Lesson 0 - Your First Android App

Introduction (Read this first)

You will complete this lesson outside of lesson time.

The instructions given in this lesson will show you how to do a “hello world” Android app. Assuming that you have a fresh installation of Android Studio, a significant amount of time is consumed in downloading components from the internet.

You should allocate 45 mins to 1 hour, together with a fast and cheap (hopefully free) internet connection for this task. Some downloads may take some time, so plan something else to do during that period.

Building android apps is computationally intensive. Your laptop can get warm very quickly and could possibly overheat (which I experienced). Do take the appropriate steps.

This tutorial was written with a freshly installed Android Studio in a new MacBook. Hence, what you see on your computer could be different, just respond accordingly to the instructions on the user interface.

Try to get an android phone if you do not have one. You may find that testing your app on a physical android mobile phone is much faster than using the emulator.

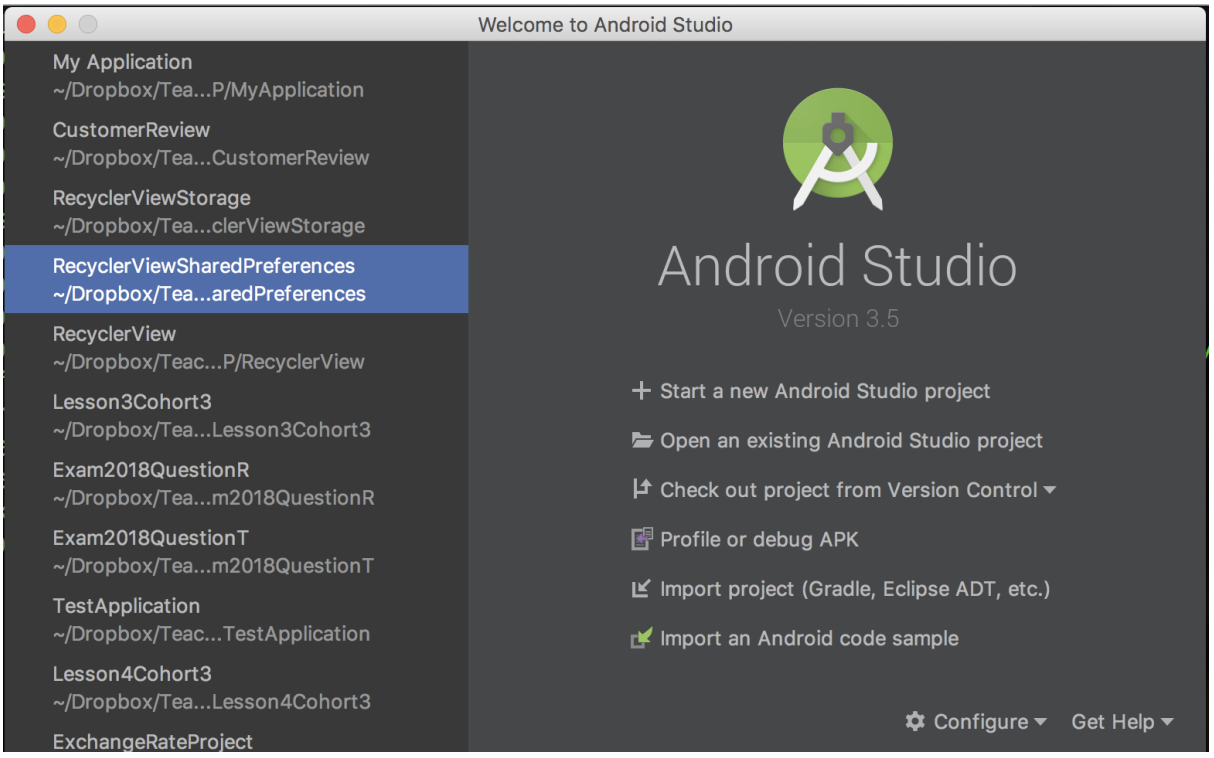
If you really want to use an emulator, you may use the default AVD manager provided by Android Studio, but very often I find that it is slow and consumes a lot of your computer’s resources. Alternatively, if you really wish to use an emulator, you could consider **Genymotion**, but I find the installation process troublesome too.

This section is based on the Android Developer Fundamentals Version 2 Lesson 1.1 Codelab.

Create a “hello world” app

Start a new android studio project

Launch Android Studio from your computer and select **Start A New Android Studio Project**



