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.

Otherwise, you may use **the emulator provided by Android studio too.** .

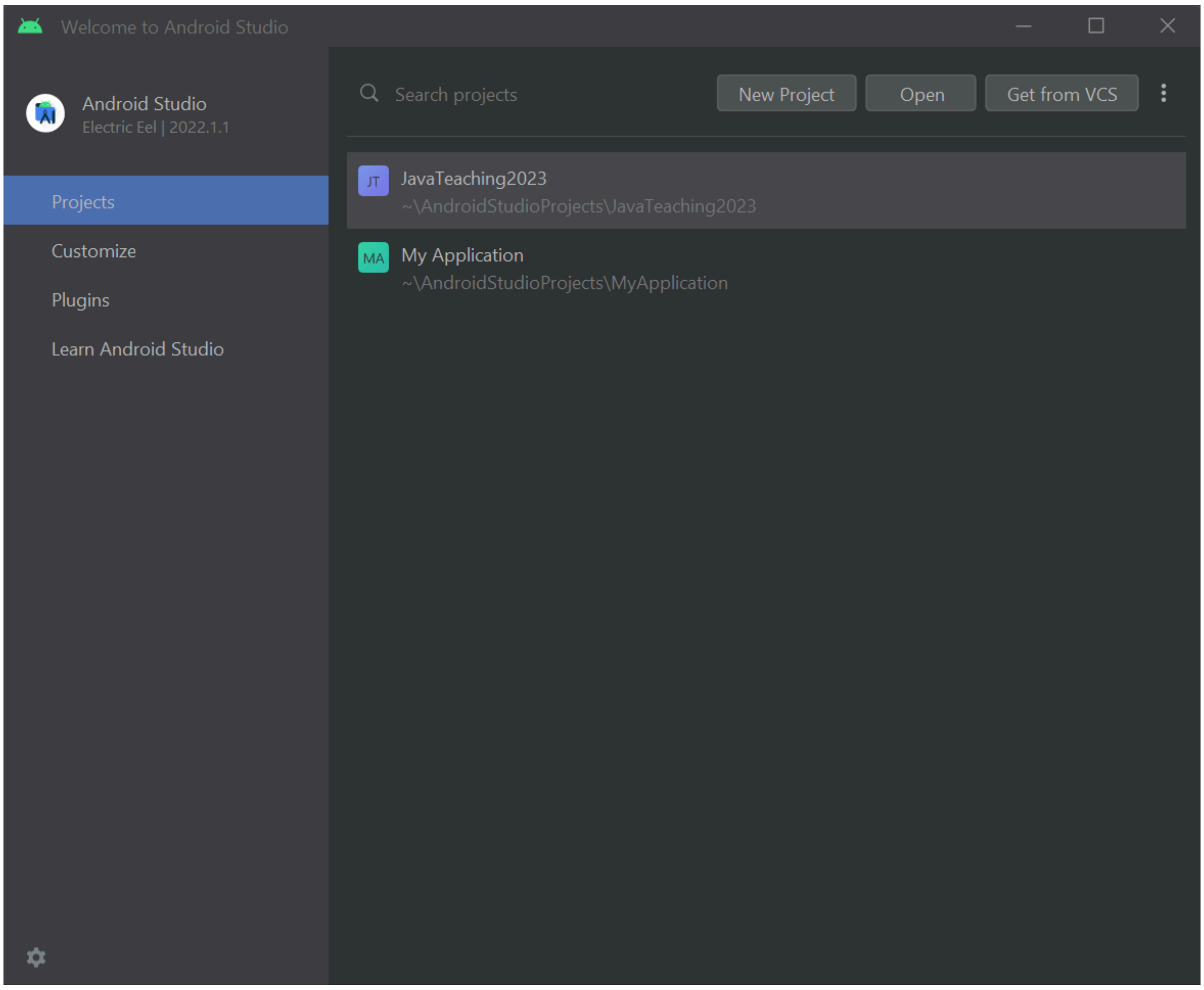
You may also explore other tools such as Genymotion, Bluestacks or Vysor.

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, cl[ose any project that is launched and select **Projects → New Project**



Start with an Activity

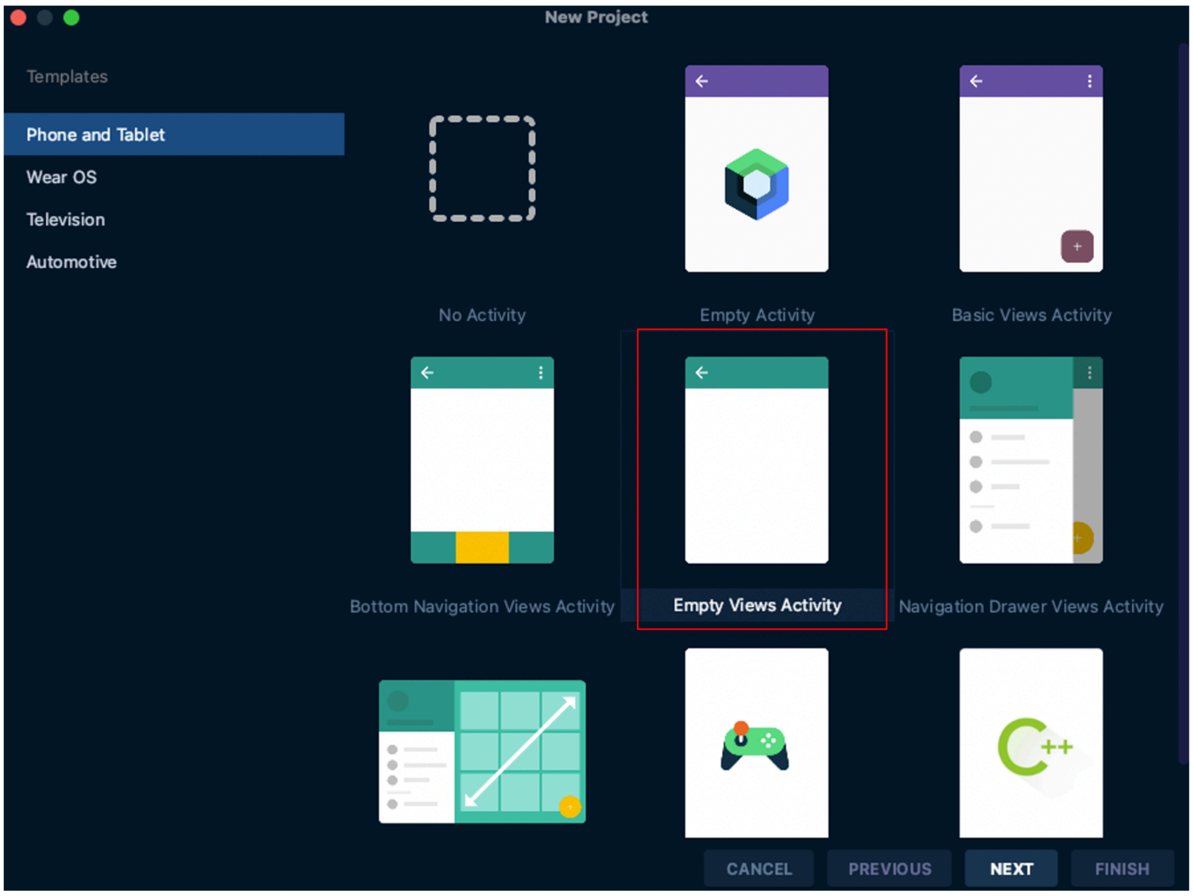
You will now have to add an **Activity**. The Android documentation defines an **Activity** as:

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.

Ensure that **Phone and Tablet** is selected. Then **Scroll down** and make sure **Empty View 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.

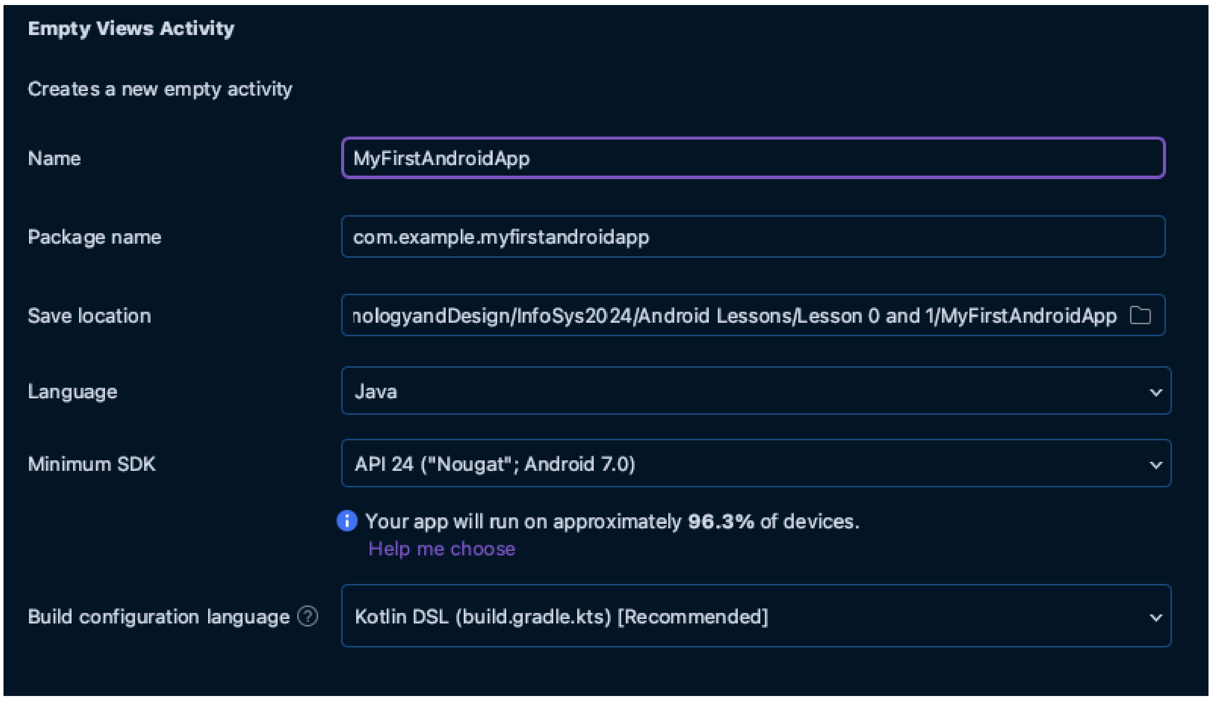


Give your project a name

Give your project a **Name** and state where you want it to be saved in the **Project location**. Otherwise just accept the default location.

- Select **Java** as the language (the default seems to be Kotlin. We are not teaching Kotlin.)
- **The Minimum SDK** or the **API Level** is the version of the Android operating system. As of Jan 2024, the latest API level is 34 but it is safe to leave the default choice as it is.

Click **Finish**.



