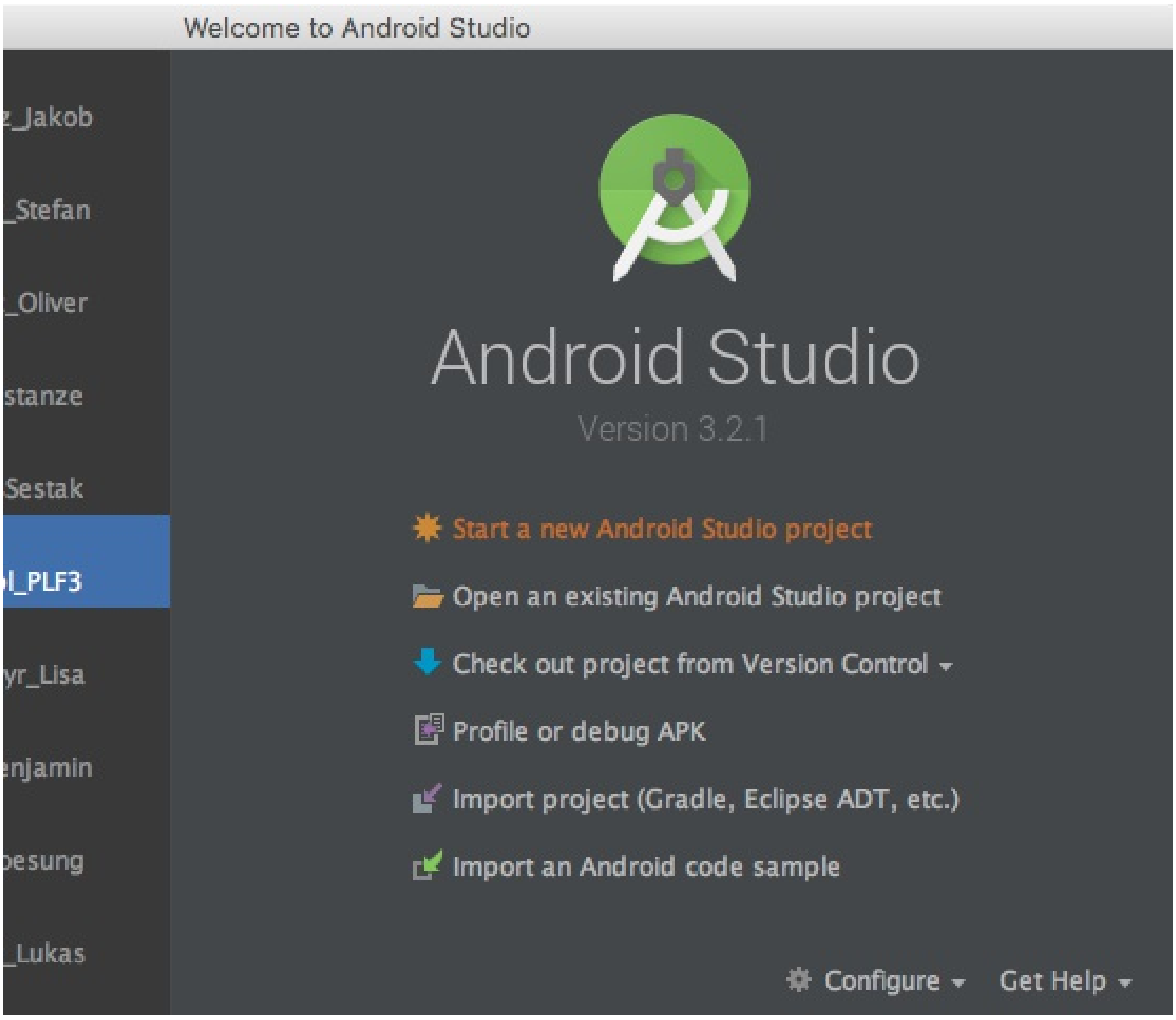
Using Android Studio

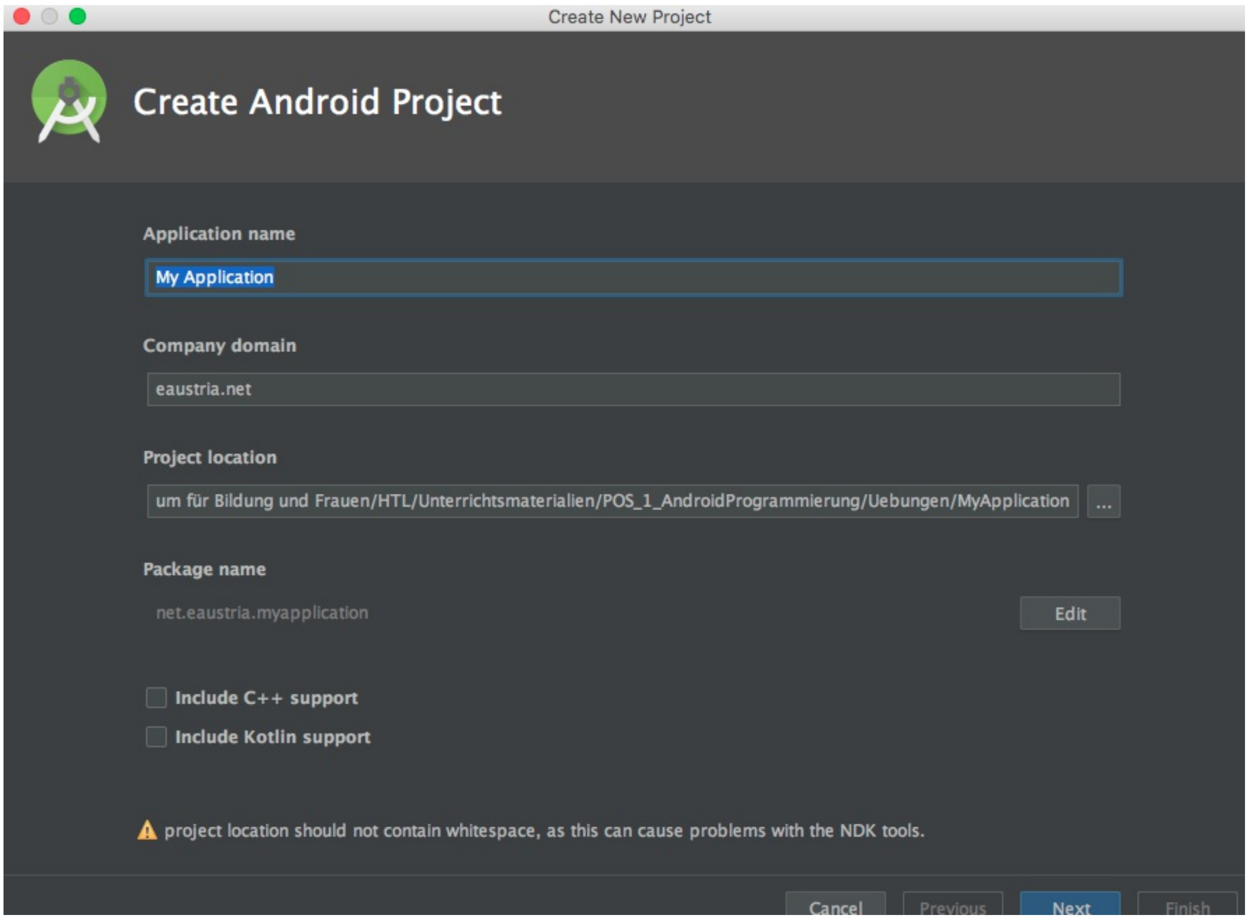
This script discusses several feature and usage of Android Studio.

