Mobile Application Architecture

Hua Huang

Assignments

- Read Chapter 1 of *Head start for Android Programming.* Getting Started: Diving In

- Complete the Android development environment setup. Generate your first running Android app.

Computer Requirement

- Installing Android Studio:

Windows

★ **Note:** Windows machines with ARM-based CPUs aren't currently supported.

Here are the system requirements for Windows:

Requirement	Minimum	Recommended
OS	64-bit Microsoft Windows 10	Latest 64-bit version of Windows
RAM	**Studio:** 8 GB **Studio & Emulator:** 16GB	32GB
CPU	Virtualization support Required (Intel VT-x or AMD-V, enabled in BIOS). CPU microarchitecture after 2017. Intel 8th Gen Core i5 / AMD Zen Ryzen (e.g., Intel i5-8xxx, Ryzen 1xxx).	Virtualization support Required (Intel VT-x or AMD-V, enabled in BIOS). Latest CPU microarchitecture. Look for CPUs from the Intel Core i5, i7, or i9 series and or the suffixes H/HK/HX for laptop or suffixes S/F/K for desktop, or the AMD Ryzen 5, 6, 7, or 9 series. Please be aware that Intel® Core™ N-Series and U-Series processors are not recommended due to insufficient performance.
Disk space	**Studio:** 8 GB of free space. **Studio & Emulator:** 16GB of free space	Solid state drive with 32 GB or more
Screen resolution	1280 x 800	1920 x 1080
GPU	**Studio:** None **Studio & Emulator:** GPU with 4GB VRAM such as Nvidia Geforce 10 series or newer, or AMD Radeon RX 5000 or newer with the latest drivers	GPU with 8GB VRAM such as Nvidia Geforce 20 series or newer, or AMD Radeon RX6600 or newer with the latest drivers.

Resources for Laptops

- Technology Resources Program (TRP)
  - Short term laptop loan
  - https://ue.ucmerced.edu/student-resources/technology-resources
- School of Engineering Laptop policy
  - https://engr-advising.ucmerced.edu/policies/soe-policies
  - https://engr-advising.ucmerced.edu/sites/engr-advising.ucmerced.edu/files/page/documents/2019policyonlaptopssoecomputer.pdf

