CSE2MAD TUTORIAL 1: HELLO ANDROID

LEARNING OBJECTIVE: INTRODUCTION TO ANDROID STUDIO IDE, SIMPLE ANDROID APP

PREPARATION FOR LAB
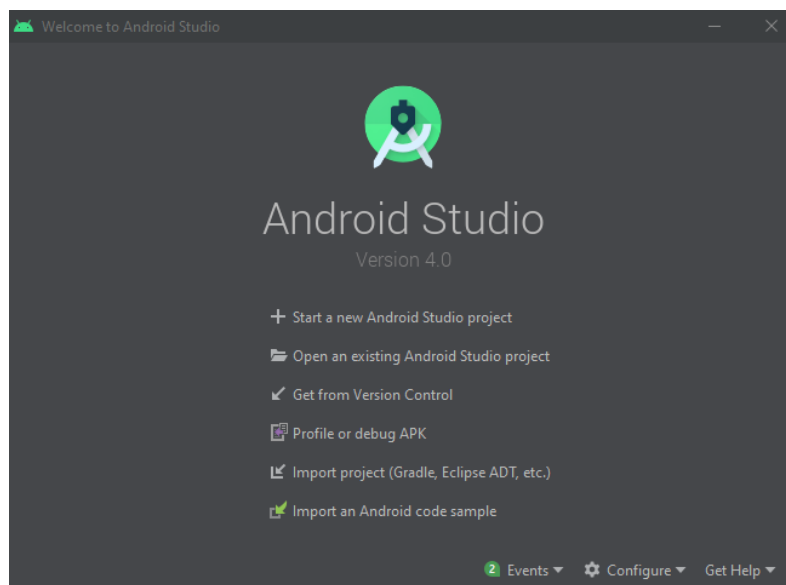
Download & Install Android Studio from https://developer.android.com/studio/install.html

LAB 2 TASKS: BUILD YOUR FIRST ANDROID APPLICATION - A HELLO ANDROID APP

Further info: https://developer.android.com/training/basics/firstapp/index.html

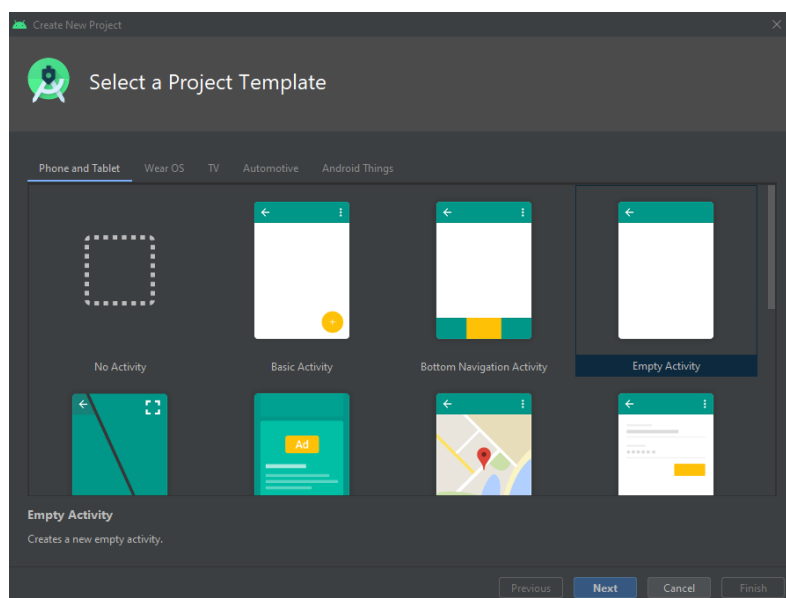**STEP 1- CREATE A NEW PROJECT & AN EMPTY ACTIVITY:**
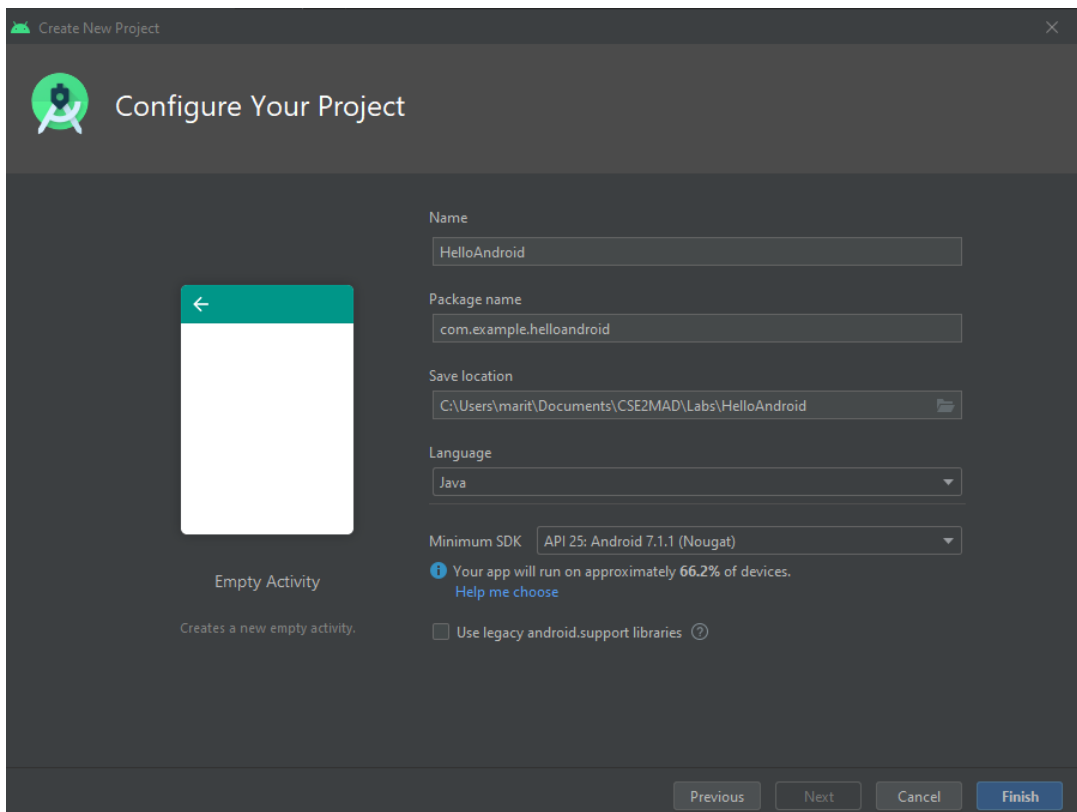
a) Create a New Android Project: Click on 'Start a new Android Studio project'

b) Select empty activity in the "Phone and Tablet" tab and press "NEXT"

c) Type Application name, domain, location to save the project (chose where ever suits your file system). Select 'Phone & Tablet' as the form factor and "API 25: Android 7.1.1" as the minimum SDK. Click Next.

Your Android project is now ready and will open to the MainActivity.java file, which is the empty activity created during the project set-up.

