MOBILE APPLICATION TECHNOLOGIES

LESSON TWO:FUNDAMENTALS OF ANDROID PROGRAMMING

Prepared by Anastase MUVANDIMWE

CONTENTS

- Android Project Structure
- Project files
- App components
- Activity life cycle
- First android project
- Android Virtual Device (AVD)

- Android Project Structure
- Project files
- App components
- Activity life cycle
- First android project
- Android Virtual Device (AVD)

ANDROID PROJECT STRUCTURE

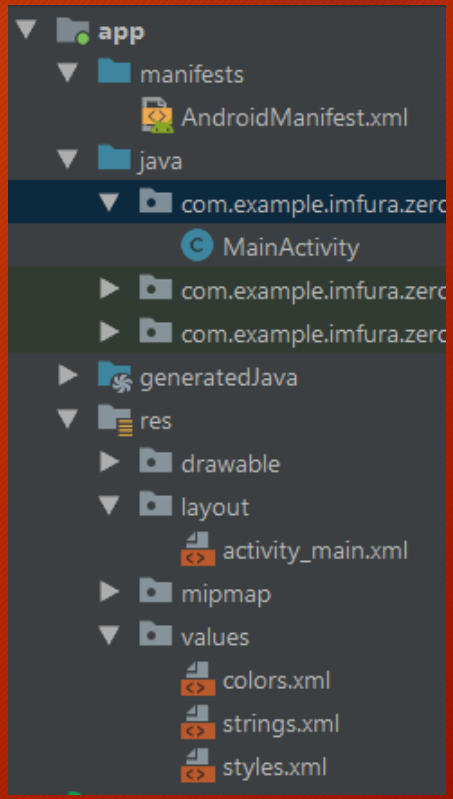
- A *module* is a collection of source files and build settings that allow you to divide your project into discrete units of functionality.
- Your project can have one or many modules and one module may use another module as a dependency.
- Each module can be independently built, tested, and debugged.
- You can add a new module to your project by clicking **File > New > New Module**.

Type of modules

- Android Studio offers a few distinct types of module:
  1. Android app module
  2. Library module
  3. Google cloud module

Android app module

- **Android app module**
  - Provides a container for your app's source code, resource files, and app level settings such as the module-level build file and Android Manifest file.
  - When you create a new project, the default module name is "app".



App module

- In the **Create New Module** window, Android Studio offers the following app modules:
  - Phone & Tablet Module
  - Wear OS Module
  - Android TV Module
  - Glass Module
- They each provide essential files and some code templates that are appropriate for the corresponding app or device type.

Library module

- Library module provides a container for your reusable code, which you can use as a dependency in other app modules or import into other projects.
- Structurally, a library module is the same as an app module, but when built, it creates a code archive file instead of an APK, so it can't be installed on a device.

Library module

- In the **Create New Module** window, Android Studio offers the following library modules:
  - **Android Library**: This type of library can contain all file types supported in an Android project, including source code, resources, and manifest files. The build result is an Android Archive (AAR) file that you can add as a dependency for your Android app modules.
  - **Java Library:** This type of library can contain only Java source files. The build result is an Java Archive (JAR) file that you can add as a dependency for your Android app modules or other Java projects.

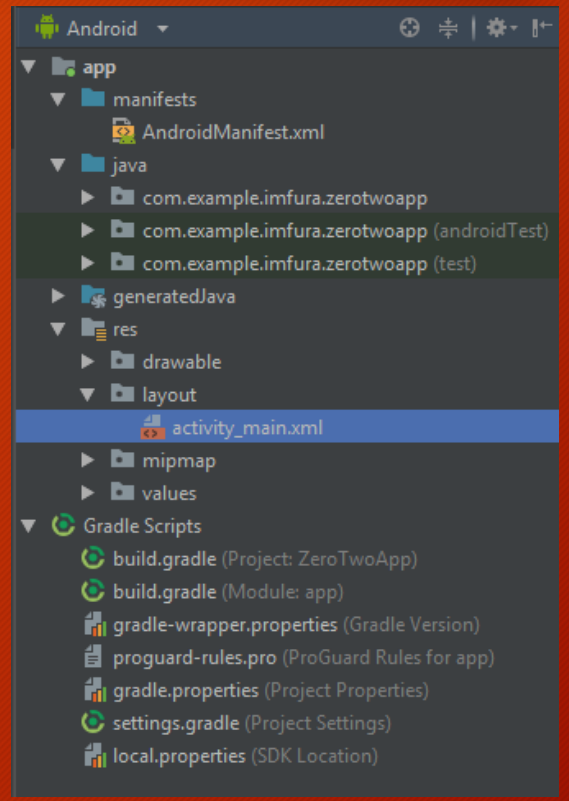
Google Cloud Module

- **Google Cloud module** provides a container for your Google Cloud backend code.
- This module has the required code and dependencies for a Java App Engine backend that uses simple HTTP, Cloud Endpoints, and Cloud Messaging to connect to your app.
- You can develop your backend to provide cloud services your app needs.
- Using Android Studio to develop your Google Cloud module lets you manage app code and backend code in the same project.

- Android Project Structure
- Project files
- App components
- Activity life cycle
- First android project
- Android Virtual Device (AVD)

Project files (Android View)

- By default, Android Studio displays your project files in the **Android** view.
- This view does not reflect the actual file hierarchy on disk, but is organized by modules and file types to simplify navigation between key source files of your project, hiding certain files or directories that are not commonly used.

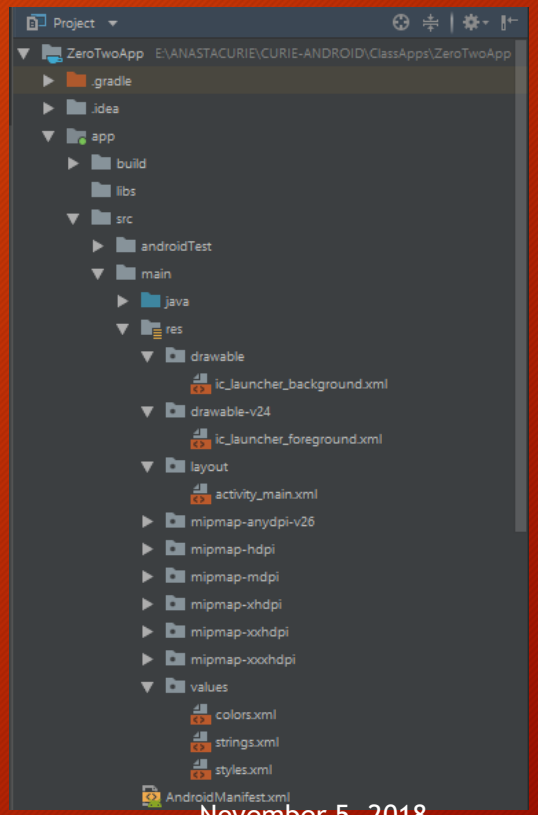


Project files

- Within each Android app module, files are shown in the following groups:
  - **manifests**
    - Contains the AndroidManifest.xml file.
  - **java**
    - Contains the Java source code files, separated by package names, including JUnit test code.
  - **res**
    - Contains all non-code resources, such as
      - XML layouts,
      - UI strings, and
      - bitmap images, divided into corresponding sub-directories.

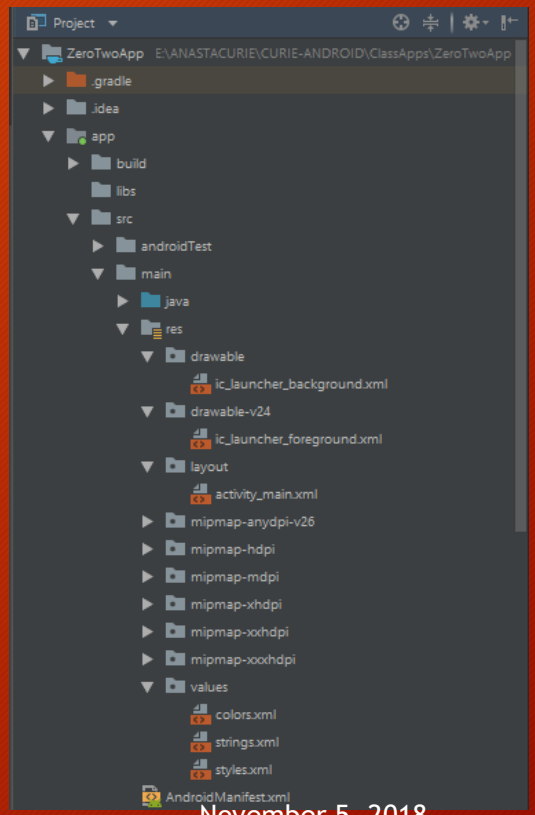
Project view

- To see the actual file structure of the project including all files hidden from the Android view, select **Project** from the dropdown at the top of the **Project** window.
- When you select **Project** view, you can see a lot more files and directories. The most important of which are the following:



Project view

- *module-name/*
  - **build/**
    - Contains build outputs.
  - **libs/**
    - Contains private libraries.
  - **src/**
    - Contains all code and resource files for the module in the following subdirectories:
    - **androidTest/**
      - Contains code for instrumentation tests that run on an Android device.
    - **main/**
      - Contains the "main" sourceset files: the Android code and resources shared by all build variants (files for other build variants reside in sibling directories, such as src/debug/ for the debug build type).