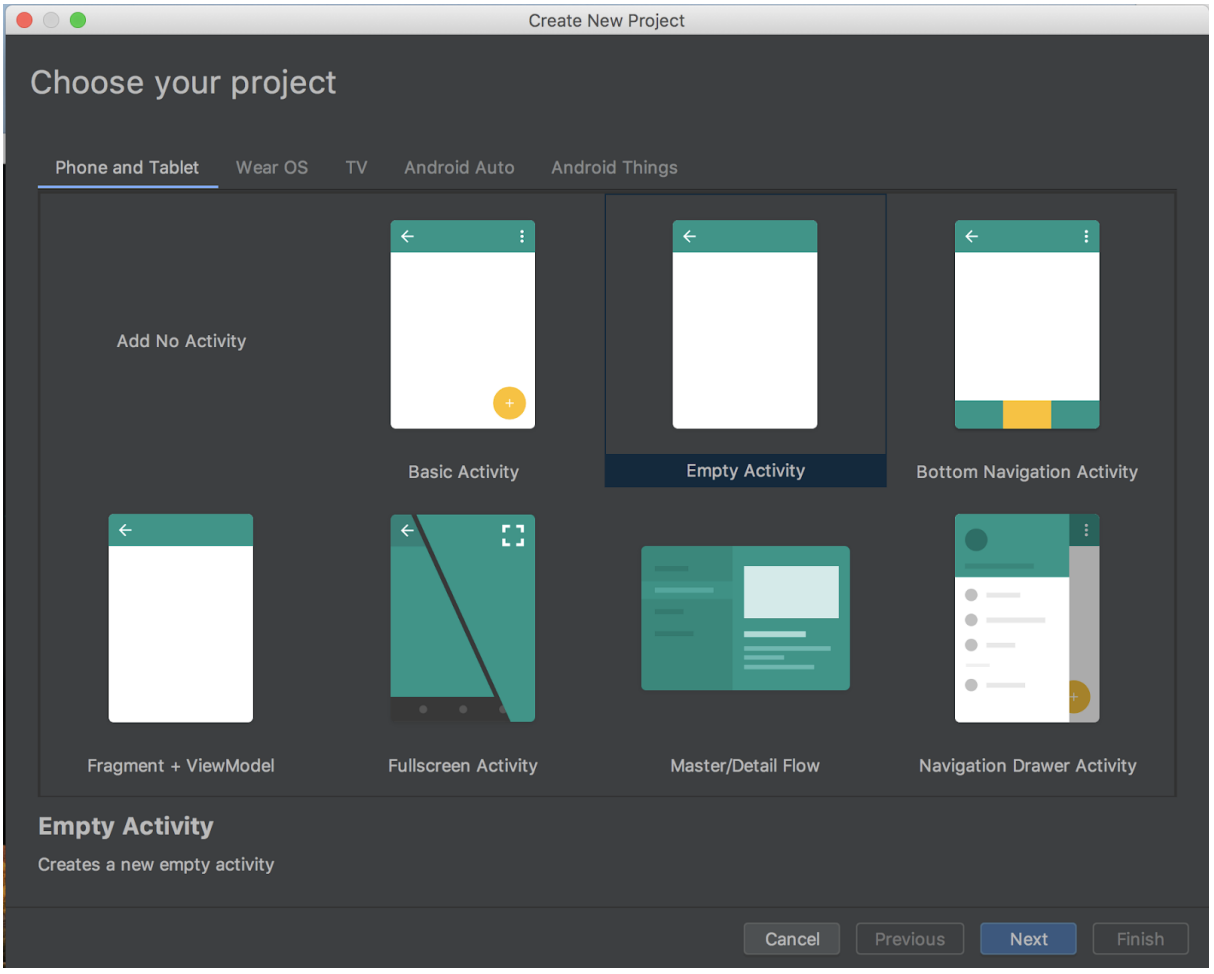
Start with an Activity

You will now have to add an **Activity**. The Android documentation defines an **Activity** as:

An activity is a single, focused thing that the user can do

In other words, it is a screen where the user interface (UI) resides on, for the user to interact with.

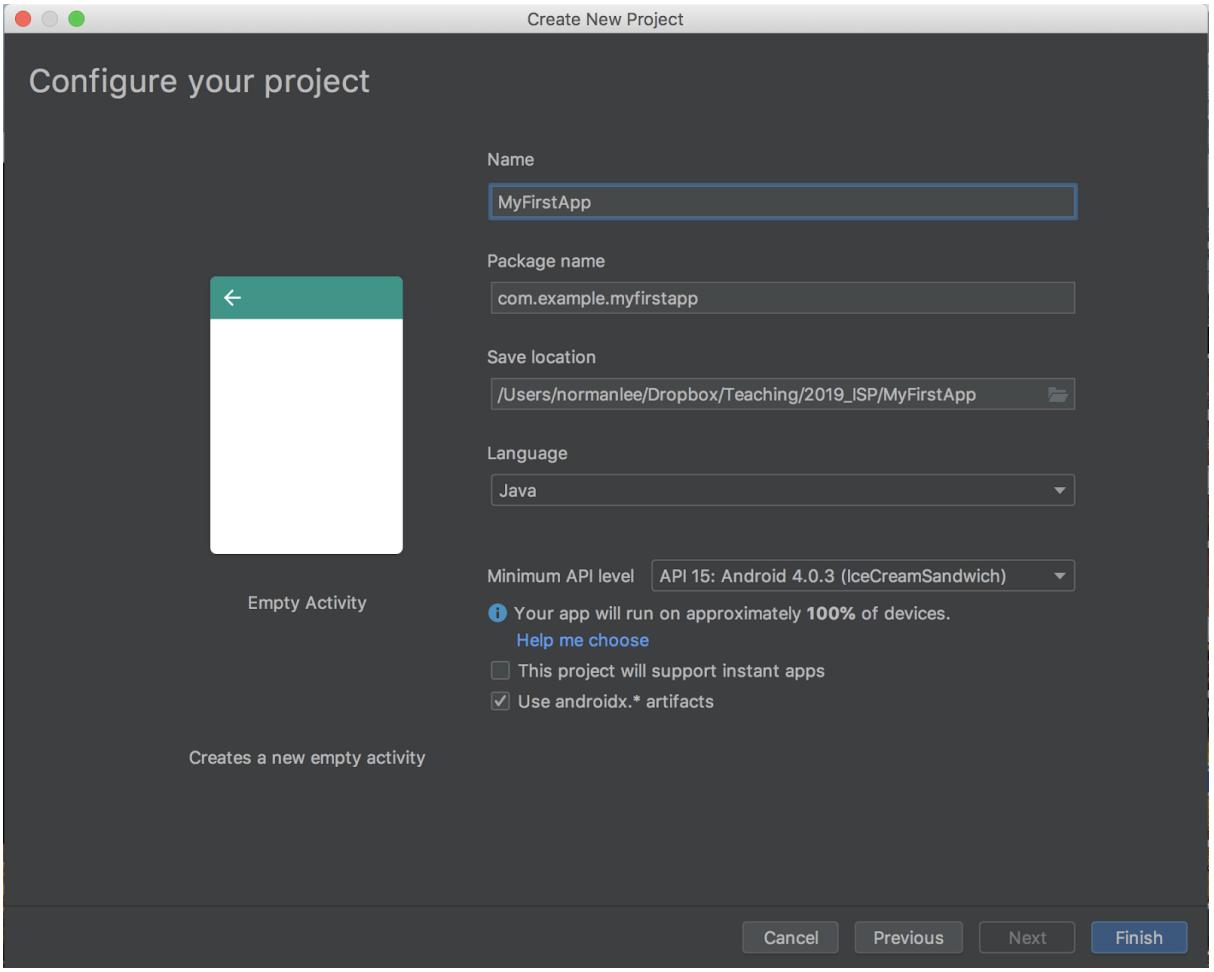
Make sure **Empty Activity** is selected, as it is the option with the least amount of code. The other options come pre-loaded with some android app features, and they are useful when you are more familiar with android app programming.