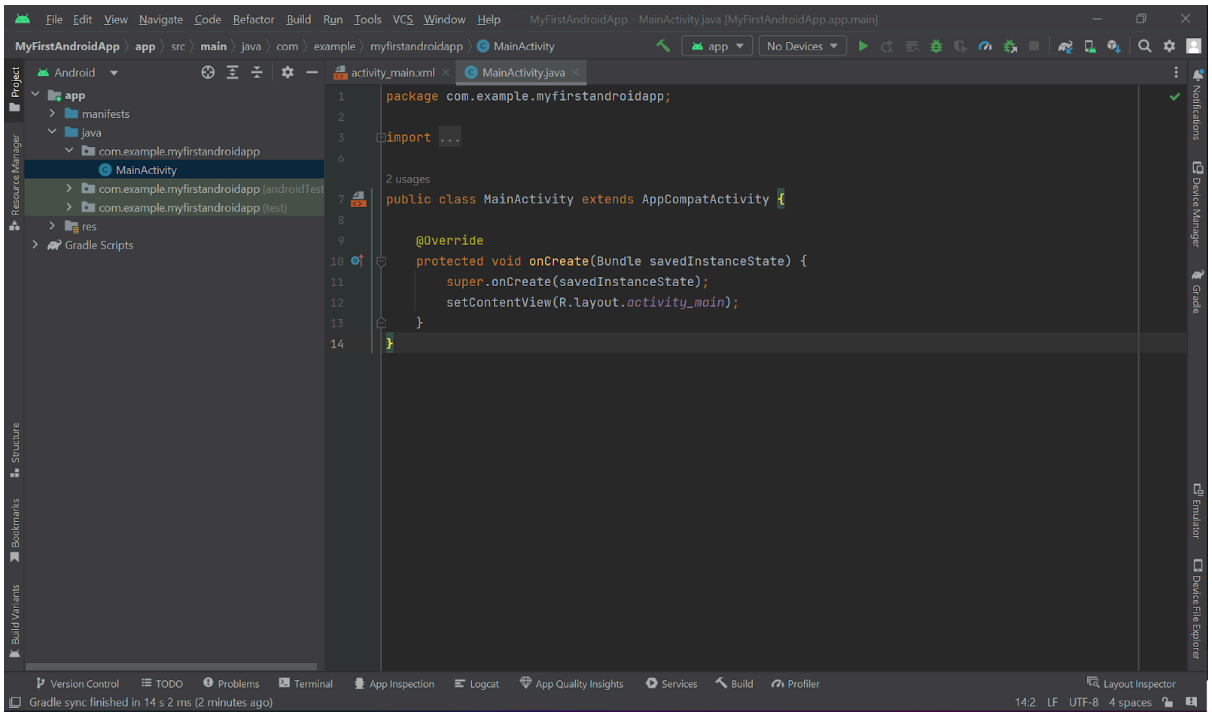
Empty Views Activity

Creates a new empty activity

Name	MyFirstAndroidApp
Package name	com.example.myfirstandroidapp
Save location	nologyandDesign/InfoSys2024/Android Lessons/Lesson 0 and 1/MyFirstAndroidApp
Language	Java
Minimum SDK	API 24 ("Nougat"; Android 7.0)
	Your app will run on approximately **96.3%** of devices. Help me choose
Build configuration language	Kotlin DSL (build.gradle.kts) [Recommended]

When the project is ready

Give a few minutes for the project to be ready. When the project is ready, you should see something like this.