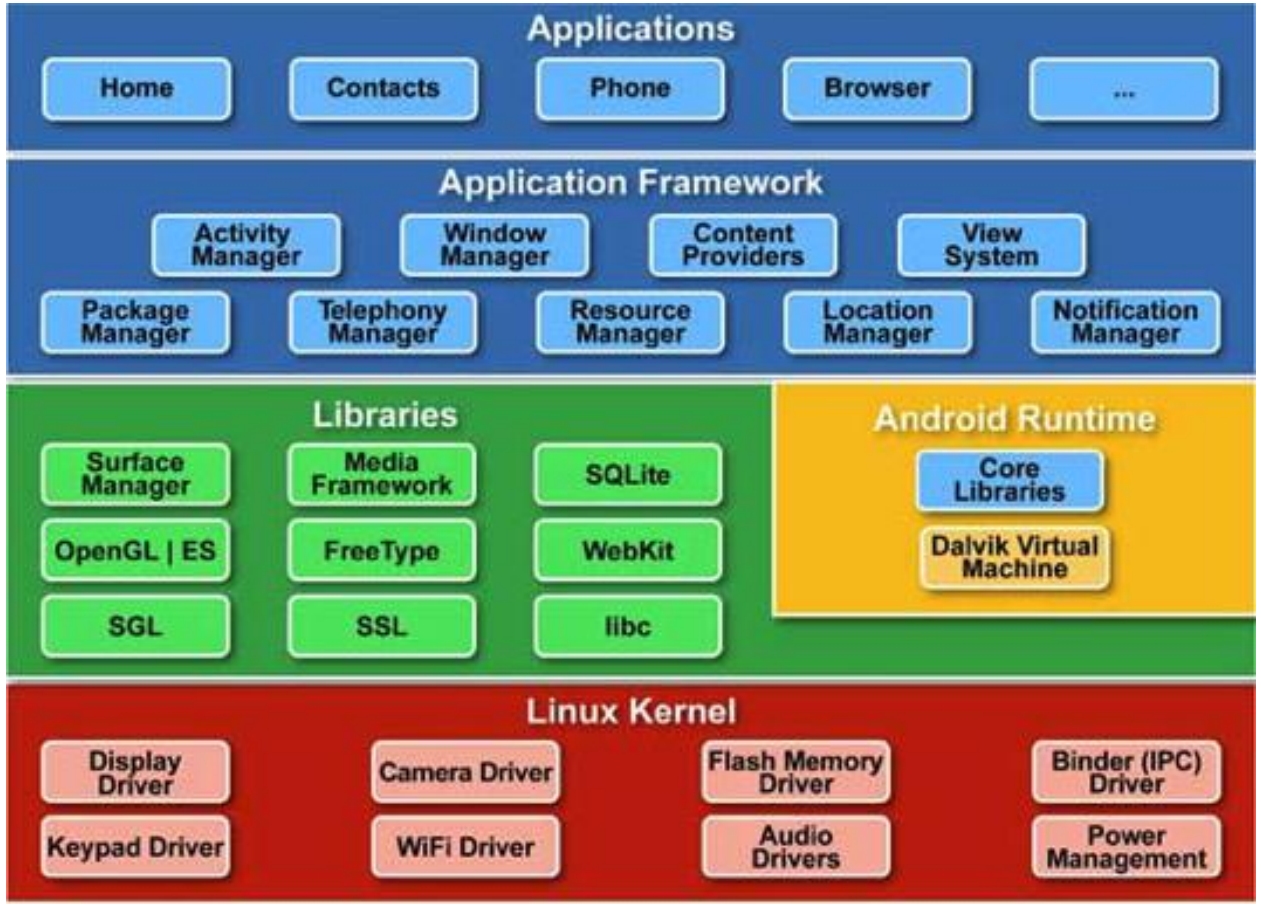
Android Architecture

The Basics

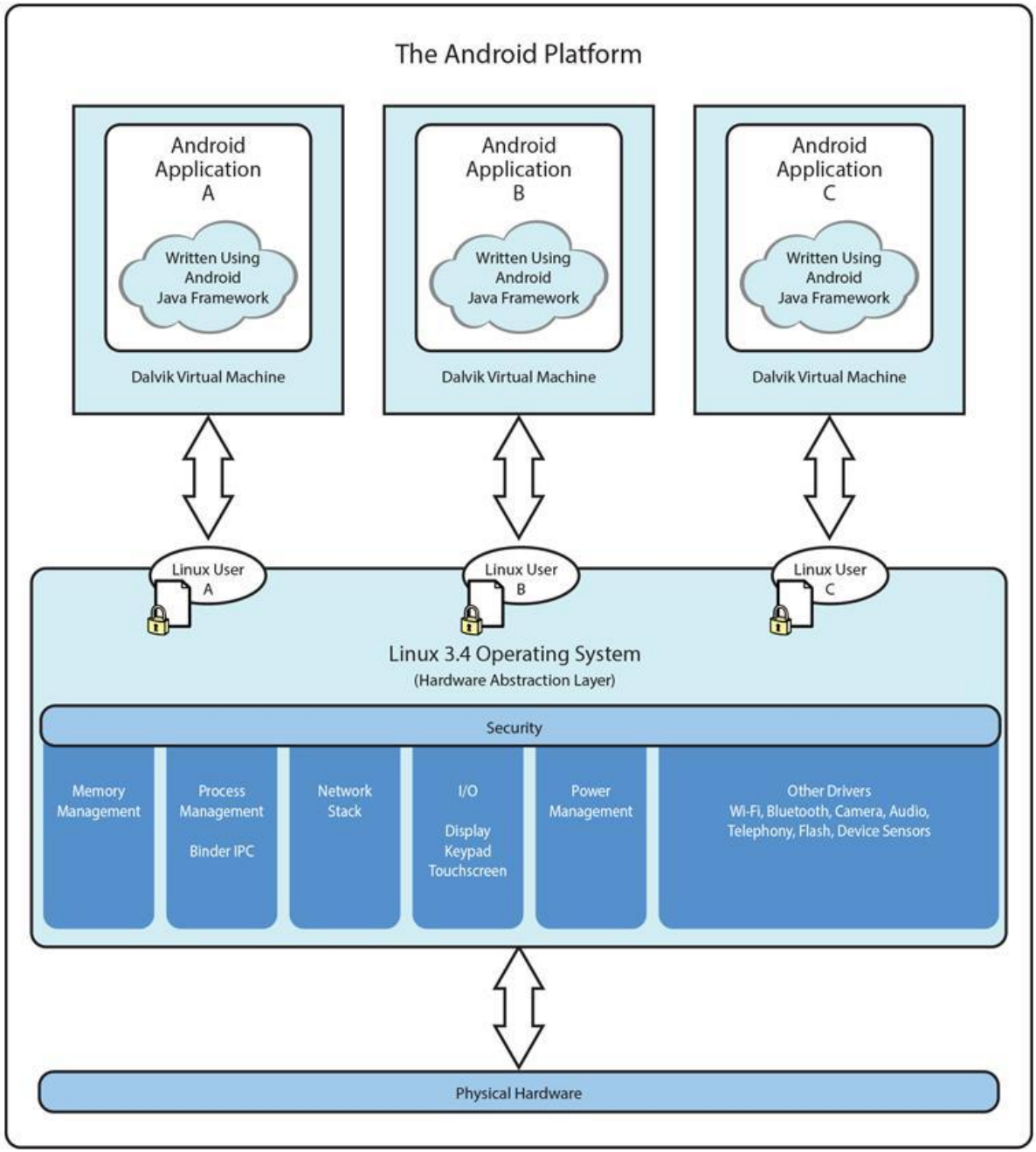
- In general, all apps are written in Java or Kotlin
- A compiled Android app is an .apk
- Android apps must be digitally signed in some way to execute
  - so that we know who releases the apk
- This digital signature can be a debug certificate that comes default with any installation

