Android: Basics on development

Desenvolvimento de Software e Sistemas Móveis (DSSMV)
Licenciatura em Engenharia de Telecomunicações e Informática
LETI/ISEP

2025/26

Paulo Baltarejo Sousa and Carlos Filipe Freitas
{pbs,caf}@isep.ipp.pt

Disclaimer

Material and Slides

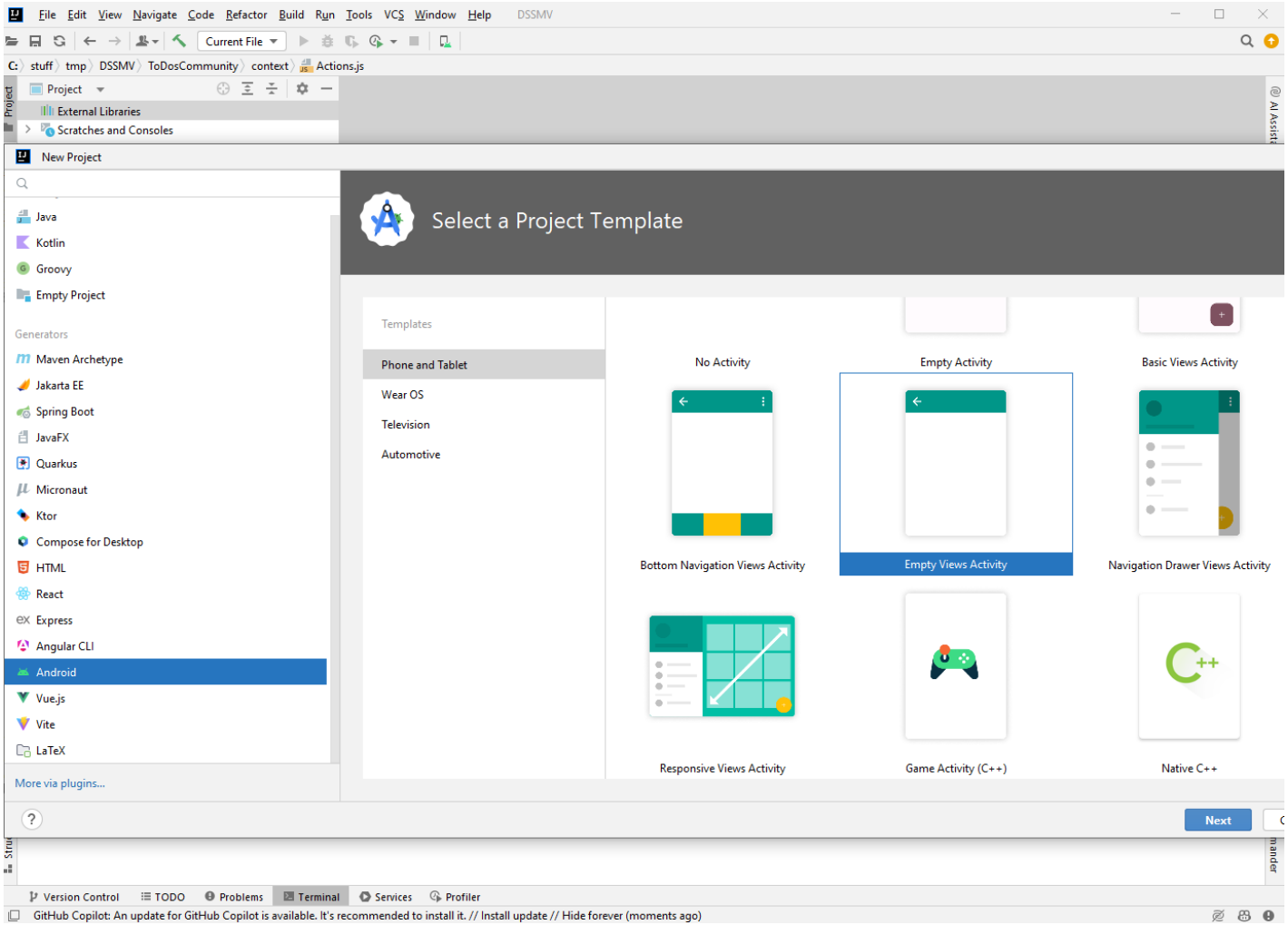
Some of the material/slides are adapted from various:

- Presentations found on the internet;
- Books;
- Web sites;
- ...

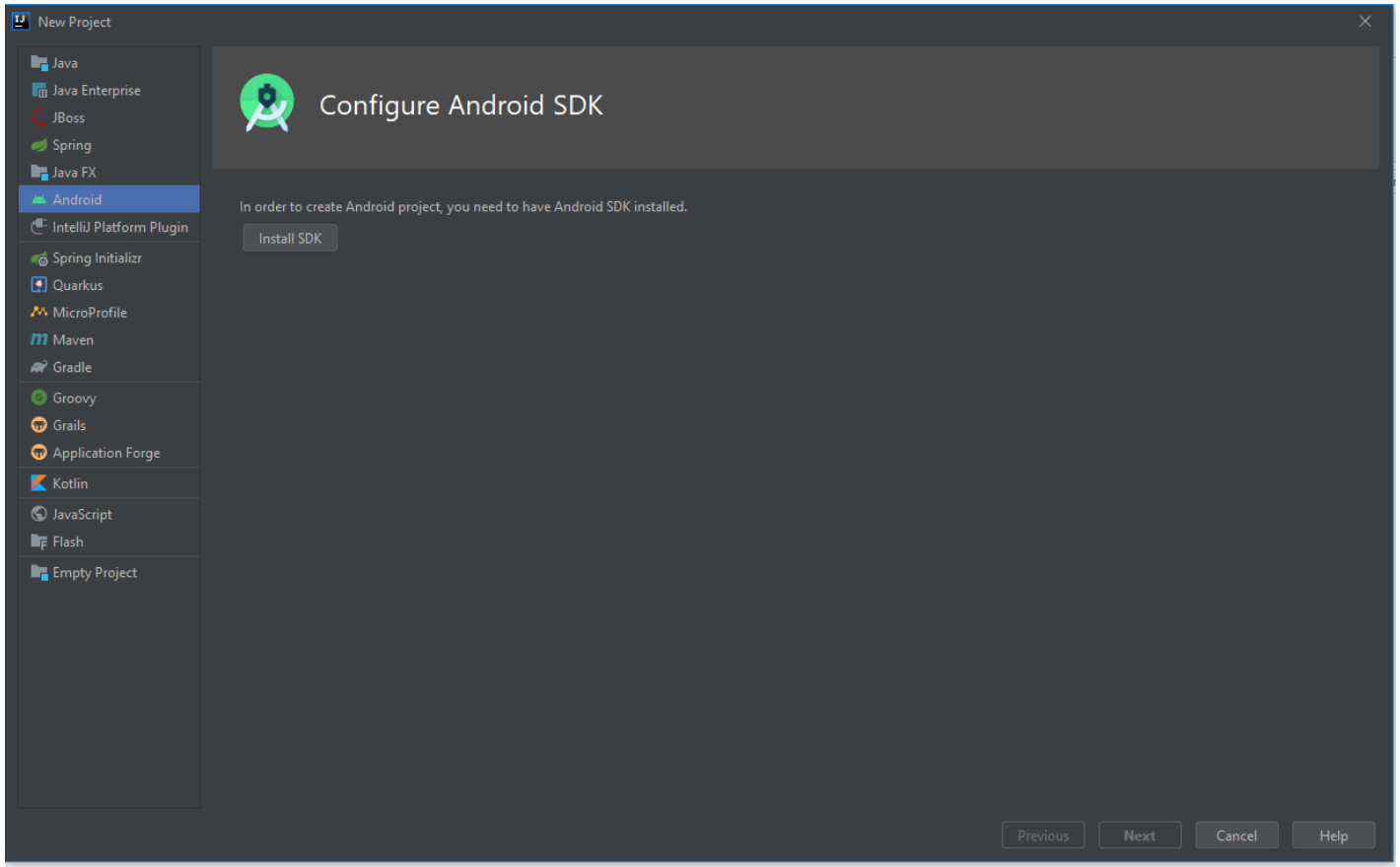
Outline

Setting up IntelliJ IDEA

Create a Project (I)

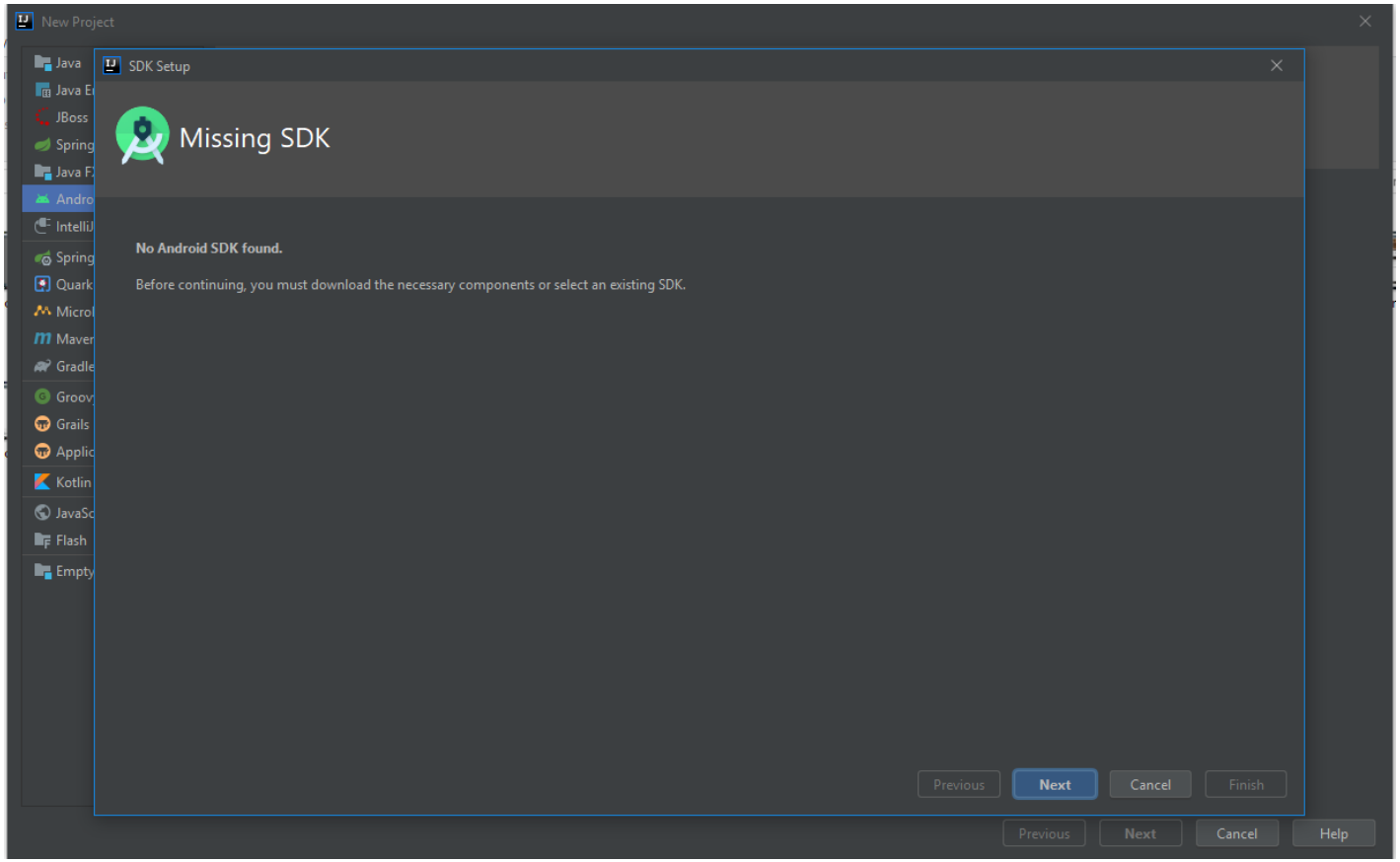


Install Android SDK (I)