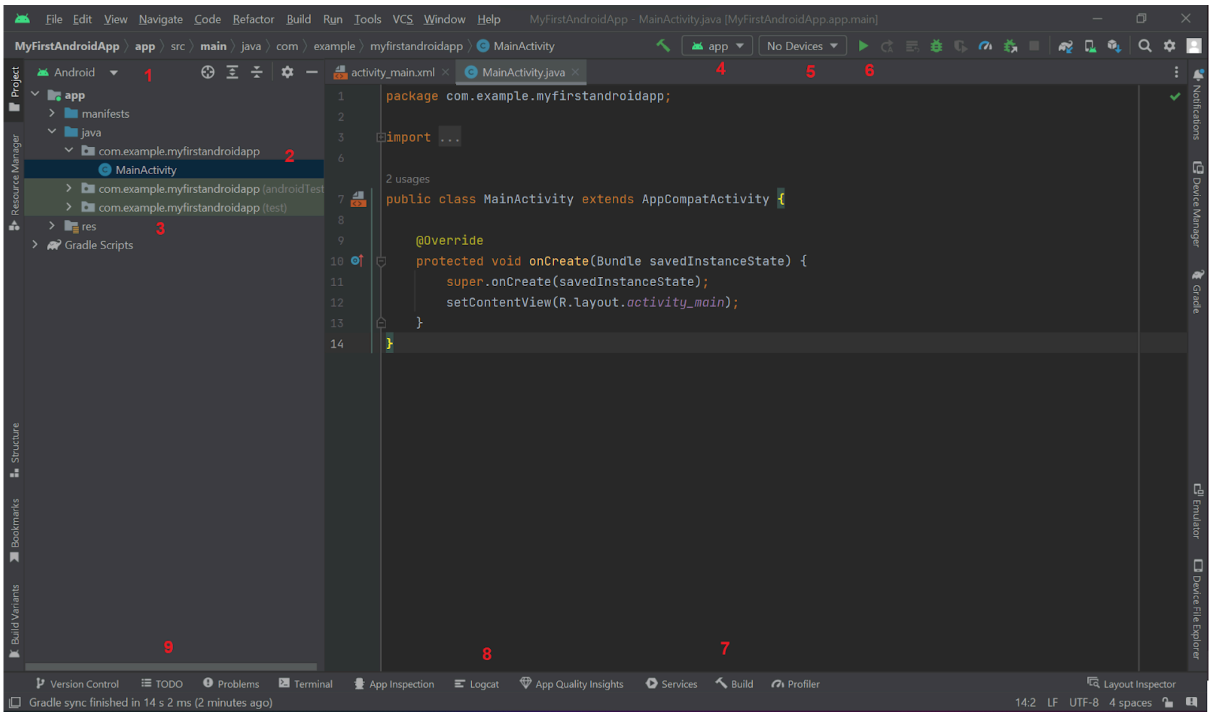

File Edit View Navigate Code Refactor Build Run Tools VCS Window Help
MyFirstAndroidApp - MainActivity.java [MyFirstAndroidApp.app.main]
MyFirstAndroidApp 〉 app 〉 src 〉 main 〉 java 〉 com 〉 example 〉 myfirstandroidapp 〉 MainActivity
app
No Devices
Android
app
manifests
java
com.example.myfirstandroidapp
MainActivity
com.example.myfirstandroidapp (androidTest)
com.example.myfirstandroidapp (test)
res
Gradle Scripts
activity_main.xml
MainActivity.java
Version Control
TODO
Problems
Terminal
App Inspection
Logcat
App Quality Insights
Services
Build
Profiler
Layout Inspector
14:2 LF UTF-8 4 spaces

```java
package com.example.myfirstandroidapp;

import ...

2 usages
public class MainActivity extends AppCompatActivity {

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

Gradle sync finished in 14 s 2 ms (2 minutes ago)

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.


File Edit View Navigate Code Refactor Build Run Tools VCS Window Help
MyFirstAndroidApp - MainActivity.java [MyFirstAndroidApp.app.main]
MyFirstAndroidApp 〉 app 〉 src 〉 main 〉 java 〉 com 〉 example 〉 myfirstandroidapp 〉 MainActivity
app
No Devices
Android
1
app
manifests
java
com.example.myfirstandroidapp
2
MainActivity
com.example.myfirstandroidapp (androidTest)
com.example.myfirstandroidapp (test)
res
3
Gradle Scripts
activity_main.xml
MainActivity.java
4
5
6
9
8
7
Version Control
TODO
Problems
Terminal
App Inspection
Logcat
App Quality Insights
Services
Build
Profiler
Layout Inspector
14:2 LF UTF-8 4 spaces

```
package com.example.myfirstandroidapp;

import ...

2 usages
public class MainActivity extends AppCompatActivity {

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

Gradle sync finished in 14 s 2 ms (2 minutes ago)

1 You can select different views. What 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).
Notice that there are three folders with very similar names. The other two are for writing code to test your app.

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 **"No Devices"** 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.

There are also other views that you can launch. This part is for your information and you just need to note them for now.
Clicking **Build** at **7** enables you to see any Build messages.
The **Logcat** at **8** enables you to see any debug messages that you write in your code.
The **TODOs** at **9** enables you to see all the comment statements prefixed with TODO in your code here.

Test your App

It is recommended that you get 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.

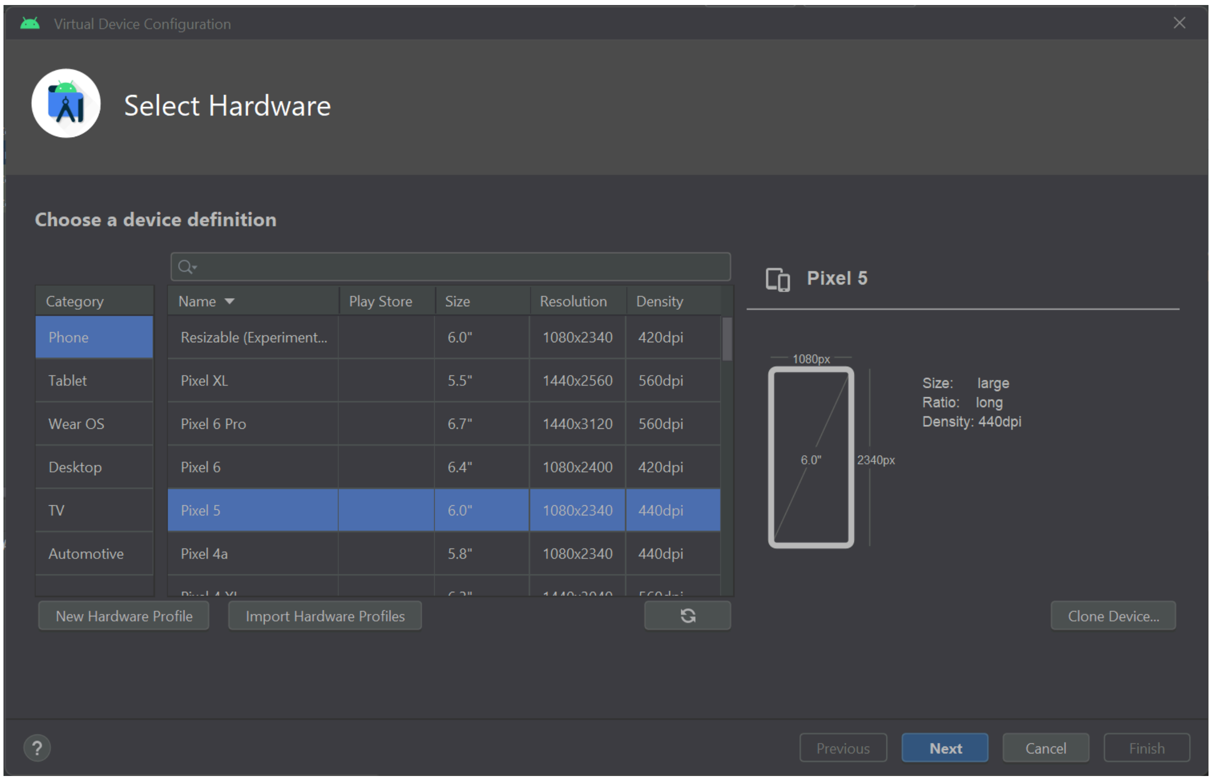
Note: Some mobile games don't work if you have USB Debugging mode enabled.

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 **Device Manager.**

You should see a list of devices. Select one, I suggest **Pixel 5** and click on **Next**.