An Activity is a single application entity that is used to perform actions. An application may have many separate activities, but the user interacts with them one at a time. The onCreate() method is called by the Android system when your Activity starts — it is where you should perform all initialization and user interface setup.
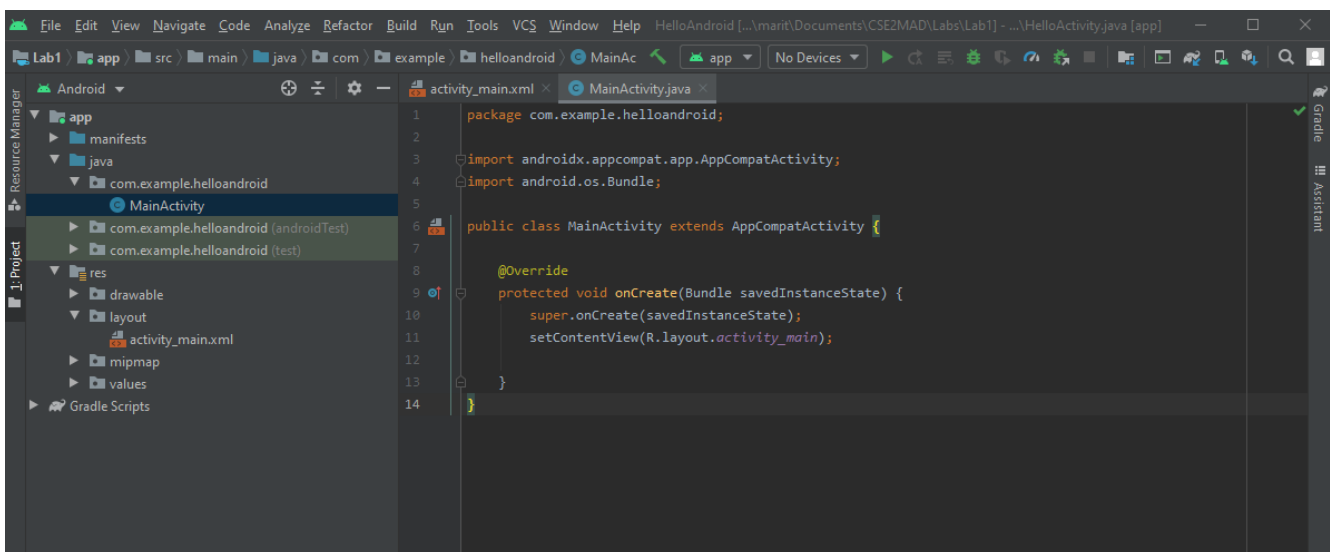
STEP 2- ADD A UI ELEMENT:

An Android user interface is composed of hierarchies of objects called Views. A View is a drawable object used as an element in your UI layout, such as a button, image, or (in this case) a text label. Each of these objects is a subclass of the View class and the subclass that handles buttons is android.widget.Button.

a) In the Project Tab of your Android Studio IDE, click on the activity_main.xml located in the res/layout folder. Double click to open in the editor.

b) Make sure 'Design' tab on the top right is selected. Can you see the UI displayed?
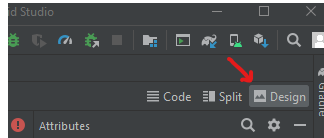
c) Locate the placeholder TextView "Hello World" in the Component Tree and press delete to remove it so we can begin with a clear screen.
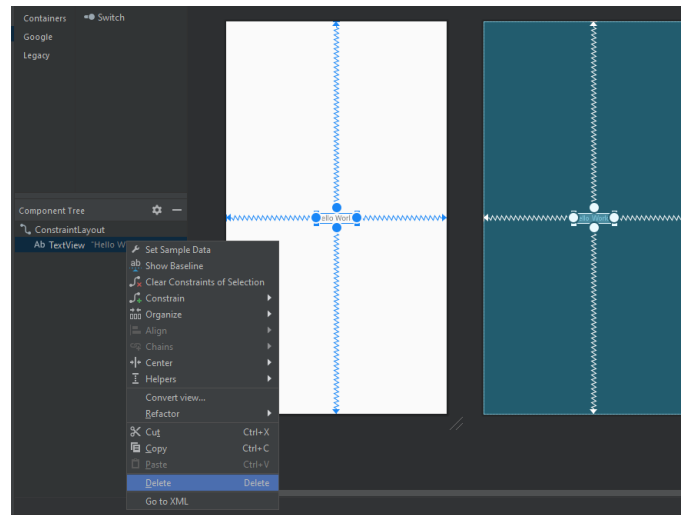
d) Select a 'Button' widget from the Pallet and drag it on to the UI.
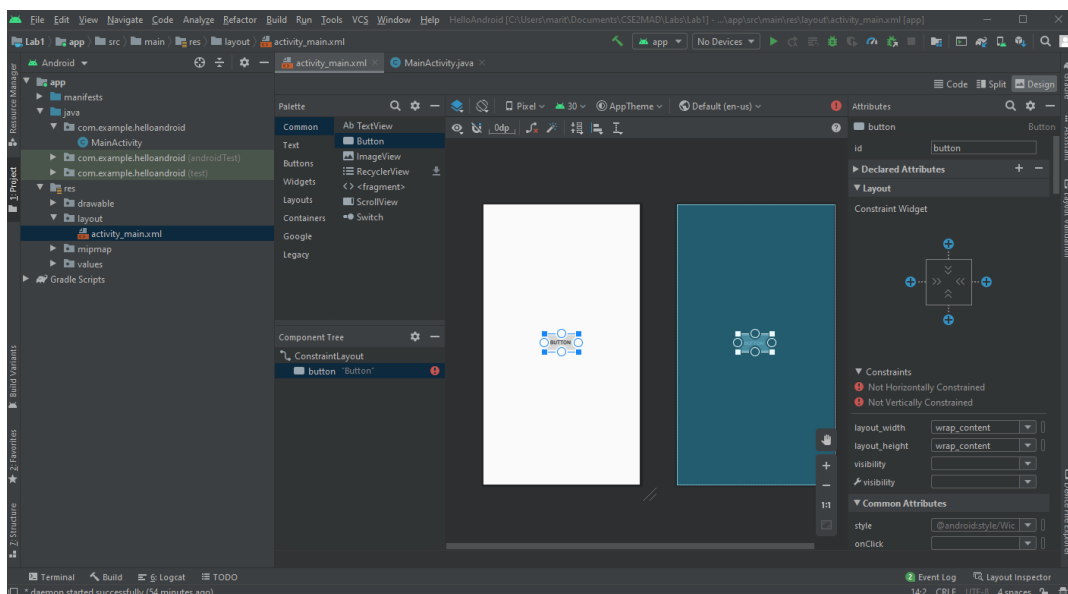
e) ClickOn the 'Attributes' panel on the right, locate the id field(top) and click on the autogenerated 'button' text and change the id to 'btnhello'.
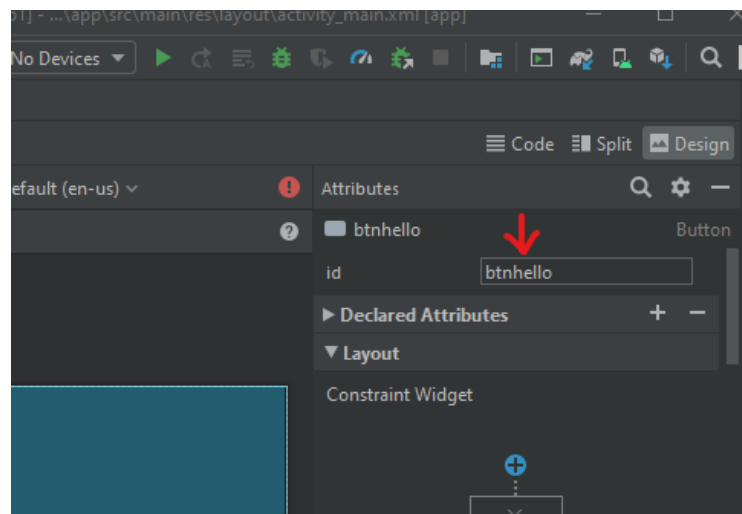
f) Look further down the Attributes panel and change the 'text' property to 'Click Me'.