Give your project a name

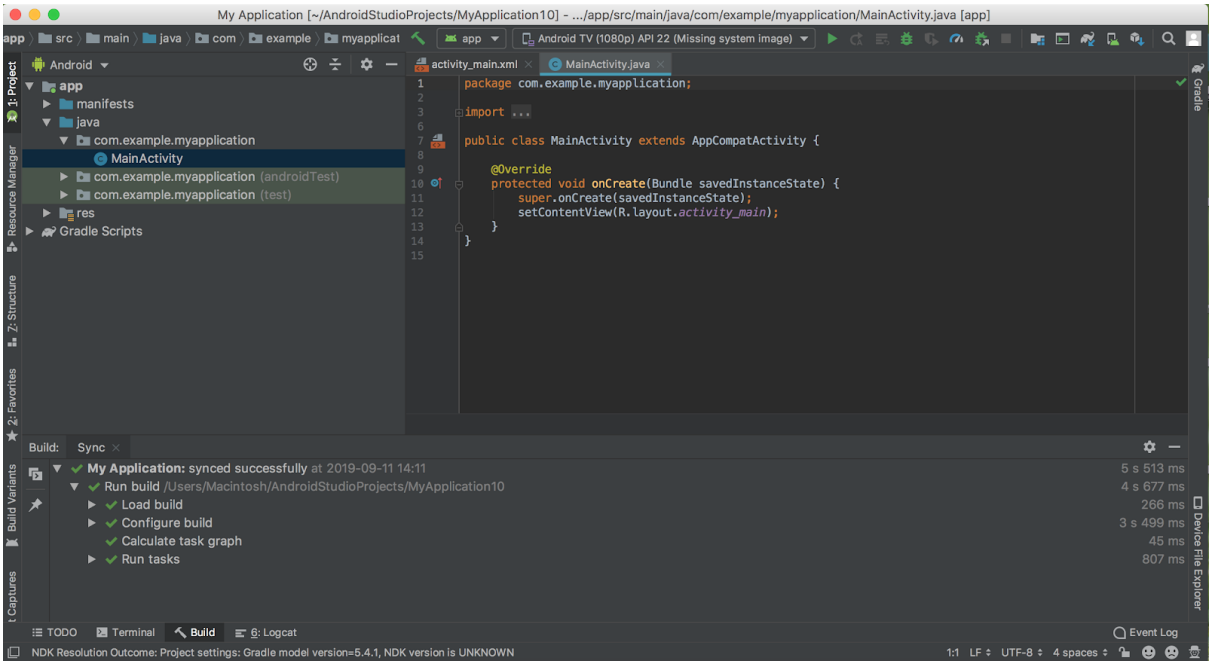
Give your project an **Application name** and state where you want it to be saved in the **Project location**.

- Select **Java** as the language (the default seems to be Kotlin. We are not teaching Kotlin.)
- Ensure the **Use androidx.* artifacts** box is checked. On some installations, you are not able to uncheck it anyway.
- The **API Level** is the version of the Android operating system. As of November 2019, the latest API level is 29 but it is safe to leave API Level 15 as a default choice.



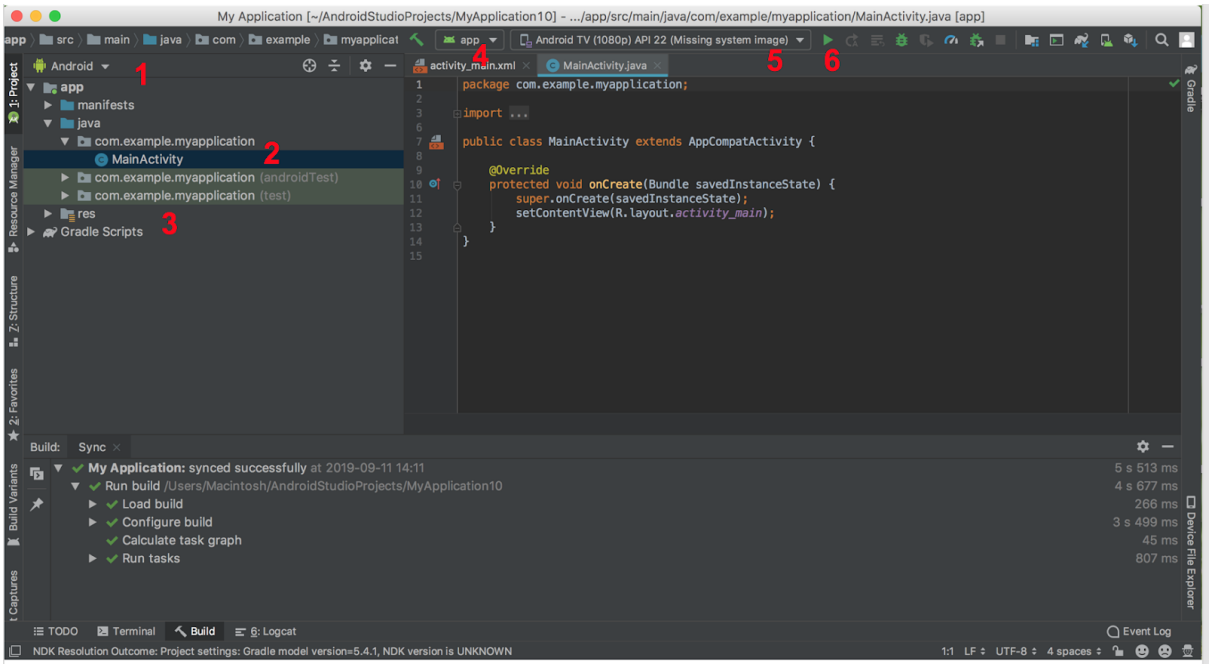
When the project is ready

When the project is ready, you should see something like this.



Examine the User Interface

If you are successful in generating a project, you should see the features in the user interface. There are six main parts that you should be familiar with.



1 You can select different views. What you should be selected is the **Android** view.
There are two folders, **app** (see point **2** below) and **res** (see point **3** below).

When an **activity** is created, you get two files:

- An xml file named `activity_main.xml`
- A java class in a file named `MainActivity.java`

The xml file specifies the user interface layout and the java class is where you write code to provide interactivity to your UI.

The **app** folder contains the java code and the java code `MainActivity.java` is found in the package folder within the java folder (see point **2**).

The **res** folder (see point **3**) contains resources for the app. This includes xml files, images and icons.
Expand the **res** folder and look for `activity_main.xml` file under the **layout** subfolder.

At **4**, ensure that **app** is shown here, without a red cross. If you do not see this, you may have a buggy installation and the first thing you could try is to update your installation
Windows: Help → Check For Updates
Mac: Android Studio (beside the Apple icon) → Check For Updates

At **5**, the emulators that are available are shown here. If your android phone is connected, you will see it here as well. See next section on what to do.
If you see **"missing system image"** then you need to install the emulator for the device that you are interested in. See next section.
If you don't, the emulator has been installed.

At **6**, this is the **Run** button (or on the menu, Run → Run). once you have an android phone connected or an emulator ready, you press this button to deploy your app.

Test your App

It is recommended that you get an an android phone. It saves you a lot of trouble with the emulators. With one, you are ready to test your app.