Starting a new Android Project

After starting Android Studio a welcomescreen is shown to the user. On this screen you can reopen former projects on the left side or start a new project on the right side.


Welcome to Android Studio
z_Jakob
_Stefan
_Oliver
stanze
Sestak
l_PLF3
yr_Lisa
enjamin
pesung
_Lukas
Android Studio
Version 3.2.1
Start a new Android Studio project
Open an existing Android Studio project
Check out project from Version Control
Profile or debug APK
Import project (Gradle, Eclipse ADT, etc.)
Import an Android code sample
Configure
Get Help

If you have already created a project on another PC and you want to open it in school, please use `import project`.
If you create a new project you want to use the wizzard to create the project stub.


Create New Project

Create Android Project

Application name: My Application

Company domain: eaustria.net

Project location: um für Bildung und Frauen/HTL/Unterrichtsmaterialien/POS_1_AndroidProgrammierung/Uebungen/MyApplication

Package name: net.eaustria.myapplication

Include C++ support

Include Kotlin support

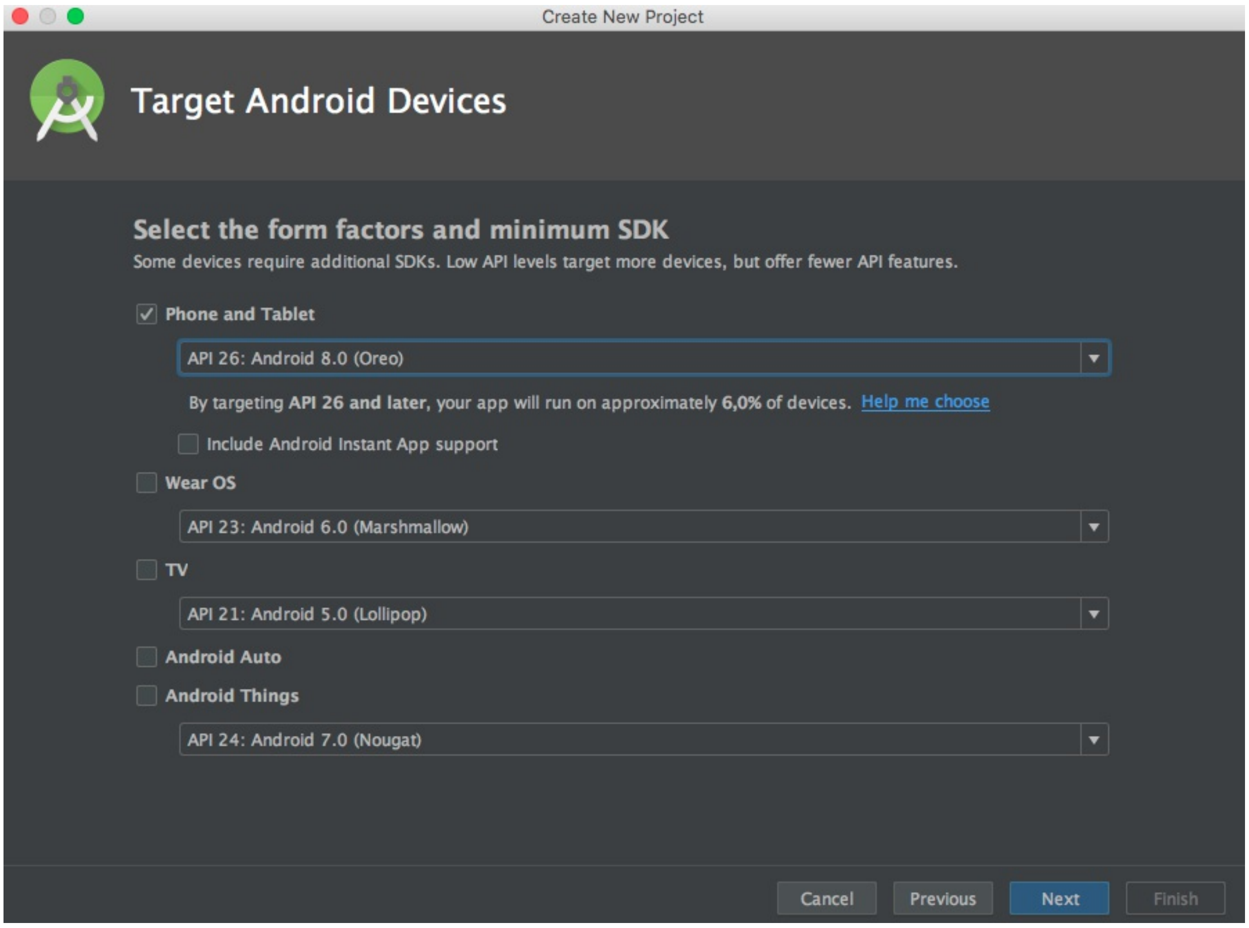
project location should not contain whitespace, as this can cause problems with the NDK tools.

Define your application name (this will be the name of your app if you deploy to google play store).

Use a domain name for `company domain`. This doesn't have to be a real domain but will effect your package structure later on.

The field `Project location` is the directory into which android studio will put all the project files.