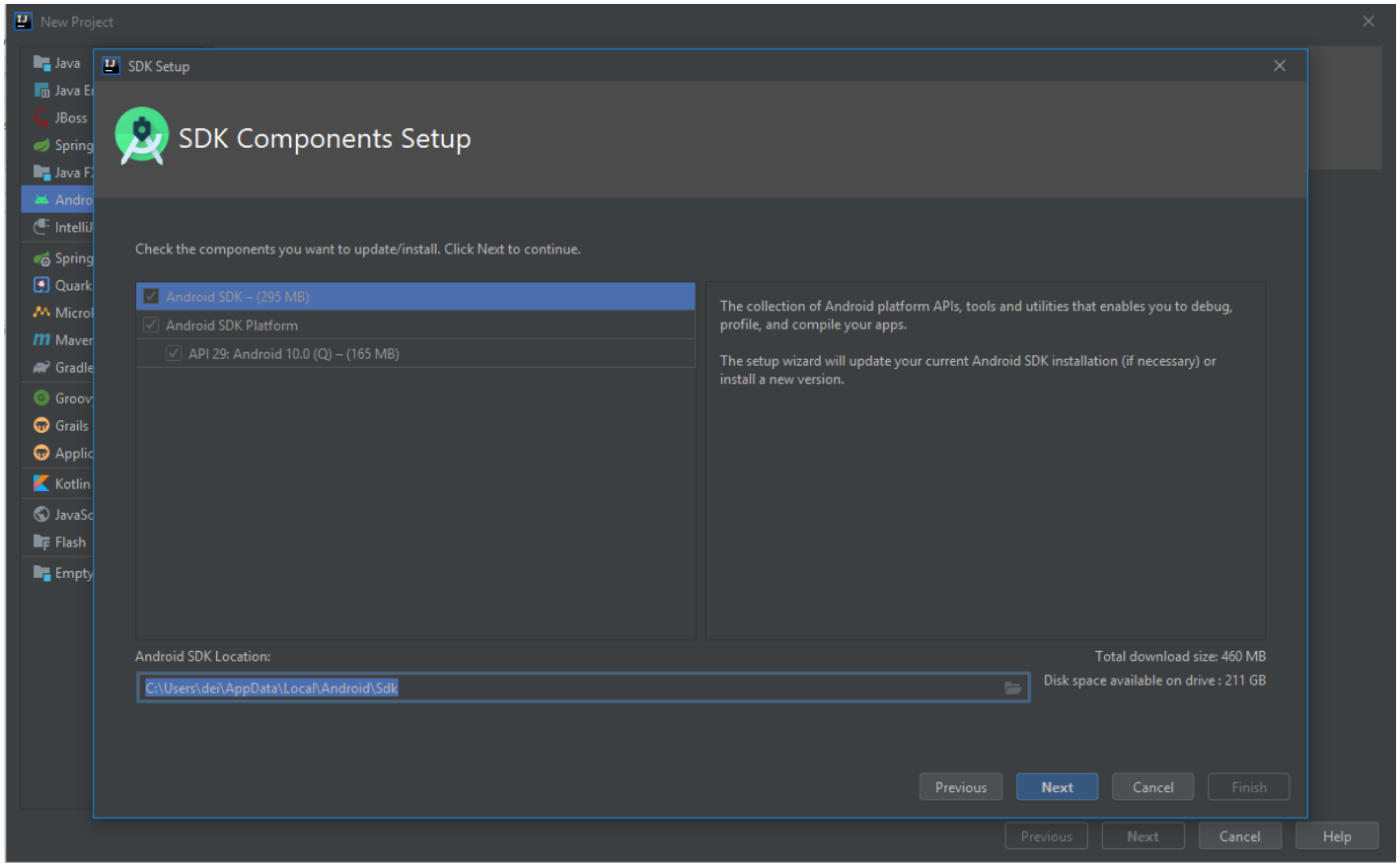
- Select `Android`
- Click on `Install SDK`

Install Android SDK (II)



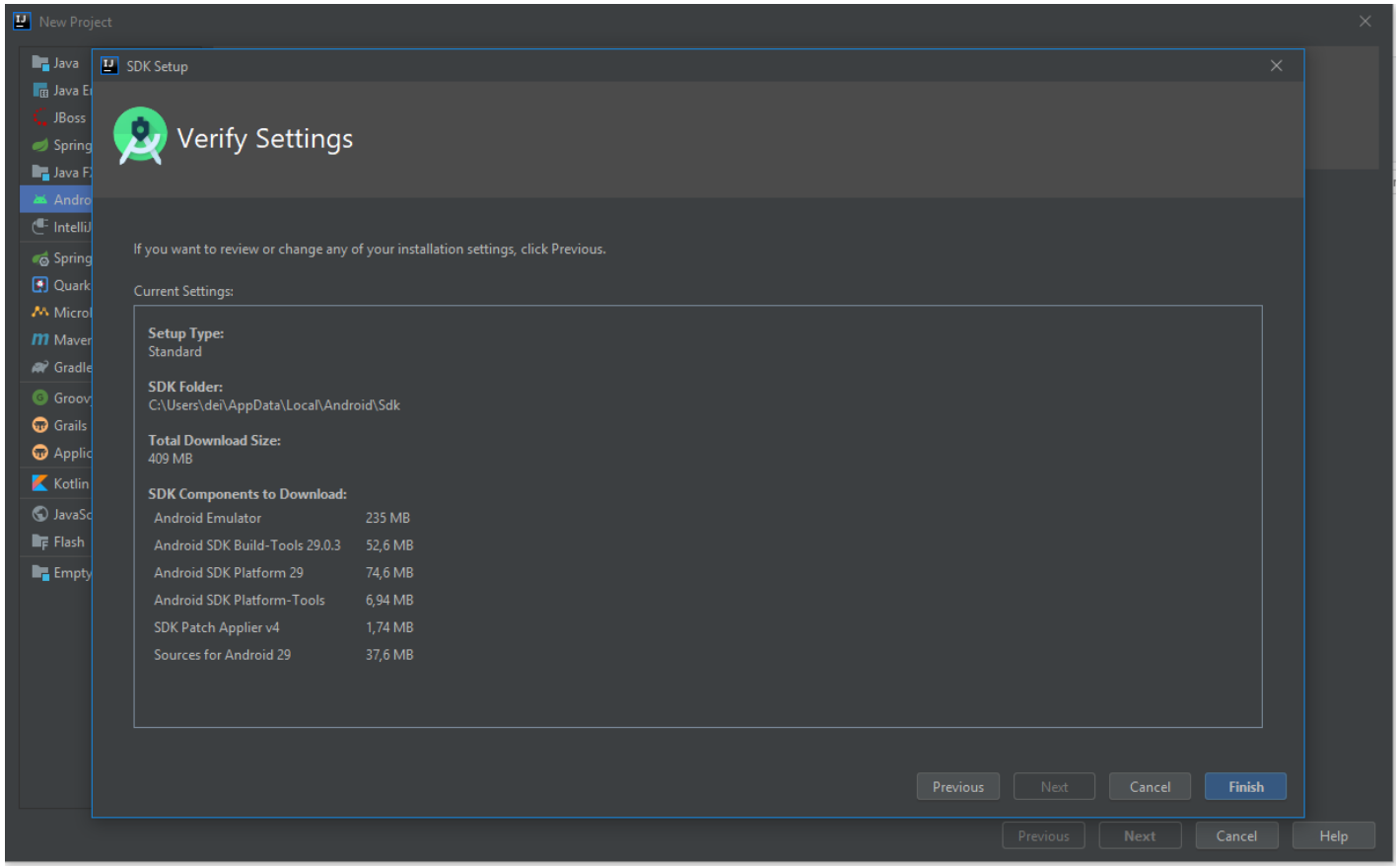
- Click on `Next`

Install Android SDK (III)



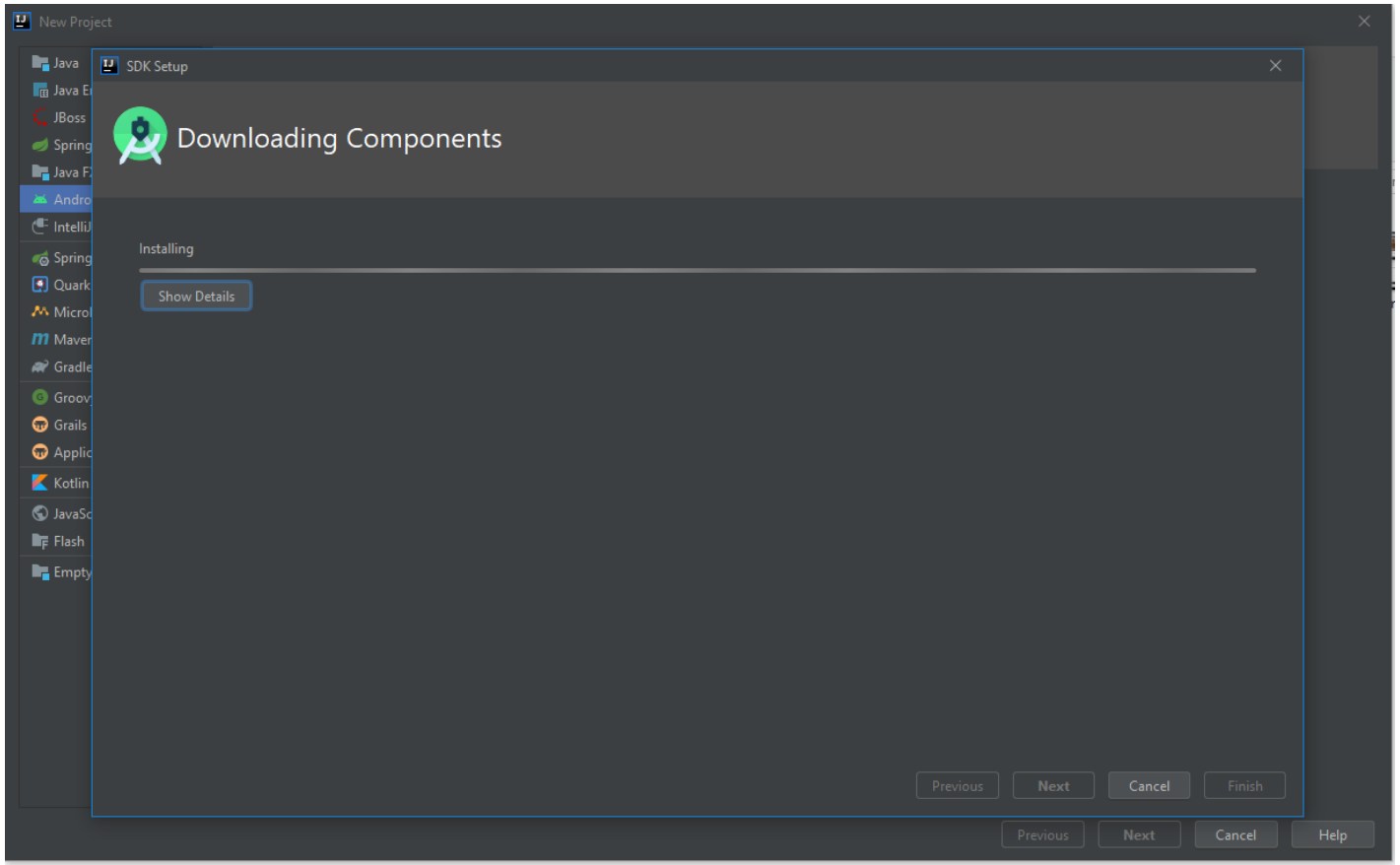
- Click on `Next`

Install Android SDK (IV)