Enable **USB Debugging** Mode on your phone. You will find instructions on how to do so here. https://developer.android.com/studio/run/device

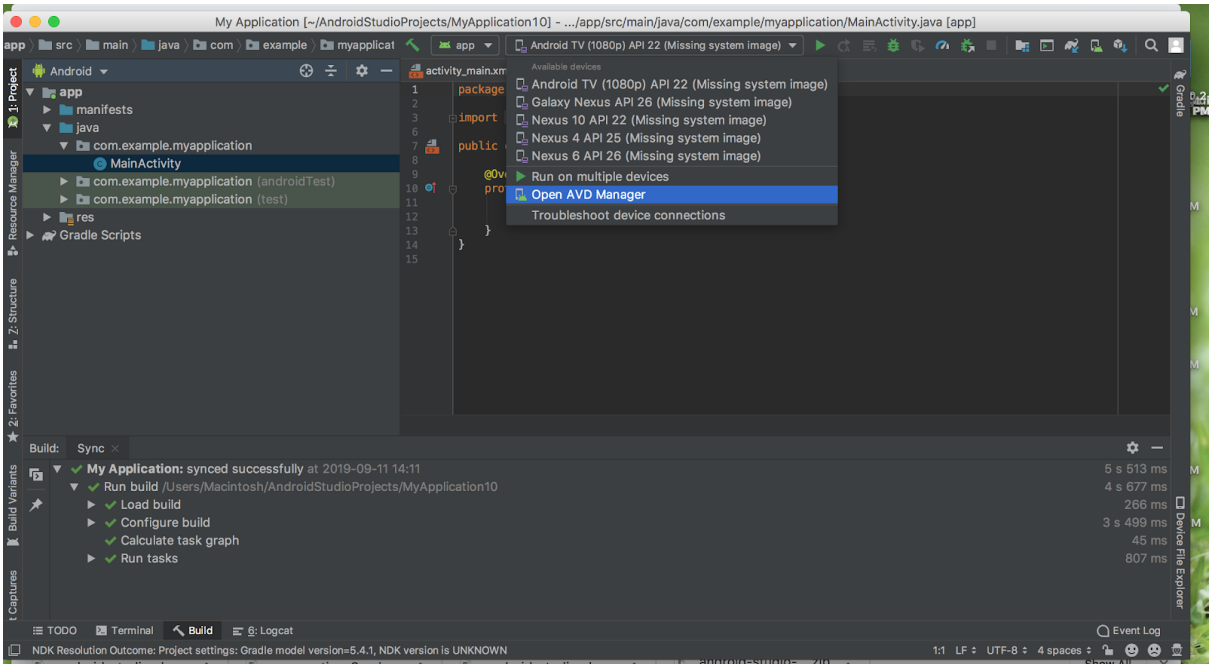
Connect your phone to the device and you should see it listed at **5**. Then press the run button at **6** and see your app appear.

Note: Some mobile games don't work if you have USB Debugging mode enabled.

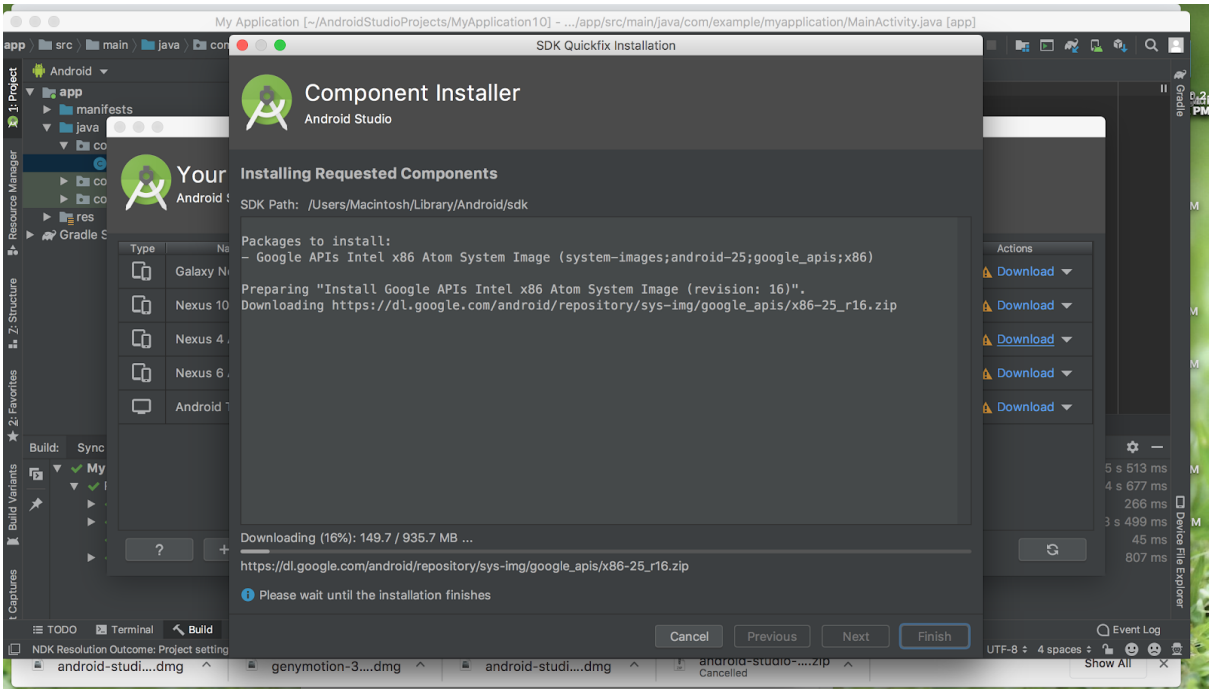
If you wish to use the emulator, continue with the rest of this guide. You will need at least 30 minutes from now if you are doing a fresh installation.

Installing the Android Studio Emulator

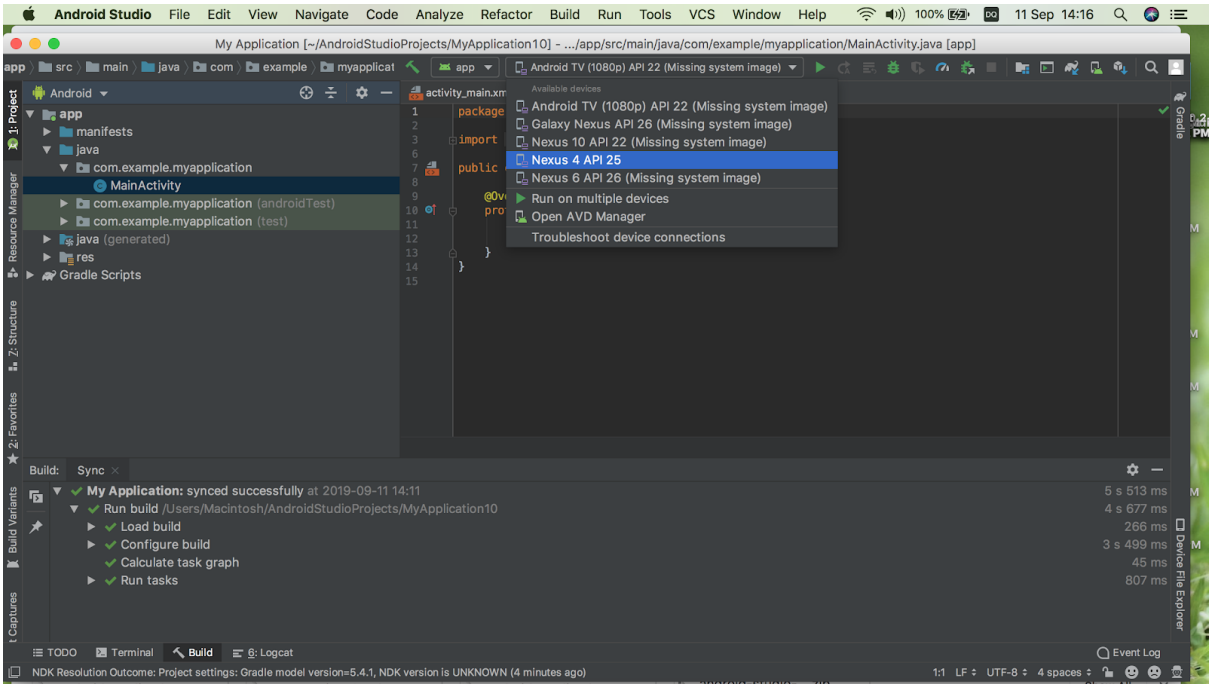
Go to the Devices section at **5** and click it and look for **Open AVD Manager**.