g) Now your UI should display as below and is now ready for testing.

STEP 3A- RUNNING YOUR APPLICATION ON A VIRTUAL DEVICE

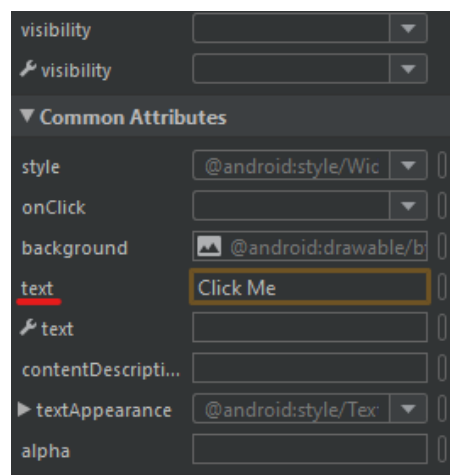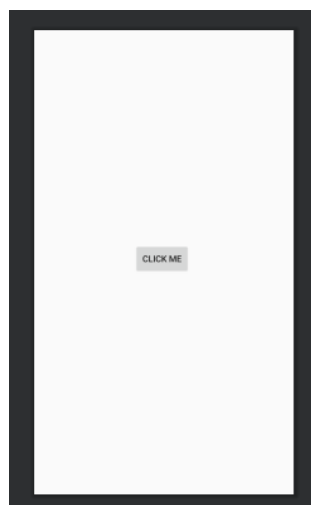a) To run your app on the emulator you need to first create an Android Virtual Device (AVD). An AVD is a device configuration for the Android emulator that allows you to model different devices.

To create an AVD click the 'Avd Manager' button on the top toolbar and select "Create Virtual Device"
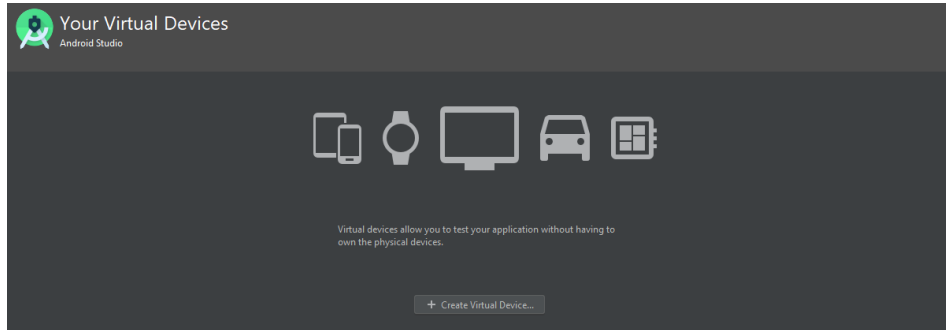
b) Select the Hardware you would like to emulate (Nexus 5X) and press Next
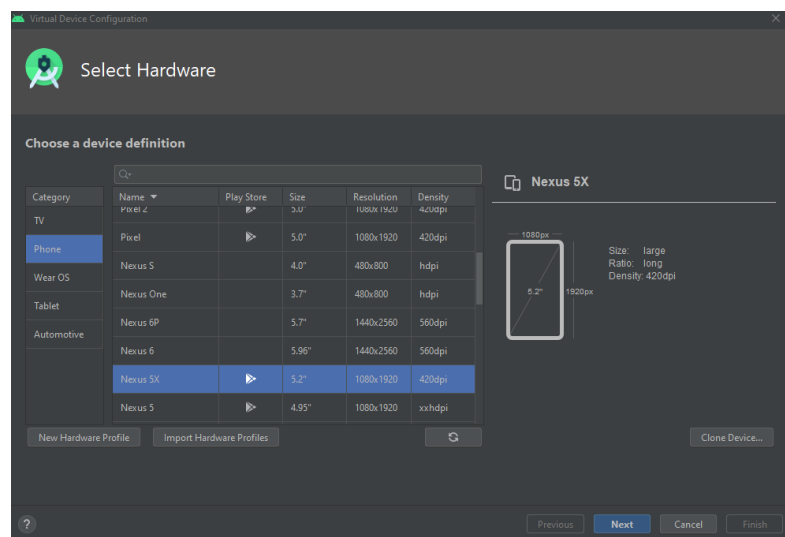
c) Click on 'download" and accept the license agreement. The Component installer will then download the System Image. Then press 'Next' to continue.