The Android Architecture

- **OS**: Linux kernel, drivers
- **Apps**: programmed & UI in Java
- **Libraries**: OpenGL ES (graphics), SQLite (database), etc.

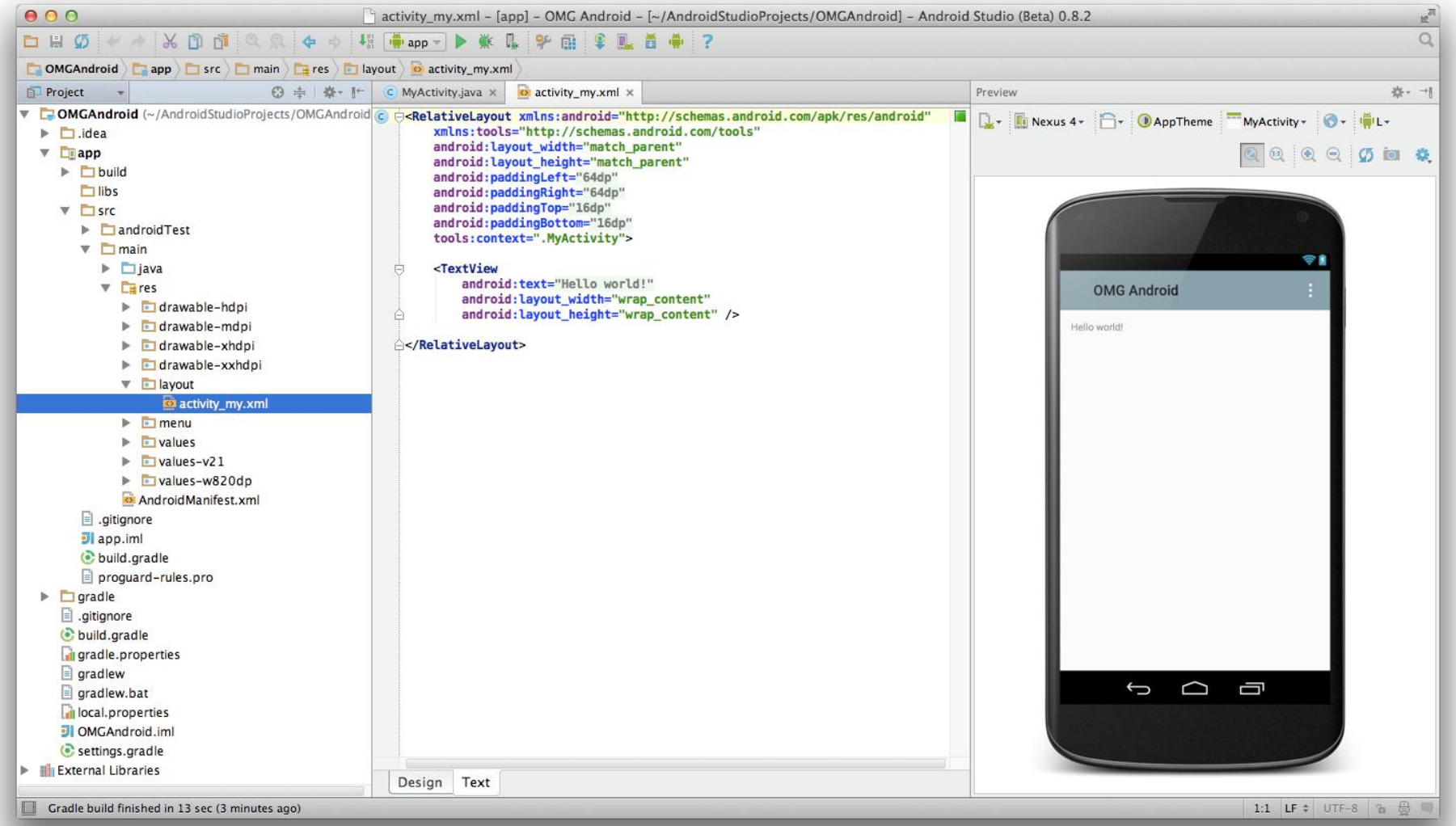


- Each Android app runs in its own security sandbox (VM, minimizes complete system crashes)
- Android OS multi-user Linux system
- Each app is a different user (assigned unique Linux ID)
- Access control: only process with the app's user ID can access its files