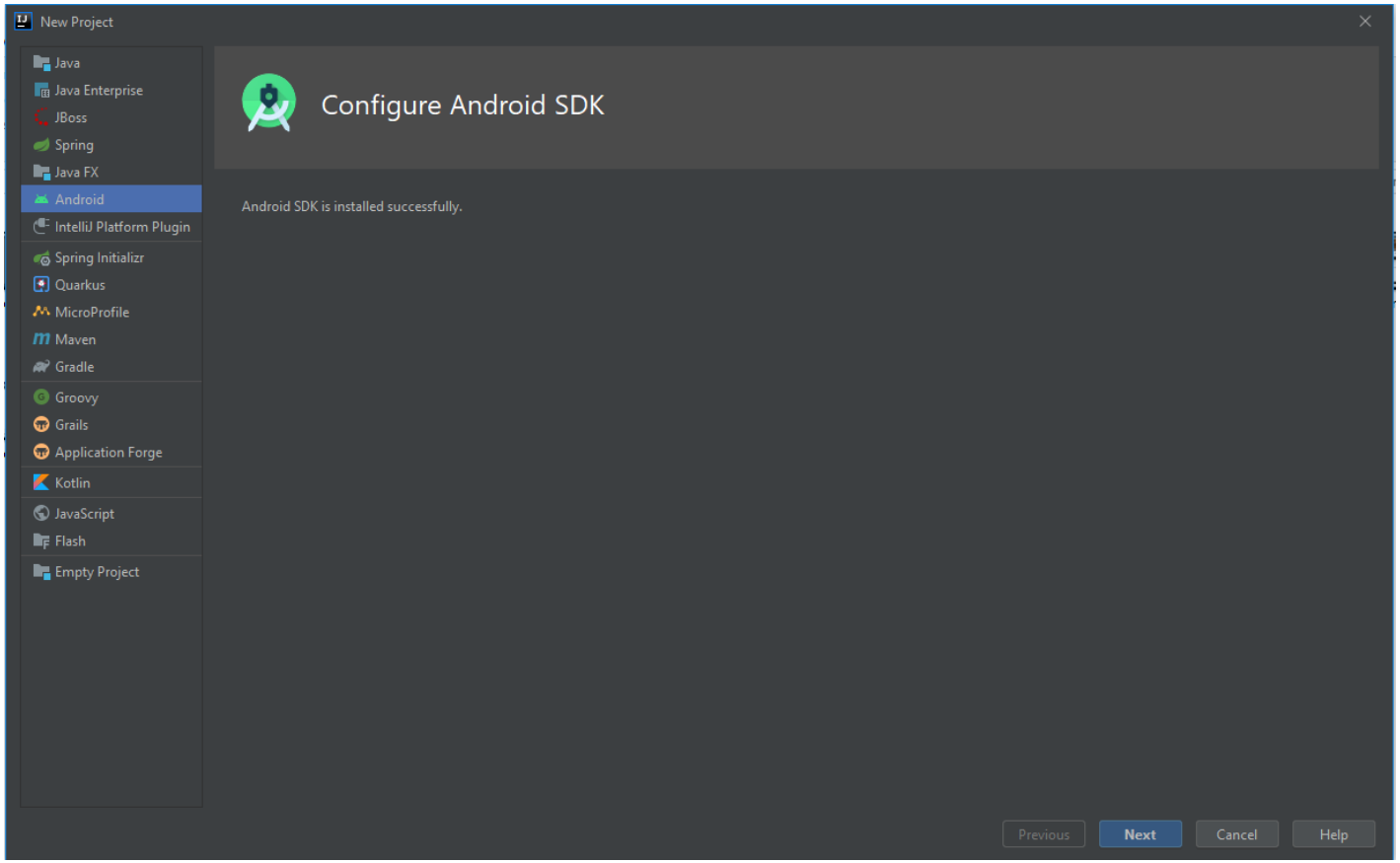


- Click on `Finish`

Install Android SDK (V)

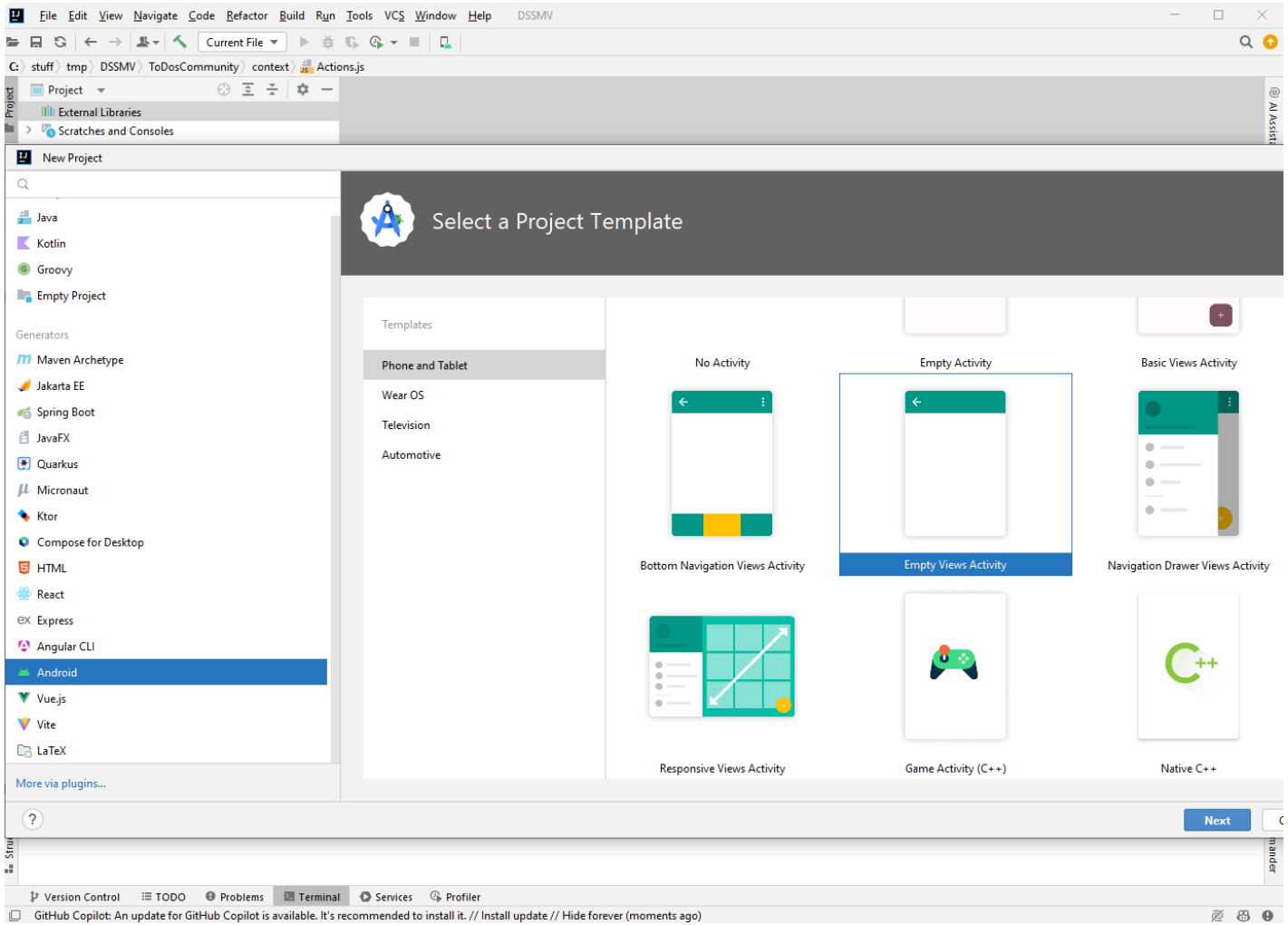


Install Android SDK (VI)



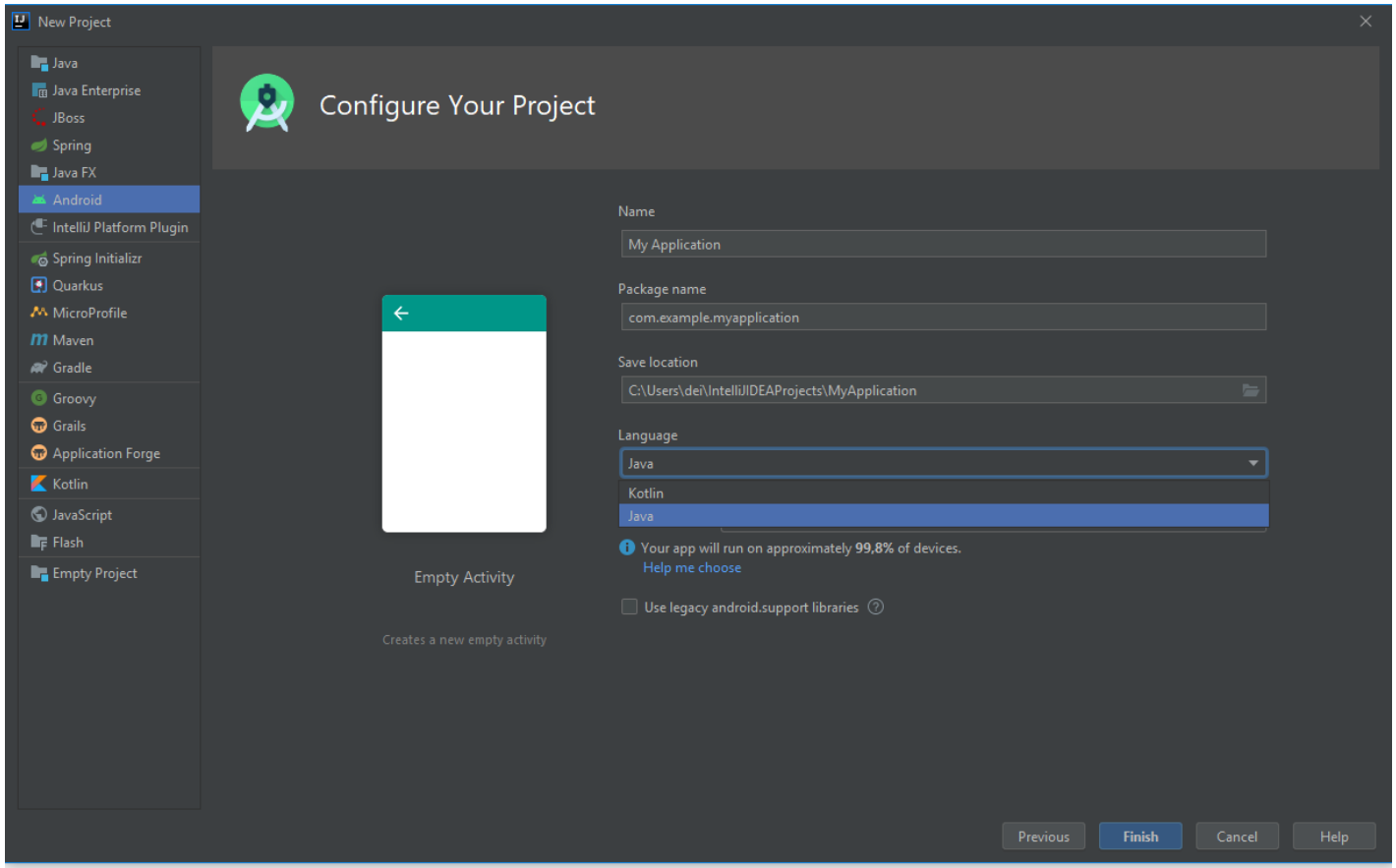
- Click on `Next`

Create a Project (II)



