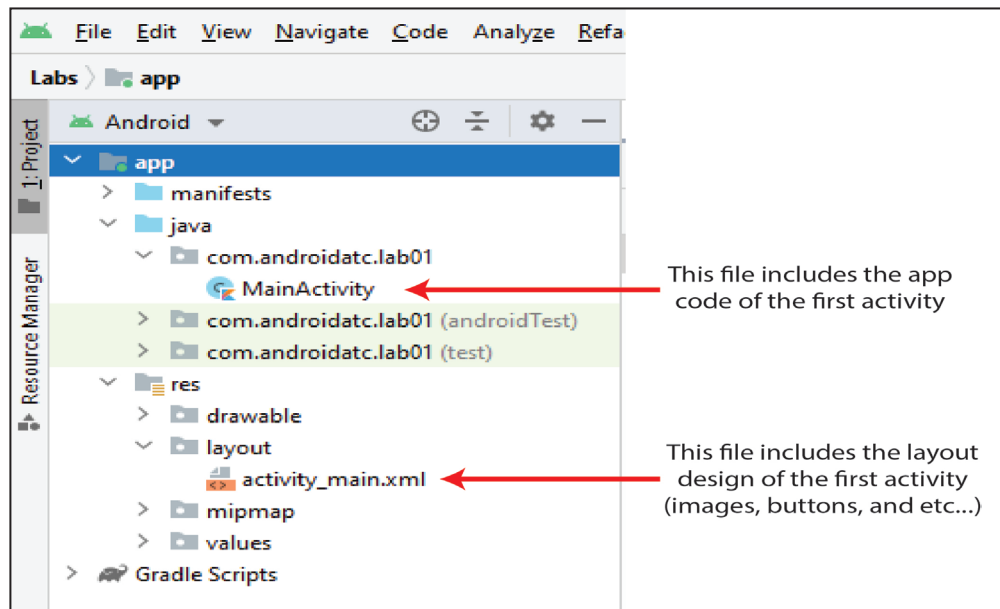
6- Open **activity_main.xml** file (**app** → **res** → **layout** → **activity_main.xml**). You will get the following figure:



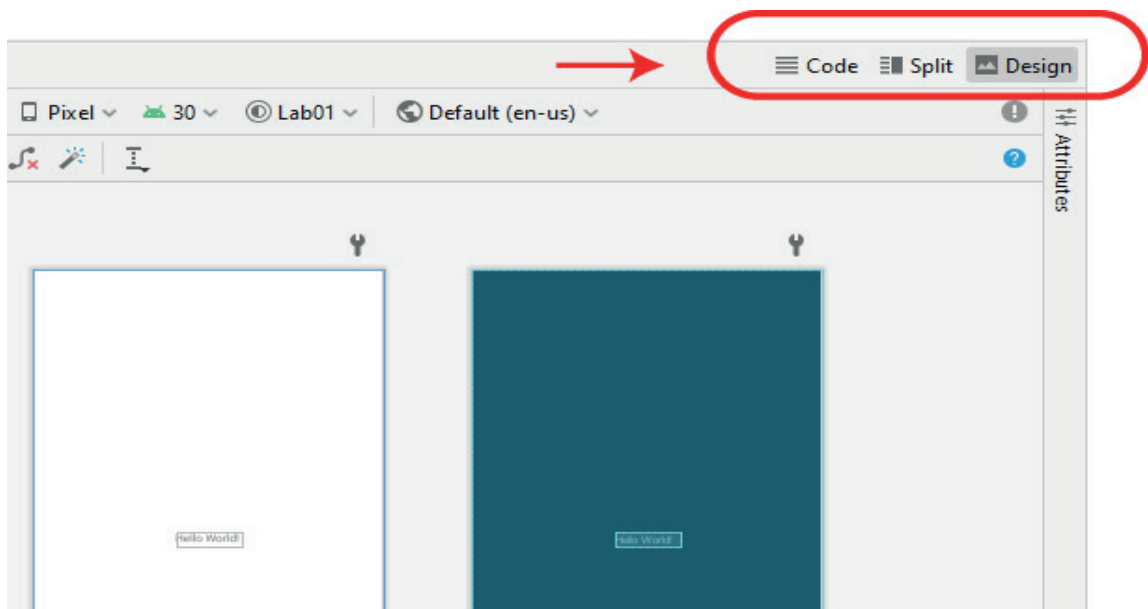
The app consists of many interfaces (activities). Usually, the startup interface for any Android app is the MainActivity file.