Android Development Environment

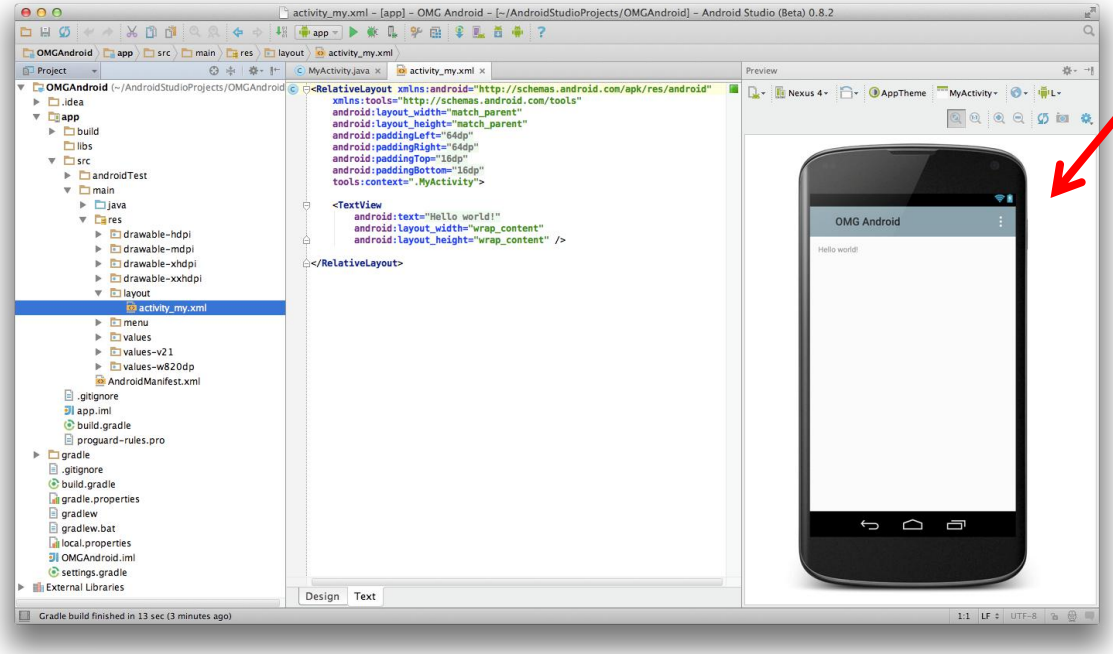
Android Studio Layout



Where to run Android Apps

- Android app can run on:
  - Real phone (or device)
  - Emulator (software version of phone)