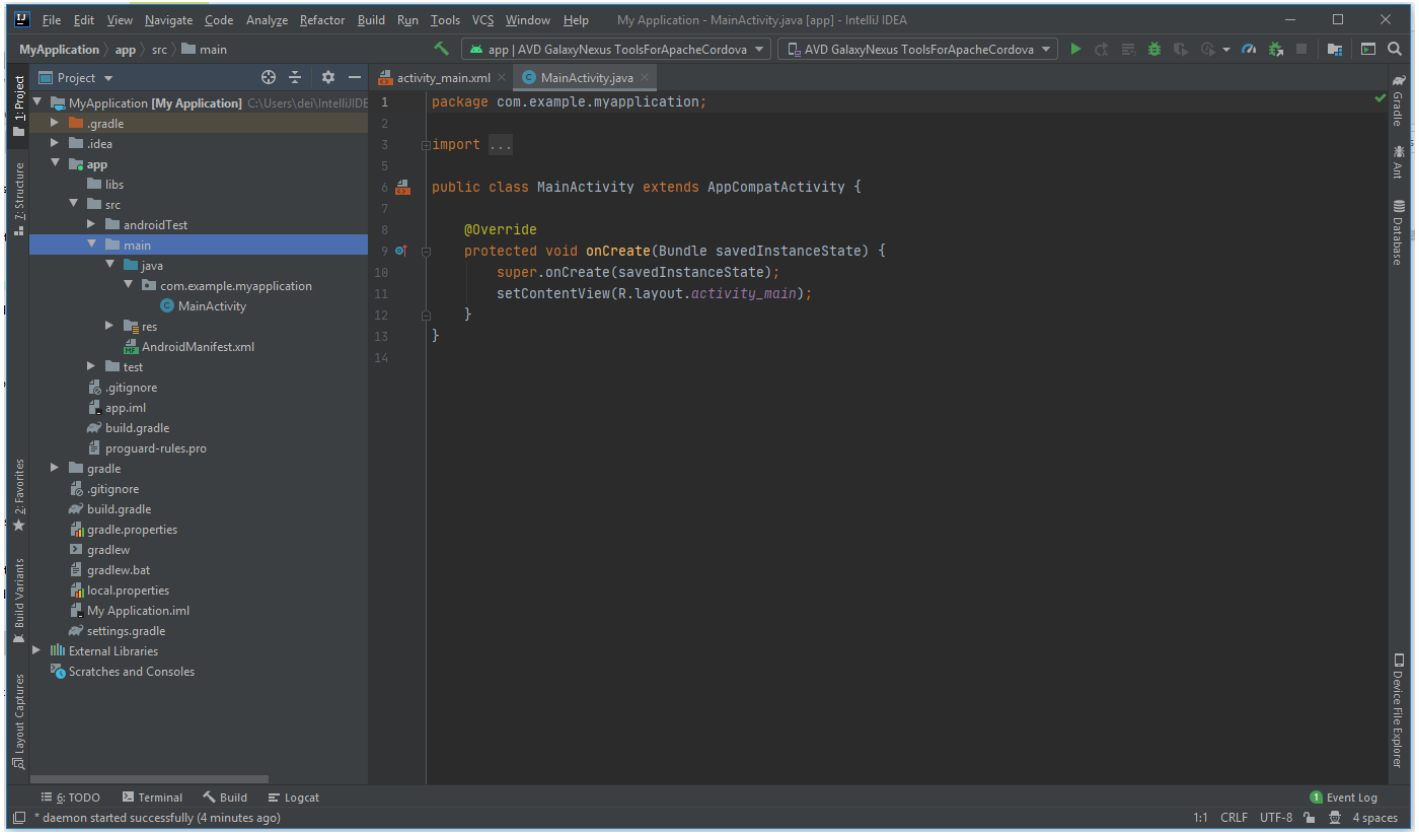
- Select `Empty Views Activity`

Create a Project (III)

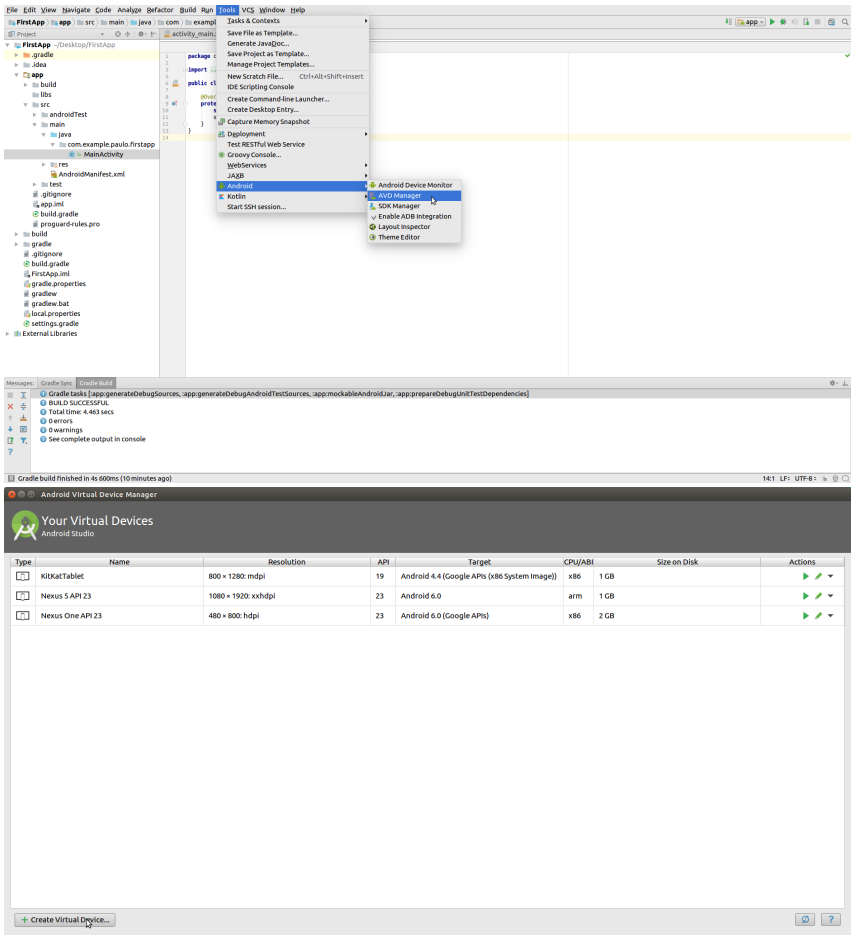


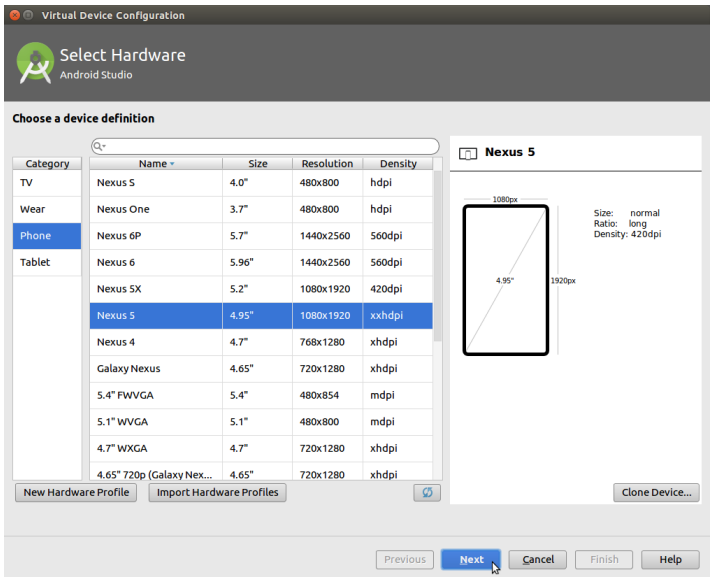
- Select Language `Java`

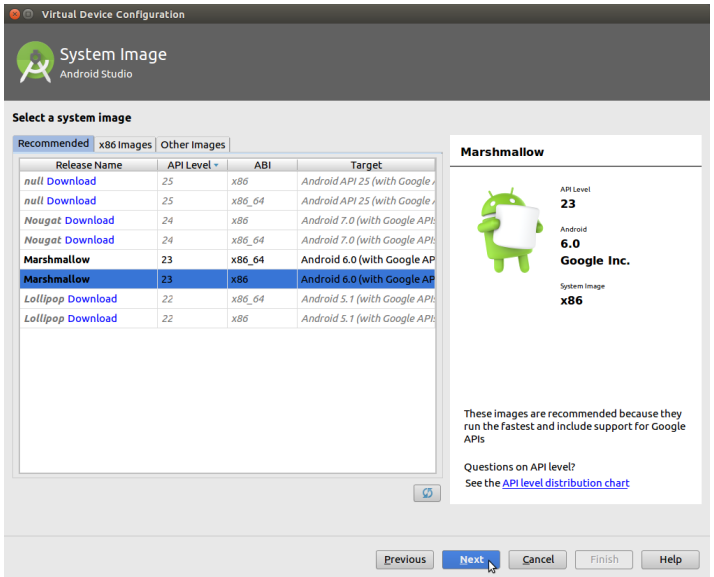
Create a Project (IV)



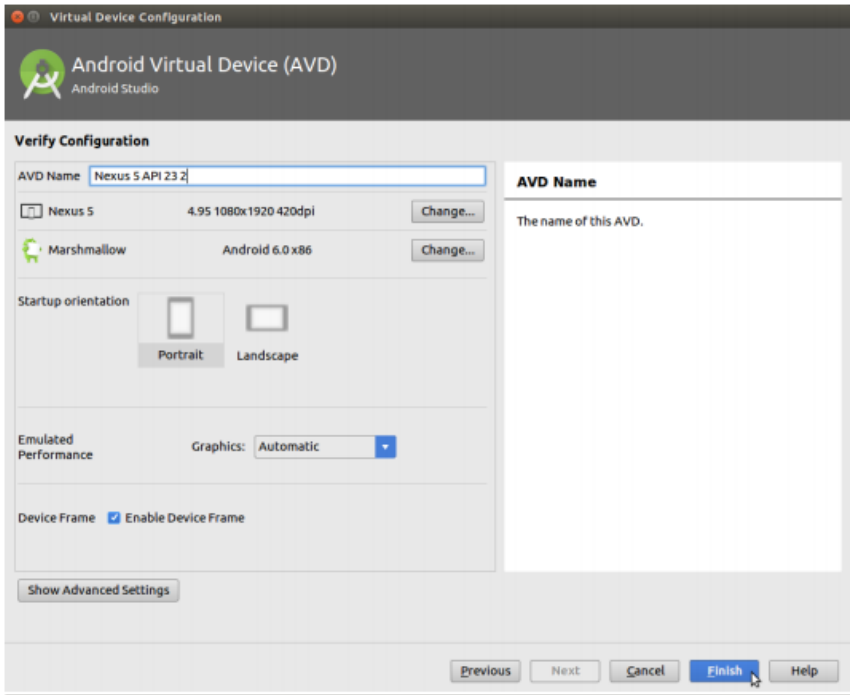
Create an Android Virtual Device (AVD) (I)