Each app interface has two files, XML and Kotlin files. We use the XML file to build the interface design, while we use the Kotlin file to build the code which controls on all the app workflow.

The first app interface (MainActivity) has an XML file is called: **activity_main.xml** which is used to build the user interface (layout) for this app. You can add a lot of widgets to this layout such as image, textbox, button and others. Then we will use the Kotlin file **MainActivity.kt** file to configure the calculator code as you will see in the next steps.



In the right side and as illustrated in the figure below, you can switch the view of the XML file between the design and code view, where you may use the drag and drop technique to add some widget such as textbox or buttons to your app interface, and then change the view to the code to configure how these items respond or react if the app user taps this button or enters a specific data type.



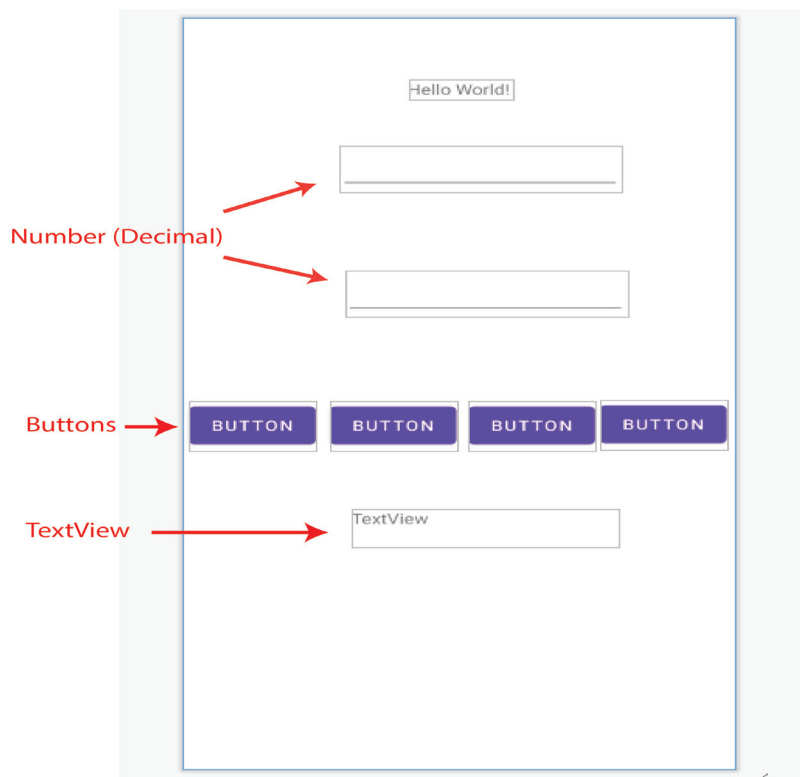
Building the Android Calculator app

In this section, you will create a simple calculator. This app will be able to add, subtract, divide or multiply two numbers and display the result.

First, you will design the **user interface (UI)** of the app using **activity_main.xml** file and then write the necessary Kotlin application code.

1- Navigate to **app** → **res** → **layout** → **activity_main.xml**, and open the file.

2- Be sure this XML file is opened in the **Design** mode. Then, move the default text : "Hello World!" to the top of this activity. From the Palette panel and using the drag and drop technique add two **Number (Decimal)** widgets, add four **Button** widgets to represent the calculator operators (Plus, Minus, Multiply and Divide), and one **TextView** widget to your activity to display the calculation result as illustrated in the figure below:



3- Open the **activity_main.xml** file in the code view, and change the value of the attribute value of the `android:text` attribute for each item as illustrated in highlight gray part of the following code:

```
<?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=".MainActivity">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Calculator App"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.084" />

    <EditText
        android:id="@+id/editTextNumberDecimal"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:ems="10"
        android:inputType="numberDecimal"
        tools:layout_editor_absoluteX="117dp"
        tools:layout_editor_absoluteY="124dp" />

    <EditText
        android:id="@+id/editTextNumberDecimal2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:ems="10"
        android:inputType="numberDecimal"
        tools:layout_editor_absoluteX="121dp"
        tools:layout_editor_absoluteY="246dp" />

    <Button
        android:id="@+id/button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="+"
        tools:layout_editor_absoluteX="5dp"
        tools:layout_editor_absoluteY="373dp" />
```

```
<Button
    android:id="@+id/button2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="-"
    tools:layout_editor_absoluteX="110dp"
    tools:layout_editor_absoluteY="373dp" />

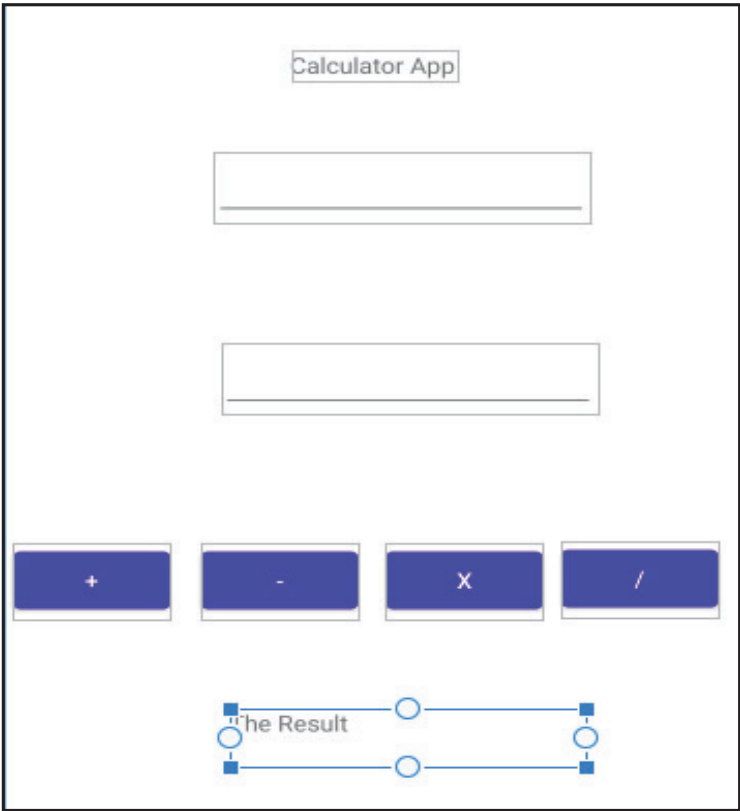
<Button
    android:id="@+id/button3"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="X"
    tools:layout_editor_absoluteX="213dp"
    tools:layout_editor_absoluteY="373dp" />

<Button
    android:id="@+id/button5"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="/"
    tools:layout_editor_absoluteX="311dp"
    tools:layout_editor_absoluteY="372dp" />

<TextView
    android:id="@+id/textView"
    android:layout_width="199dp"
    android:layout_height="37dp"
    android:text="The Result"
    tools:layout_editor_absoluteX="126dp"
    tools:layout_editor_absoluteY="478dp" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

The design view should have the following figure:



4- Now, you should give an **id** for each of these items on your app interface. You need this ID to use when you want to create the Kotlin code for this calculator. You should use a unique ID for each item. For example, in this lab you may use the following IDs:

id :Number1 hint : Enter The First Number	id : button_plus text : +	id : button_ multiply text : *	id : result_view
id : Number2 hint : Enter The Second Number	id : button_ minus text : -	id : button_ divide text : /	

The following is the **activity_main.xml** file after modifying the ID of each of these pallets as they are illustrated in the gray highlighted colors in the following code:

```
<?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=".MainActivity">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Calculator App"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.084" />

    <EditText
        android:id="@+id/Number1"
        android:hint="Enter Number 1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:ems="10"
        android:inputType="numberDecimal"
        tools:layout_editor_absoluteX="117dp"
        tools:layout_editor_absoluteY="124dp" />

    <EditText
        android:id="@+id/Number2"
        android:hint="Enter Number 2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:ems="10"
        android:inputType="numberDecimal"
        tools:layout_editor_absoluteX="121dp"
        tools:layout_editor_absoluteY="246dp" />