5
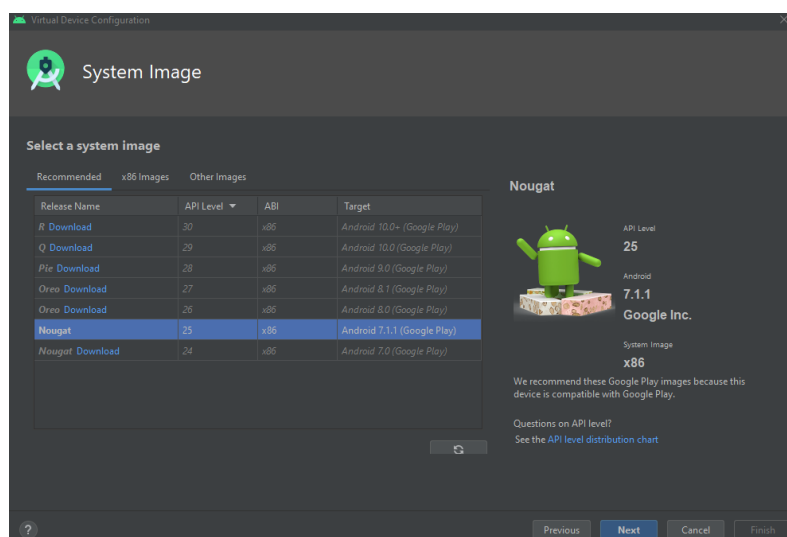
d) Now you can press 'Finish' to complete the creation of your first emulator.
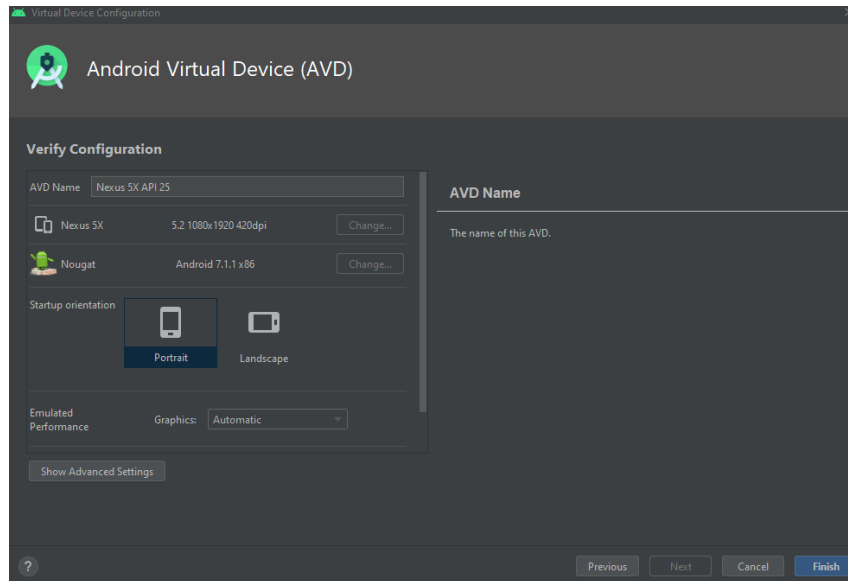
e) Note the Virtual Devices are LARGE files (As you can see below), I would suggest initially only creating one or 2, an older model such as we are creating now to give us coverage, and in your own time perhaps a Pixel with Android 10 to experience cutting edge functionality.

Type	Name	Play Store	Resolution	API	Target	CPU/ABI	Size on Disk	Actions
📱	Nexus 5X API 25	▷	1080 × 1920: 420dpi	25	Android 7.1.1 (Google Play)	x86	1.0 GB	▶ ✎ ▾
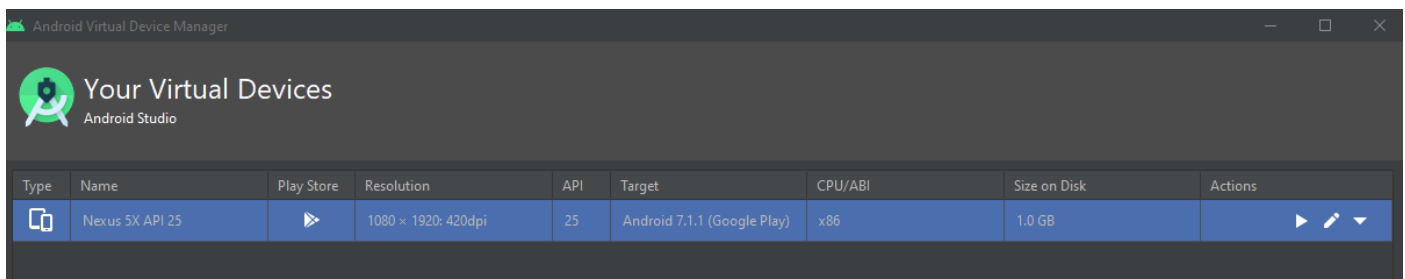
f) Now click the 'Run' button ▶ (in the top toolbar) to launch the Virtual Device you have created.

6
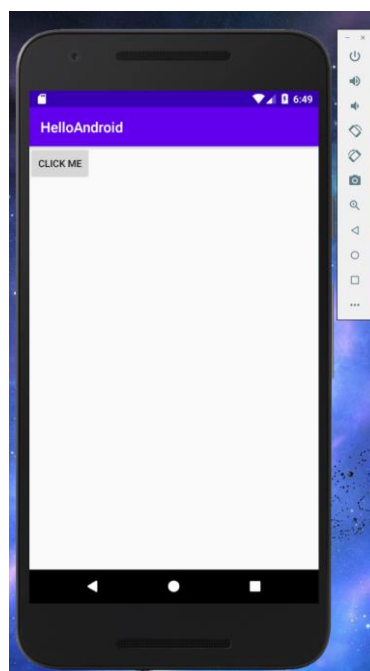
STEP 3B- RUNNING YOUR APPLICATION ON A DEVICE

If you have a real Android Powered device, you can also install and run your app from Android Studio.

a) Prepare your device for development by opening **Settings** > **Developer Options** and enable **USB Debugging**. If you do not have developer options please follow the instructions at https://developer.android.com/studio/debug/dev-options.

b) Prepare your System for development.

WINDOWS

You may need to install the appropriate USB driver for your device, however I would try without doing so first then if it doesn't work follow the instructions at https://developer.android.com/studio/run/oem-usb

MAC

No config required

LINUX

- **Ubuntu Linux**: Use `apt-get install` to install the `adb` package. This gives you a community-maintained default set of `udev` rules for all Android devices.

  Make sure that you are in the plugdev group. If you see the following error message, adb did not find you in the plugdev group:

  ```
  error: insufficient permissions for device: udev requires plugdev group membership
  ```

  Use `id` to see what groups you are in. Use `sudo usermod -aG plugdev $LOGNAME` to add yourself to the plugdev group.

  The following example shows how to install the Android adb tools package.

  ```
  apt-get install adb
  ```

Linux Instructions [2]

c) When you are set up and plugged in over USB, you can click Run ▶ in Android Studio to build and run your app on the device.

d) Android Studio installs the app on your connected device and starts it. The "HelloWorld" you see in the blue bar is the application title. Android Studio creates this automatically (the string is defined in the res/values/strings.xml)

** SEE THE OPTIONAL ADVANCED CONCEPTS SECTION ON PAGE 12 FOR SOME MORE INFORMATION ON LAYOUT AND ERRORS.

STEP 4- MODIFYING SOME CODE

a) When you run your application, try clicking on the 'Click Me' button we added. Does anything happen? This is because we did not specify what the app should do when the button is clicked. Lets do this now. Let us now write some code to print a message every time the button is clicked. For the purposes of this exercise, we will print to the Logcat display. https://developer.android.com/studio/command-line/logcat.html
Logcat is a very useful tool that lets developers print/log messages as they develop/debug android applications. (This is something worth noting as you will use this very often!)