Click next to get to the selection of the form factors of your app. Select `Phone and Tablet` on all our training and assignment examples. We will use API level 26 (*Android 8.0 Oreo*) on all our projects to be able to use the most recent features.


Create New Project

Target Android Devices

Select the form factors and minimum SDK

Some devices require additional SDKs. Low API levels target more devices, but offer fewer API features.

[✓] Phone and Tablet

API 26: Android 8.0 (Oreo)

By targeting **API 26 and later**, your app will run on approximately **6,0%** of devices. Help me choose

[ ] Include Android Instant App support

[ ] Wear OS

API 23: Android 6.0 (Marshmallow)

[ ] TV

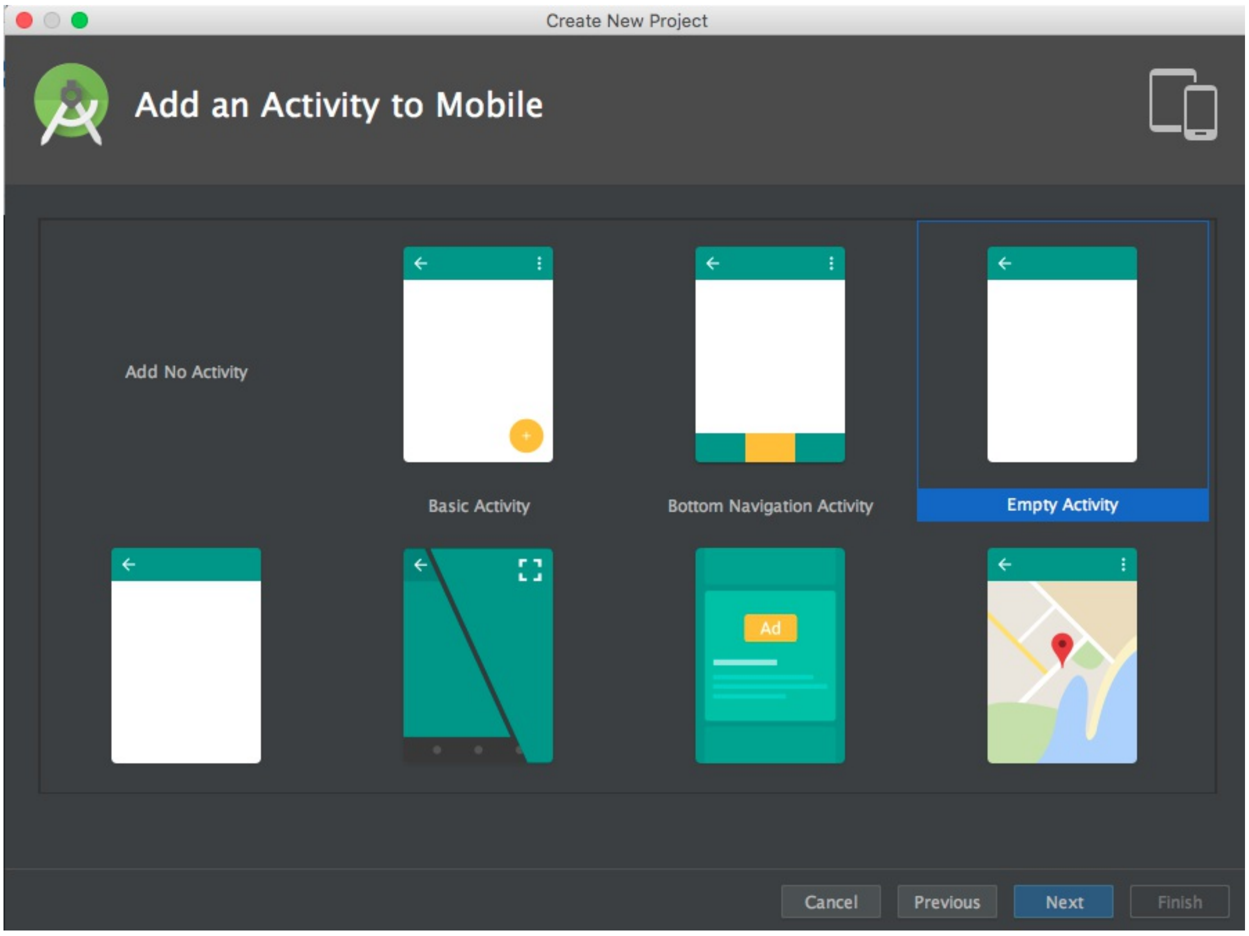
API 21: Android 5.0 (Lollipop)