Project view (Cont.)

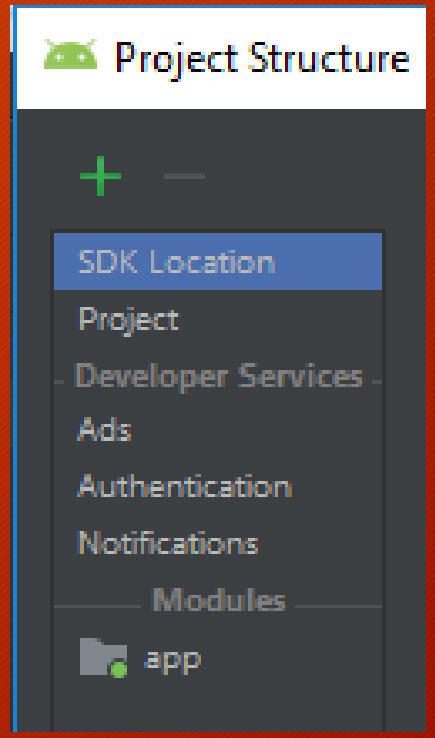
- **AndroidManifest.xml**
  - Describes the nature of the application and each of its components.
- **java/**
  - Contains Java code sources.
- **jni/**
  - Contains native code using the Java Native Interface (JNI).
- **gen/**
  - Contains the Java files generated by Android Studio, such as your R.java file and interfaces created from AIDL files.

Project view (Cont.)

- res/
  - Contains application resources, such as drawable files, layout files, and UI string.
- assets/
  - Contains file that should be compiled into an .apk file as-is. You can navigate this directory in the same way as a typical file system using URIs and read files as a stream of bytes using the **AssetManager** . For example, this is a good location for textures and game data.
- test/
  - Contains code for local tests that run on your host JVM.
- build.gradle **(module)**
  - This defines the module-specific build configurations.

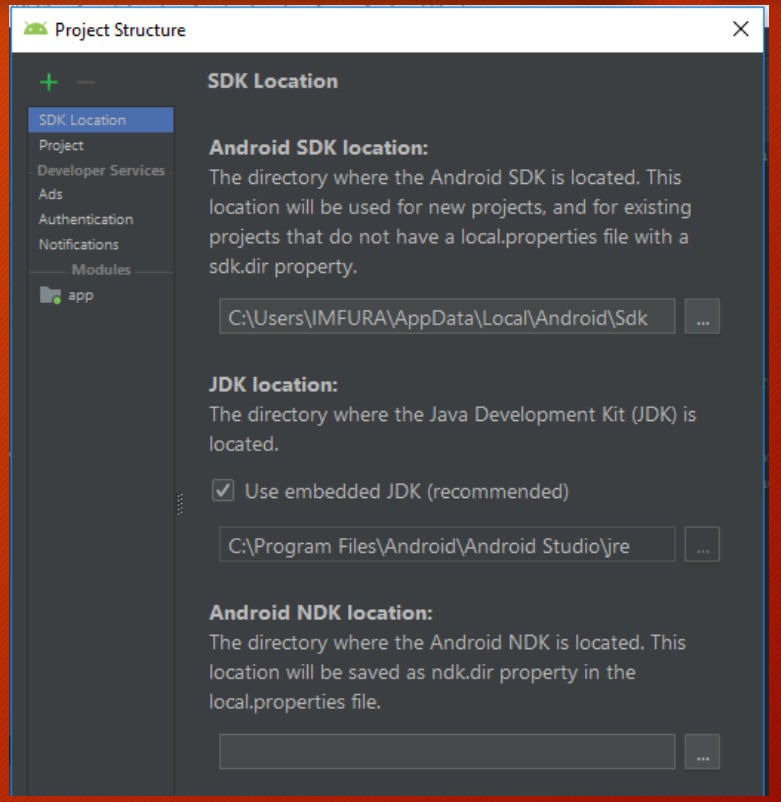
Project Structure Settings

- To change various settings for your Android Studio project, open the **Project Structure** dialog by clicking **File > Project Structure**.
- It contains the following sections:
  - **SDK Location**
  - **Project**
  - **Developer services**
  - **Modules**



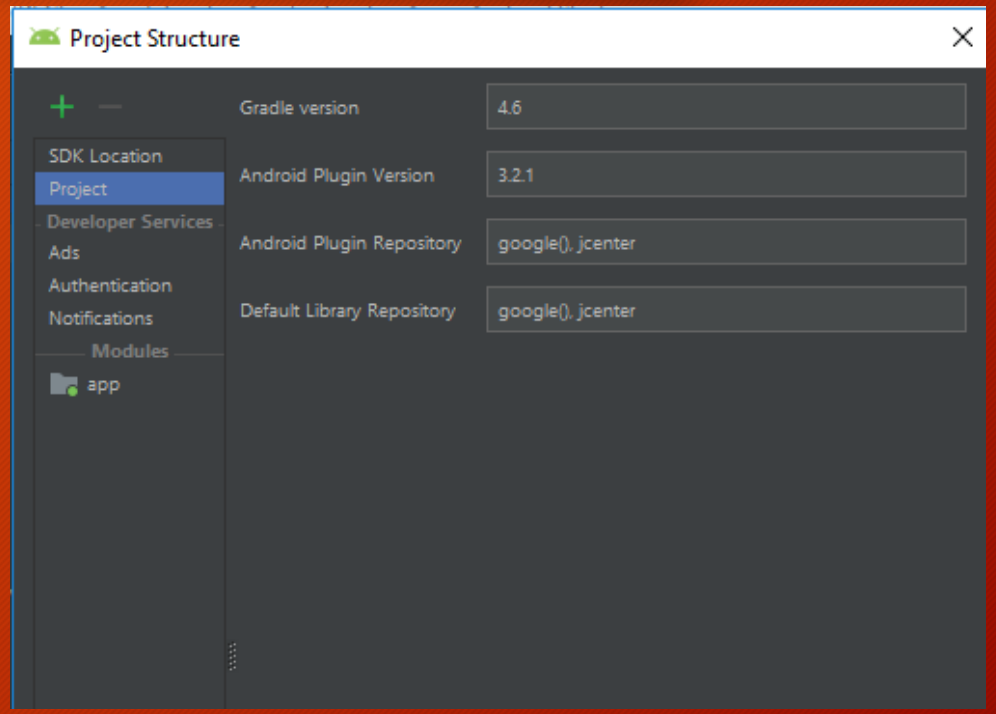
Project Structure Settings

- **SDK Location**
  - **It** sets the location of the JDK, Android SDK, and Android NDK that your project uses.