b) In the file **res/layout/activity_main.xml**, add the **android:onClick** attribute to the <Button>element as shown below:

```xml
activity_main.xml        MainActivity.java
1    <?xml version="1.0" encoding="utf-8"?>
2    <androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
3        xmlns:app="http://schemas.android.com/apk/res-auto"
4        xmlns:tools="http://schemas.android.com/tools"
5        android:layout_width="match_parent"
6        android:layout_height="match_parent"
7        tools:context=".MainActivity">
8
9        <Button
10           android:id="@+id/btnhello"
11           android:layout_width="wrap_content"
12           android:layout_height="wrap_content"
13           android:text="@string/click_me"
14           android:onClick="sayHello"
        ✕ Suppress: Add tools:ignore="OnClick" attribute    tomOf="parent"
        💡 Create 'sayHello(View)' in 'MainActivity'    ▶    "parent"
17           app:layout_constraintStart_toStartOf="parent"
18           app:layout_constraintTop_toTopOf="parent" />
19   </androidx.constraintlayout.widget.ConstraintLayout>
```

Here, we are saying that, when the button's onClick event happens, execute the method 'sayHello'. However, we have not yet declared the 'sayHello' method in code - > We will do so in the next step. Notice how Android Studio actually notices this and displays a red light bulb to warn us of the error. You can use the code assist by clicking on the Create 'sayHello…' option **OR** manually type the 'sayHello' method in the HelloActivity.java file. For today I would recommend using the code assist. https://developer.android.com/guide/topics/ui/controls/button#HandlingEvents

c) Open HelloActivity.java in java folder. You should see the newly created sayHello method:

```java
activity_main.xml        MainActivity.java
1    package com.example.helloworld;
2
3    import androidx.appcompat.app.AppCompatActivity;
4
5    import android.os.Bundle;
6    import android.view.View;
7
8    public class MainActivity extends AppCompatActivity {
9
10       @Override
11       protected void onCreate(Bundle savedInstanceState) {
12           super.onCreate(savedInstanceState);
13           setContentView(R.layout.activity_main);
14       }
15
16       public void sayHello(View view) {
17       }
18   }
```

Modify the sayHello method to include the following code:

```java
//Called when the user clicks on the button
public void sayHello(View view) {
    Log.d( tag: "Test",  msg: "You Clicked");
}
```

You may need to hover over 'Log' to automatically import the java library using the Alt-Enter shortcut, you will see it appear at the top of the file. This method will fire every time you click on the 'Click Me' button

8

d) Run the modified app. You should see the "You Clicked" message on the logcat output at the bottom of your IDE when you click the button.
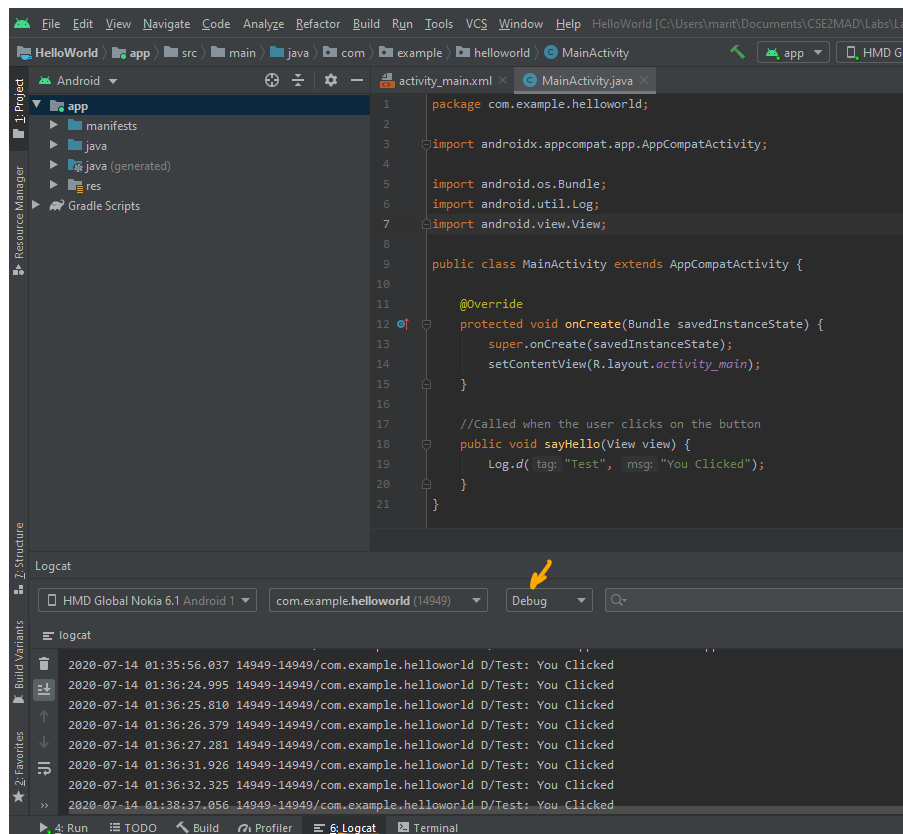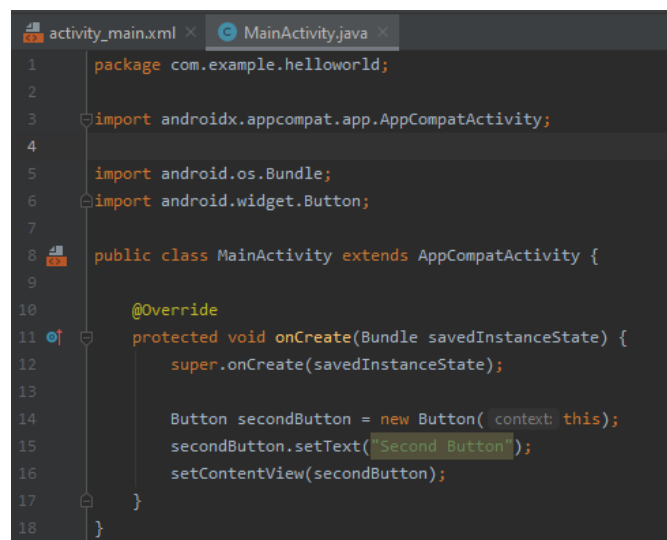

STEP 5 - ALTERNATE METHODS TO ADD A UI ELEMENT

In this example, we used the graphical designer to add UI elements.
However, note that you can also do this:

- Programmatically, or
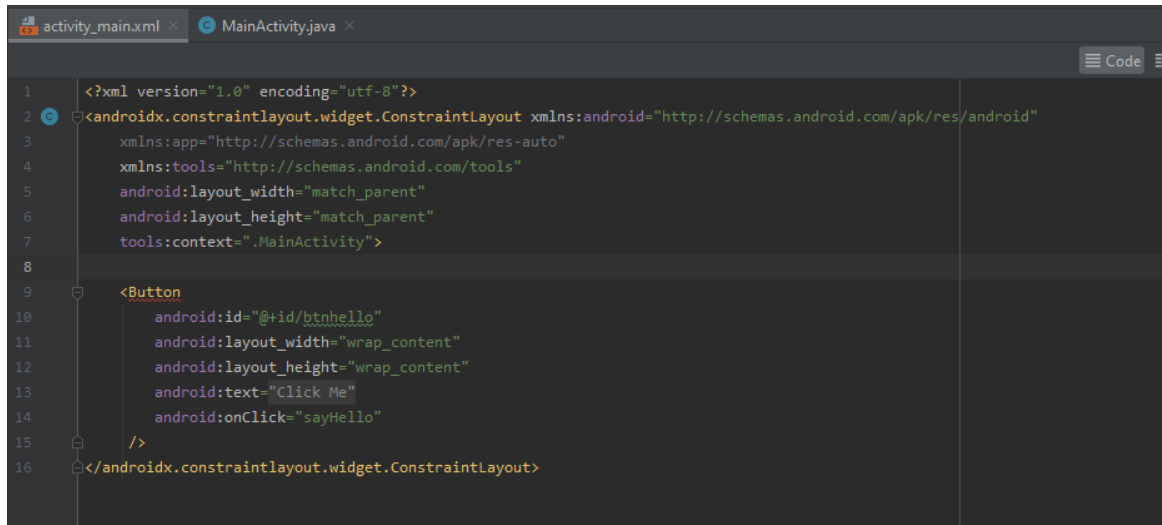- By editing the XML-based layout file