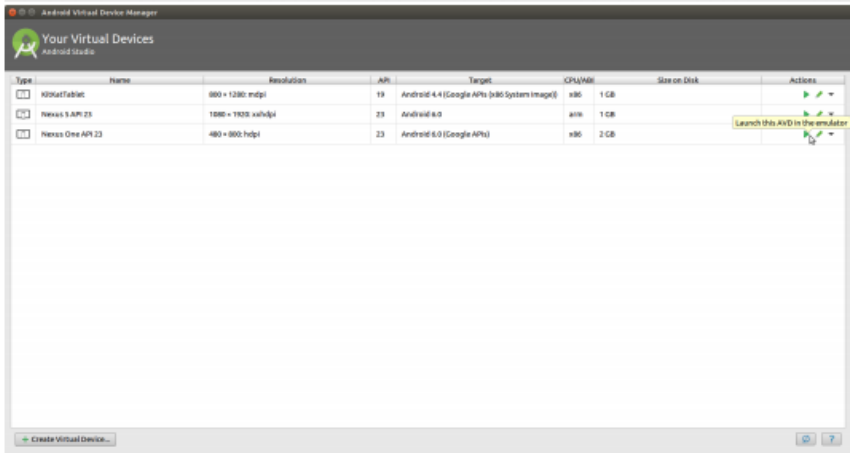


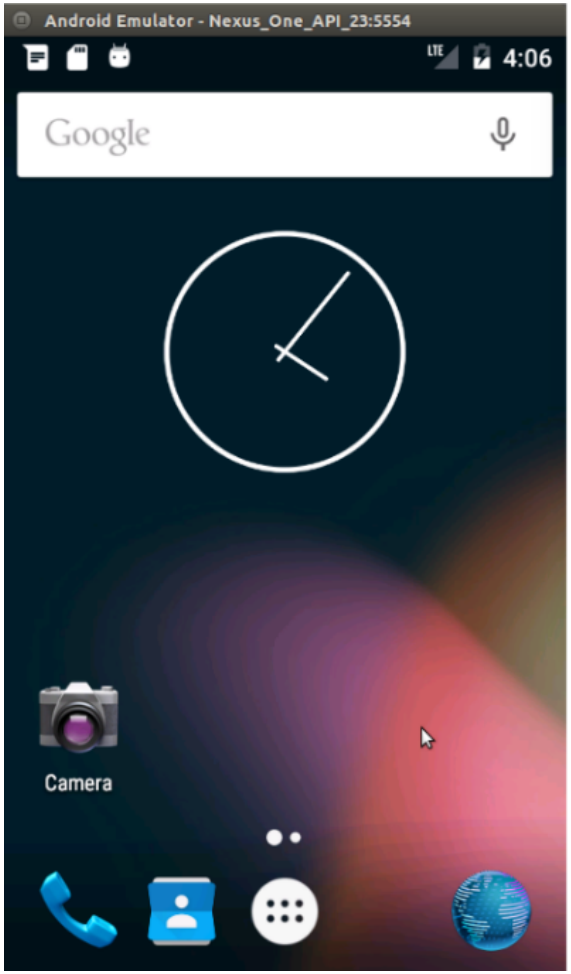




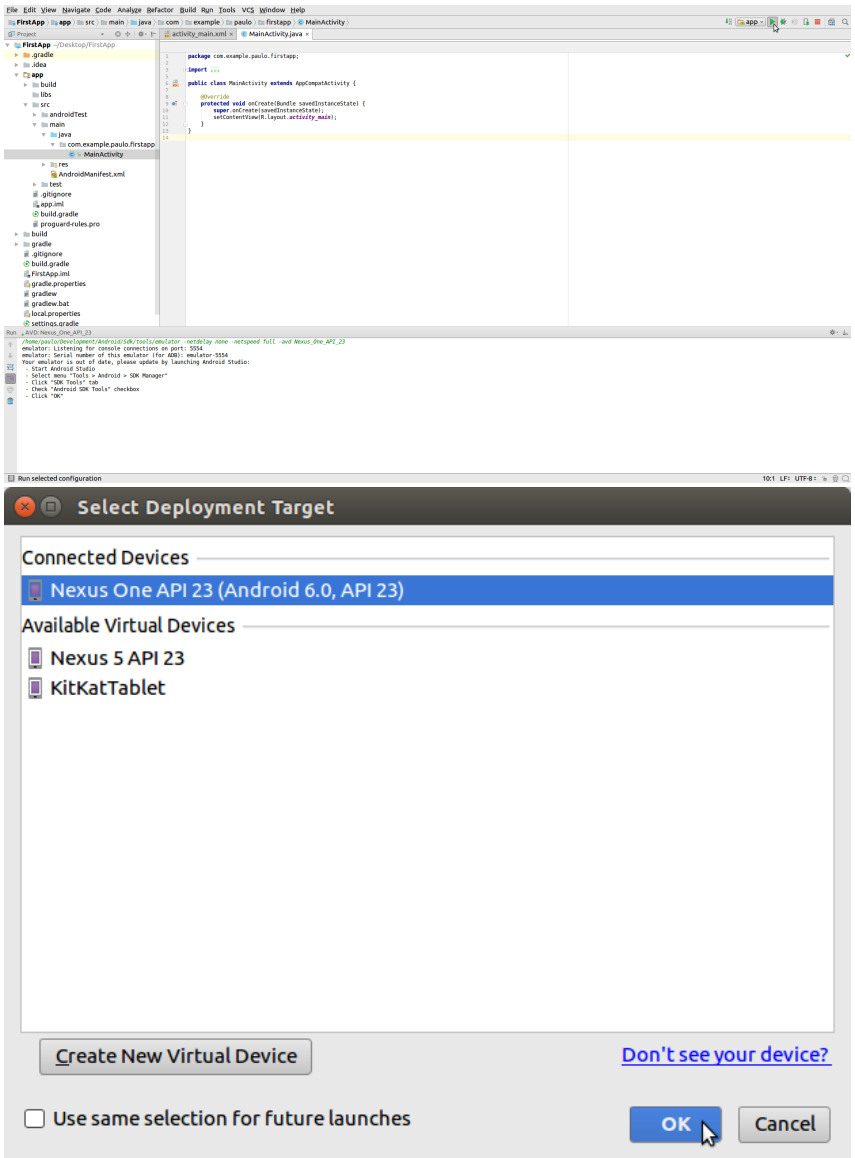
Create an Android Virtual Device (AVD) (II)