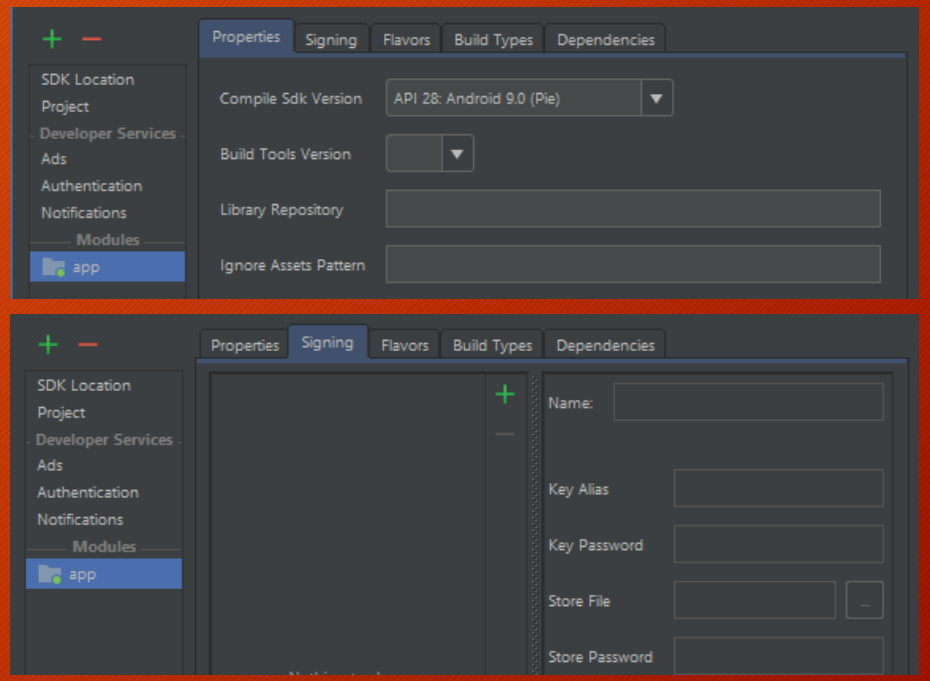
Project Structure Settings

- **Project**
  - It sets the version for **Gradle and the Android plugin for Gradle**, and the repository location name.
- **Developer Services**
  - **It** contains settings for Android Studio add-in components from Google or other third parties.



Project Structure Settings

- **Modules**
  - It allows you to edit module-specific build configurations, including the target and minimum SDK, the app signature, and library dependencies.



- Android Project Structure
- Project files
- App components
- Activity life cycle
- First android project
- Android Virtual Device (AVD)

App components

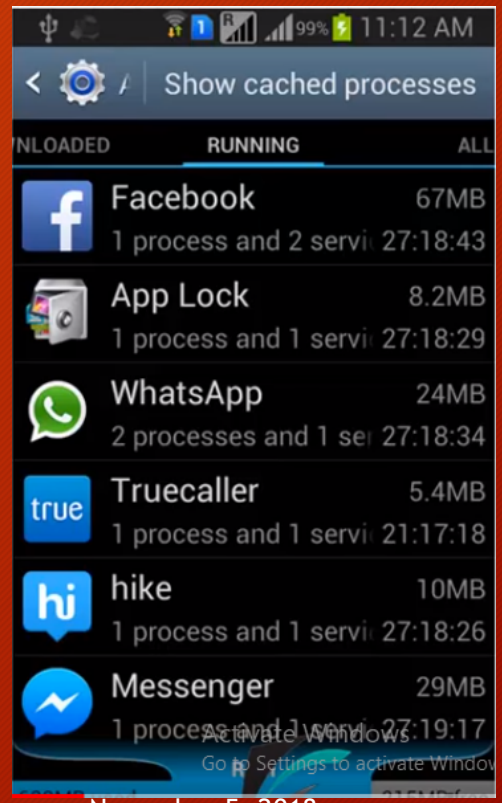
- There are four different types of app components:
  - Activities
  - Services
  - Broadcast receivers
  - Content providers

Activities

- An ***activity*** is the entry point for interacting with the user.
- It represents a single screen with a user interface.
- For example, an email app might have one activity that shows a list of new emails, another activity to compose an email, and another activity for reading emails.
- Although the activities work together to form a cohesive user experience in the email app, each one is independent of the others.

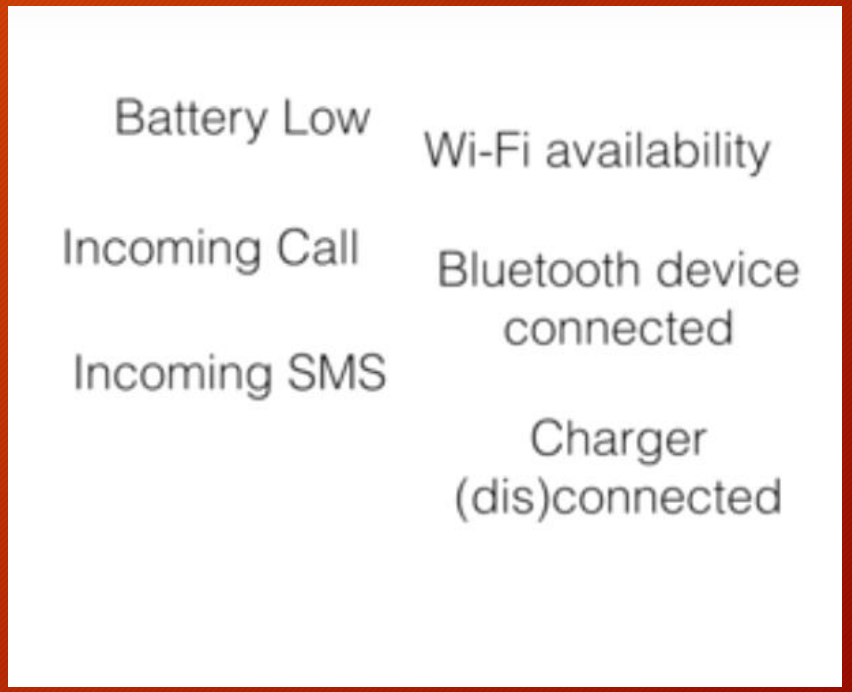
Services

- A ***service*** is a general-purpose entry point for keeping an app running in the background for all kinds of reasons.
- It is a component that runs in the background to perform long-running operations or to perform work for remote processes.
- A service does not provide a user interface.
- For example, a service might play music in the background while the user is in a different app, or it might fetch data over the network without blocking user interaction with an activity.



Broadcast receiver

- A ***broadcast receiver*** is a component that enables the system to deliver events to the app outside of a regular user flow, allowing the app to respond to system-wide broadcast announcements.
- You can launch an Android application to process a specific element of data or respond to an event, such as **receiving a text message**



Content Provider

- A ***content provider*** manages a shared set of app data that you can store in the file system, in a SQLite database, on the web, or on any other persistent storage location that your app can access.
- Through the content provider, other apps can query or modify the data if the content provider allows it.
- For example, the Android system provides a content provider that manages the user's contact information.

- Android Project Structure
- Project files
- App components
- Activity life cycle
- First android project
- Android Virtual Device (AVD)

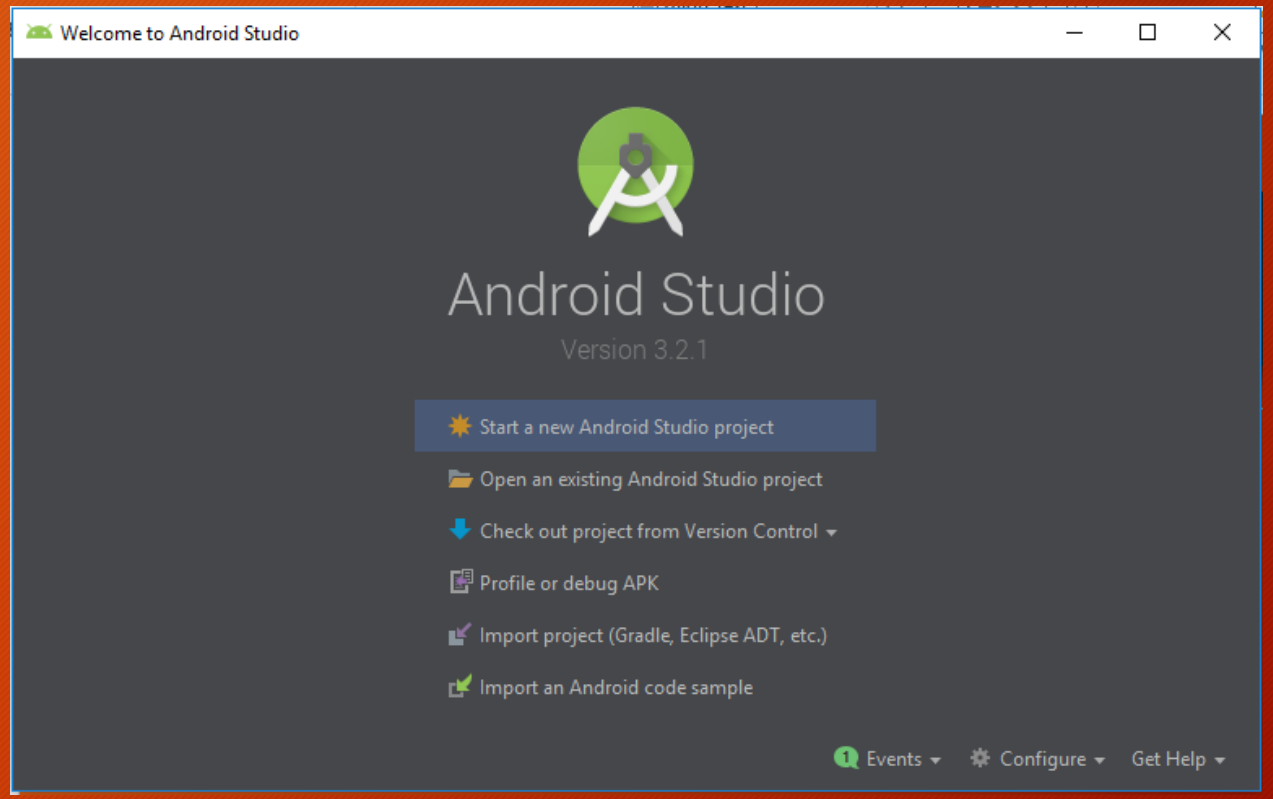
Activity life cycle

- To navigate transitions between stages of the activity lifecycle, the Activity class provides a core set of six callbacks:
  - onCreate(),
  - onStart(),
  - onResume(),
  - onPause(),
  - onStop(), and
  - onDestroy().
- The system invokes each of these callbacks as an activity enters a new state.