PROGRAMMATICALLY ADDING A BUTTON:

Clear the exiting code in MainActivity.java & try out the following code, what happens?

ADD A BUTTON VIA EDITING THE XML-BASED LAYOUT FILE:

Open the activity_main.xml file and look at its contents(In CODE mode) You should see something like this

```xml
<?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">

    <Button
        android:id="@+id/btnhello"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Click Me"
        android:onClick="sayHello"
    />
</androidx.constraintlayout.widget.ConstraintLayout>
```
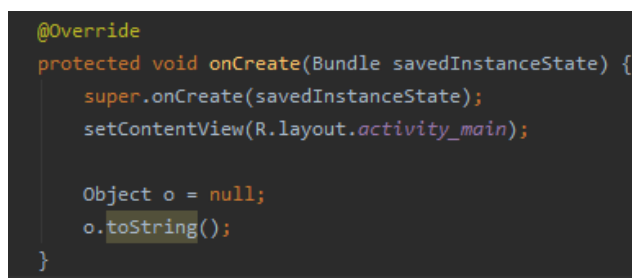
Instead of dragging from the graphical design editor, you can directly add UI elements as XML. The general structure of an Android XML layout file is simple: it's a tree of XML elements, wherein each node is the name of a View class (this example, however, is just one View element). You can use the name of any class that extends View as an element in your XML layouts, including custom View classes you define in your own code. This structure makes it easy to quickly build up UIs, using a more simple structure and syntax than you would use in a programmatic layout. This model is inspired by the web development model, wherein you can separate the presentation of your application (its UI) from the application logic used to fetch and fill in data.

STEP 6 - DEBUGGING

https://developer.android.com/studio/debug/index.html
Debugging is an essential skill that is vital for a programmer.
The Android Studio also has excellent debugger. To demonstrate this, introduce a bug into your code. Change your onCreate() method in source code to look like this:

```java
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    Object o = null;
    o.toString();
}
```

Run your code, what happens?

This change simply introduces a NullPointerException into your code.

To find out more about the error, set a breakpoint in your source code on the line Object o = null; (click on the marker bar next to the source code line). Then select Run > Debug from the menu to enter debug mode. Your app will restart in the emulator, but this time it will suspend when it reaches the breakpoint you set. You can then step through the code in Android Studio's Debug Perspective, just as you would for any other application.
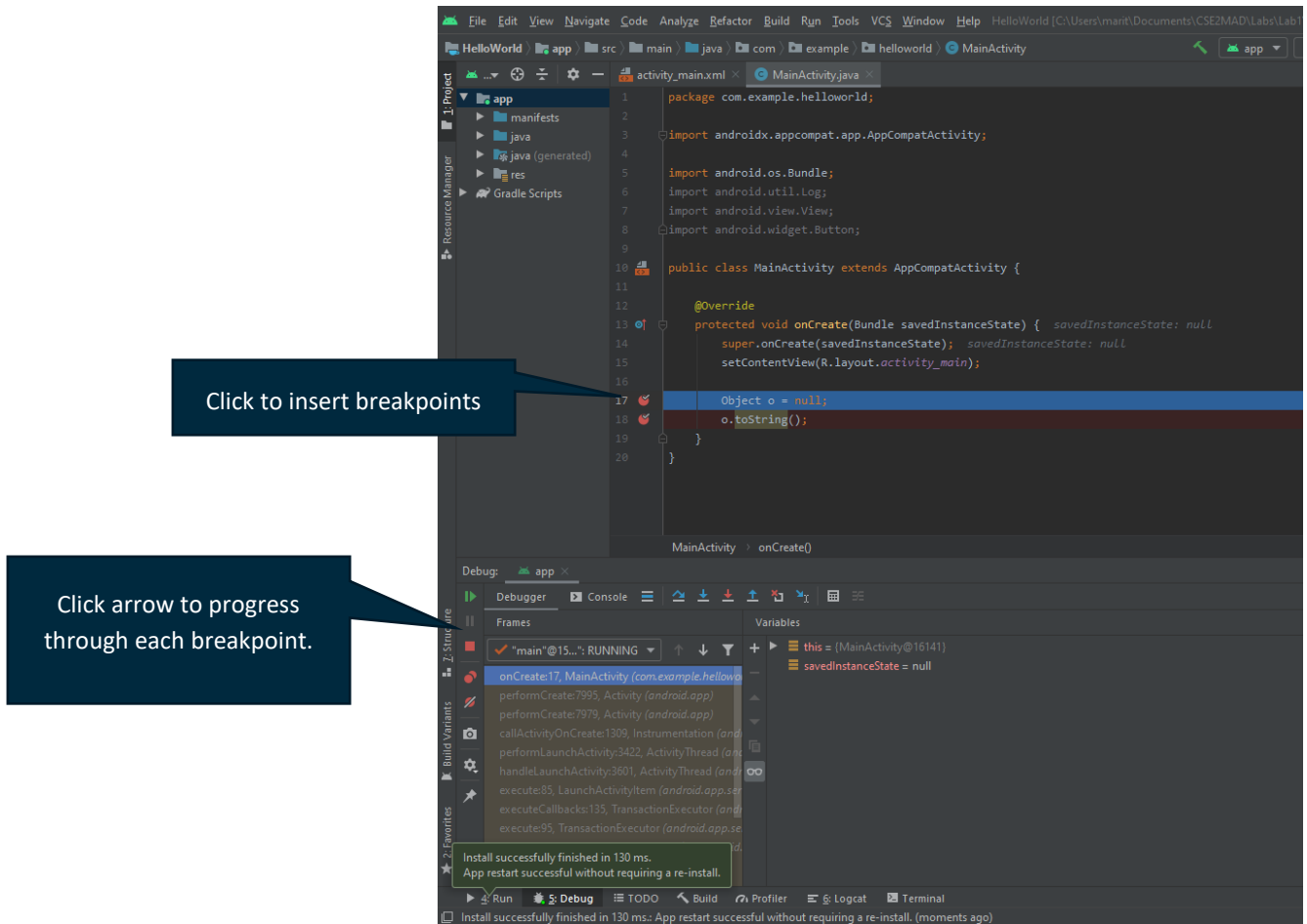
We will be discussing these concepts in the upcoming weeks, however if you were wondering why your button was floating around and what is going on with all the alarming exclamation marks read on 😊