[ ] Android Auto

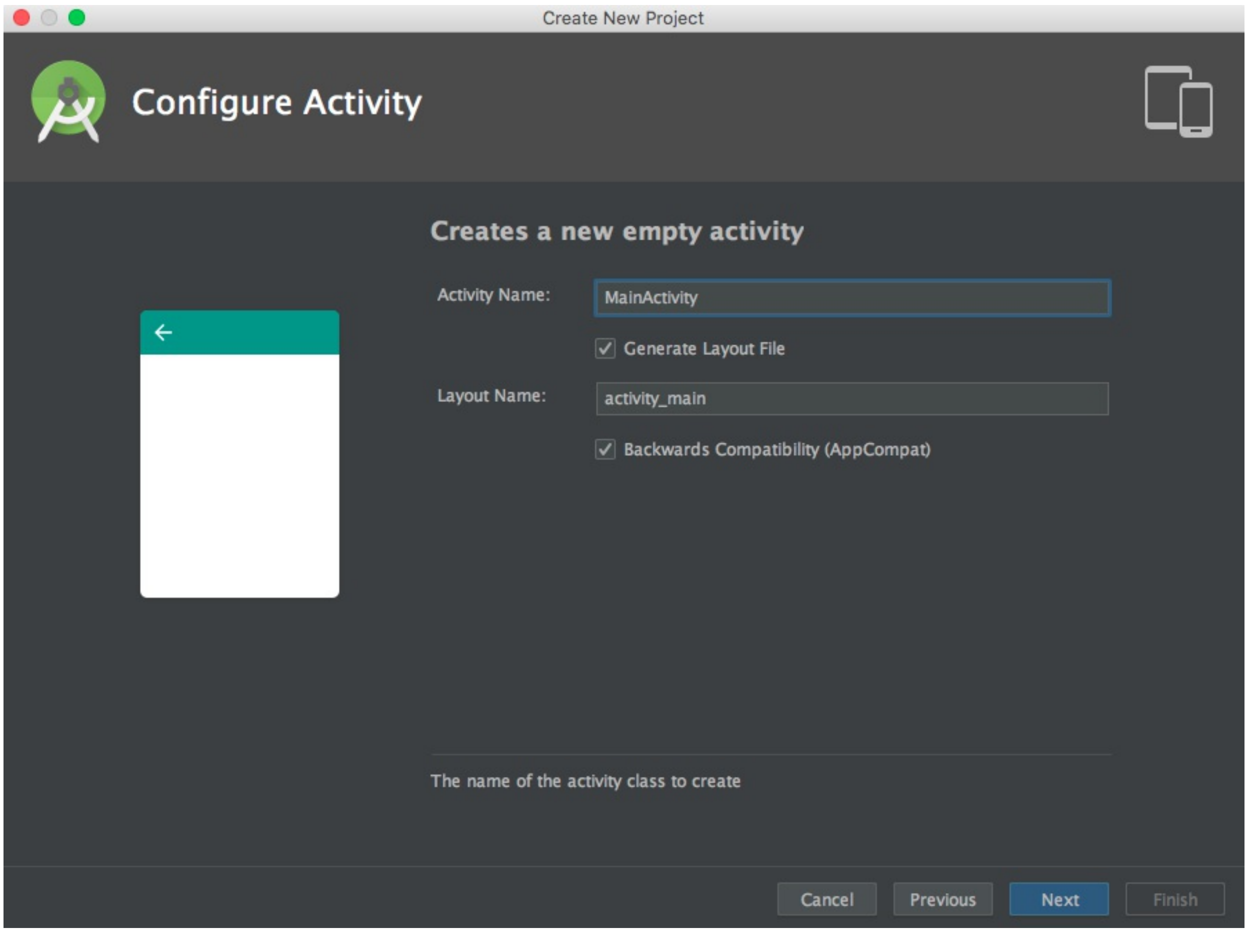
[ ] Android Things

API 24: Android 7.0 (Nougat)

Next choose the activity format the wizzard should prepare. We will start choosing an empty activity.



On the next window we could change the names of some of the files which the wizzard will auto-generate for us. For the sake of simplicity we will keep the defaults.



Activity Name: MainActivity

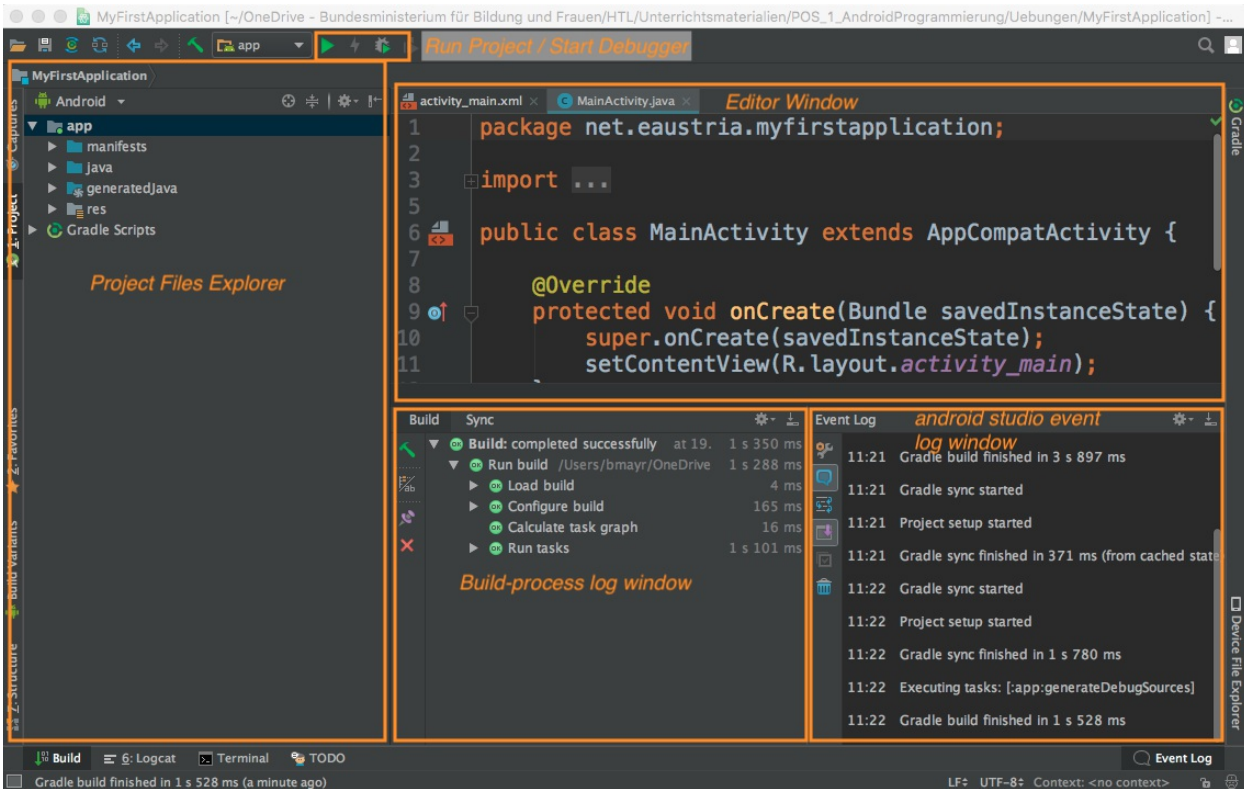
☑ Generate Layout File

Layout Name: activity_main

☑ Backwards Compatibility (AppCompat)