Activity life cycle

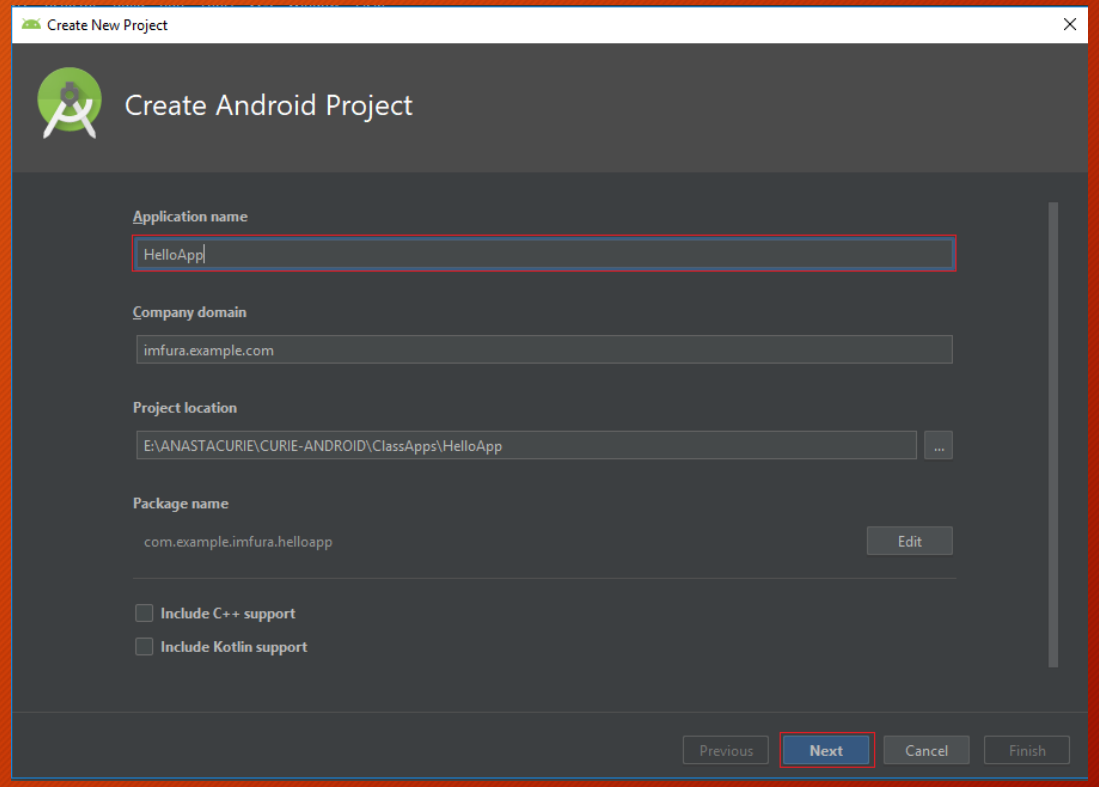
First Android Project

- Launch android studio then choose start a new android studio project



First Android Project

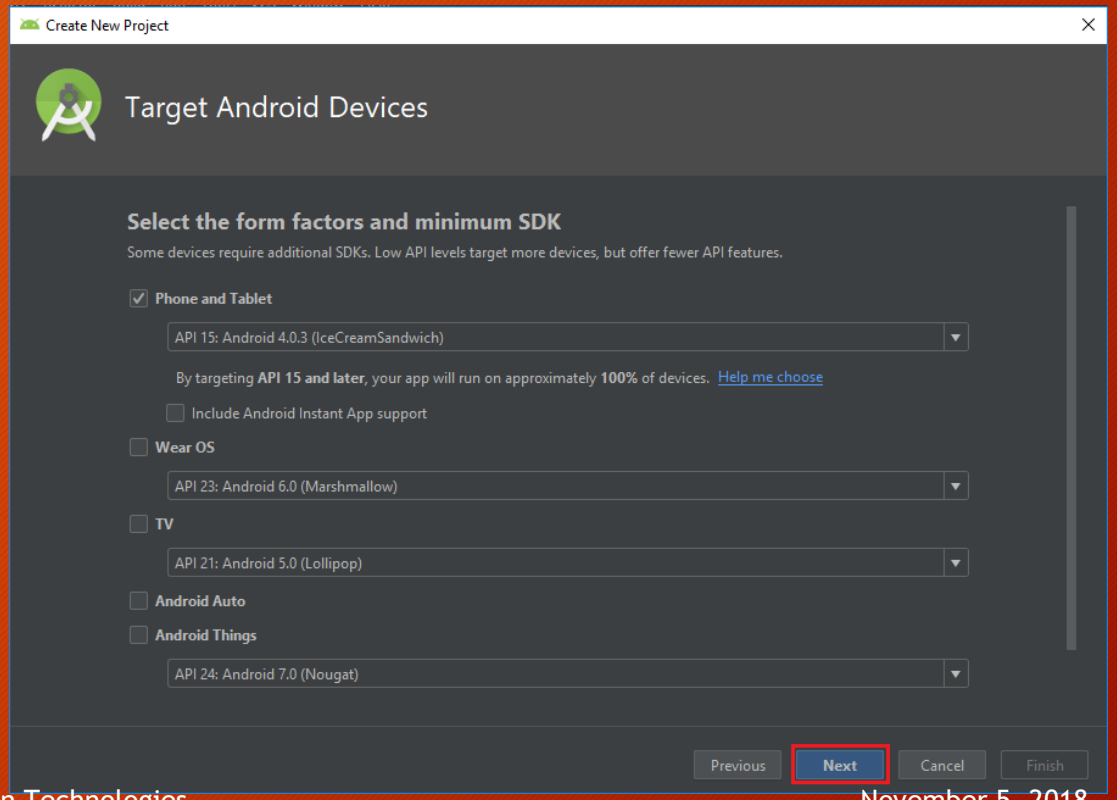
- In Create new project windows give application the name



- Android Project Structure
- Project files
- App components
- Activity life cycle
- First android project
- Android Virtual Device (AVD)