    <Button
        android:id="@+id/button_plus"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="+"
        tools:layout_editor_absoluteX="5dp"
        tools:layout_editor_absoluteY="373dp" />
```

```

<Button
    android:id="@+id/button_minus"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="-"
    tools:layout_editor_absoluteX="110dp"
    tools:layout_editor_absoluteY="373dp" />

<Button
    android:id="@+id/button_multiply"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="X"
    tools:layout_editor_absoluteX="213dp"
    tools:layout_editor_absoluteY="373dp" />

<Button
    android:id="@+id/button_divide"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="/"
    tools:layout_editor_absoluteX="311dp"
    tools:layout_editor_absoluteY="372dp" />

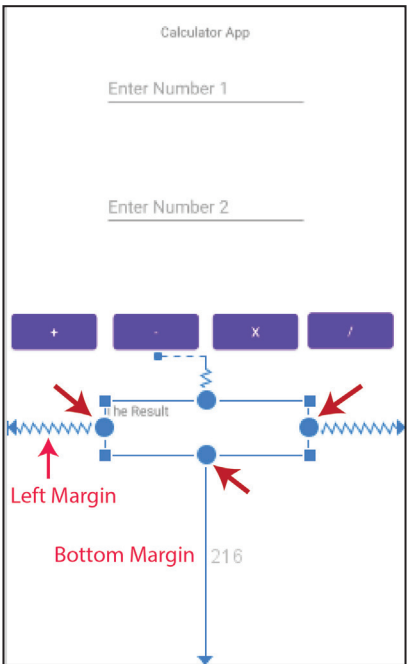
<TextView
    android:id="@+id/result_view"
    android:layout_width="199dp"
    android:layout_height="37dp"
    android:text="The Result"
    tools:layout_editor_absoluteX="126dp"
    tools:layout_editor_absoluteY="478dp" />