The name of the activity class to create

Now the build process will start and after a while you should see your android studio ide looking something like this:



```
package net.eaustria.myfirstapplication;

import ...

public class MainActivity extends AppCompatActivity {

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

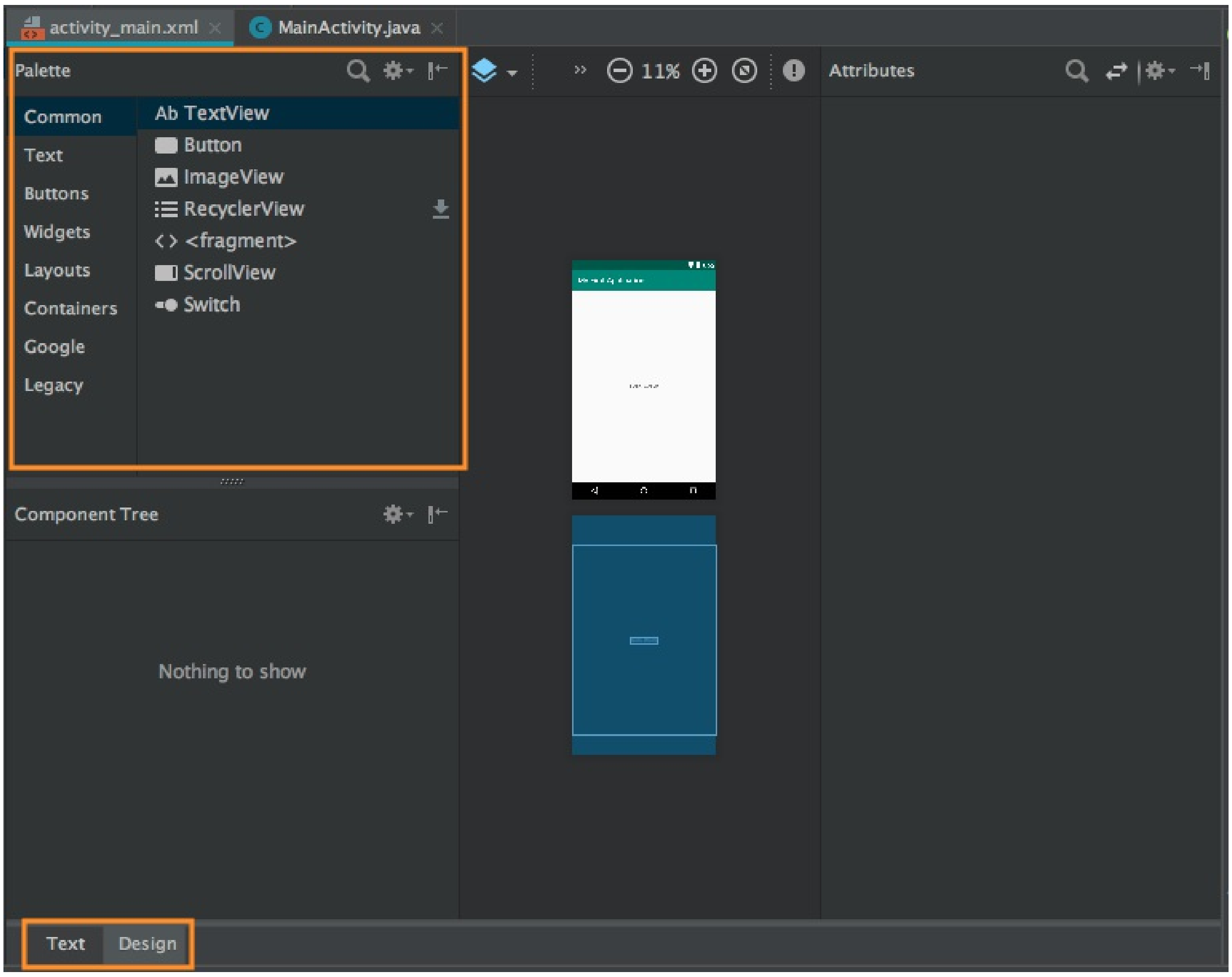
```
Build: completed successfully   at 19.   1 s 350 ms
  Run build  /Users/bmayr/OneDrive   1 s 288 ms
    Load build                             4 ms
    Configure build                      165 ms
    Calculate task graph                  16 ms
    Run tasks                       1 s 101 ms
```

```
11:21  Gradle build finished in 3 s 897 ms
11:21  Gradle sync started
11:21  Project setup started
11:21  Gradle sync finished in 371 ms (from cached state
11:22  Gradle sync started
11:22  Project setup started
11:22  Gradle sync finished in 1 s 780 ms
11:22  Executing tasks: [:app:generateDebugSources]
11:22  Gradle build finished in 1 s 528 ms
```