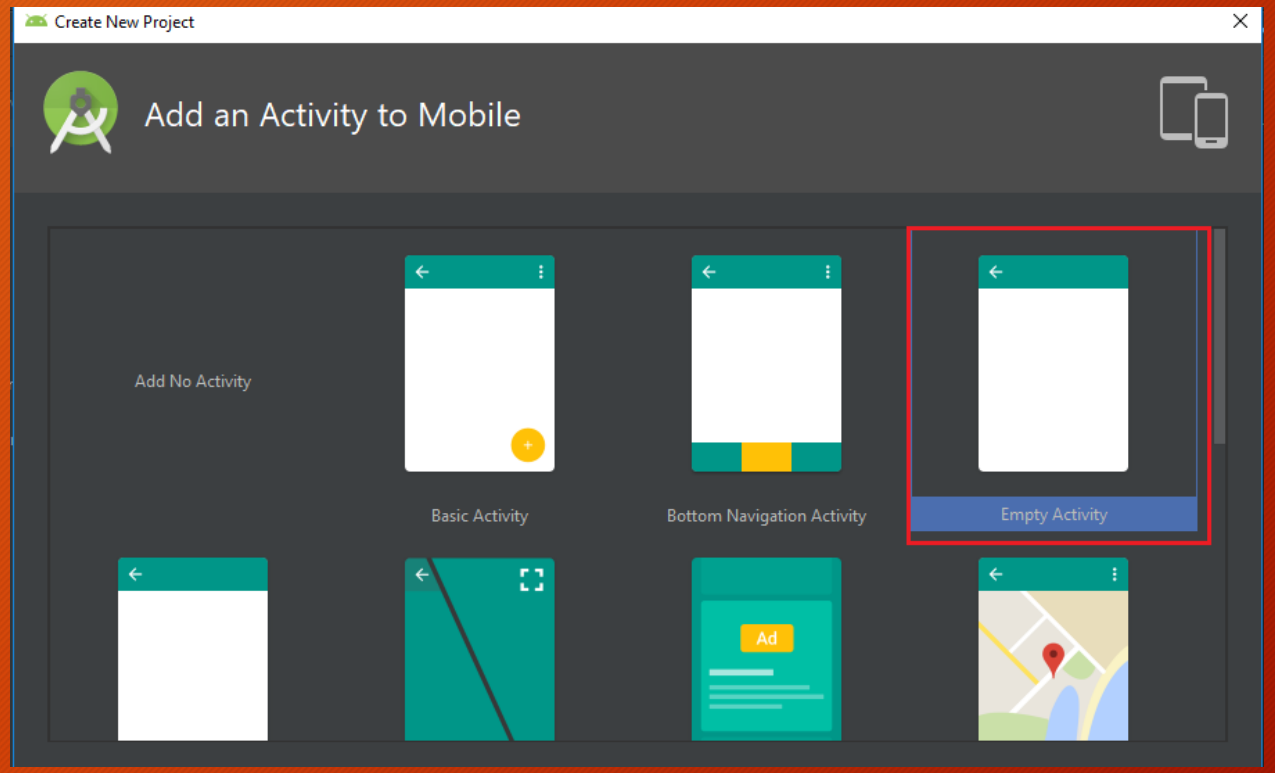
First Android Project

- Select the API Level then click NEXT



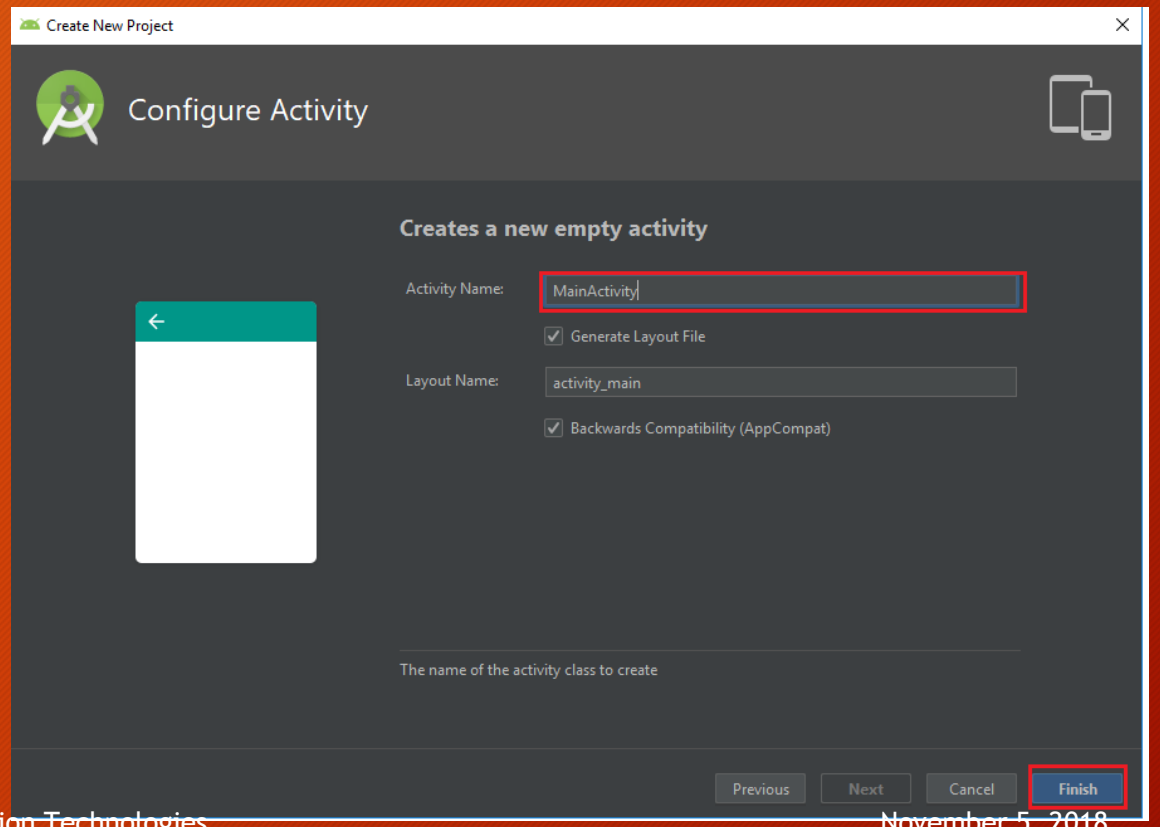
First Android Project

- Choose the preferable activity (here choose Empty Activity) depending on the app



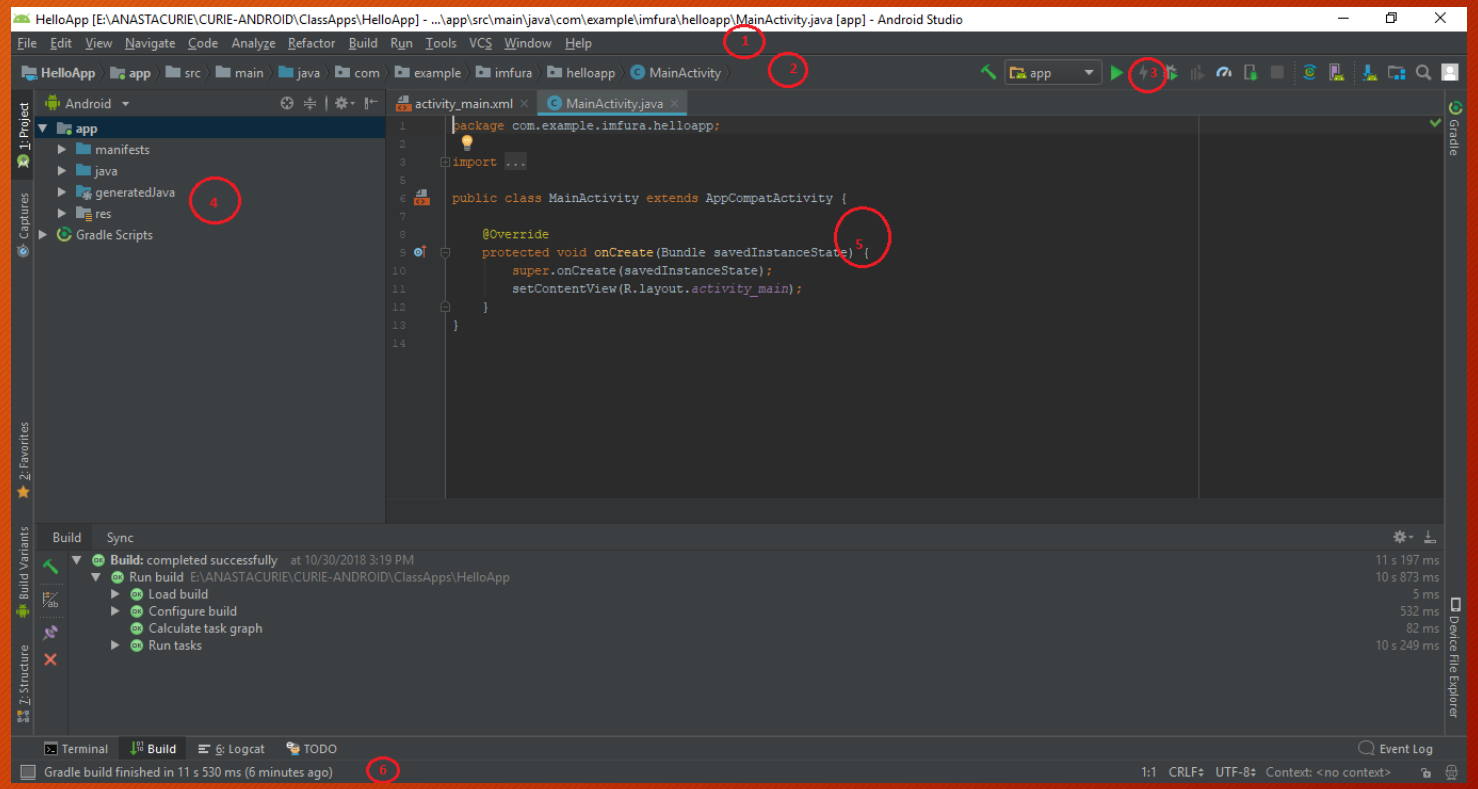
First Android Project

- Use configure activity window to set the name of activity (don’t change the default name) then click FINISH