Virtual Device Configuration

Select Hardware

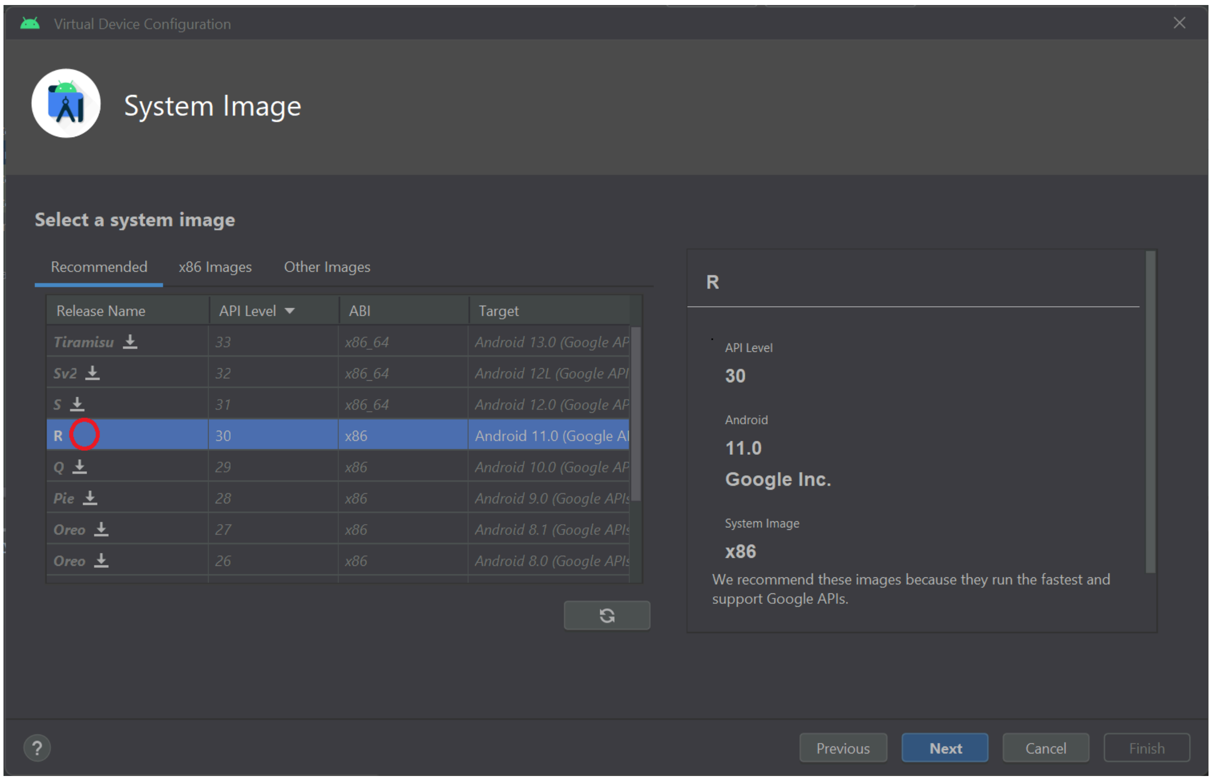
Choose a device definition

Category	Name	Play Store	Size	Resolution	Density
Phone	Resizable (Experiment...		6.0"	1080x2340	420dpi
Tablet	Pixel XL		5.5"	1440x2560	560dpi
Wear OS	Pixel 6 Pro		6.7"	1440x3120	560dpi
Desktop	Pixel 6		6.4"	1080x2400	420dpi
TV	Pixel 5		6.0"	1080x2340	440dpi
Automotive	Pixel 4a		5.8"	1080x2340	440dpi

Pixel 5

Size: large
Ratio: long
Density: 440dpi

If it's your first time installing a device in the Emulator, you should see the download button beside **R**, where the red circle is.
Click on that **download button** first. When the download is completed, you should be then be able to click **Next**.



Release Name	API Level	ABI	Target
Tiramisu	33	x86_64	Android 13.0 (Google AP
Sv2	32	x86_64	Android 12L (Google API
S	31	x86_64	Android 12.0 (Google AP
R	30	x86	Android 11.0 (Google A
Q	29	x86	Android 10.0 (Google AP
Pie	28	x86	Android 9.0 (Google APIs
Oreo	27	x86	Android 8.1 (Google APIs
Oreo	26	x86	Android 8.0 (Google APIs

R

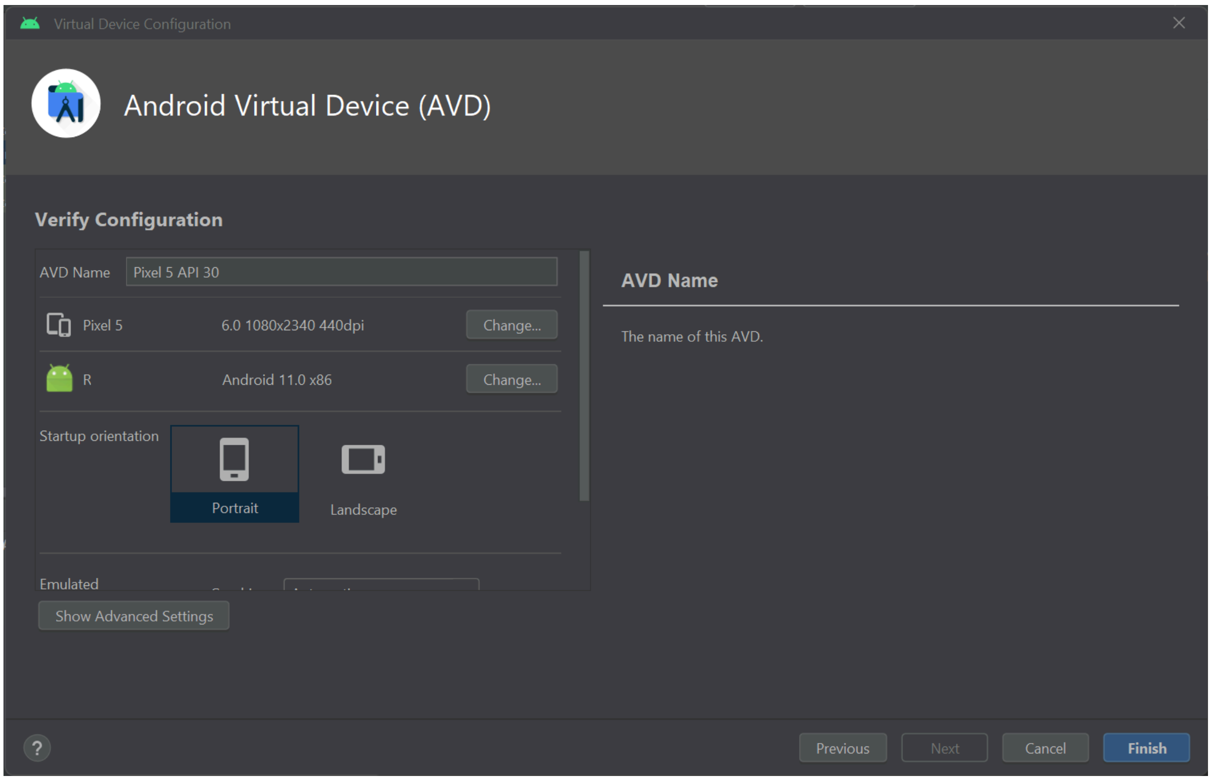
API Level
30

Android
11.0
Google Inc.