FIXING THE ERROR NOTIFICATIONS

a) You will note that your button is not sitting nicely in the centre of the screen in either your virtual / real device, why? Have a look at the top right, you will notice a red exclamation mark which indicated there is an **Error**, click on it to expand the Warning and Errors in the bottom panel.
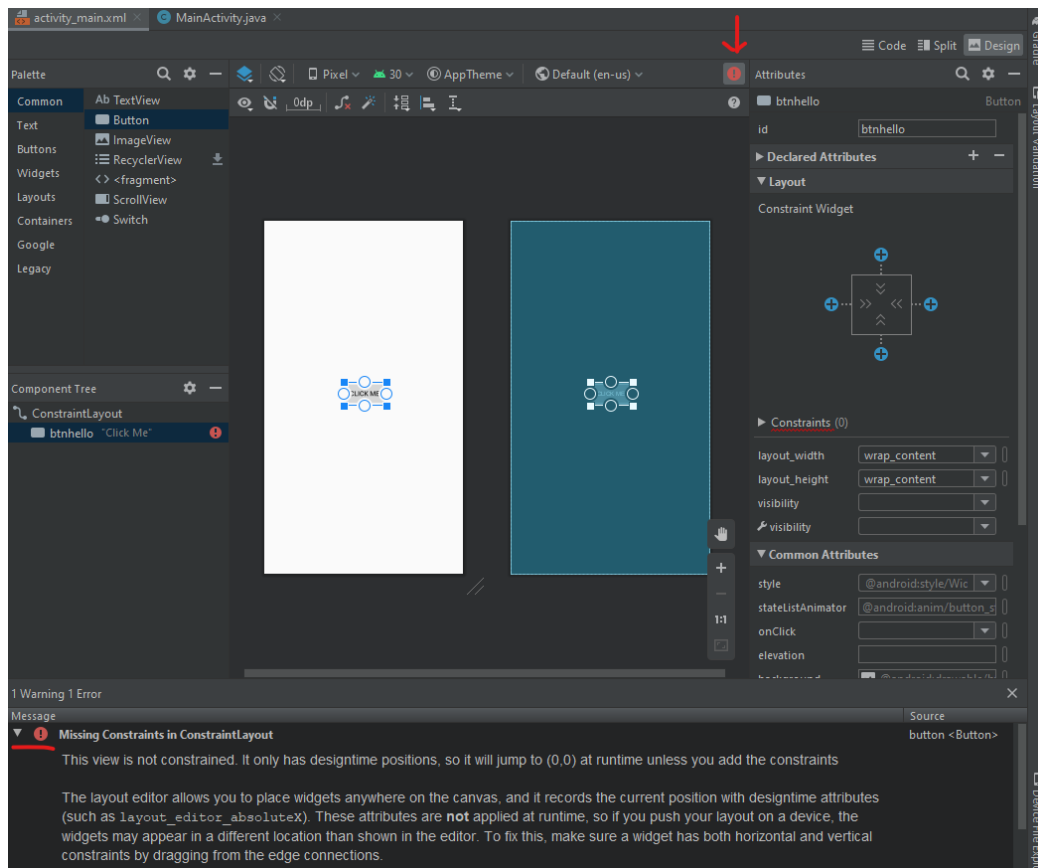

b) As the error indicates, we are missing the Constraints in ConstraintLayout, so what is this? We will cover Layouts in more detail later but what is beneficial now is that you get comfortable in experimenting with the Android Studio Layouts. To make a functional layout every view must have at least two constraints: one horizontal and one vertical. In this case we will set 4 constraints by clicking on the + symbols in the Layout > Constraint Widget on the right. Use the number fields to set your desired margins, I have used 0. For more information about the Constraint Layout you can view https://developer.android.com/training/constraint-layout?hl=en
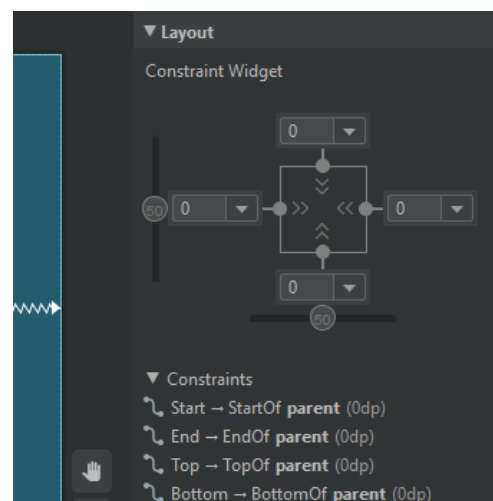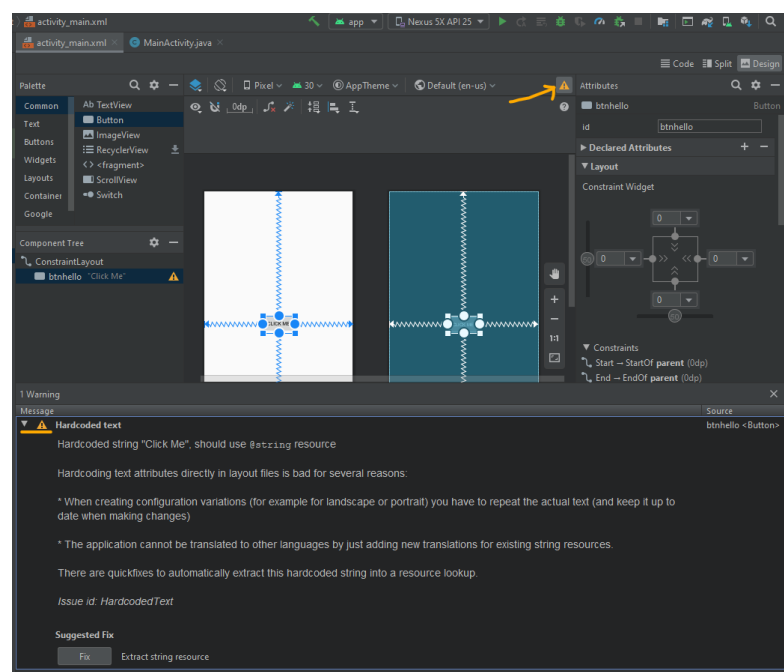
c) Check your new layout using the Layout Validation Tool available by clicking 'Layout Validation on the right border of Android Studio.
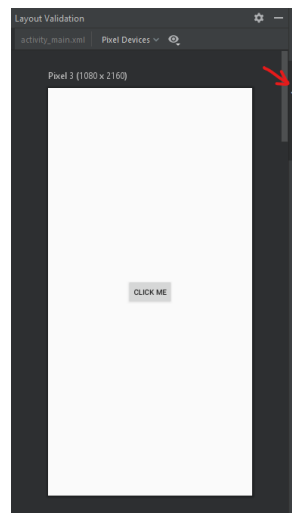

d) Have we finished fixing the problems? No? Why? Look at the top right, you will see a yellow exclamation mark, which is a Warning, like before click on it to expand the error details.