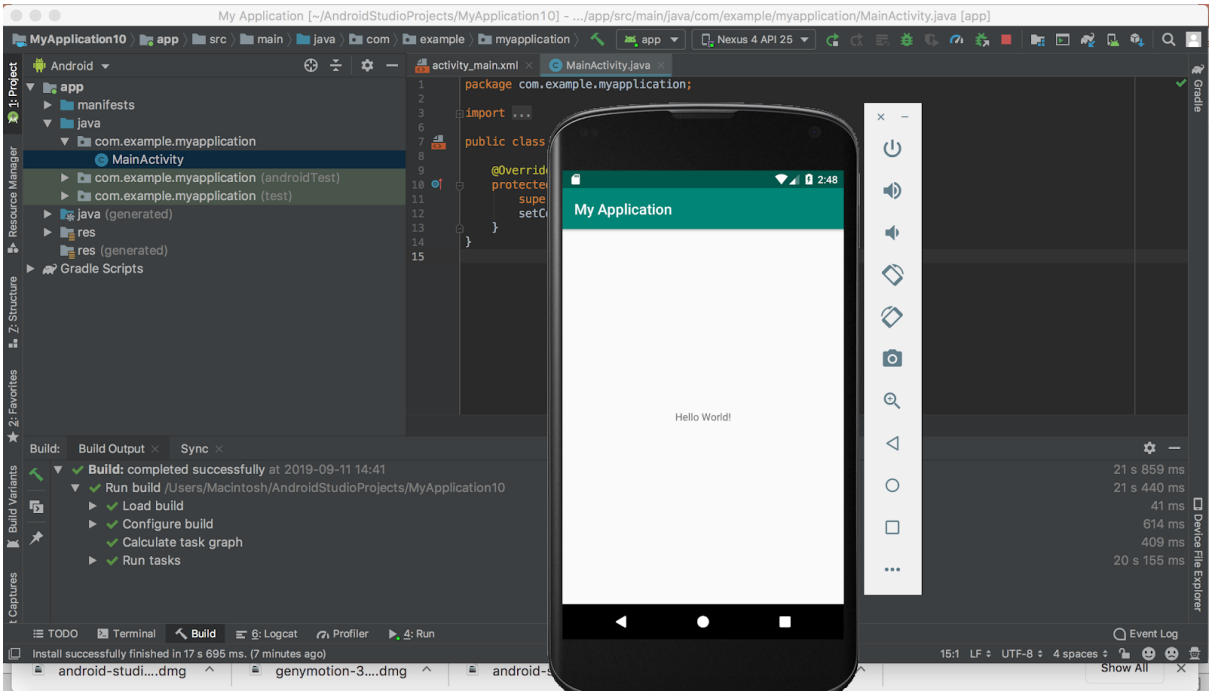
You should see a list of devices. Select one, I suggest **Nexus 4** and click on **Download**. The installation will take a couple of minutes.



When the installation is over, you should see that your chosen devices is ready to be selected. You may now click the Run button to compile and deploy your app on the emulator. Be aware that this process will be slow and consume quite a lot of CPU effort on your laptop. You may want to take steps to prevent your laptop from overheating.



The emulator should now display the app. Congratulations, you have just completed your very first android app!



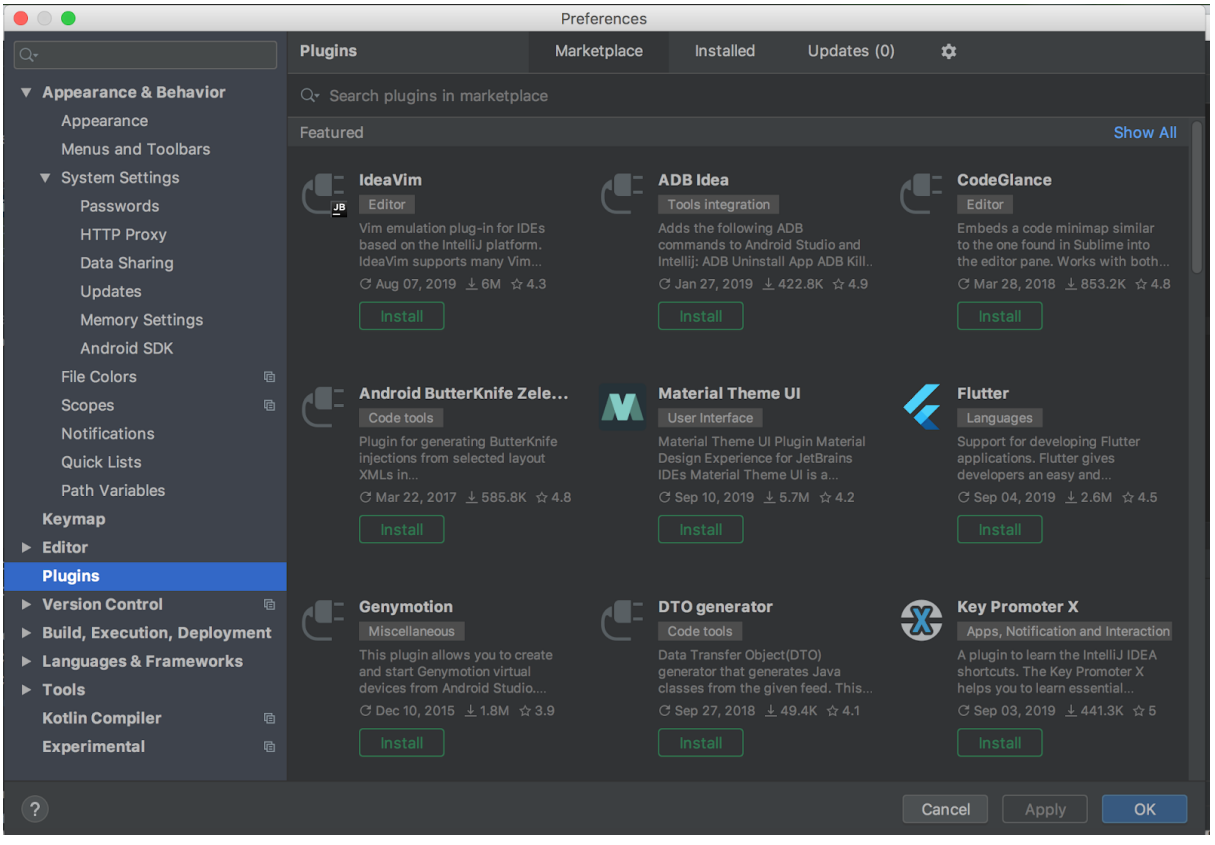
Using Genymotion instead of Android Studio Emulator (optional)