System Image
x86

We recommend these images because they run the fastest and support Google APIs.

You should then see this. Accept the defaults and click **Finish**.



Virtual Device Configuration

Android Virtual Device (AVD)

Verify Configuration

AVD Name: Pixel 5 API 30

Pixel 5 — 6.0 1080x2340 440dpi — Change...

R — Android 11.0 x86 — Change...

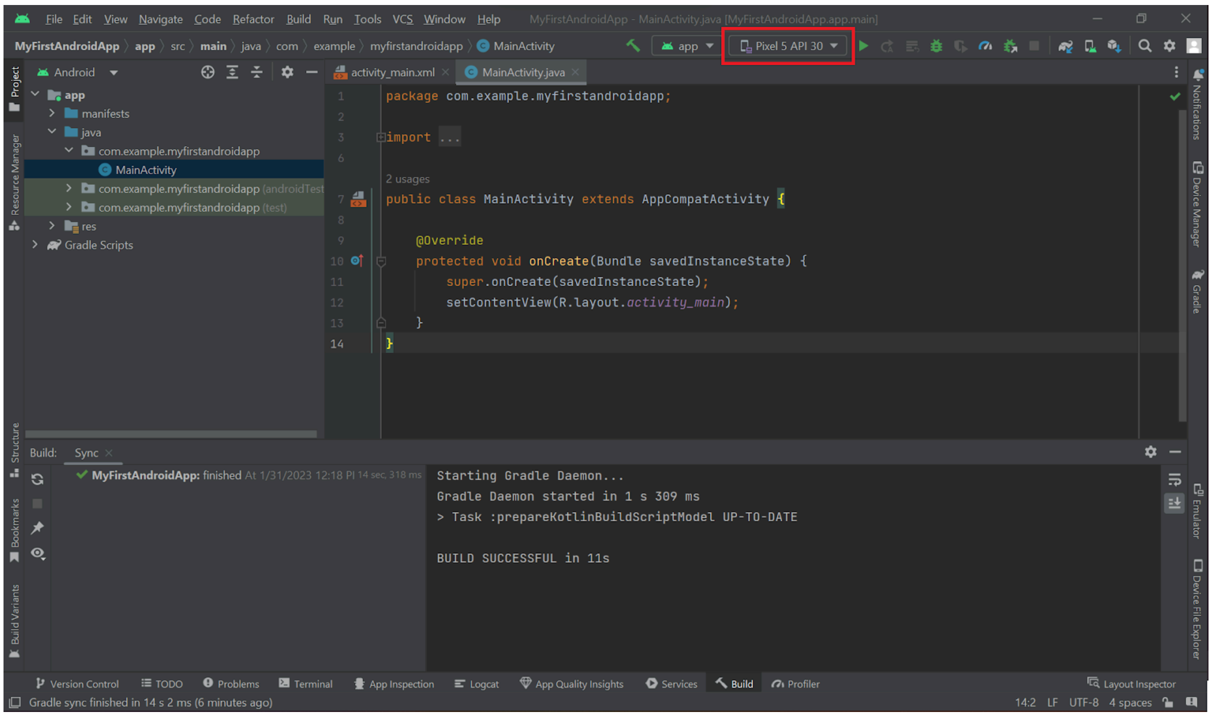
Startup orientation: [✓ Portrait] Landscape

Emulated

AVD Name

The name of this AVD.

When the installation is over, you should see that your chosen device 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.


File Edit View Navigate Code Refactor Build Run Tools VCS Window Help
MyFirstAndroidApp - MainActivity.java [MyFirstAndroidApp.app.main]
MyFirstAndroidApp 〉 app 〉 src 〉 main 〉 java 〉 com 〉 example 〉 myfirstandroidapp 〉 MainActivity
app
Pixel 5 API 30
Android
app
manifests
java
com.example.myfirstandroidapp
MainActivity
com.example.myfirstandroidapp (androidTest)
com.example.myfirstandroidapp (test)
res
Gradle Scripts
activity_main.xml
MainActivity.java
2 usages
Build:
Sync
Version Control
TODO
Problems
Terminal
App Inspection
Logcat
App Quality Insights
Services
Build
Profiler
Layout Inspector
14:2 LF UTF-8 4 spaces