</androidx.constraintlayout.widget.ConstraintLayout>

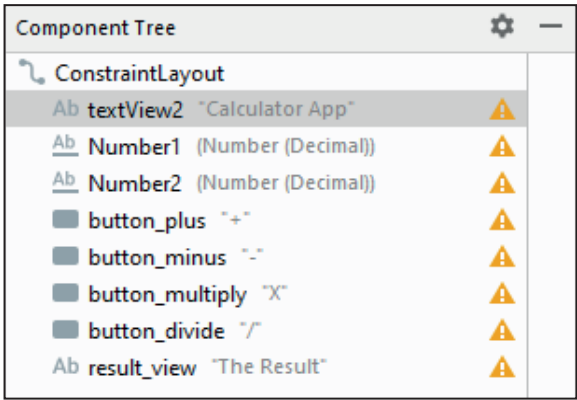
```

5- Now for importance, you must set the margins (constraints) of each button, textbox or any palette you have already added to this design layout. At least you must set the top and left margins of these items.

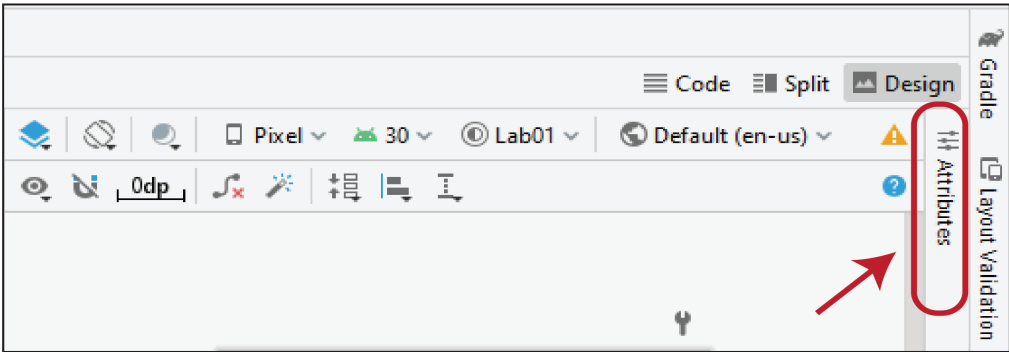
You can easily do this by moving the margin arrow of any of these items and using the blue circle which exists on the item border as illustrated in the figure below to the left phone screen edge. This will determine the left margin for this palette. Also, you must move the arrow for the palette to the bottom phone screen border to set the bottom margin. You can set the margins for any palette related to the neighbor palettes too. This will motivate Android Studio to set the margins (constraints) of each item you have moved. If you do not do this, buttons and other items will appear in unsuitable and different locations depending on the device's (phone or tablet) screen size or screen resolution which is used to run your app. This is a very important step for the design usability.



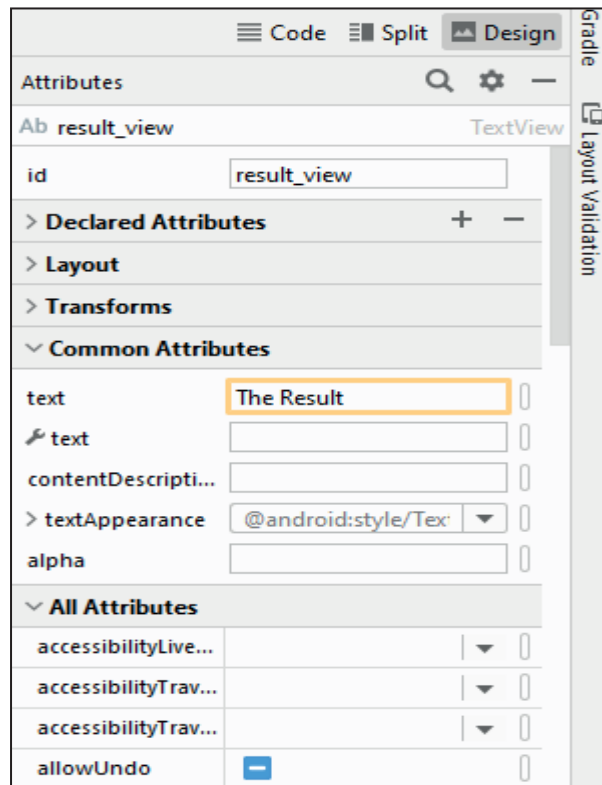