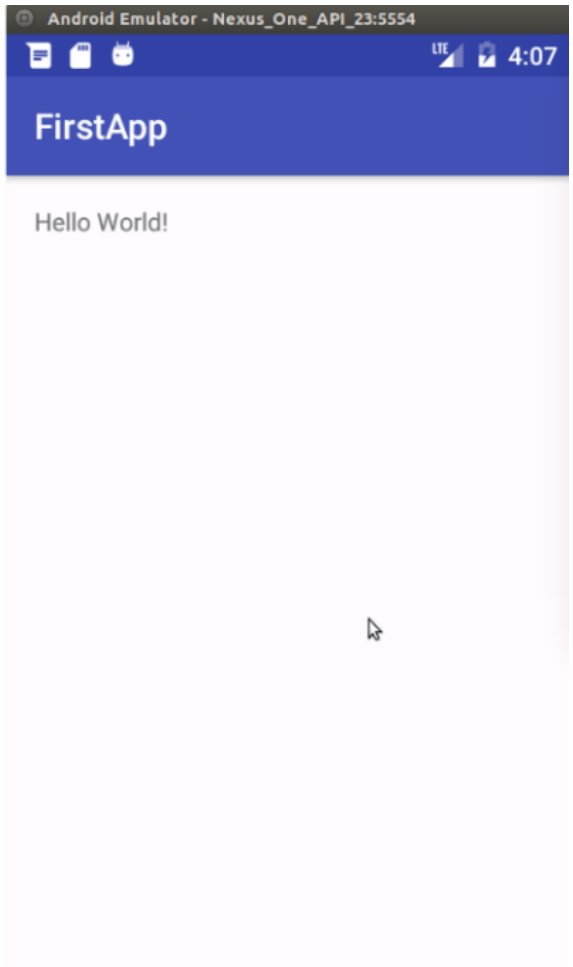




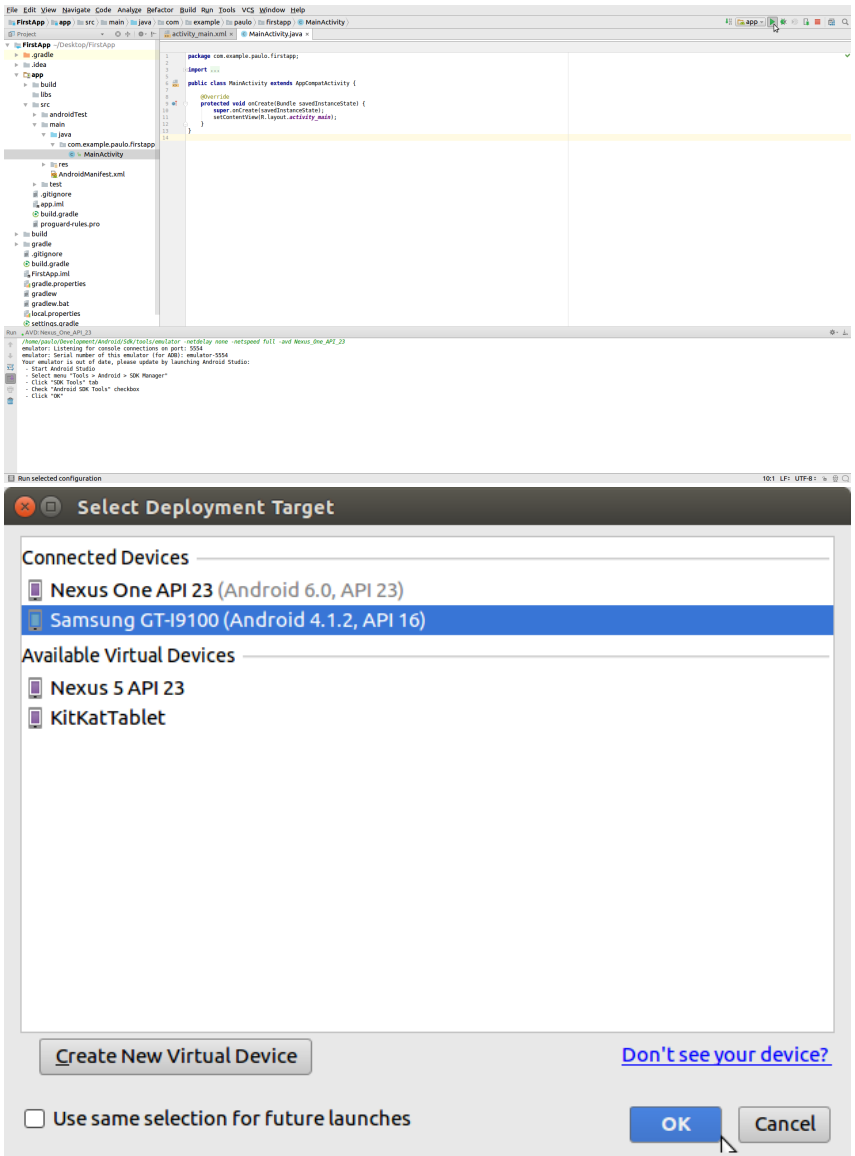


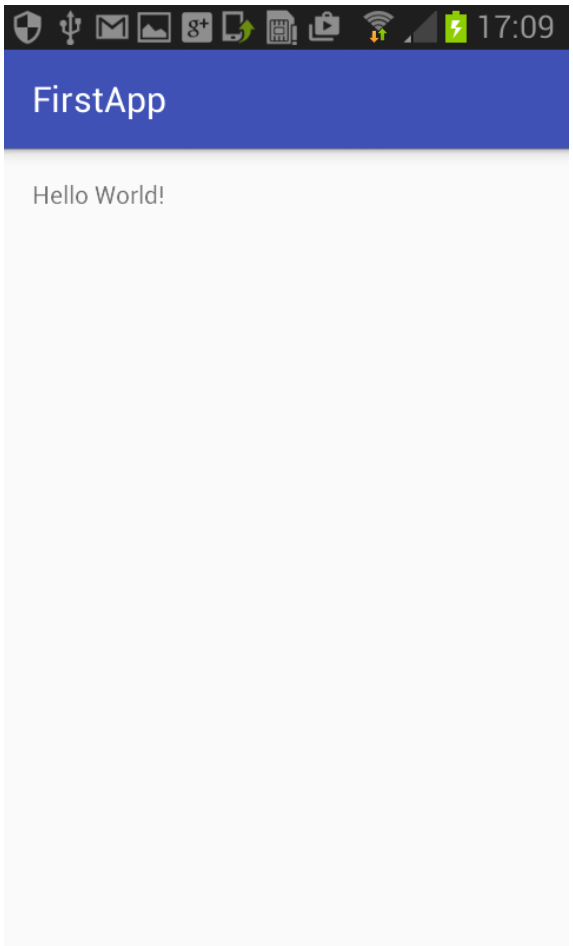
Running the FirstApp on AVD





Running the FirstApp on Real Device





IntelliJ IDEA

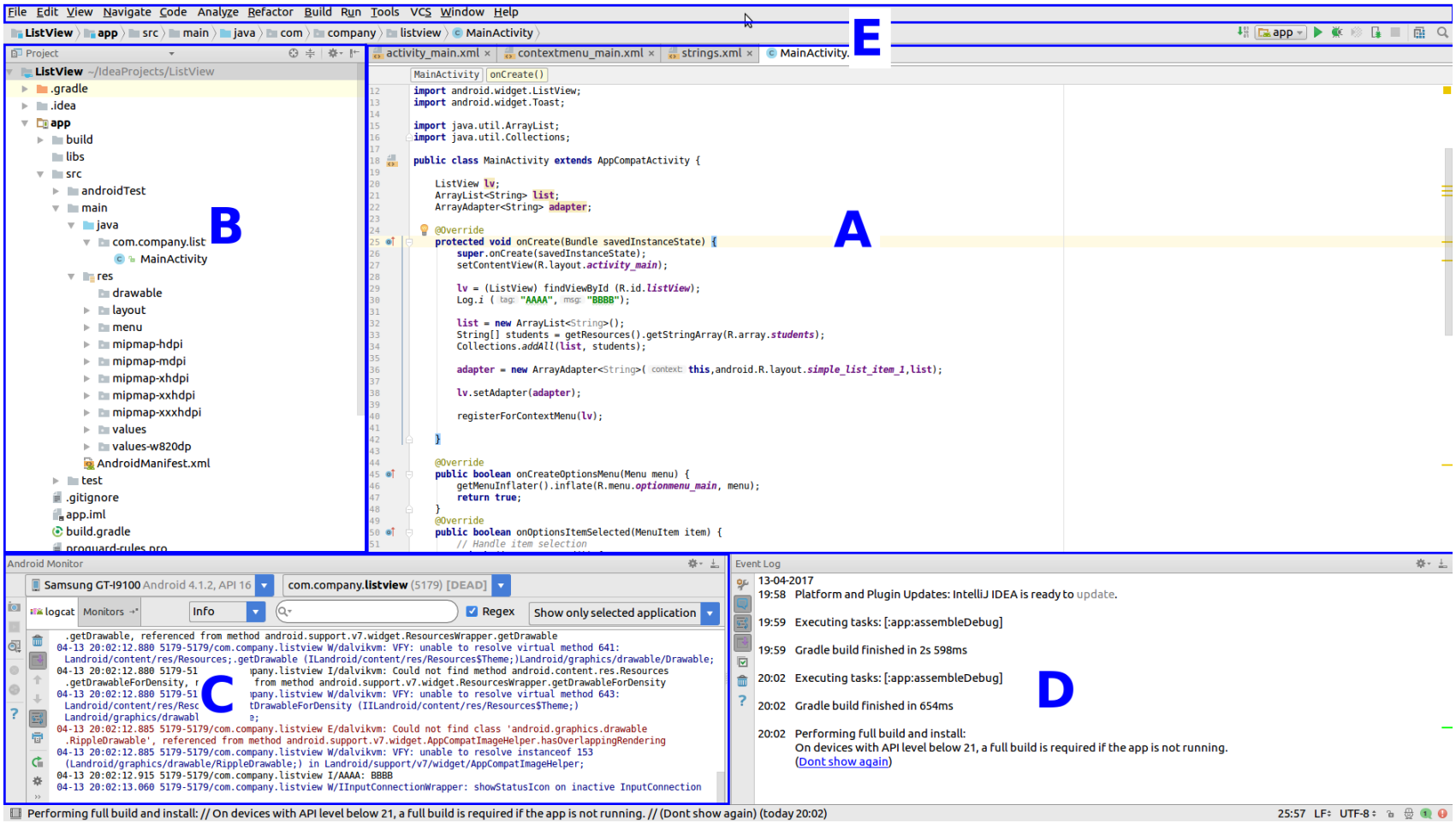
Overview

- Android Software Development Kit (SDK) is required for developing Android apps.
  - It includes everything required for developing Android apps:
    - Android SDK Tools
    - Android Platform-tools
    - Android platform versions
    - Android system images (for the emulator)
- Android Studio [1] is the official IDE.
- IntelliJ IDEA could also be used for developing Android Apps
  - Setting up IntelliJ IDEA for developing Android Apps [2].

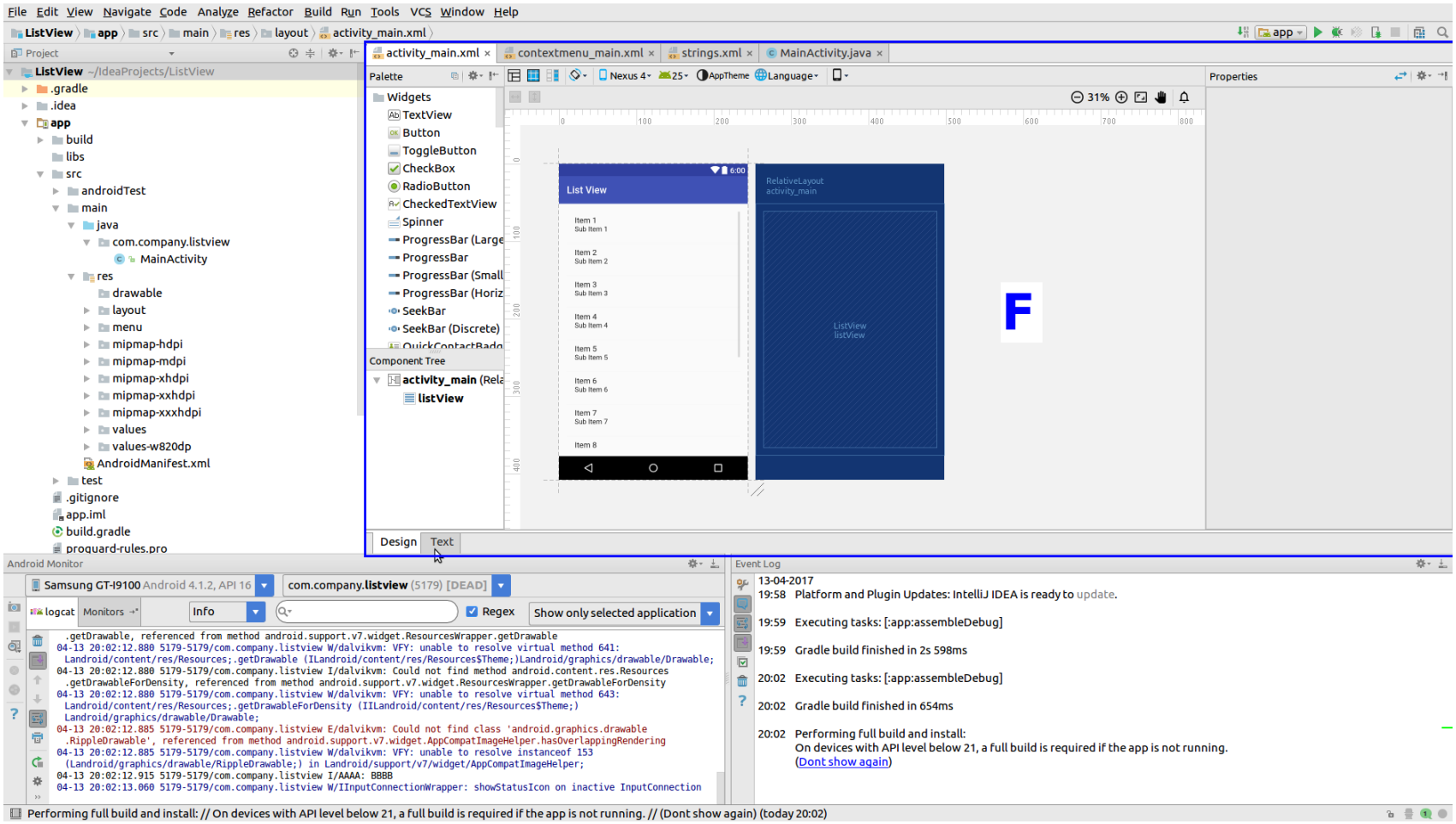
[1] developer.android.com/sdk/index.html

[2] all information can be found in `PLA1-1.pdf` file available at moodle

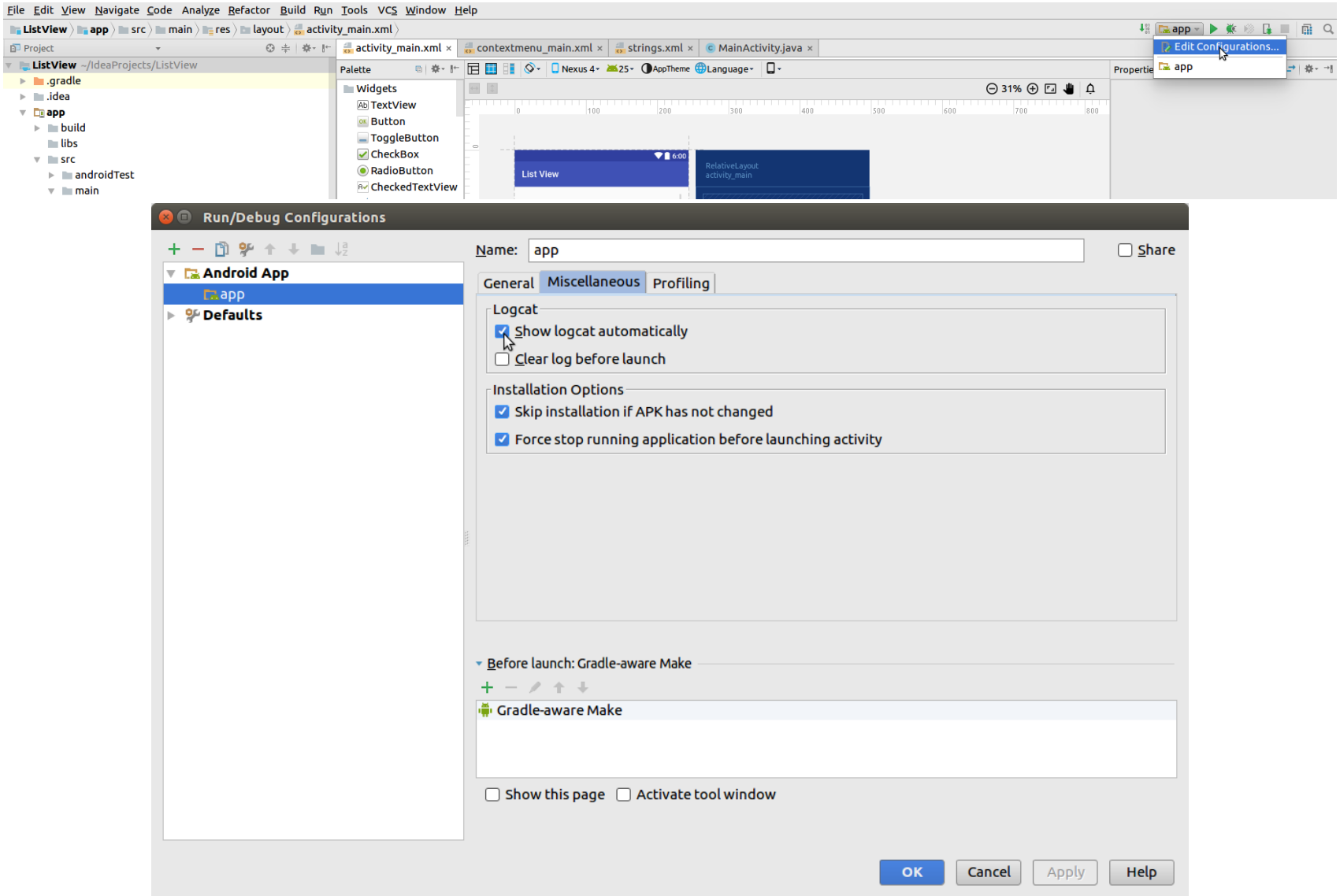
IntelliJ IDEA: General view (I)



IntelliJ IDEA: General view (II)