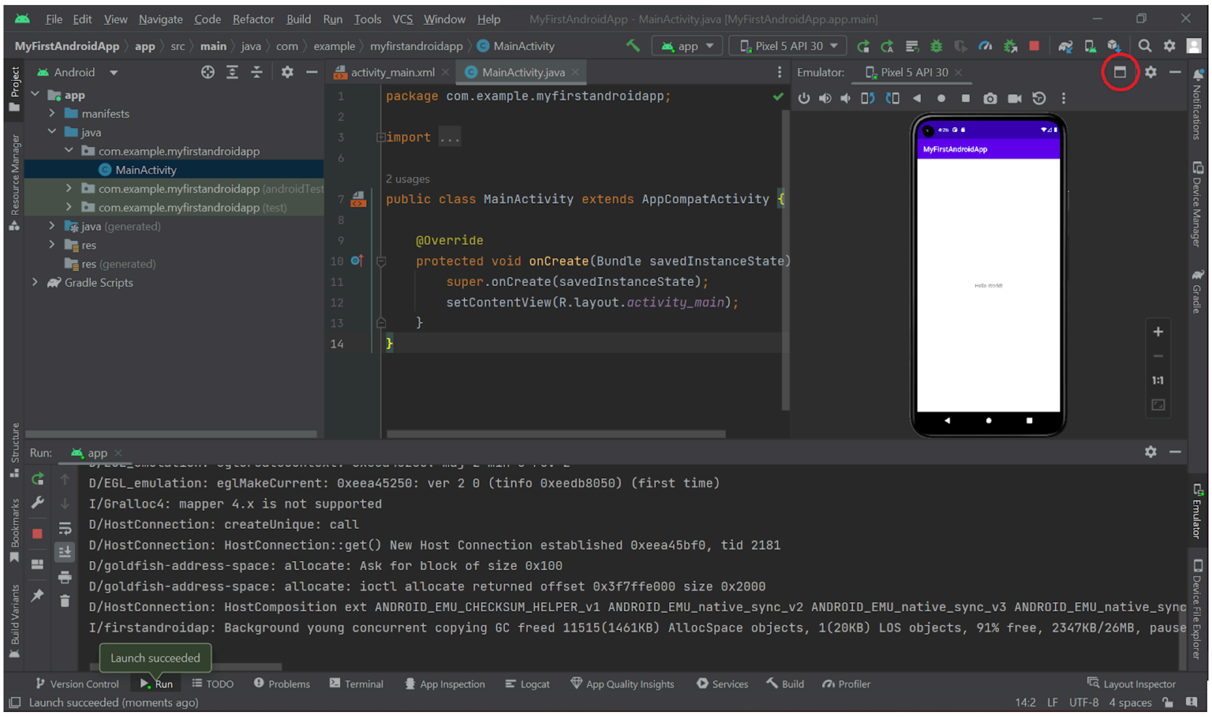
```java
package com.example.myfirstandroidapp;

import ...

public class MainActivity extends AppCompatActivity {

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

MyFirstAndroidApp: finished At 1/31/2023 12:18 PI 14 sec, 318 ms

```
Starting Gradle Daemon...
Gradle Daemon started in 1 s 309 ms
> Task :prepareKotlinBuildScriptModel UP-TO-DATE

BUILD SUCCESSFUL in 11s
```

Gradle sync finished in 14 s 2 ms (6 minutes ago)

The emulator should now display the app. Congratulations, you have just completed your very first android app! If you find that the phone looks rather small on your screen, you may click on the circled button to view it larger.



```
package com.example.myfirstandroidapp;

import ...

2 usages
public class MainActivity extends AppCompatActivity {

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

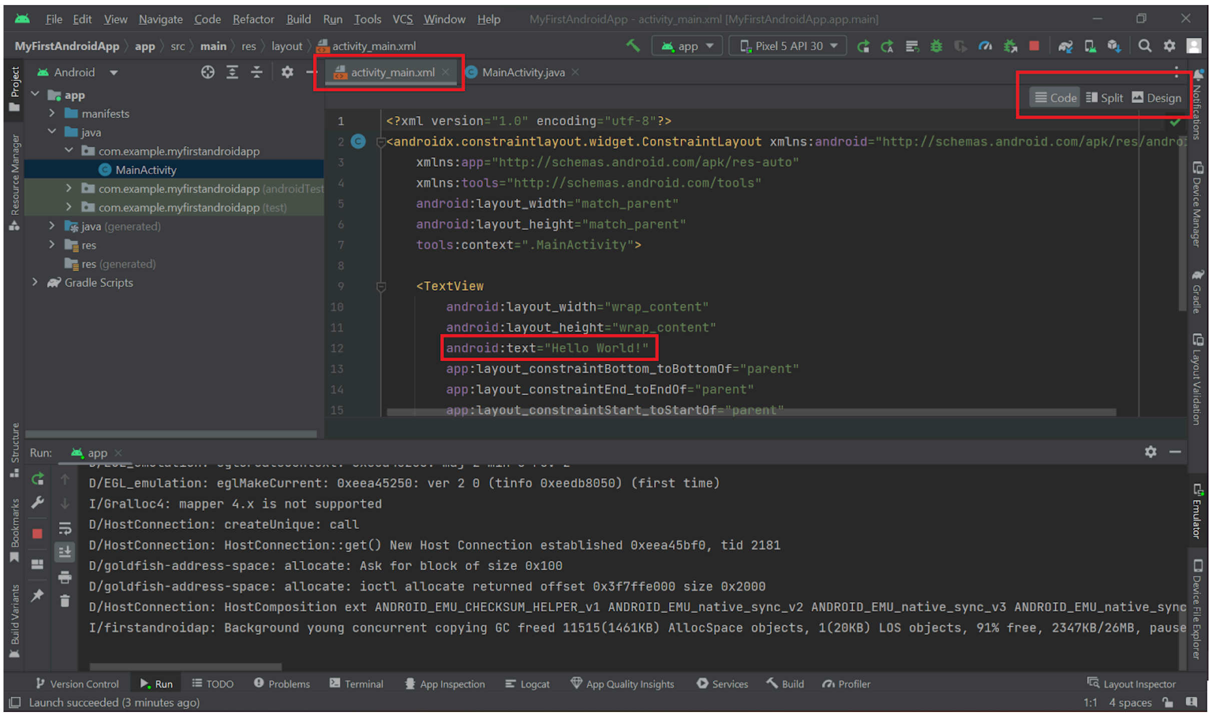
```
D/EGL_emulation: eglMakeCurrent: 0xeea45250: ver 2 0 (tinfo 0xeedb8050) (first time)
I/Gralloc4: mapper 4.x is not supported
D/HostConnection: createUnique: call
D/HostConnection: HostConnection::get() New Host Connection established 0xeea45bf0, tid 2181
D/goldfish-address-space: allocate: Ask for block of size 0x100
D/goldfish-address-space: allocate: ioctl allocate returned offset 0x3f7ffe000 size 0x2000
D/HostConnection: HostComposition ext ANDROID_EMU_CHECKSUM_HELPER_v1 ANDROID_EMU_native_sync_v2 ANDROID_EMU_native_sync_v3 ANDROID_EMU_native_sync
I/firstandroidap: Background young concurrent copying GC freed 11515(1461KB) AllocSpace objects, 1(20KB) LOS objects, 91% free, 2347KB/26MB, pause
```

Launch succeeded

Launch succeeded (moments ago)

Explore activity_main.xml further

Select **activity_main.xml**, followed by **Code**. You should see the following screen.
You'll see that the layout has one **TextView widget**, which is specified by an XML tag.
A TextView widget is a UI element that is responsible for displaying text on the **Activity**.
Notice that one setting within the XML tag specifies the text to be displayed.
Try changing the text, click on Run and see the change in your layout.