Please note that the component tree window on Android Studio gives a brief description of the **ID** of each used item on your interface as it is illustrated in the figure below:



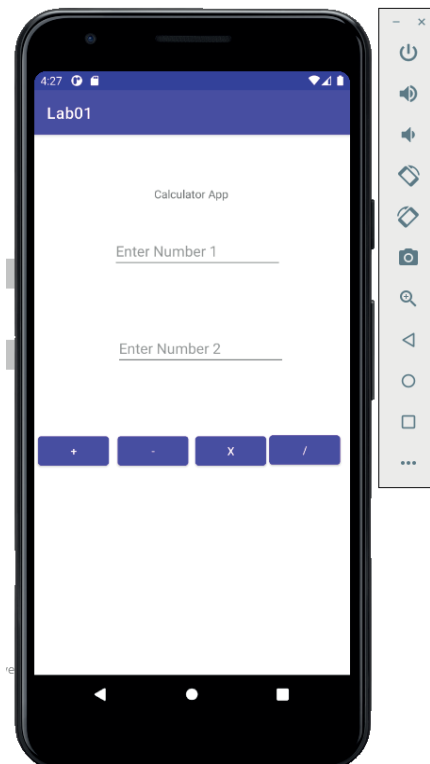
You can also manage all the attribute values for any palette by selecting the palette, then clicking the attribute tab which exists on the top right side of the Android Studio as illustrated in the figure below:



Then, you will get the following window which includes all the palette attributes. You may change the palette ID from here too.



5- Run your app and you should get a similar copy to the following figure:

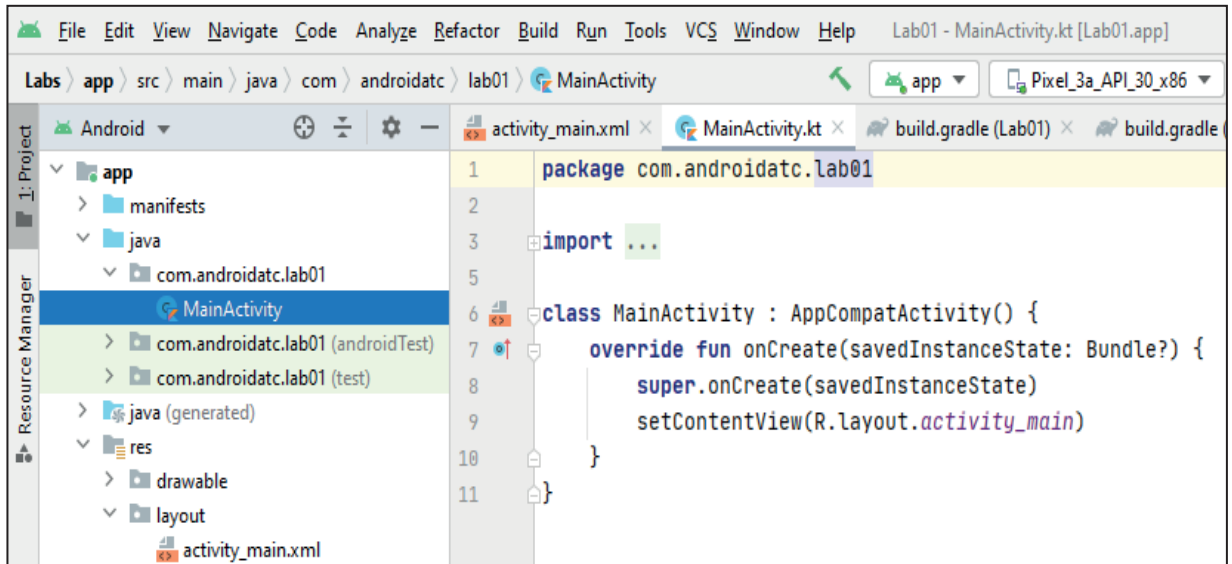


Writing the application code:

The application code is written in the **MainActivity.kt** file. The activity in Android application represents only one screen of your app, and the main activity is the first screen (interface) which appears when your app starts, like the home page of any website.

Now, start writing your calculator app code by following the steps below:

1- Navigate to the **MainActivity** file as illustrated in the following figure:

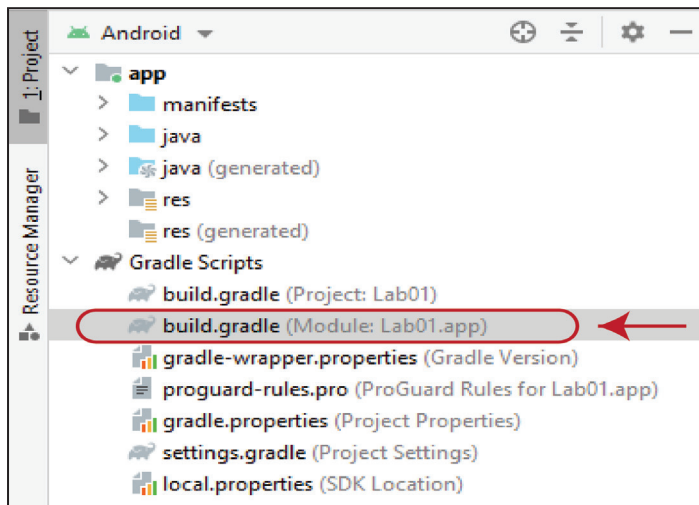