```
Gradle build finished in 1 s 528 ms (a minute ago)          LF÷  UTF-8÷  Context: <no context>
```

Android Editor Window

Let's have a closer look on the editor window. This is a code editor for your java files but a dual-editor for your layout files. If you open a layout file (e.g. activity_main) you will see two tabs (*Text / Design*) on the lower edge. There you can switch between text and WYSIWG editing of your layout files. When choosing the *Design* Tab you see your palette of UI elements on the left and a preview of the layout on the right side.



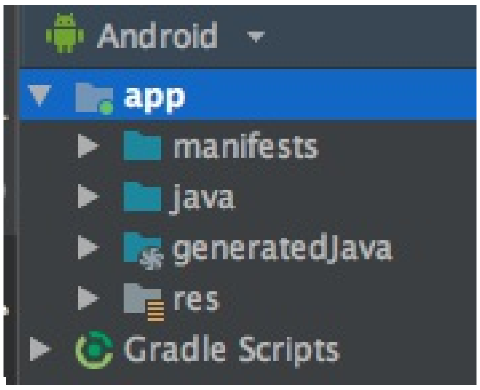
Project Files Organisation

When working on Android projects we will have to deal with these categories of files:

- Gradle files
- Source Code files
- Resource files

1. **Gradle files**: Gradle files are the scripts that define how to compile and build your apps. We won't discuss these in much detail right now but I'll show you where they are for the

time being.
2. **Source Code Files**: These are the Java source code files that define the logic of your application. The third major group of files we'll show you are the Resource Files.
3. **Resource Files**: These are files that your app makes use of while it is running. Such as the user interface layout files that define what different screens of your app look like. Any images that are included within your app, including the app's icon itself. And XML files containing values used by your app. One example XML file resource is called `strings.xml`. This file is where you put all the words displayed by your app. That way you can easily translate every word into a different language, and easily swap in and out which language should be used based on the users language settings.



If we choose the `Android` Project view we will see two main categories. `app` and `Gradle Scripts`.

The **app** portion is were your project files exist. There are three main categories of files in the Android Studio project. Manifest which are special XML files that describe how the Android platform should interact with your project.

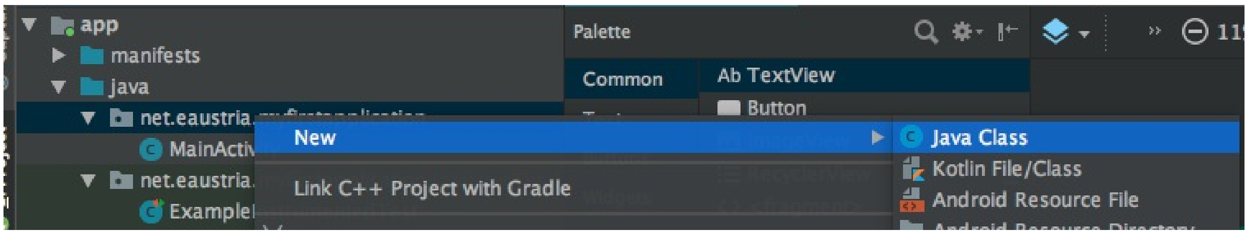
1. **manifests**: The manifest folder will only contain a single file for all projects you work on for this course. That file is called `AndroidManifest.xml`. This file defines how the Android platform, on each user's device, will interact with our app.
2. **java**: The java folder contains the java packages for your source code and your unit- and instrumented-tests. More on the difference between unit and instrumented tests later on.
3. **generatedJava**: Autogenerated code. Do not modify the files in this folder.
4. **res**: The Resources folder, abbreviated *res*. The Resources folder contains things such as images, user interface layout files, and other files that hold special value for us. Within the `res` folder we will find the `/values/strings.xml` file. This file contains all the language specific files of your android project like a *ressouce bundle* in javaFX applications.

Creating and Editing Files in Android Projects

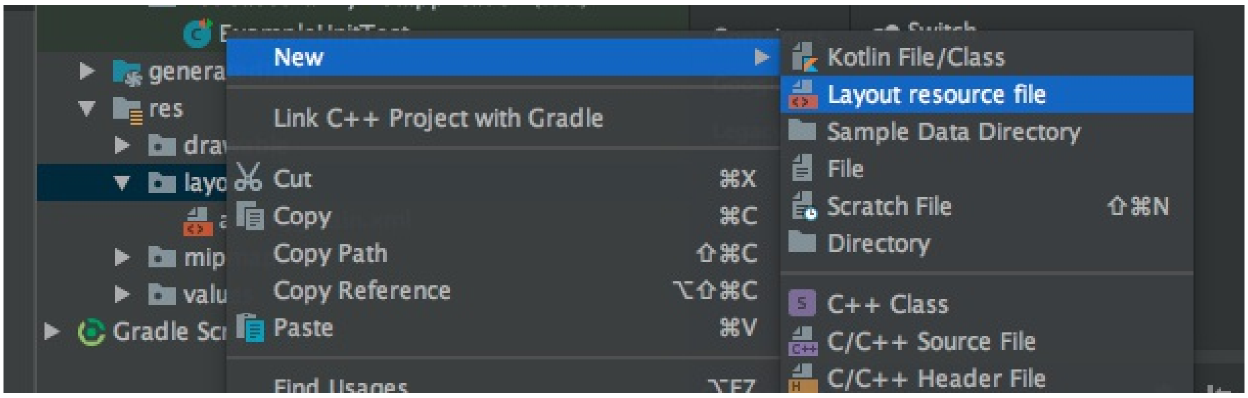
Android Studio has two main sets of files that you will edit. **Java source code files__ that**

contain the logic of your app. And __resource files, such as the user interface layout files.

To create a new java activity file just right click on the package you want your class file to reside and choose `New` --> `Java Class` .



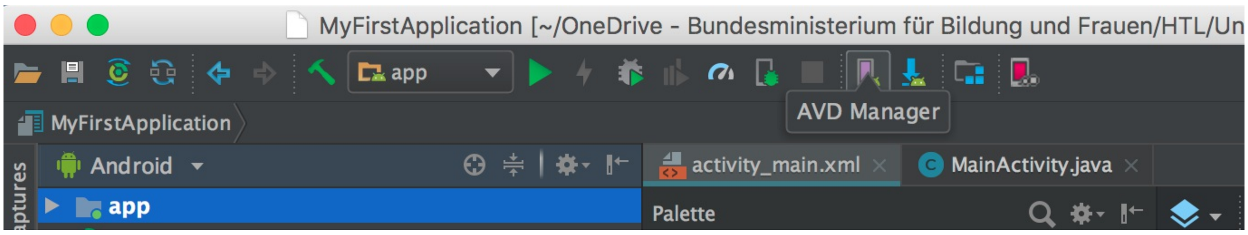
To create a new layout file is just as easy as creating a java class file. Just right click on the layout folder within our res portion in the project explorer.



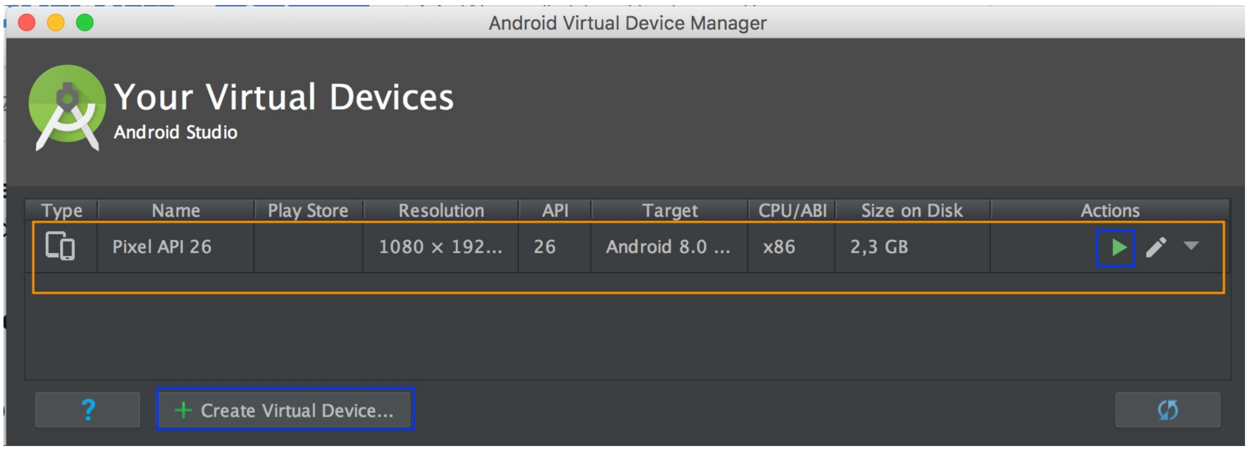
Creating and Running a Virtual device