Emulated Phone in Android Studio



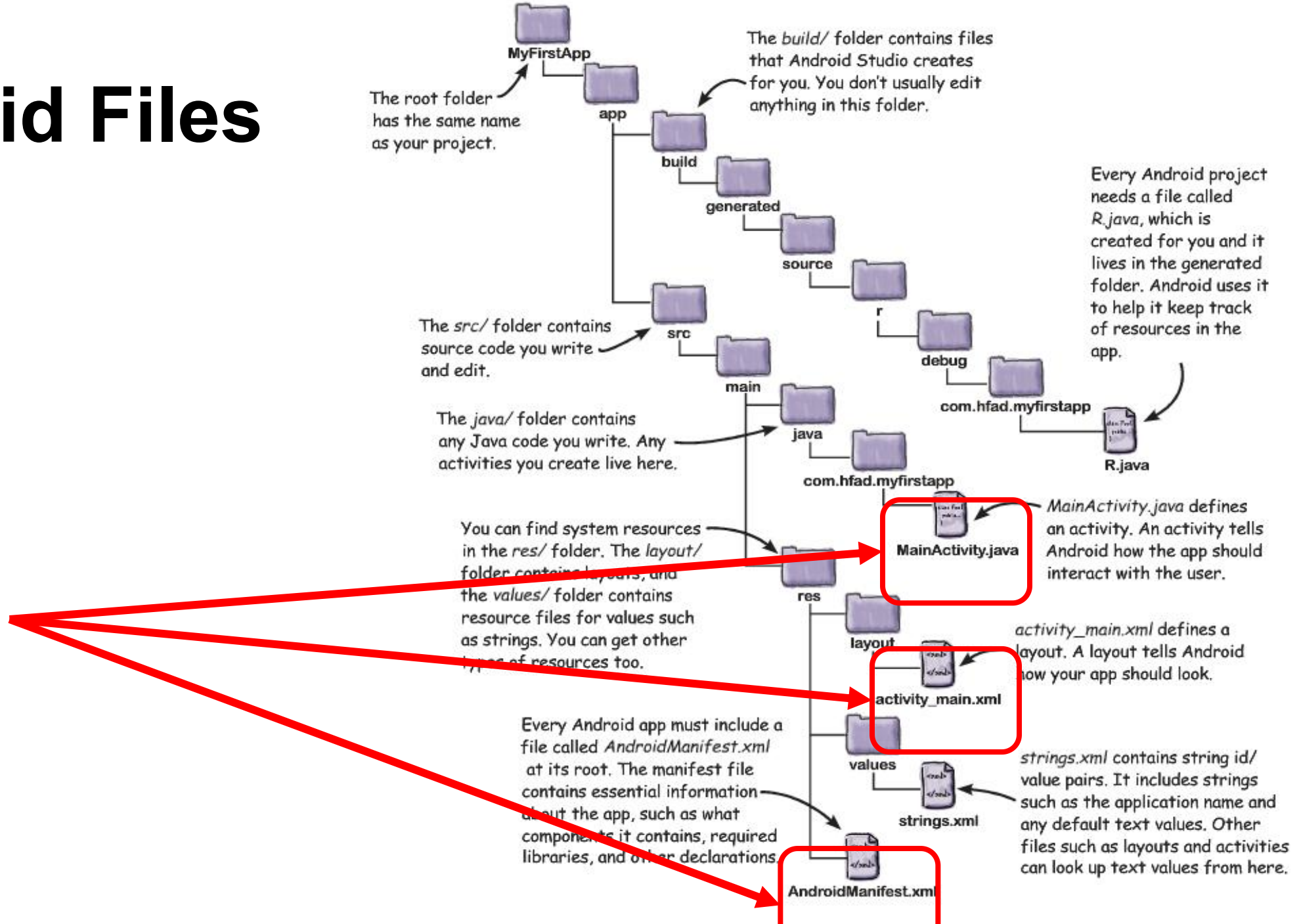
Run Android App on Real Phone

- Computer side: install Android Debug Bridge (ADB)
- Phone side: enable USB debugging
- checkout lab0