Some of you may want to consider **Genymotion** as an alternative to the Android studio emulator. This is because you might find that the Android studio emulator will take a longer time to load. **I put in this section as an option for those of you who would like to explore another Emulator. I cannot guarantee that it will work for you and I'm not familiar with the steps for Windows.**

You should reserve about 30 mins for this part.

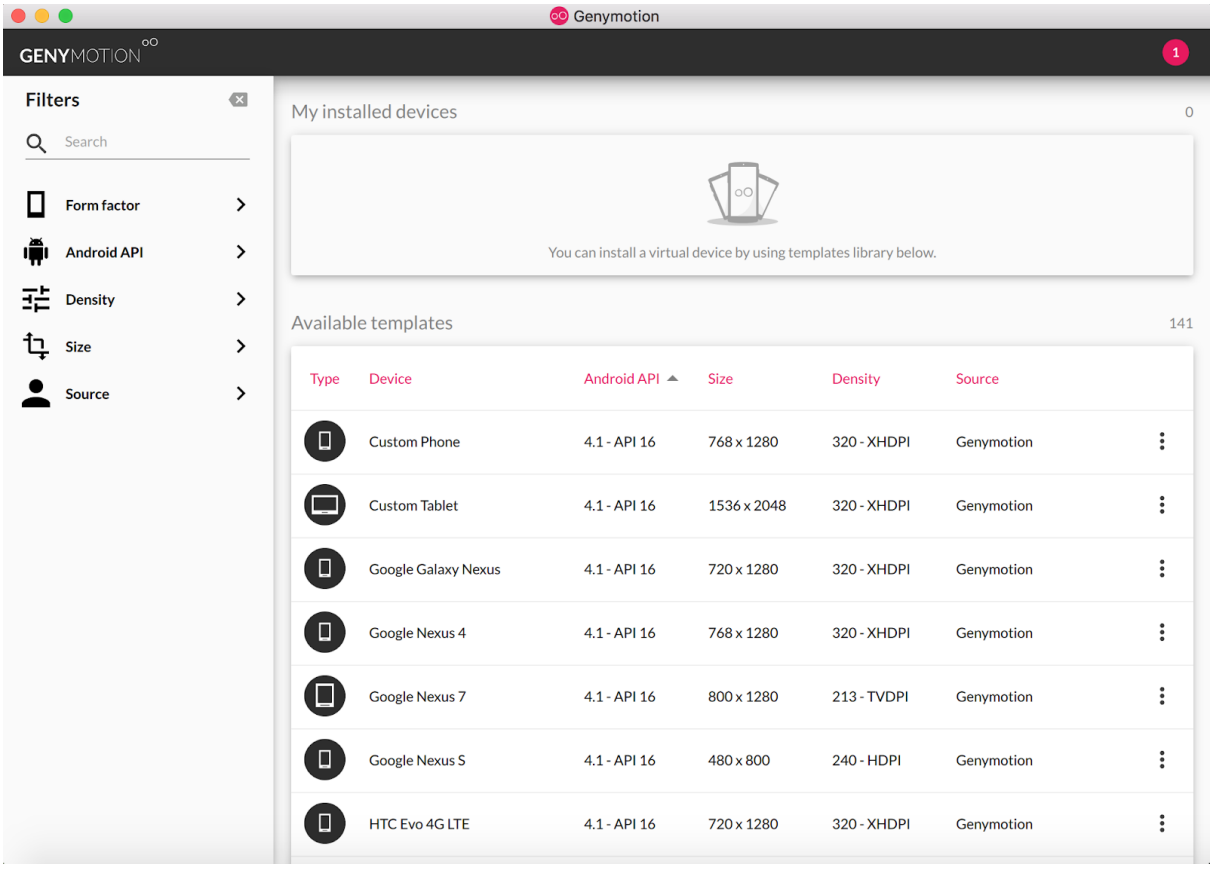
1. Go to https://genymotion.com/fun_zone/ to download the **Genymotion personal edition** and install it on your own computer. You may refer to the official documentation

2. Go to the Settings section and install the Genymotion plugin.
   Mac: Android Studios → Preferences → Plugins
   You will be asked to restart your IDE.



3. Open Genymotion and Mac users will be asked to install **VirtualBox** as well. When the installation is successful, you will be asked to create an account and declare that it is for personal use. (sigh).

   When everything is done, you should see this screen and you are ready to select any available device below to appear in “My Installed Devices”. I used **Samsung Galaxy S8** and have been using it since without any issues.