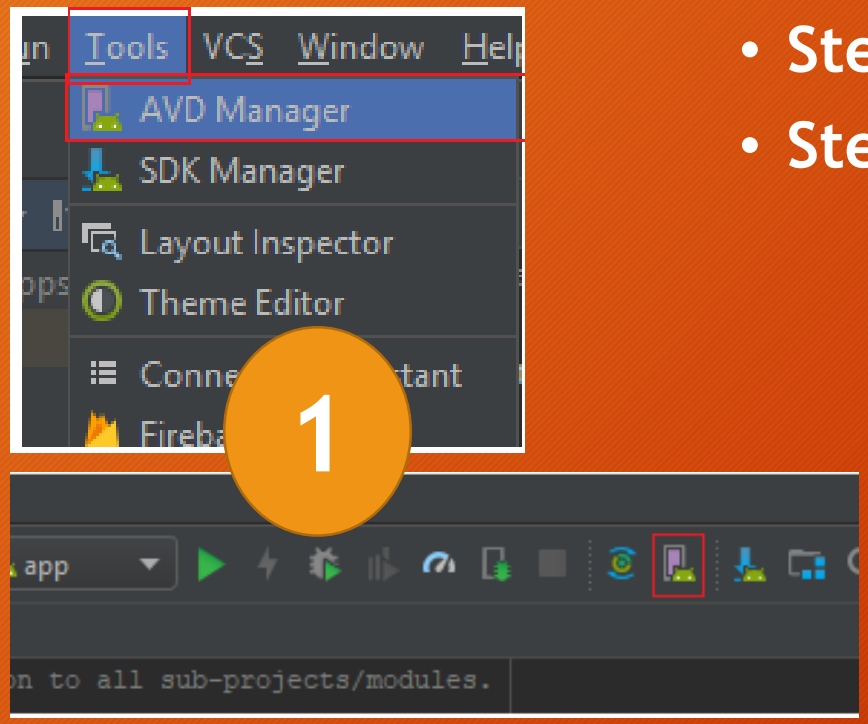
Explore Android studio

1. Menu bar
2. Navigation bar
3. Tool bar
4. Project Tool window
5. Editor window
6. Status bar

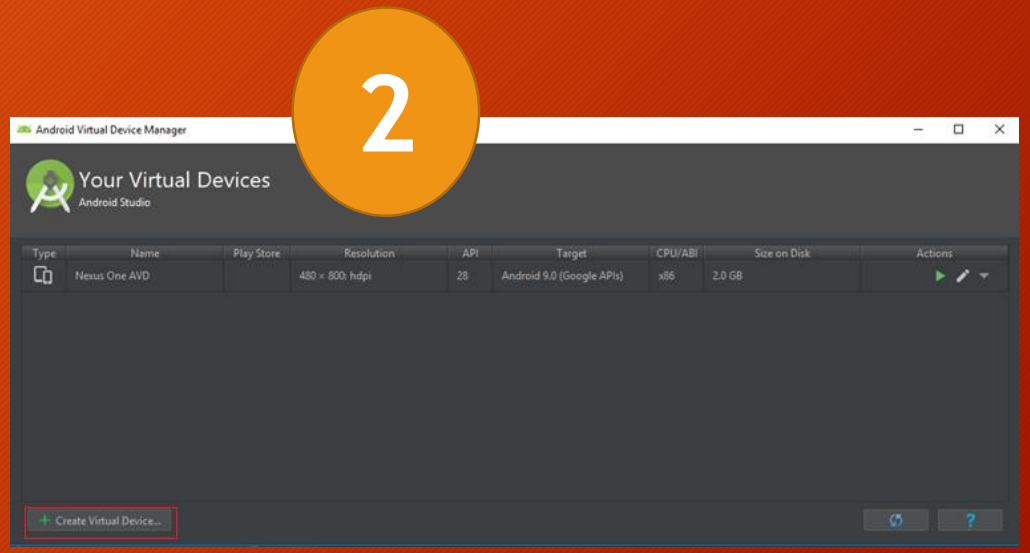


- Android Project Structure
- Project files
- App components
- Activity life cycle
- First android project
- Android Virtual Device (AVD)

Android Virtual Device



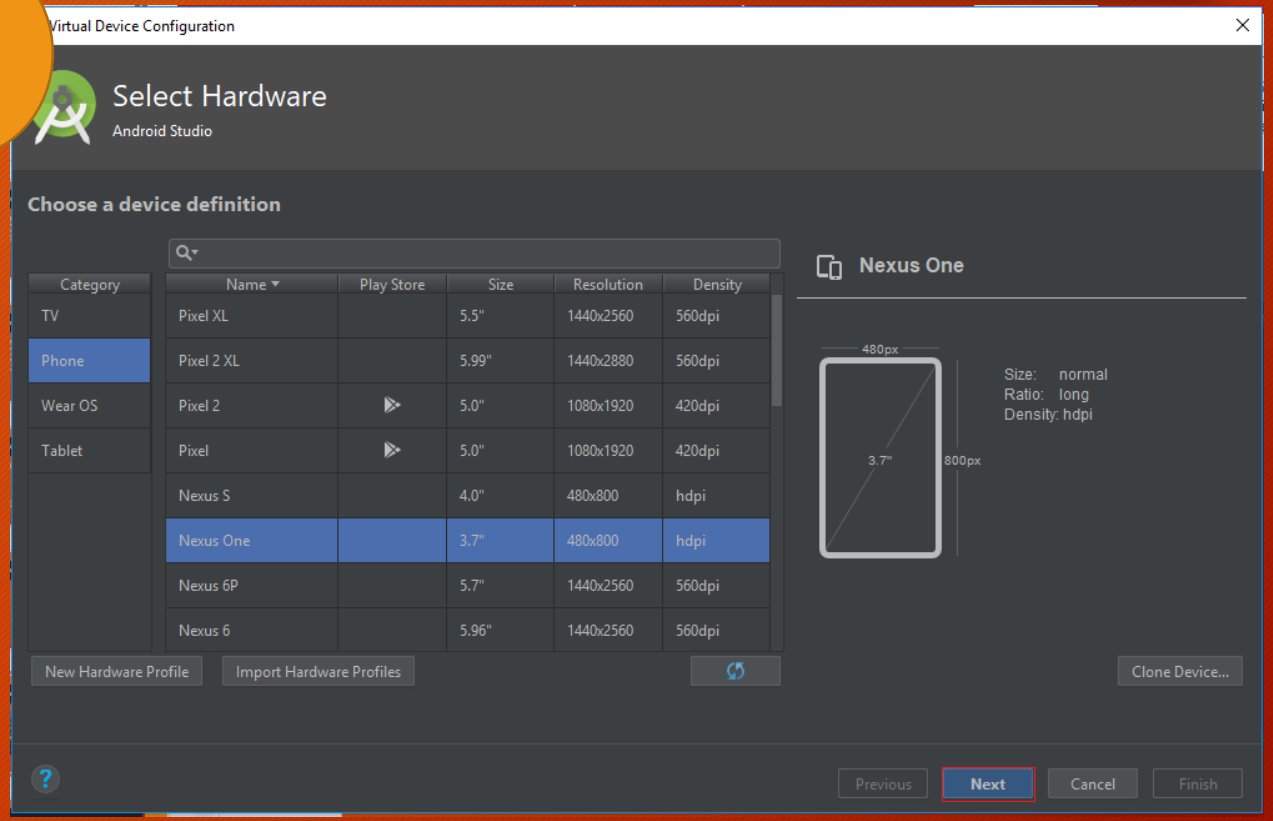
- **Step 1:** Open AVD Manager
- **Step 2:** click Create Virtual Device