Our First Android App

Android Files

3 Main Files to Write Android app

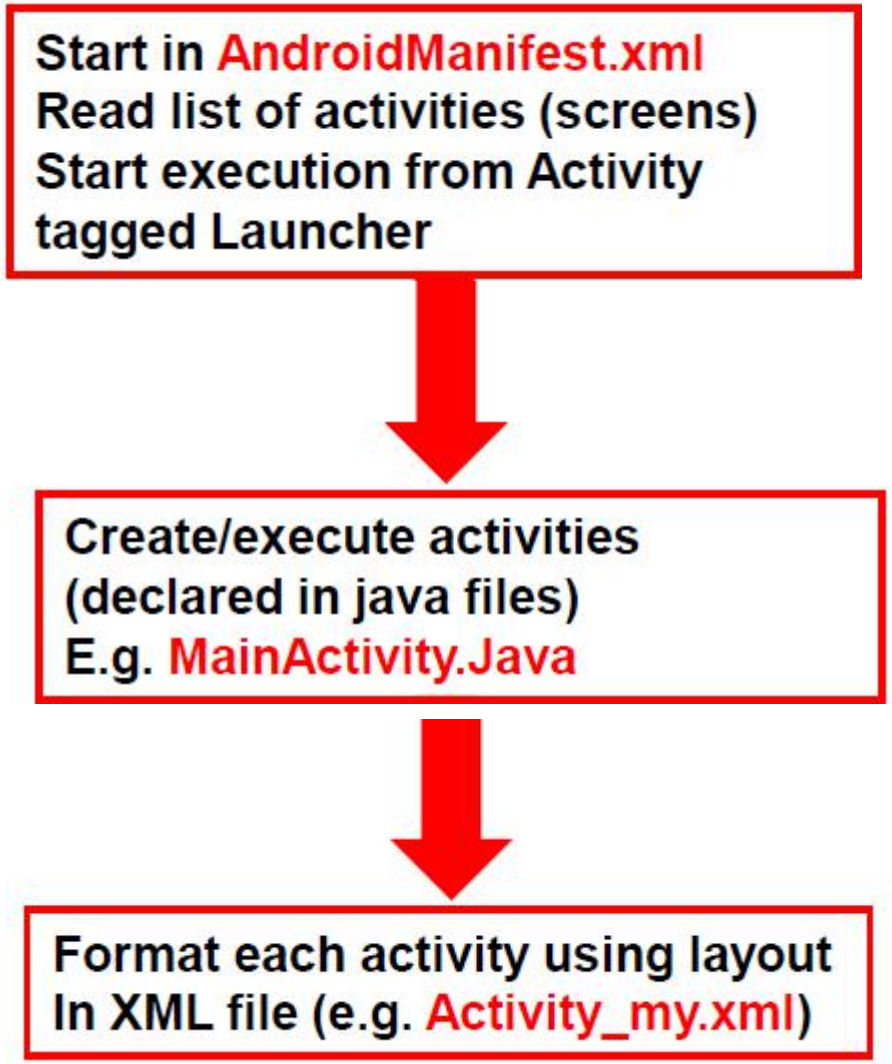



Android
app
manifests
AndroidManifest.xml
java
com.example.watchsensor
MainActivity
SensorDescriptor
SensorInfo
java (generated)
res
layout
activity_main.xml
mipmap
values
dimens.xml
strings (2)
strings.xml
strings.xml (round)
Gradle Scripts
Manifest
Java Program
Layout

3 Files in “Hello World” Android Project

- **Activity_my.xml:** XML file specifying screen layout
- **MainActivity.Java:**Java code to define behavior, actions taken when button clicked (intelligence)
- **AndroidManifest.xml:**
  - Lists all screens, components of app
  - Analogous to a table of contents for a book
  - E.g. Hello world program has 1 screen, so AndroidManifest.xml has 1 item listed
  - App starts running here (like main( ) in C)
- **Note:** Android Studio creates these 3 files for you