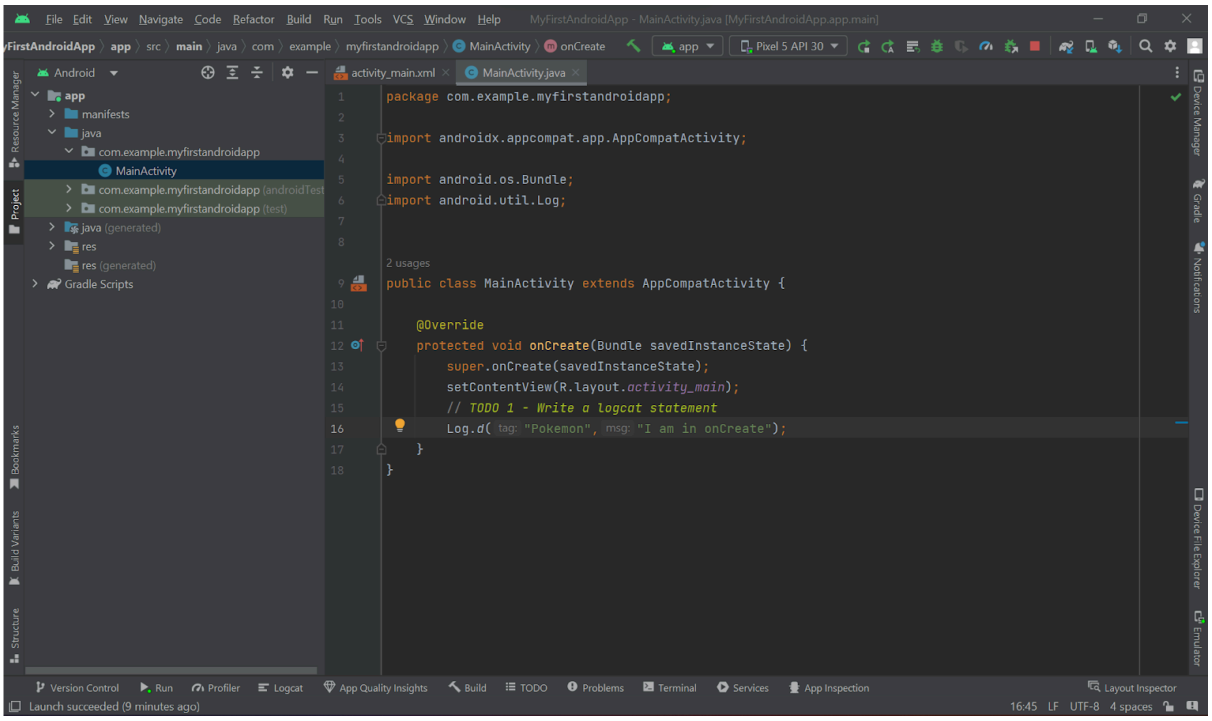
```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/andro
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello World!"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
```

```
D/EGL_emulation: eglMakeCurrent: 0xeea45250: ver 2 0 (tinfo 0xeedb8050) (first time)
I/Gralloc4: mapper 4.x is not supported
D/HostConnection: createUnique: call
D/HostConnection: HostConnection::get() New Host Connection established 0xeea45bf0, tid 2181
D/goldfish-address-space: allocate: Ask for block of size 0x100
D/goldfish-address-space: allocate: ioctl allocate returned offset 0x3f7ffe000 size 0x2000
D/HostConnection: HostComposition ext ANDROID_EMU_CHECKSUM_HELPER_v1 ANDROID_EMU_native_sync_v2 ANDROID_EMU_native_sync_v3 ANDROID_EMU_native_sync
I/firstandroidap: Background young concurrent copying GC freed 11515(1461KB) AllocSpace objects, 1(20KB) LOS objects, 91% free, 2347KB/26MB, pause
```

Launch succeeded (3 minutes ago)

Write some code in MainActivity.java

Click on **MainActivity.java** and write the following code as shown in the screenshot. This shows you how to write TODO comment statements and Logcat statements. These statements are useful in managing your project. Android Studio has tabs below to help you keep track of these statements.



```java
package com.example.myfirstandroidapp;

import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;
import android.util.Log;

2 usages
public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        // TODO 1 - Write a logcat statement
        Log.d( tag: "Pokemon", msg: "I am in onCreate");
    }
}
```

Further Reading

- **Building your first Android App in Java**
  https://developer.android.com/codelabs/build-your-first-android-app#0

- **Android Developer Fundamentals Concepts**
  - Section 1.0
  - Section 1.1

- **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 further explains the purpose of the different folder in the res folder.
  https://google-developer-training.github.io/android-developer-fundamentals-course-concepts-v2/unit-1-get-started/lesson-1-build-your-first-app/1-2-c-layouts-and-resources-for-the-ui/1-2-c-layouts-and-resources-for-the-ui.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.github.io/android-developer-fundamentals-course-concepts-v2/unit-1-get-started/lesson-1-build-your-first-app/1-3-c-text-and-scrolling-views/1-3-c-text-and-scrolling-views.html

- **Using the logcat**
  - **Why use it when you have System.out.println()**
    https://stackoverflow.com/questions/2364811/how-do-i-write-outputs-to-the-log-in-android
  - **Viewing logs with logcat -** plenty of details if you wish to know more
    https://developer.android.com/studio/debug/logcat