This file by default includes the following:

- **onCreate** : is the method responsible for starting the activity.
- **super**: is used to call the parent class constructor.
- **setContentView**: is used to set the xml file.

Importance Note: You can Not use the items IDs for the objects (textbox, numbers) which you have added on the XML activity file at the beginning of this lab. To do that, you need to make sure to add **kotlin-android-extensions** plug-in your module **gradle** file.

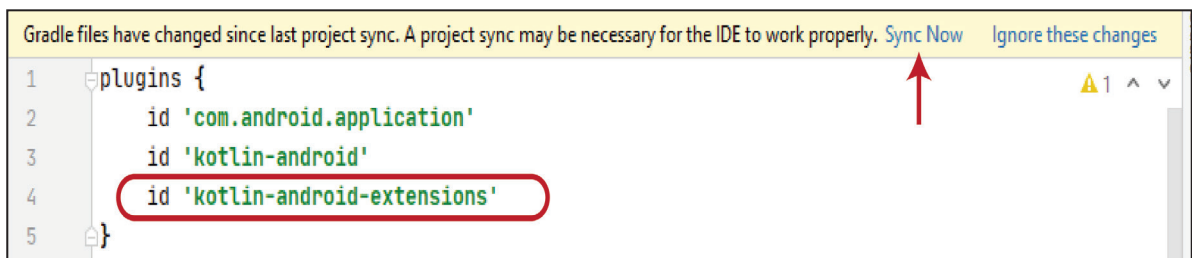
2- Open the build.gradle file (Gradle Scripts → build.gradle)



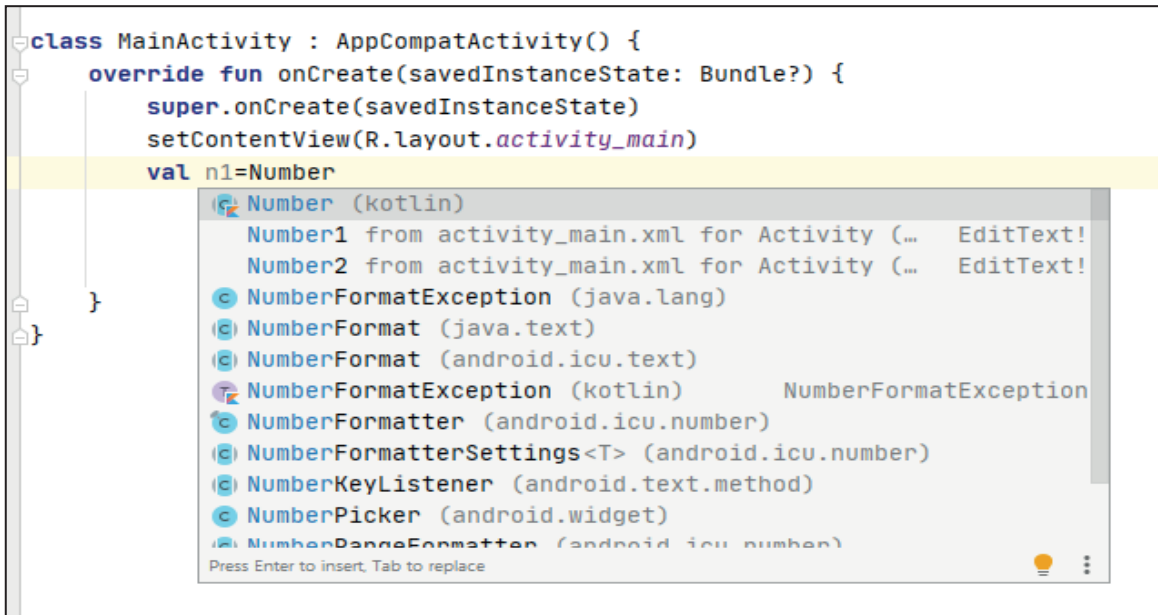
3- Check if the **kotlin-android-extensions** plug-in is already configured. As you see in the figure below, it is not added yet.



4- Add the **kotlin-android-extensions** plug-in to this gradle file, and click **Sync Now** as illustrated in the figure below:



5- Now, back to the MainActivity file, and configure variable **n1** to represent the first decimal number in your calculator app (Number1). As you see in the print shot below, when you call this number using its ID, the auto-fill features is working now:



You should add the following code:

```
class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
        val n1=Number1.text
        val n2=Number2.text
    }
}
```

Here, **n1** variable value represents the number which will be typed in the first text field. The same thing for **n2** variable value which represents the number which will be typed in the second text field of this calculator app.

6- Now, start with the code of the plus button (ID= **button_plus**) which is responsible for calculating the sum of Number1 and Number2 (n1 and n2).

The app user will tap the plus sign (button) to implement the sum calculation of **n1** and **n2** numbers. To do that, you should use a new method called **onClickListener**. When the user taps the plus sign (the button which has Id: button_plus) it performs the action which will be added as follows:

Remember here the plus button **id** is : **button_plus**


```

class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
        val n1=Number1.text
        val n2=Number2.text
        button_plus.
    }
}

```

setOnClickListener {...} (l: ((View!) -> Unit?))	Unit
setOnClickListener(l: View.OnClickListener?)	Unit
accessibilityClassName (from getAccessibi...	CharSequence!
accessibilityDelegate (from... View.AccessibilityDelegate!	
accessibilityLiveRegion (from getAccessibilityLiveR...	Int

The full code of the main activity is as follows:

```

class MainActivity : AppCompatActivity() {

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
        val N1=Number1.text
        val N2=Number2.text

        button_plus.setOnClickListener {

        }

    }
}

```

6- Inside the method `button_plus.setOnClickListener { }`, you should write the code that produces the sum of **Number1** and **Number2**. The code is as follows:

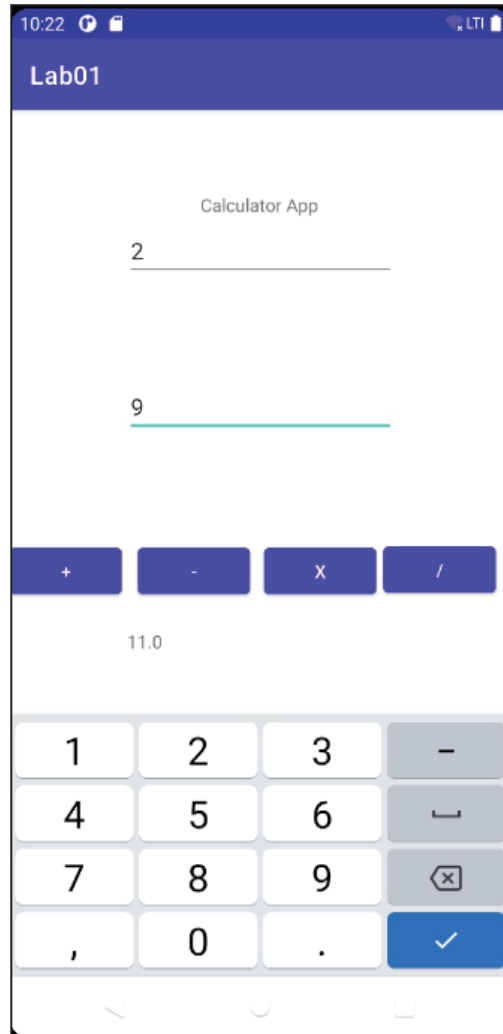
```

package com.androidatc.lab01

import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import kotlinx.android.synthetic.main.activity_main.*
class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
        val n1=Number1.text
        val n2=Number2.text
        button_plus.setOnClickListener {
            val sumResult= n1.toString().toDouble()+ n2.toString().toDouble()
            result_view.text= sumResult.toString()
        }
    }
}

```

7- Click **Run** to start your app and wait until your app is started. To test the sum button, for example, enter the First Number: **5** and Enter The Second Number : **9**, then click the **+** operator. You will get the following result as illustrated in the figure below:



8- Complete the remaining code for the other operators (-, * and /). To save the time and effort, you may use copy and paste technique for the `setOnClickListener` method and just change the button ID and the operator sign inside this method for the other operators (-, *, and /). The code is as follows:

```
class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
        val n1=Number1.text
        val n2=Number2.text
        button_plus.setOnClickListener {
            val sumResult= n1.toString().toDouble()+ n2.toString().toDouble()
            result_view.text= sumResult.toString()
        }

        button_minus.setOnClickListener {
            val sumResult= n1.toString().toDouble()- n2.toString().toDouble()
            result_view.text= sumResult.toString()
        }

        button_multiply.setOnClickListener {
            val sumResult= n1.toString().toDouble()* n2.toString().toDouble()
            result_view.text= sumResult.toString()
        }

        button_divide.setOnClickListener {
            val sumResult= n1.toString().toDouble()/ n2.toString().toDouble()
            result_view.text= sumResult.toString()
        }
    }
}
```

It is recommended to stop the running app, and then run it again.

Note: If you want to test your application more than one time, you don't need to close the emulator device. You can keep it open and when you click run again, the application will start without waiting for the Android system to reboot. This will save a lot of time during the development period.

Note: If you want to remove the “**The Result**” label which appears on the phone screen when the application starts, go to the attribute of this text in the design view, and delete only the **text** value attribute.

Congratulations! You have designed your first Android application.