Execution Order



Inside “Hello World” AndroidManifest.xml

This file is written using xml namespace and tags and rules for android

package name

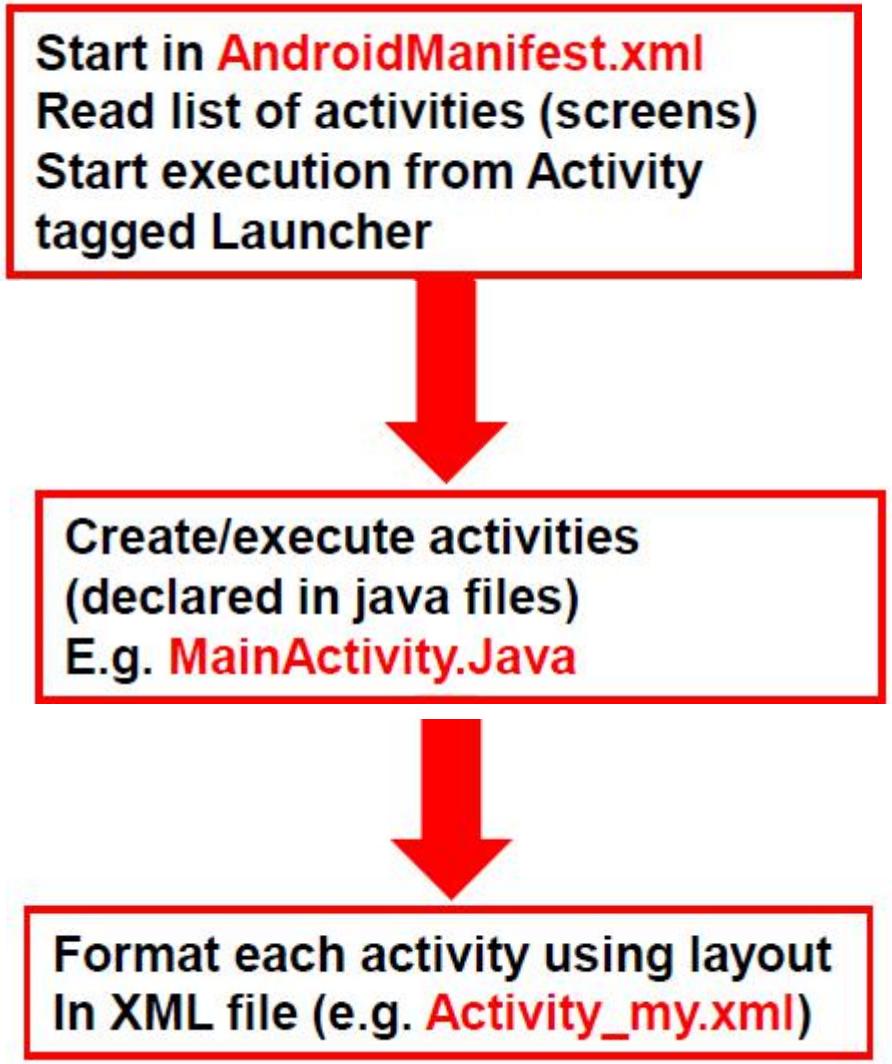
Android Version

Activity List

```
<?xml version="1.0"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
  package="com.commonsware.android.skeleton"
  android:versionCode="1"
  android:versionName="1.0">

  <application>
    <activity
      android:name="Now"
      android:label="Now">
      <intent-filter>
        <action android:name="android.intent.action.MAIN" />

        <category android:name="android.intent.category.LAUNCHER" />
      </intent-filter>
    </activity>
  </application>

</manifest>
```

One activity (screen) designated LAUNCHER.
The app starts running here

Execution Order



Example Activity Java file (E.g. MainActivity.java)

Package Declaration

Import needed classes

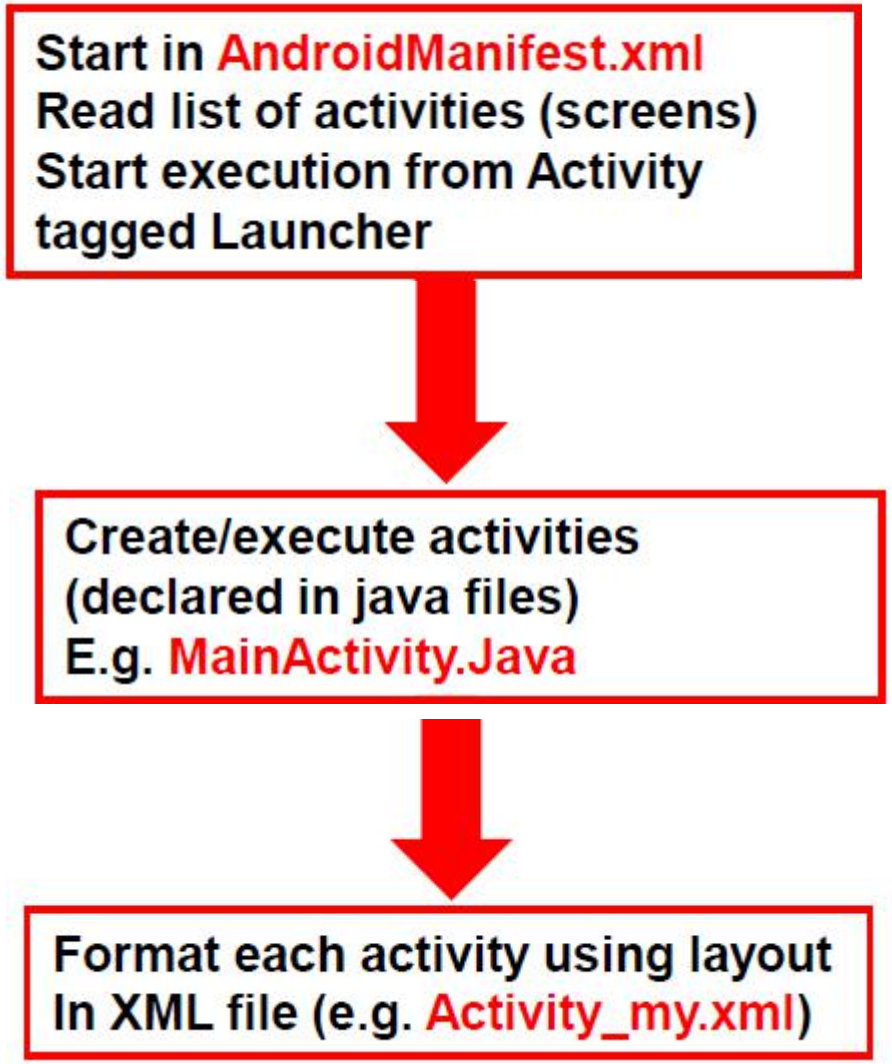
Inherits from the Android Activity Class

App initialization: onCreate()

```
package com.commonsware.empublite;

import android.app.Activity;
import android.os.Bundle;

public class EmPubLiteActivity extends Activity {
  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
  }
}
```

Use screen layout in file main.xml

Execution Order



Declare Layout

Add Widgets

Widget properties

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:tools="http://schemas.android.com/tools"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   tools:context=".EmPubLiteActivity">

   <TextView
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_centerHorizontal="true"
      android:layout_centerVertical="true"
      android:text="@string/hello_world"/>