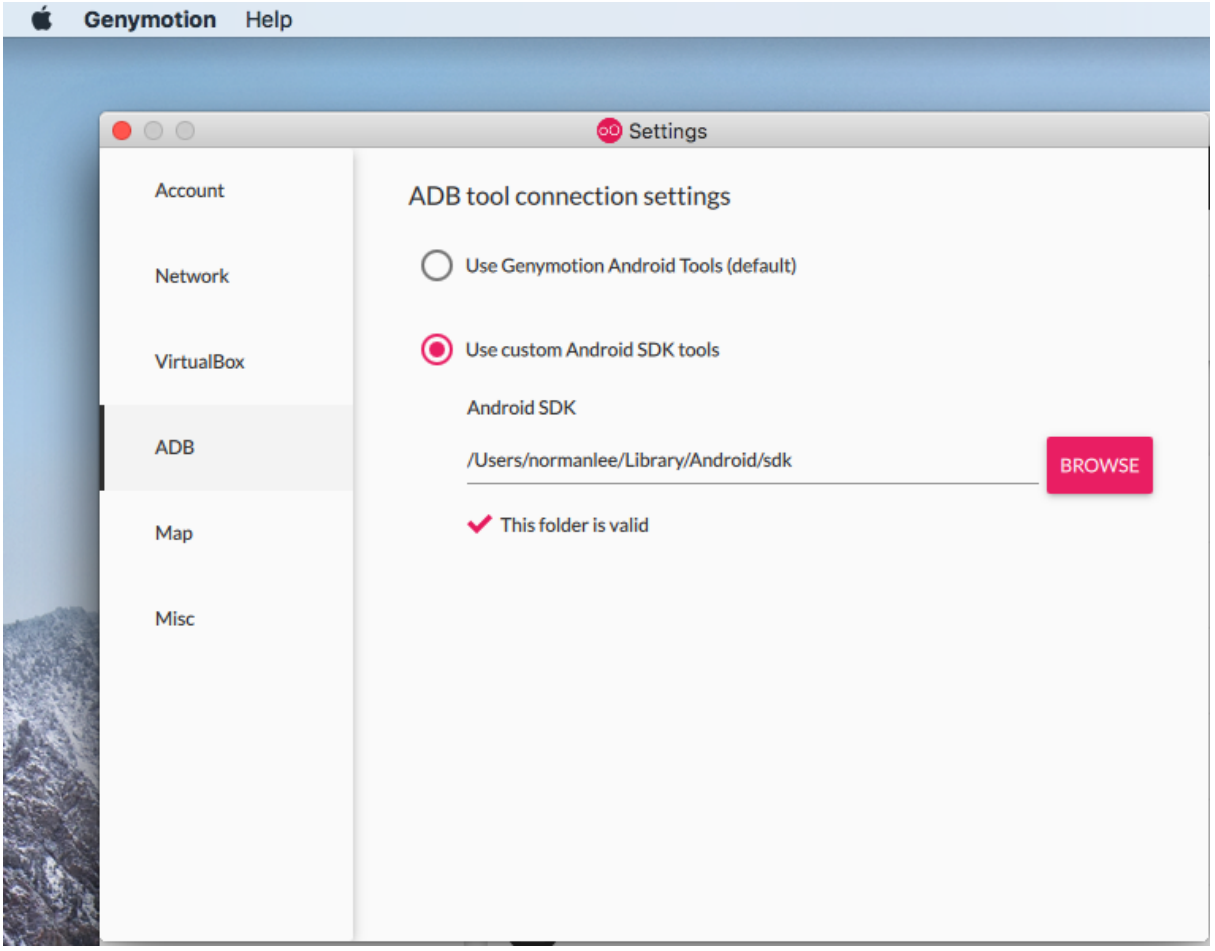


4. Now you are ready to enter the location of the Android SDK.

For Mac users:

Genymotion (beside the Apple icon) → Preferences → ADB

Your SDK location will be at: `/Users/Macintosh/Library/Android/sdk`



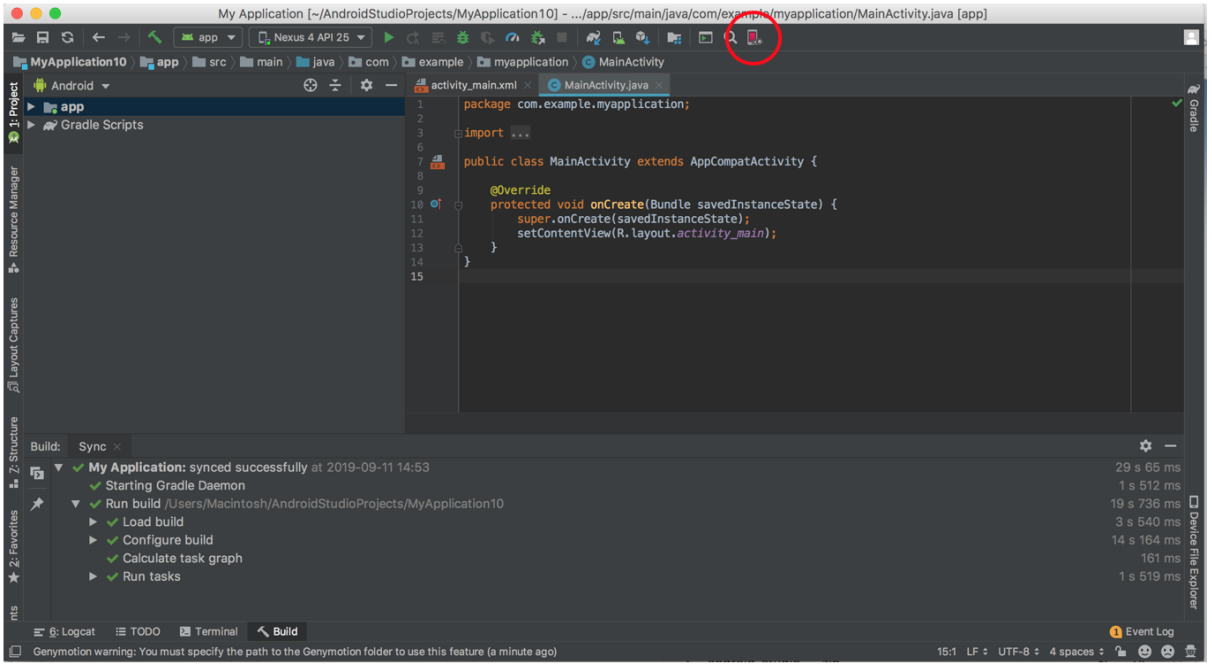
For Windows users:

You should look for a similar settings page and refer to this link for advice on where your sdk location is.

https://stackoverflow.com/questions/39563467/where-is-android-studios-default-sdk-location-on-windows