e) So, what's wrong this time? This is not a crucial error and the app will still run; however, it is warning in this instance that poor coding has occurred, with the button text being automatically hard coded into the XML. You can go back to the code by clicking the Code button at the top right. The error will be highlighted and if you hover over it with your mouse (or click the yellow lightbulb) a suggested fix will appear.
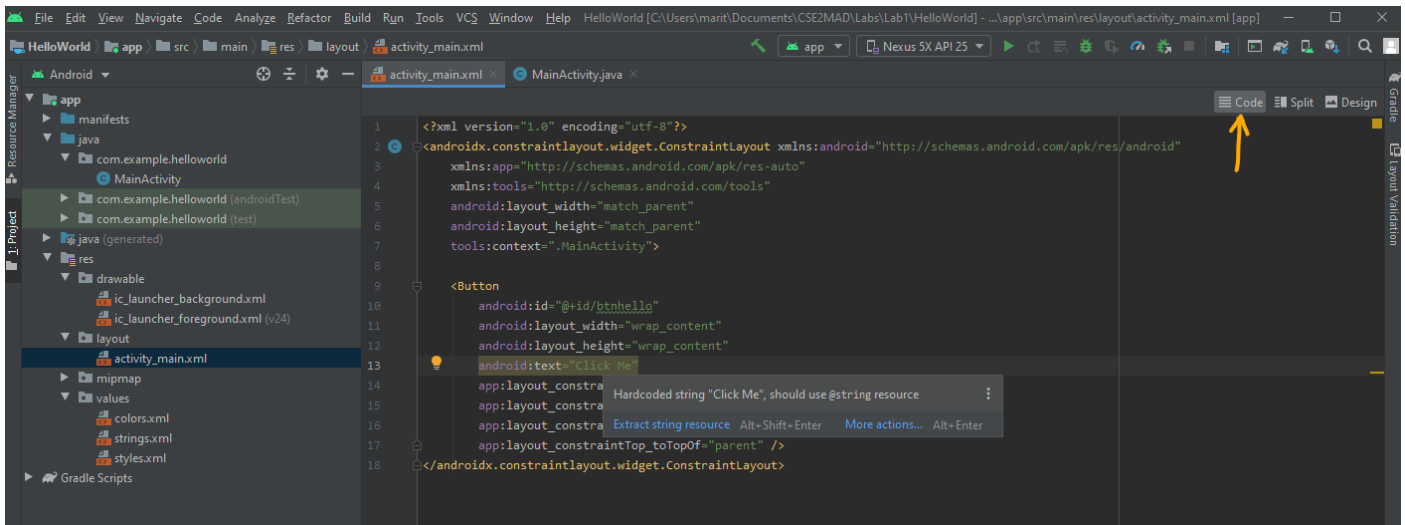
f) As the hint suggests, hover your mouse over the 'Click Me' string and use Alt + Shift + Enter to extract the string to the res/values/strings.xml file. Why does this matter? This I18N message is informing us that a potential issue exists with regard to the future internationalization of the project. The warning is reminding us that when developing Android applications, attributes and values such as text strings should be stored in the form of resources wherever possible. Doing so enables changes to the appearance of the application to be made by modifying resource files instead of changing the application source code. This can be especially valuable when translating a user interface to a different spoken language [1].

14
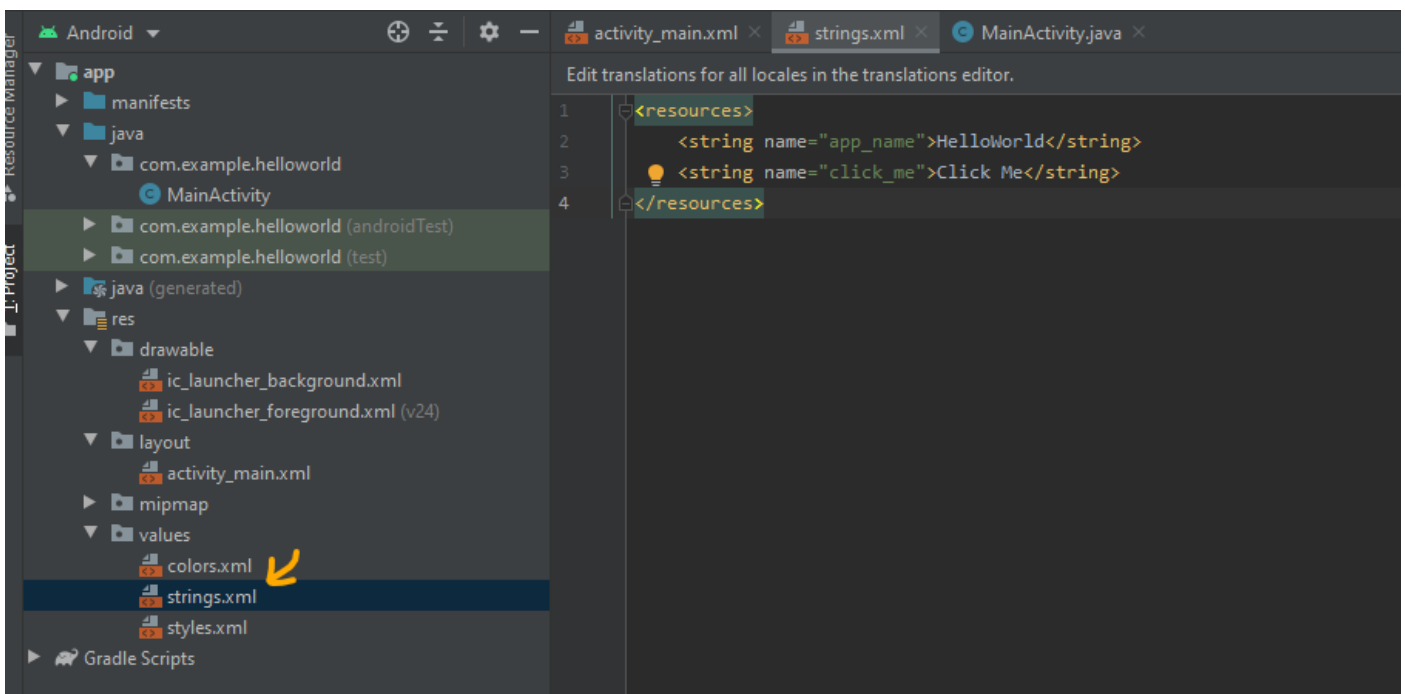
REFERENCES

[1] N. Smyth, *Android Studio 4.0 Development Essentials - Java Edition*. Payload Media, 2020.

[2] 'Run apps on a hardware device | Android Developers'. https://developer.android.com/studio/run/device (accessed Jul. 14, 2020).