Android Virtual Device

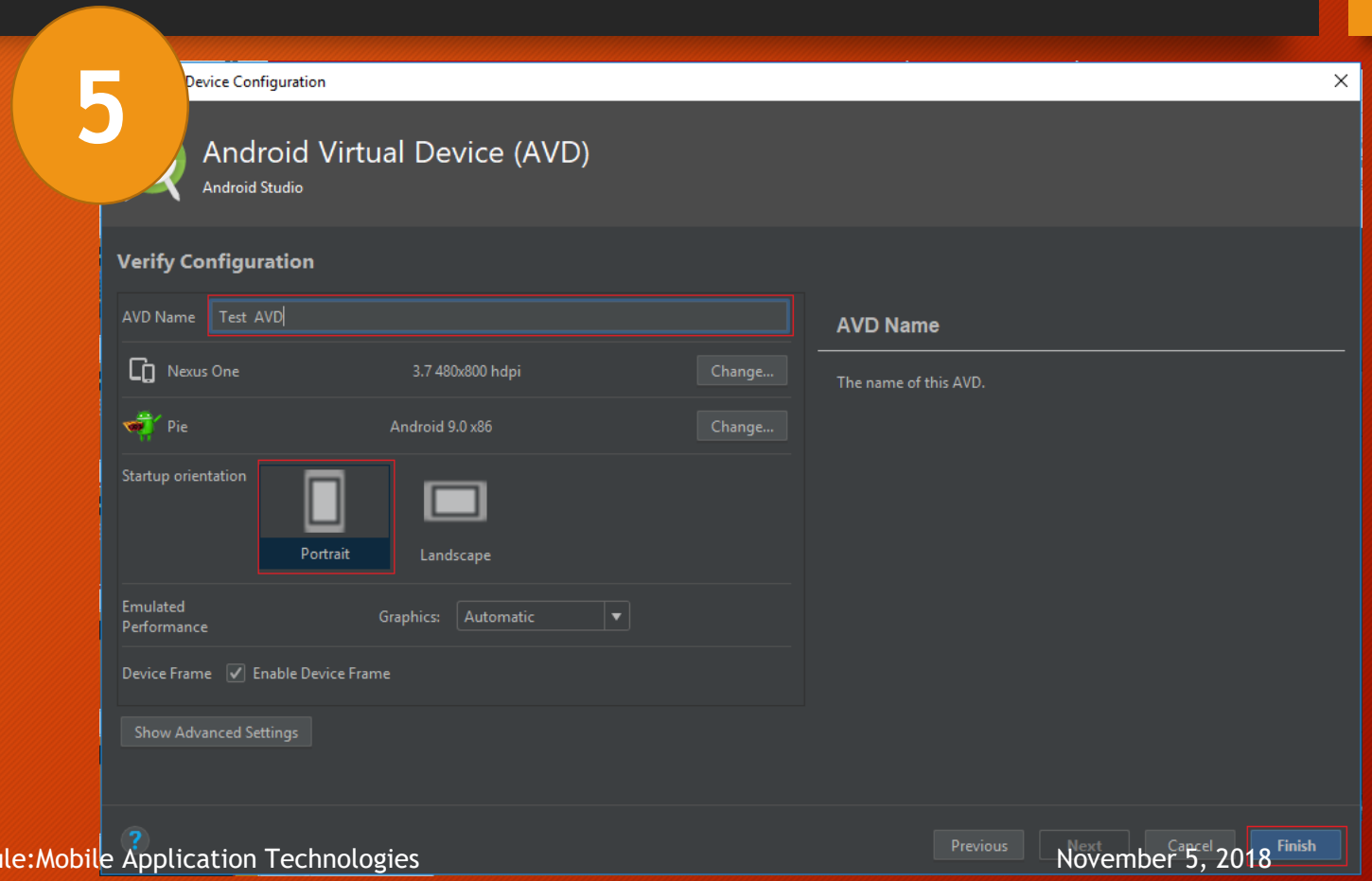
- **Step 3:** Choose device definition then click NEXT