Virtual Machines or VMs allow you to run code on your development machine as though it were inside a physical Android device. One of the reasons this is useful is for testing run time behavior of your app when you don't have a physical device to test your code on.

The first step in managing, creating or starting a virtual device running Android is to open the Android Virtual Device Manager:



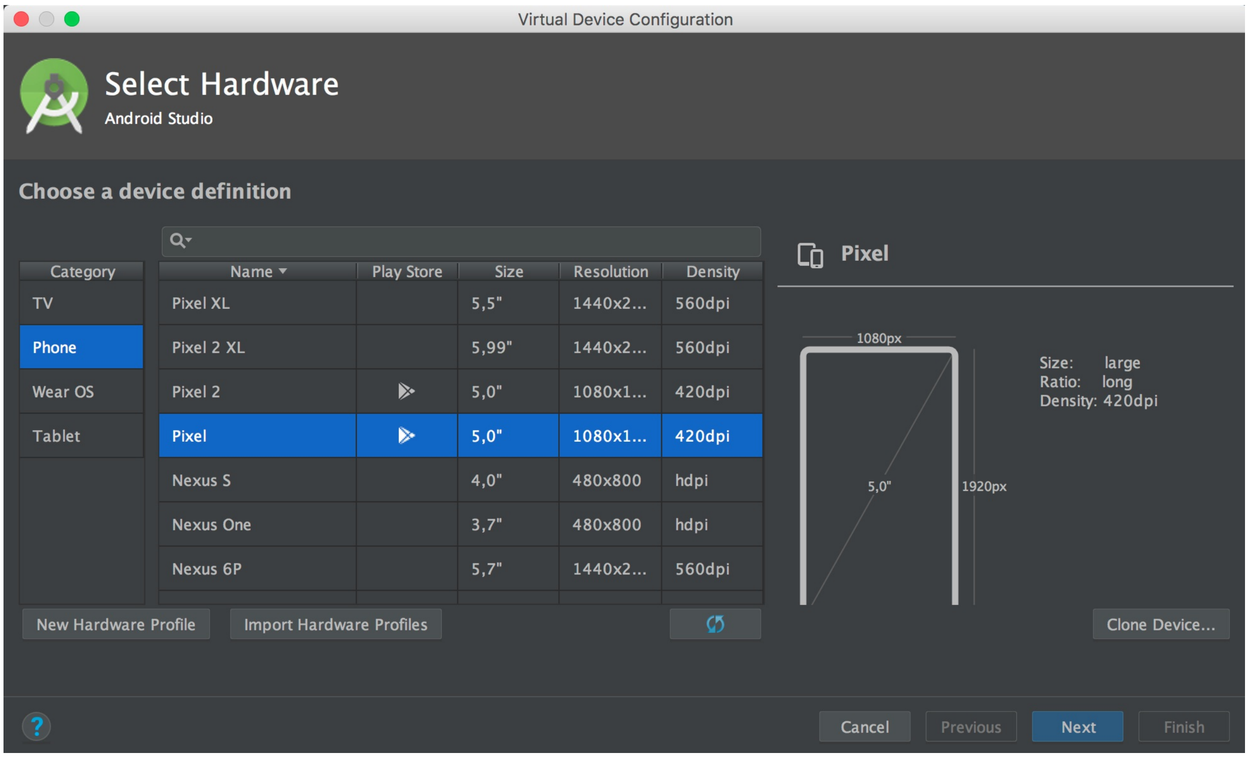
After opening the Android Virtual Device Manager you see a list of all your VMs. If you haven't created any, this list is empty. You can click `+Create Virtual Device` to create new VMs. To start an existing VM, just click on the green arrow on the right hand side of the Device.


Android Virtual Device Manager

Your Virtual Devices
Android Studio

Type	Name	Play Store	Resolution	API	Target	CPU/ABI	Size on Disk	Actions
	Pixel API 26		1080 × 192...	26	Android 8.0 ...	x86	2,3 GB	

We want to create a new device so we click on '+Create Virtual Device'. Another dialog opens and we can specifiy several aspects of our VM. We choose *category Phone* and use the *Pixel* emulator. Then just click next to proceed.


Virtual Device Configuration

Select Hardware
Android Studio

Choose a device definition

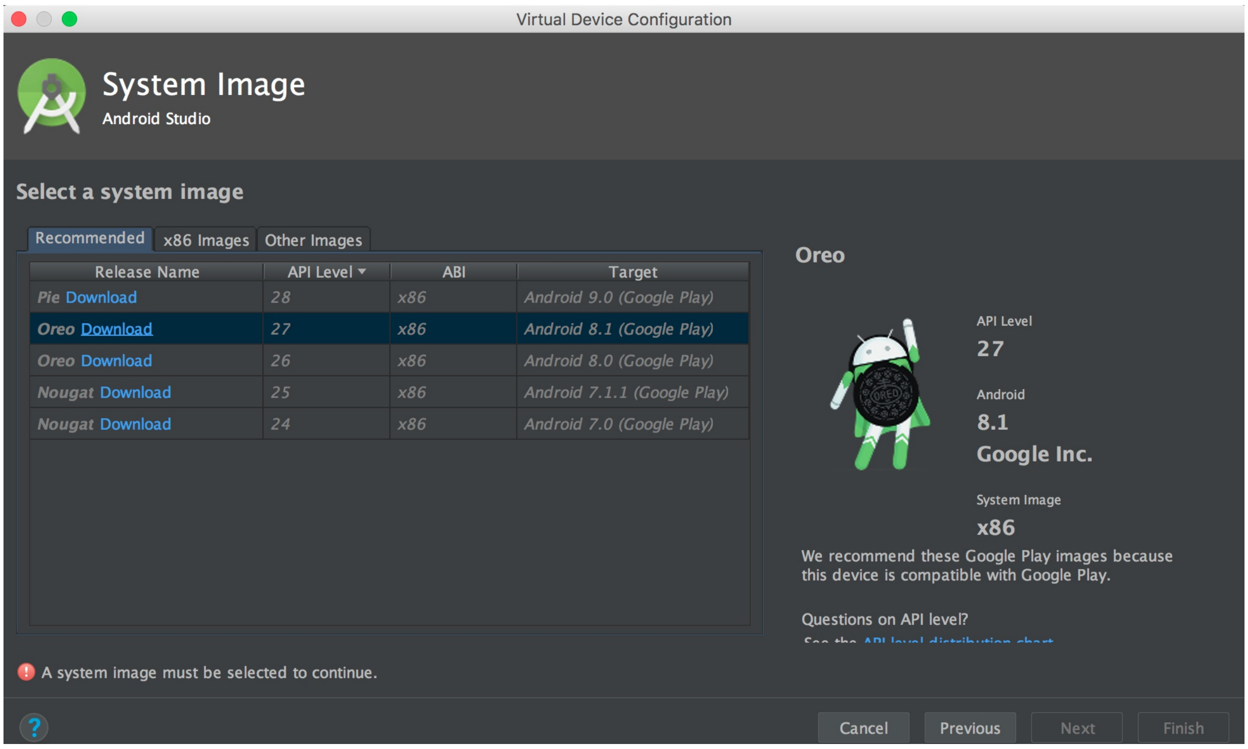
Category
TV
Phone
Wear OS
Tablet

Name	Play Store	Size	Resolution	Density
Pixel XL		5,5"	1440x2...	560dpi
Pixel 2 XL		5,99"	1440x2...	560dpi
Pixel 2	▶	5,0"	1080x1...	420dpi
Pixel	▶	5,0"	1080x1...	420dpi
Nexus S		4,0"	480x800	hdpi
Nexus One		3,7"	480x800	hdpi
Nexus 6P		5,7"	1440x2...	560dpi

Pixel

Size: large
Ratio: long
Density: 420dpi

Next we have to choose the API Level and the ABI. These are the two central columns. The **API level__ is the API level of the image or Android operating system version that you want to run on your virtual device. The __ABI, or Application Binary Interface**, is the specification of what kind of CPU architecture your virtual device will emulate. The important thing to know from this is that the Arm EABI Version 7 is the most universal. But it will run slower on our machines because it isn't of the same instruction set as our desktops or laptops. Therefore, running an X86 or an X8664 image would be faster. However, your device, meaning your laptop or desktop, needs to have a CPU that allows for virtualization, and needs to have the feature turned on. For the sake of performance we will use x86 ABI for our emulators.



Select a system image

Recommended | x86 Images | Other Images

Release Name	API Level	ABI	Target
Pie Download	28	x86	Android 9.0 (Google Play)
Oreo Download	27	x86	Android 8.1 (Google Play)
Oreo Download	26	x86	Android 8.0 (Google Play)
Nougat Download	25	x86	Android 7.1.1 (Google Play)
Nougat Download	24	x86	Android 7.0 (Google Play)

Oreo

API Level
27

Android
8.1
Google Inc.

System Image
x86

We recommend these Google Play images because this device is compatible with Google Play.

Questions on API level?

A system image must be selected to continue.

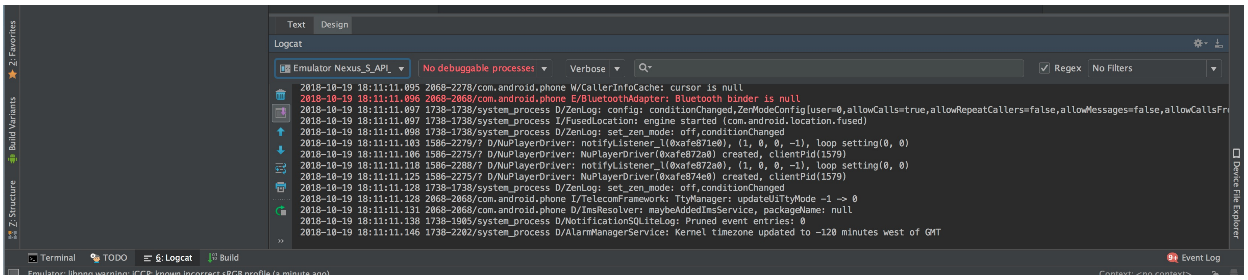
The last window will display additional features that you can change. But for now, we will not changing anything and just clicking finish.

Now the recently created emulator devices appears in the list and we start it by clicking the green arrow.

As soon as the emulator has finished the boot process we can run our app on the emulator by clicking the *run app* Button (green triangle) in Android Studio.

A dialog pops up which lets us choose the device we will run our app on.

After starting the emulator you will see many log messages in the `Logcat` window of Android Studio. This is where you can see all the logs from the emulator. We will discuss this window in a later lesson in depth.