IntelliJ IDEA: Logcat



Basics

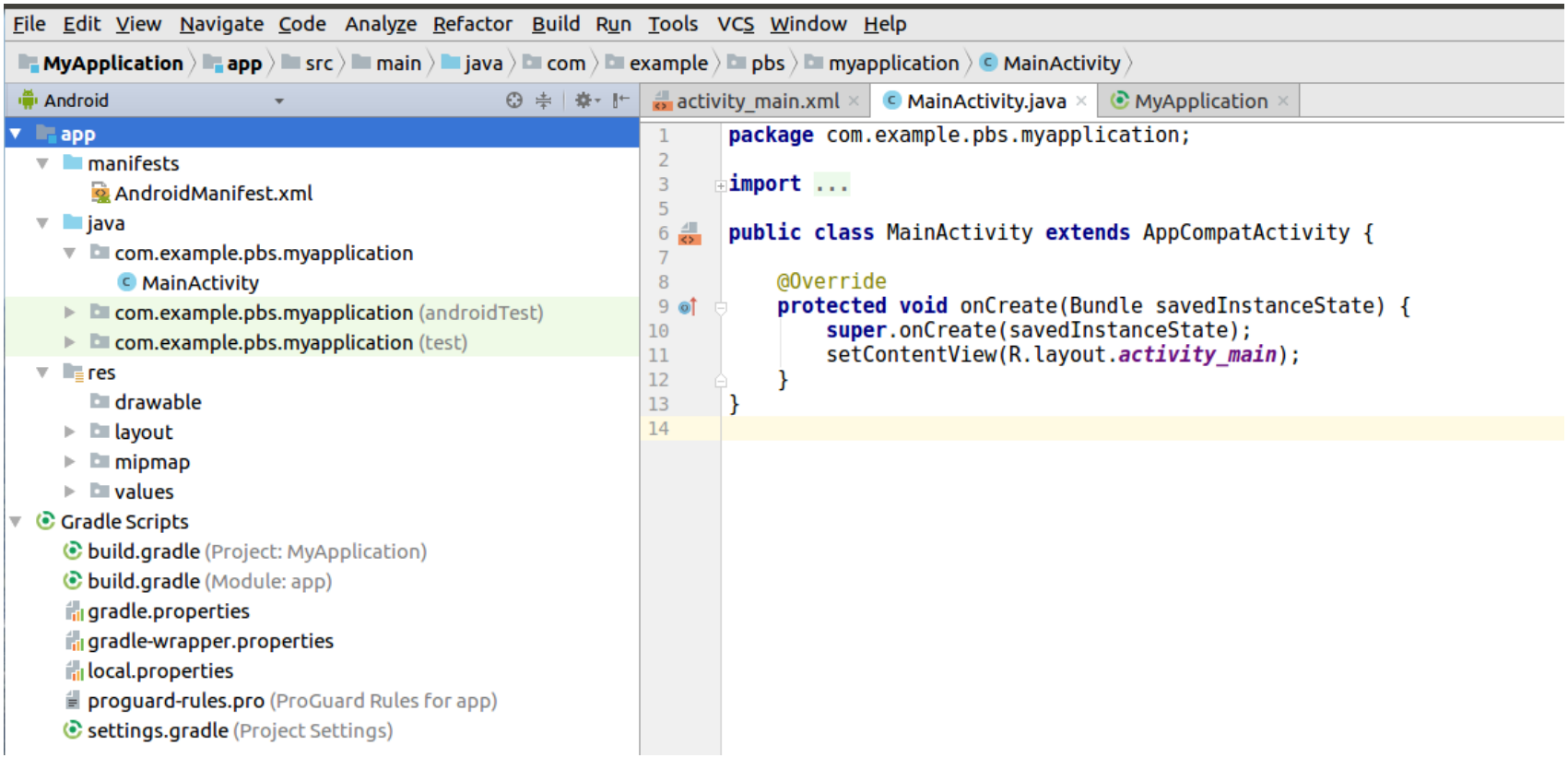
Android Application

- An Android application (**app**) is a single installable unit which can be started and used independently.
- An Android app consists of configuration files, Java source and resource files
- Android Package (APK) is the package file format used by the Android operating system for distribution and installation of mobile apps.

Android view (I)

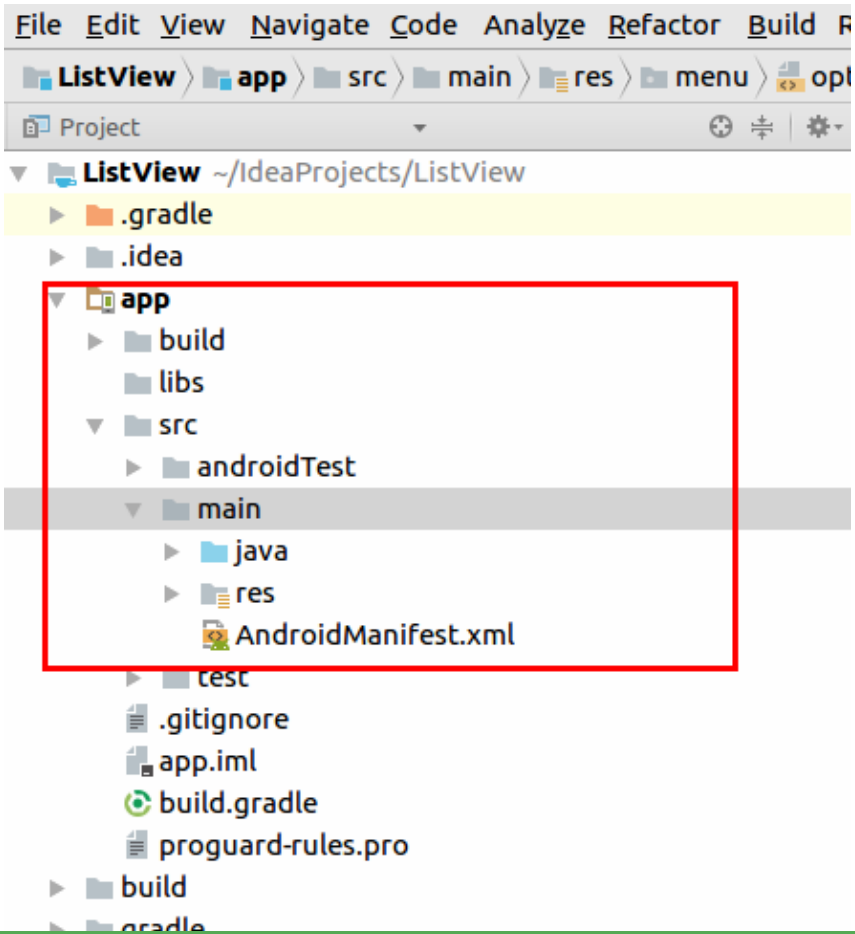
- This view shows a flattened version of your project's structure that provides quick access to the key source files of Android projects and helps you work with the Gradle-based build system.
- Within each Android app module, files are shown in the following groups:
  - `AndroidManifest.xml` file.
  - `java`
    - Contains the Java source code files, separated by package names.
  - `res`
    - Contains all non-code resources, such as XML layouts, UI strings, and bitmap images, divided into corresponding sub-directories. For more information about all possible resource types, see Providing Resources.
  - `GradleScripts`
    - Gradle build and property files.

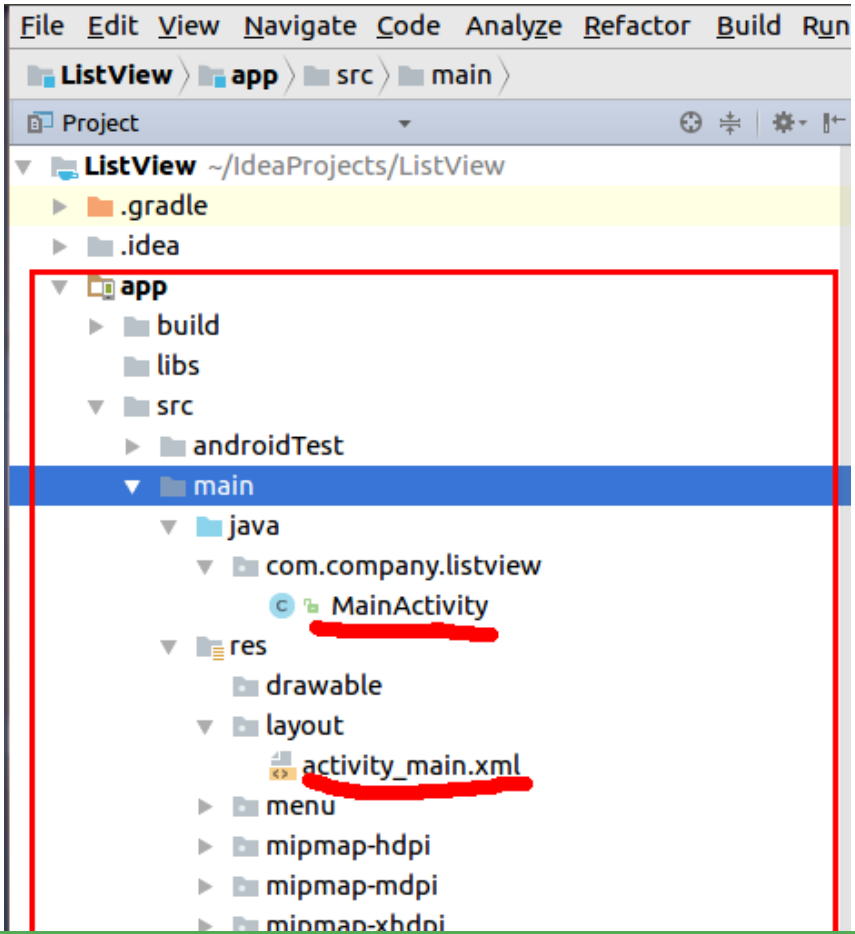
Android view (II)



Project view

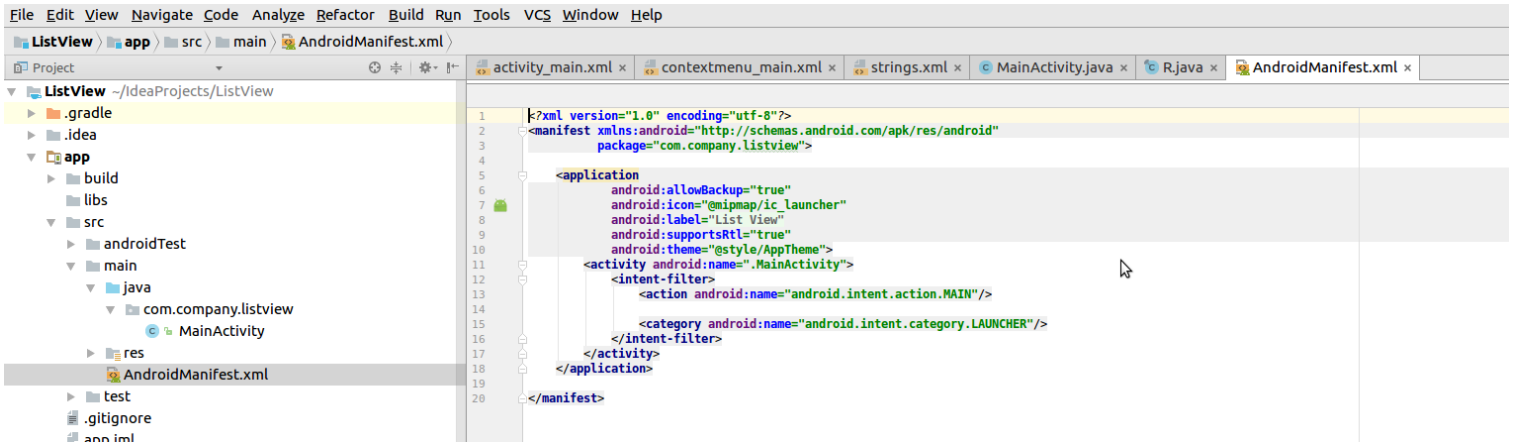
- In this view, all the project items along with their dependencies (SDKs and libraries) are shown. The emphasis is on the directory structure (though the packages are also shown).





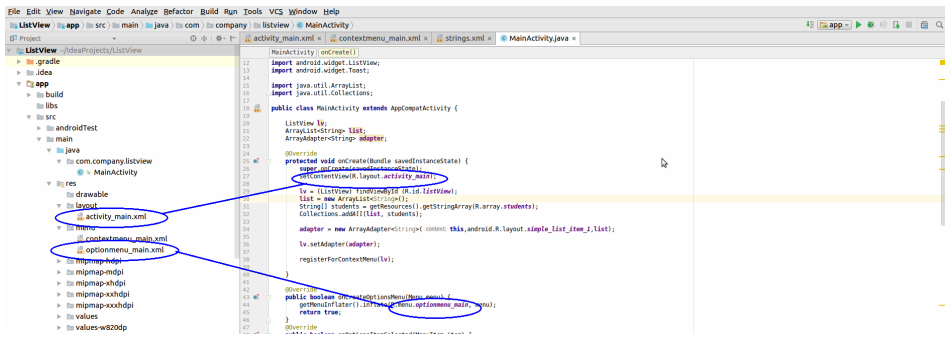
Android Manifest XML

- Every app **must have** a `AndroidManifest.xml` file in its root directory.
- The manifest file **presents essential information about app** to the Android system:
  - Java package name for the application: the package name serves as a unique identifier for the application.
  - The components of the application: the activities, services, broadcast receivers, and content providers.
  - Permissions the application.