3



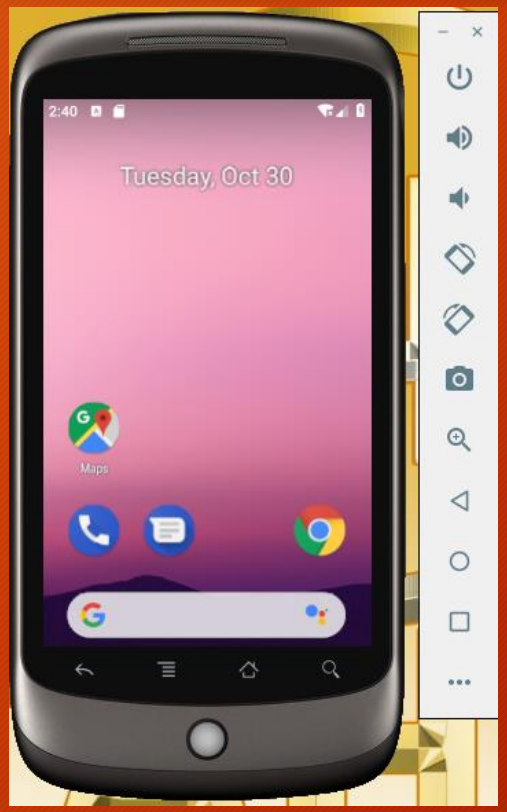
Android Virtual Device

- **Step 5:** Give AVD the name, set startup orientation the click FINISH



Launch Emulator

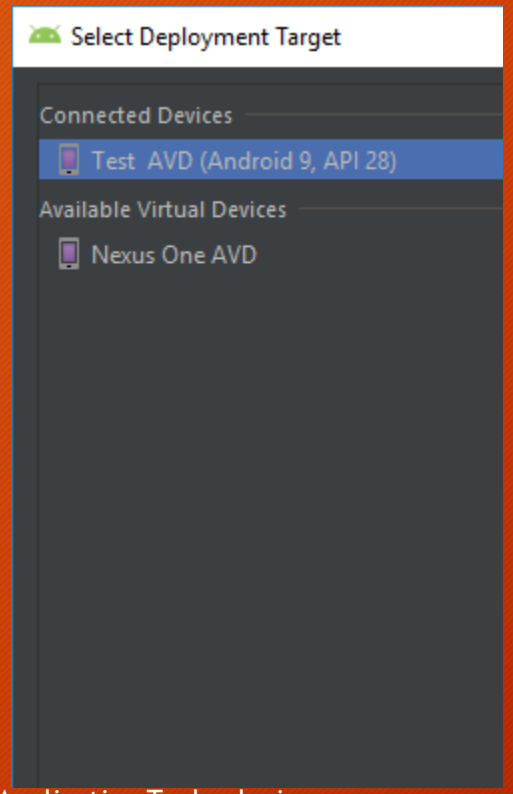
- Double-click ADV or click run

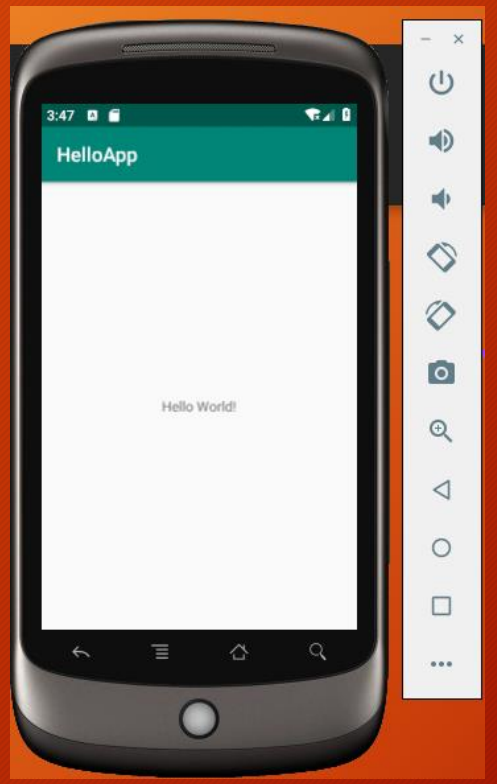


Run HelloApp project

- Click Run under Run menu or click Run icon

- Select deployment target



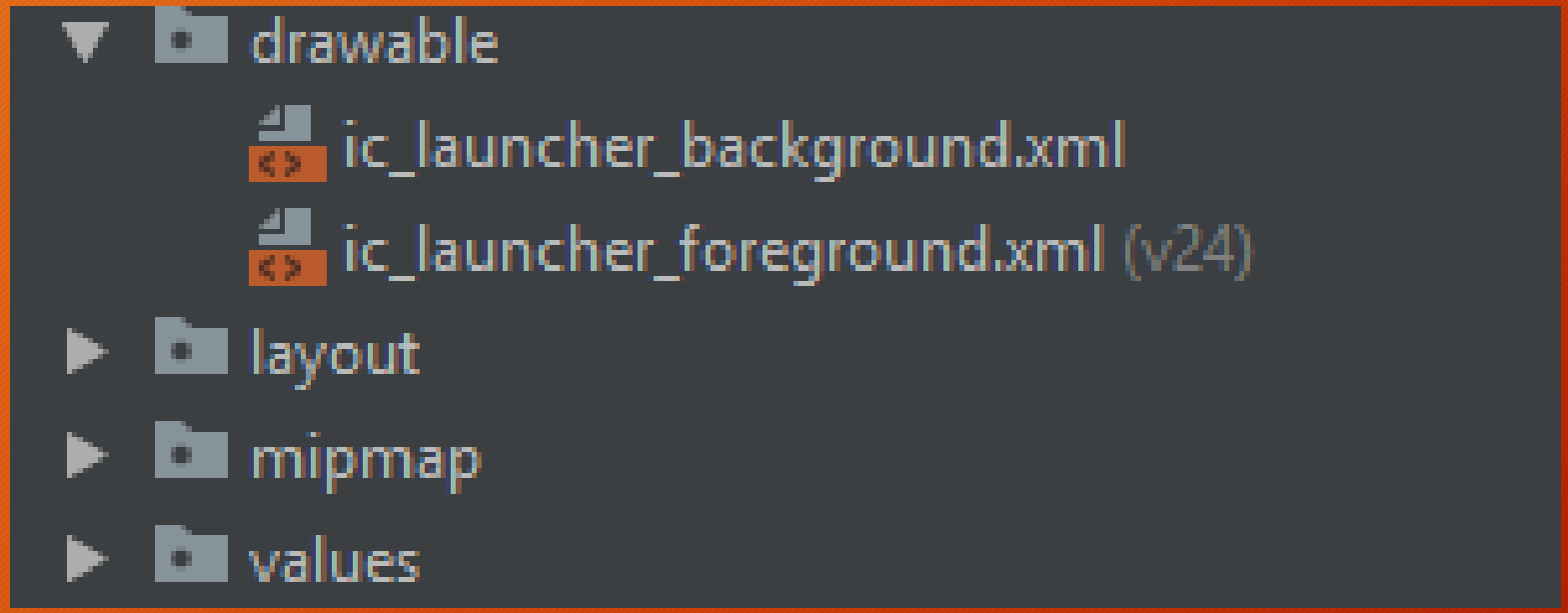


Resources on first application (HelloApp)

- Different resources exit for project HelloApp
  - Drawable
  - Layout
  - Mipmap
  - Values

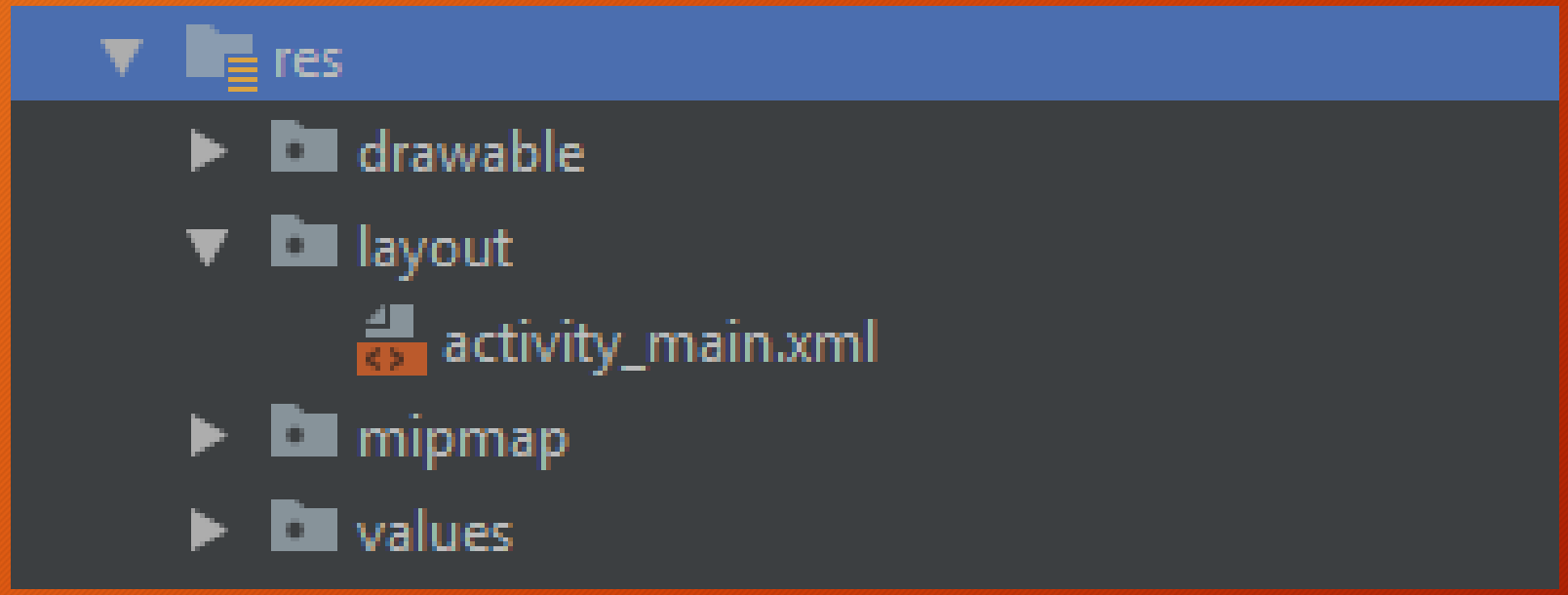
Drawable

Drawable is the container of the images



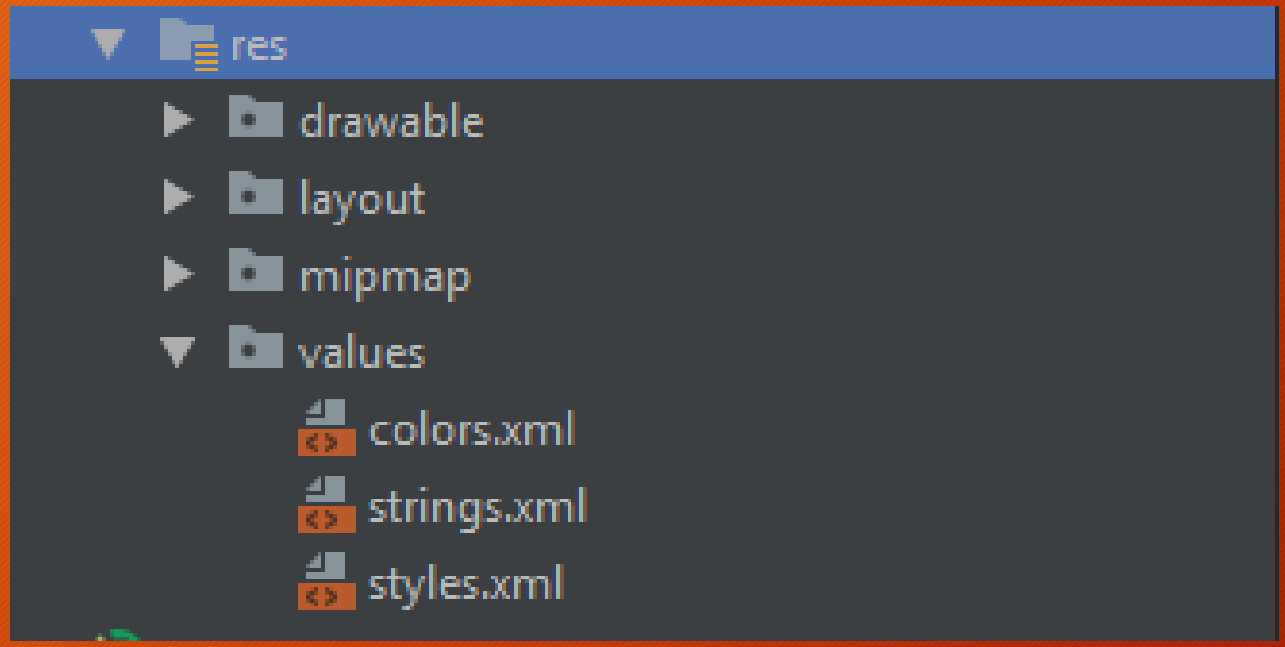
Layout

- Layout contains .xml files responsible for UI layouts of the activities
- In HelloApp there is only one file activity_main.xml in Layout



Values

- Values contain .xml files in which we define some values such as colors and strings



END

THANK YOU!