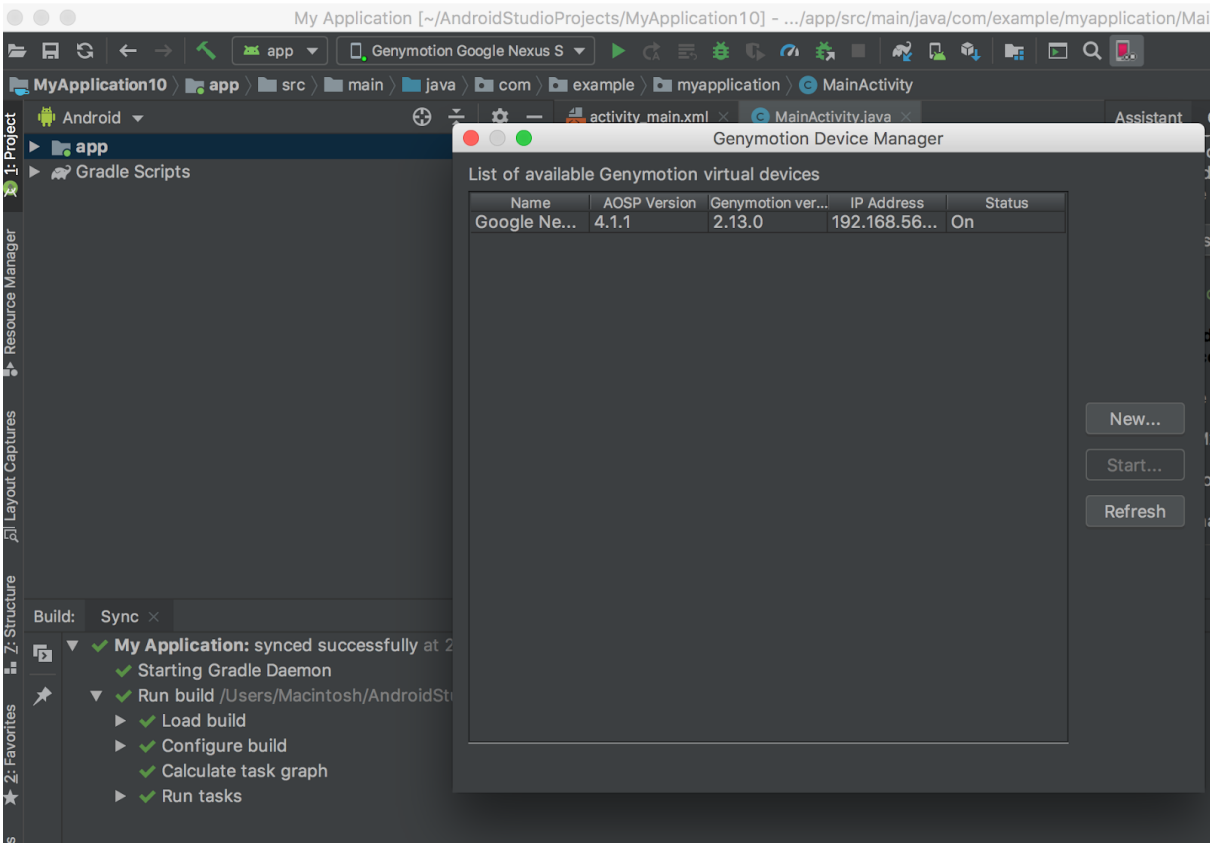
5. Go back to Android Studio and you should see the red Icon. Click on the red icon and put in the path to where Genymotion is installed.

   For most Mac users, it should be at `/Applications/Genymotion.app`



6. Click the red icon again and you should see the following window. If the text under **Status** shows Off, go to Genymotion and launch your installed virtual device.

   Upon launching, it should turn to On, and the device should appear beside the Run button. You are now ready to build your android app and it should display



Further Reading

- Android Developer Fundamentals Concepts
    - Section 1.0
    - Section 1.1
- You may want to complete Tasks 5 and Task 6 of the Lesson 1.1 Codelab on your own.

Examining the xml file and the java file

Video 1

Video 1 (~5 mins) tells you

- `MainActivity.java` is located in app/java/<package name> folder
- `activity_main.xml` is located in app/res/layout folder
- Obtain a preview of the layout
- the app/res/drawables folder is to store images
- Android studio's code completion feature
- Modify activity_main.xml file to change the base layout to LinearLayout

Transcript of Video 1 (Optional)

The actual speech in the video will differ slightly from this. It is optional for you to read this.

In this video, you will be introduced to the components of an android project. This is important as several files are required to complete an android project.

First, make sure that you are in the Android project view. If you are not, click here to change. This view allows you to see the components clearly.

Recall that we created one Activity, generating a java file named MainActivity.java, and a layout file named activity_main.xml.

I will first show you how to look for the java file.

Under the app folder, expand the folder tree. Look for a subfolder named java and expand the folder tree again. The java file is stored in the subfolder that has your package name.

There is actually more than one folder. The other folders are for java code to carry out testing.

We will come to the java code in another video.

Let’s look at the res folder.

The res folder stores resources that belong to your app. I will highlight two important subfolders in this video. Expand the folder tree.

Images go in the drawables subfolder.

The layout file is stored in the layout subfolder. Expand the folder and you will see the xml file listed there.

Open it and you will see the underlying xml code. If you don’t see the code, click “Text” on this tab here.

Click on the Preview button to see a preview of the layout, and click it again to make it disappear.

Let’s examine the xml code. The widget that stores the text “Hello World” is called a TextView widget and it sits within a base layout of Constraint Layout.

Each widget has several attributes, and you can see that the text “Hello World” is assigned to the text attribute.

Constraint Layout is quite an advanced layout, so we will start with Linear Layout instead.

Let’s make one simple change. Delete Constraint Layout using the backspace or delete button and type in LinearLayout.

As you type it in, you will see that Android’s code completion feature is activated. Press enter to choose the first option. You will see that the code is completed for you.

Press Run to see the actual app. It might take a while. If you don’t see the emulator pop up, you will need to select it from the bottom.

In the next video, I will explain what Linear Layout is, as well as what the values for the attributes height and width mean.

Further Reading

- **Android documentation on Linear Layout.** By reading this documentation, you should be familiar with how the layout_width attribute is used.
  https://developer.android.com/guide/topics/ui/layout/linear

- **Android Developer Fundamentals Concepts. Section 1.2 - Resource Files.** This section futher explains the purpose of the different folder in the res folder.
  https://google-developer-training.gitbooks.io/android-developer-fundamentals-course-concepts/content/en/Unit%201/12_c_layouts,_views_and_resources.html#adding_resources

- **Android Developer Fundamentals Concepts. Section 1.3 - TextView.** By reading this documentation, you have a reference on what other attributes for the TextView widget are available.
  https://google-developer-training.gitbooks.io/android-developer-fundamentals-course-concepts/content/en/Unit%201/13_c_text_and_scrolling_views.html#textview

Video 3

In this video, I will show you how to modify the java file to control the widgets.

What we are going to do is to modify the text of the textview widget programmatically.

For each widget type, a class of the same name exists.

Go to MainActivity and declare a TextView variable. As you type in, you see that the code completion feature is activated.

The TextView variable is red, which means the library needs to be imported first. Press Option + Enter to do so. For windows users, press Alt + Enter.

Your next task is to get this variable to point to an instance of the TextView class.

This instance is created in the memory when the app is started up.

You need to get a reference to the widget that you are interested in, and that's where the id attribute that you created in the xml file is so important.

Android has a special class called the R class. This R class is used to store the ids for all the widgets.

If the id of the widget is firstWidget, then the id is stored in memory using the R class as follows.

You get the instance of the object by using the findViewByID method, and the instance that is returned can now be assigned to a variable of the same type.

Let's type the code out.

Make sure you have declared an instance variable of the TextView type in the MainActivity class.

Within onCreate, invoke the findViewById method, and pass to it the R class.

Assign the result to the TextView instance variable.

Now that we have this instance variable, we are able to modify its properties by calling a suitable method.

Use the dot operator and from the list that pops out, look for the setText() method.

Put the string in and you are ready to run your app.