```
2018-10-19 18:11:11.095 2068-2278/com.android.phone W/CallerInfoCache: cursor is null
2018-10-19 18:11:11.096 2068-2068/com.android.phone E/BluetoothAdapter: Bluetooth binder is null
2018-10-19 18:11:11.097 1738-1738/system_process D/ZenLog: config: conditionChanged,ZenModeConfig[user=0,allowCalls=true,allowRepeatCallers=false,allowMessages=false,allowCallsFr
2018-10-19 18:11:11.097 1738-1738/system_process I/FusedLocation: engine started (com.android.location.fused)
2018-10-19 18:11:11.098 1738-1738/system_process D/ZenLog: set_zen_mode: off,conditionChanged
2018-10-19 18:11:11.103 1586-2279/? D/NuPlayerDriver: notifyListener_l(0xafe871e0), (1, 0, 0, -1), loop setting(0, 0)
2018-10-19 18:11:11.106 1586-2275/? D/NuPlayerDriver: NuPlayerDriver(0xafe872a0) created, clientPid(1579)
2018-10-19 18:11:11.118 1586-2288/? D/NuPlayerDriver: notifyListener_l(0xafe872a0), (1, 0, 0, -1), loop setting(0, 0)
2018-10-19 18:11:11.125 1586-2275/? D/NuPlayerDriver: NuPlayerDriver(0xafe874e0) created, clientPid(1579)
2018-10-19 18:11:11.128 1738-1738/system_process D/ZenLog: set_zen_mode: off,conditionChanged
2018-10-19 18:11:11.128 2068-2068/com.android.phone I/TelecomFramework: TtyManager: updateUiTtyMode -1 -> 0
2018-10-19 18:11:11.131 2068-2068/com.android.phone D/ImsResolver: maybeAddedImsService, packageName: null
2018-10-19 18:11:11.138 1738-1905/system_process D/NotificationSQLiteLog: Pruned event entries: 0
2018-10-19 18:11:11.146 1738-2202/system_process D/AlarmManagerService: Kernel timezone updated to -120 minutes west of GMT
```

Debugging in Android Studio