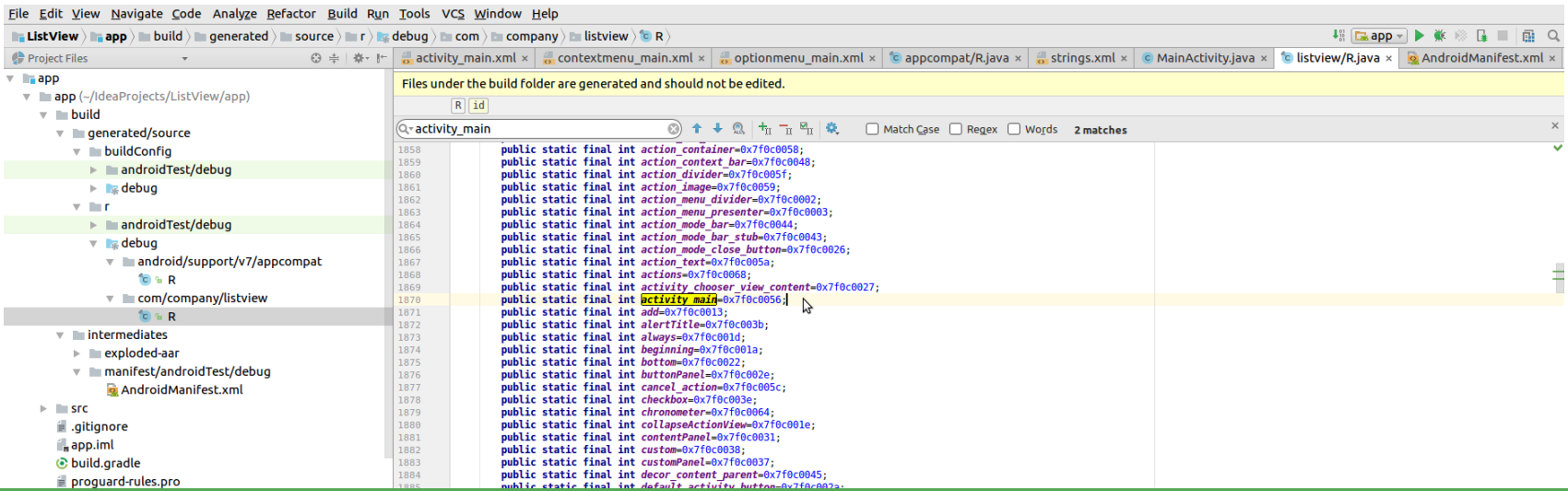
Model-View-Controller (MVC)

- Programming rules establishes a separation between User Interface (UI) and java code:
  - Models:
    - Shared Preferences
    - SQLite Databases
    - Network Connection
    - Internal and External Storage
    - Syncing to the Cloud
  - Views
    - XML files
  - Controllers
    - Java files



Class `R`

- **All resource Identifiers** (IDs) are defined in your project's `R` class, which the `aapt` tool automatically generates.
- `R` class contains resource IDs for all the resources in your `res/` directory.
  - For each type of resource, there is an `R` subclass (for example, `R.layout` for all layout resources), and for each resource of that type, there is a static integer (for example, `R.layout.activity_main`).
    - This integer is the resource ID that you can use to retrieve your resource.



Dimension Units

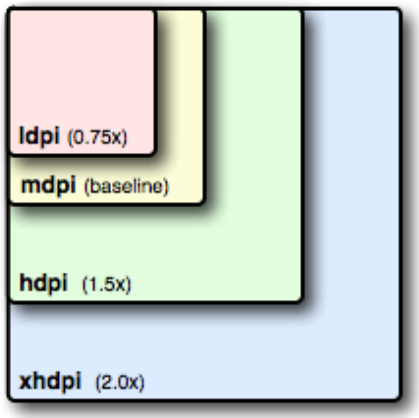
- In order to express a form of distance or length, there are several units for dimensions.
  - These can be used on elements to set widths, heights, margins, padding, and more:
    - `px`: An actual pixel on the screen. This is a density dependent unit, and the physical size of a single "px" varies depending on screen density.
    - `in`: A physical inch on the screen.
    - `mm`: A physical millimeter on the screen.
    - `pt`: A point, a common font size unit, on the screen.
    - `dp`: This is a density independent unit, however the physical size of a single "dp" is only approximately the same on every screen density. There are approximately 160 "dp" in an inch.
    - `sp`: A scale independent pixel, specially designated for text sizes.

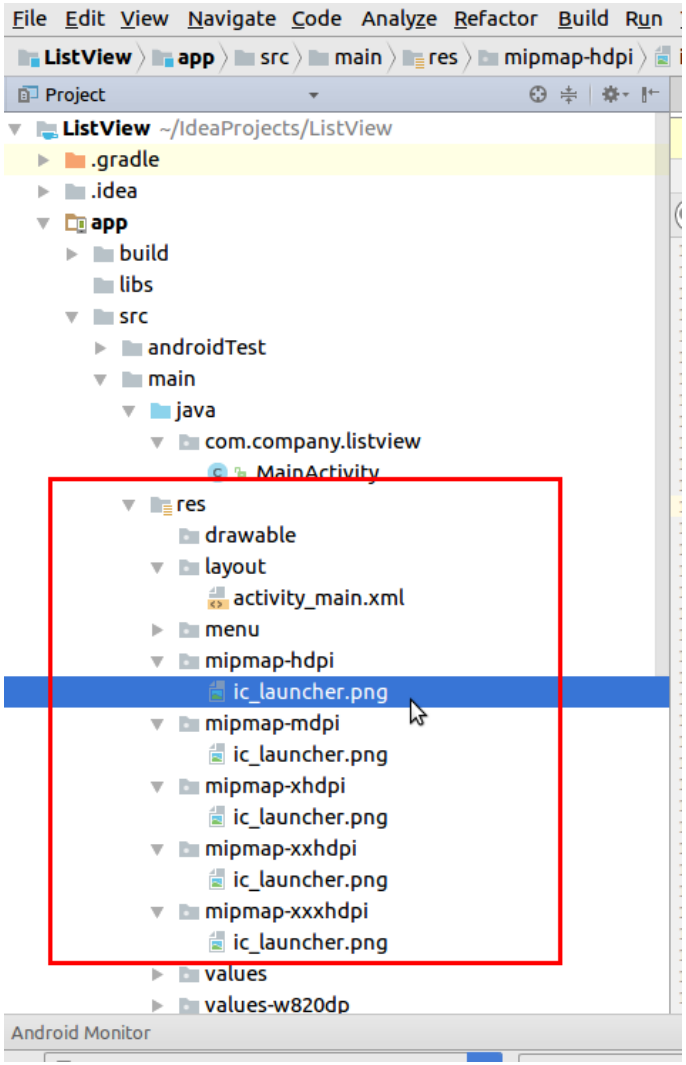
Preferrable

It must be used the **Independent** units

Images

- Since Android runs in devices with a wide variety of screen densities, you should always provide your bitmap resources tailored to each of the generalized density buckets: low, medium, high and extra-high density.
  - This will help you achieve good graphical quality and performance on all screen densities.





`Context`

- Instances of the class `android.content.Context` provide the **connection to the Android system and actual device**.
- It gives access to the system and application resources and services.
  - For example, you can check the size of the current device display via it.
- Activities and services extend the `Context` class. Therefore, they can be directly used to access the `Context`.
  - `getApplicationContext()`
  - `getContext()`
  - `getBaseContext()`
  - `this` (when in the activity class)

Populate an `AdapterView` (III)

- Whenever the data source is changed, the `notifyDataSetChanged()` must be invoked.
  - Notifies the adapter that the underlying data source has been changed and any View reflecting the data set should refresh itself.

```
dataSource.add(str);
adapter.notifyDataSetChanged();
```

Populate an `AdapterView` (III)

- Whenever the data source is changed, the `notifyDataSetChanged()` must be invoked.
  - Notifies the adapter that the underlying data source has been changed and any View reflecting the data set should refresh itself.

```
dataSource.add(str);
adapter.notifyDataSetChanged();
```

Populate an `AdapterView` (III)

- Whenever the data source is changed, the `notifyDataSetChanged()` must be invoked.
  - Notifies the adapter that the underlying data source has been changed and any View reflecting the data set should refresh itself.

```
dataSource.add(str);
adapter.notifyDataSetChanged();
```

Populate an `AdapterView` (III)

- Whenever the data source is changed, the `notifyDataSetChanged()` must be invoked.
  - Notifies the adapter that the underlying data source has been changed and any View reflecting the data set should refresh itself.

```
dataSource.add(str);
adapter.notifyDataSetChanged();
```

Populate an `AdapterView` (III)

- Whenever the data source is changed, the `notifyDataSetChanged()` must be invoked.
  - Notifies the adapter that the underlying data source has been changed and any View reflecting the data set should refresh itself.

```
dataSource.add(str);
adapter.notifyDataSetChanged();
```

Graphical Interface

Populate an `AdapterView` (III)

- Whenever the data source is changed, the `notifyDataSetChanged()` must be invoked.
  - Notifies the adapter that the underlying data source has been changed and any View reflecting the data set should refresh itself.

```
dataSource.add(str);
adapter.notifyDataSetChanged();
```

Populate an `AdapterView` (III)

- Whenever the data source is changed, the `notifyDataSetChanged()` must be invoked.
  - Notifies the adapter that the underlying data source has been changed and any View reflecting the data set should refresh itself.

```
dataSource.add(str);
adapter.notifyDataSetChanged();
```

Populate an `AdapterView` (III)

- Whenever the data source is changed, the `notifyDataSetChanged()` must be invoked.
  - Notifies the adapter that the underlying data source has been changed and any View reflecting the data set should refresh itself.

```
dataSource.add(str);
adapter.notifyDataSetChanged();
```

Populate an `AdapterView` (III)

- Whenever the data source is changed, the `notifyDataSetChanged()` must be invoked.
  - Notifies the adapter that the underlying data source has been changed and any View reflecting the data set should refresh itself.

```
dataSource.add(str);
adapter.notifyDataSetChanged();
```

Populate an `AdapterView` (III)

- Whenever the data source is changed, the `notifyDataSetChanged()` must be invoked.
  - Notifies the adapter that the underlying data source has been changed and any View reflecting the data set should refresh itself.

```
dataSource.add(str);
adapter.notifyDataSetChanged();
```

Populate an `AdapterView` (III)

- Whenever the data source is changed, the `notifyDataSetChanged()` must be invoked.
  - Notifies the adapter that the underlying data source has been changed and any View reflecting the data set should refresh itself.

```
dataSource.add(str);
adapter.notifyDataSetChanged();
```

Populate an `AdapterView` (III)

- Whenever the data source is changed, the `notifyDataSetChanged()` must be invoked.
  - Notifies the adapter that the underlying data source has been changed and any View reflecting the data set should refresh itself.

```
dataSource.add(str);
adapter.notifyDataSetChanged();
```

Populate an `AdapterView` (III)

- Whenever the data source is changed, the `notifyDataSetChanged()` must be invoked.
  - Notifies the adapter that the underlying data source has been changed and any View reflecting the data set should refresh itself.

```
dataSource.add(str);
adapter.notifyDataSetChanged();
```

Populate an `AdapterView` (III)

- Whenever the data source is changed, the `notifyDataSetChanged()` must be invoked.
  - Notifies the adapter that the underlying data source has been changed and any View reflecting the data set should refresh itself.

```
dataSource.add(str);
adapter.notifyDataSetChanged();
```

Populate an `AdapterView` (III)

- Whenever the data source is changed, the `notifyDataSetChanged()` must be invoked.
  - Notifies the adapter that the underlying data source has been changed and any View reflecting the data set should refresh itself.

```
dataSource.add(str);
adapter.notifyDataSetChanged();
```

Populate an `AdapterView` (III)

- Whenever the data source is changed, the `notifyDataSetChanged()` must be invoked.
  - Notifies the adapter that the underlying data source has been changed and any View reflecting the data set should refresh itself.

```
dataSource.add(str);
adapter.notifyDataSetChanged();
```

AdapterView

Populate an `AdapterView` (III)

- Whenever the data source is changed, the `notifyDataSetChanged()` must be invoked.
  - Notifies the adapter that the underlying data source has been changed and any View reflecting the data set should refresh itself.

```
dataSource.add(str);
adapter.notifyDataSetChanged();
```

Populate an `AdapterView` (III)

- Whenever the data source is changed, the `notifyDataSetChanged()` must be invoked.
  - Notifies the adapter that the underlying data source has been changed and any View reflecting the data set should refresh itself.

```
dataSource.add(str);
adapter.notifyDataSetChanged();
```

Populate an `AdapterView` (III)

- Whenever the data source is changed, the `notifyDataSetChanged()` must be invoked.
  - Notifies the adapter that the underlying data source has been changed and any View reflecting the data set should refresh itself.

```
dataSource.add(str);
adapter.notifyDataSetChanged();
```

Bibliography

Resources

- "Mastering Android Application Development", by Antonio Pachon Rui, 2015
- `https://developer.android.com/index.html`
- `http://simple.sourceforge.net/home.php` `http://simple.sourceforge.net/download/stream/doc/tutorial/tutorial.php`