</RelativeLayout>
```

Simple XML file Designing UI

After choosing the layout, then widgets added to design UI

- XML Layout files consist of:
  - UI components (boxes) called **Views**
  - Different types of views. E.g
    - **TextView:** contains text,
    - **ImageView:** picture,
    - **WebView:** web page
  - **Views** arranged into layouts or **ViewGroups**

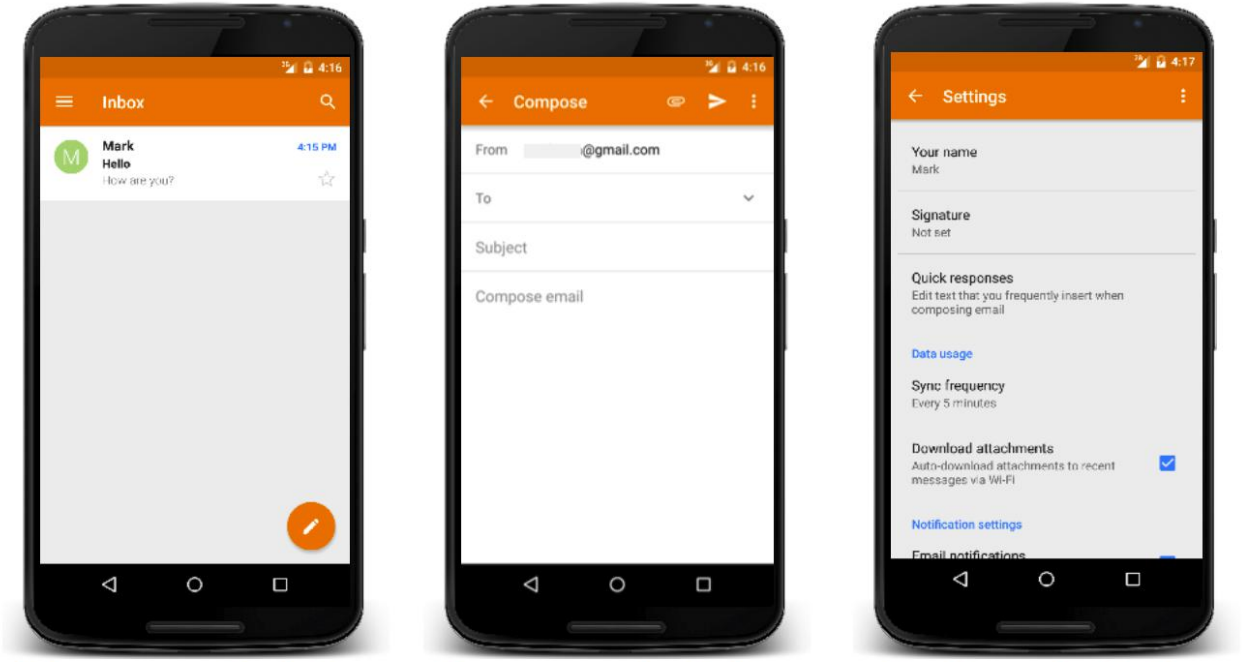
Android Software Components

Consider

- What is a mobile app made of?
  - graphical UI: buttons, texts, video etc
  - play sounds?
  - services: sensors, locations, etc
  - more
- If one module fails, we don’t want to quit entire app
  - network is disconnected and fails to download audio, but we still want to play local resources
  - Encasulation

Activity

- Conceptually, an Activity shows a single screen of your application
- In other words, an App really is a collection of related Activities



Messages Activity Compose Activity Settings Activity

Activity

- Consider each Activity both a screen and a feature
- Apps can activate Activities in other Apps

Service

- A Service is a component that runs in the background to perform long-running operations
- A Service has no UI
- Examples of Services:
  - Playing music in background
  - Gathering GPS data
  - Downloading a data set from the server

Intent

- An Intent is a message that requests an action from another component of the system
- This includes the “please start up your App” Intent that the system sends when a user clicks on your App icon

Broadcast Receiver

- A Broadcast Receiver responds to system-wide announcements (which are manifested as Intents)
- System status information is delivered this way
  - e.g., device turned on, screen off, low battery, phone call incoming, etc.
- Broadcast Receivers typically don’t have a UI, but could have a status bar icon

Connected Apps

- Due to the component nature of Apps (made up of Activities, Services, etc.), it is easy to build features of your App using existing system components
- For example, if your App needs to take a picture, you can query the Camera Activity to handle that request and return the resulting image

Put them all together

- If an App is made up of all these disparate parts, what holds them all together?
- The AndroidManifest.xml file!
  - Sets up all permissions the user has to agree to (i.e. Internet, GPS, contacts, etc.)
  - Declares the API level of the App
  - Requests hardware features needed
  - Needed libraries
  - Which Activities are part of this App