Native app development with android

Part I

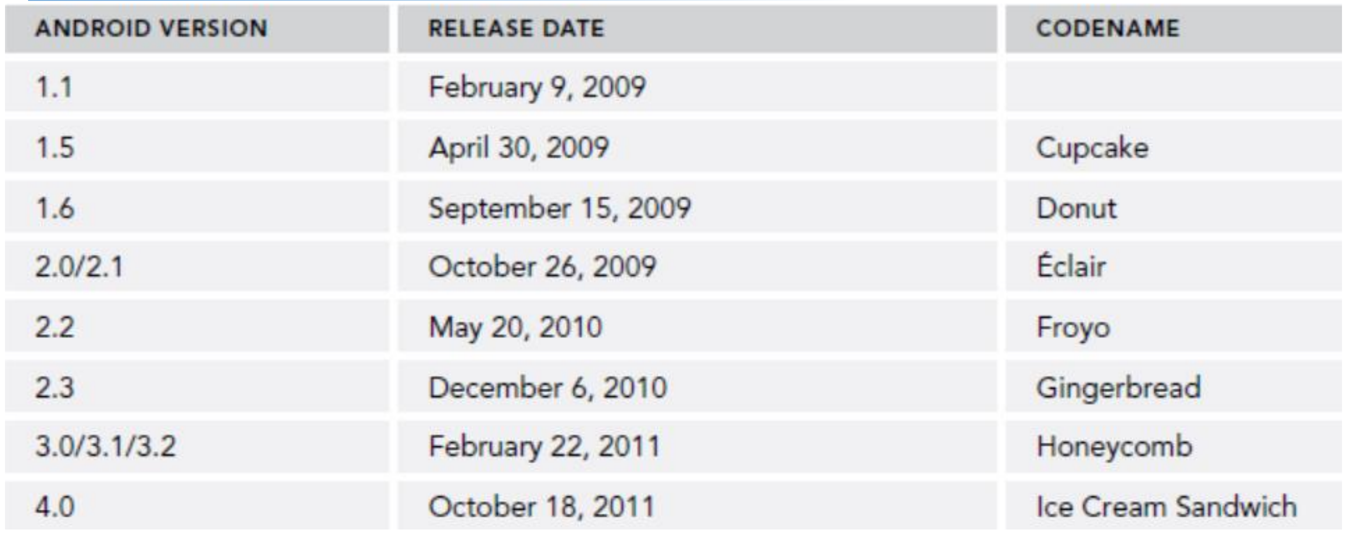
What is android?

- **Android is a mobile operating system that is based on a modified version of Linux.**
- **It was originally developed by a startup of the same name, Android, Inc.**
- **In 2005, as part of its strategy to enter the mobile space, Google purchased Android, Inc. and took over its development work, and also their team!**
- **Google wanted Android OS to be free.**
- **Thus it was released under Apache License.**
- **Anyone can download Android OS.**
- **Vendors do that, they also add their own extensions to differentiate their products from others.**
- **In 2017, there was Two Billion active Android devices…!!!**

Why android?

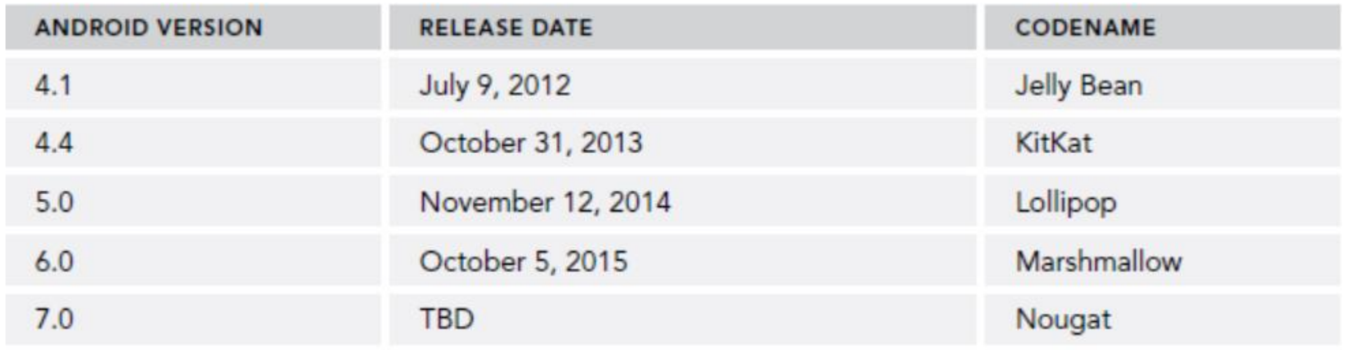
- **When iPhone first made a great success, many vendor saw Android as a great opportunity and solution.**
- **This included Motorola and Sony Ericson, both been developing their own OS for many years.**
- **And lets not forget HUAWEI, the Chinese dragon!**
- **Only iPhone and SAMSUMG have more market share than HUAWEI..!!**
- **The main advantage when adopting Android is that it offers a unified approach to development.**
- **Developers need only to develop for Android.**
- **And their apps will run on numerous Android devices**

Android versions



ANDROID VERSION	RELEASE DATE	CODENAME
1.1	February 9, 2009	
1.5	April 30, 2009	Cupcake
1.6	September 15, 2009	Donut
2.0/2.1	October 26, 2009	Éclair
2.2	May 20, 2010	Froyo
2.3	December 6, 2010	Gingerbread
3.0/3.1/3.2	February 22, 2011	Honeycomb
4.0	October 18, 2011	Ice Cream Sandwich

One important thing to keep in mind as you are looking at Android versions is that each version has its own features and APIs..!



ANDROID VERSION	RELEASE DATE	CODENAME
4.1	July 9, 2012	Jelly Bean
4.4	October 31, 2013	KitKat
5.0	November 12, 2014	Lollipop
6.0	October 5, 2015	Marshmallow
7.0	TBD	Nougat

8.0	Oreo
9.0	Pie

Features of android

- **Android is free to be customized by vendors.**
- **No fixed hardware or software configurations!**
- **However, Android supports the following features:**
  - **Storage: such as SQLite**
  - **Connectivity: GSM/EDGE, IDEN, CDMA, EV-DO, UMTS, Bluetooth (includes A2DP and AVRCP), Wi-Fi, LTE, and WiMAX.**
  - **Messaging**
  - **All Media Support**
  - **Hardware Support: Accelerometer sensor, camera, digital compass, proximity sensor, and GPS.**
  - **Multi-Touch**
  - **Multi Tasking**
  - **Tethering**

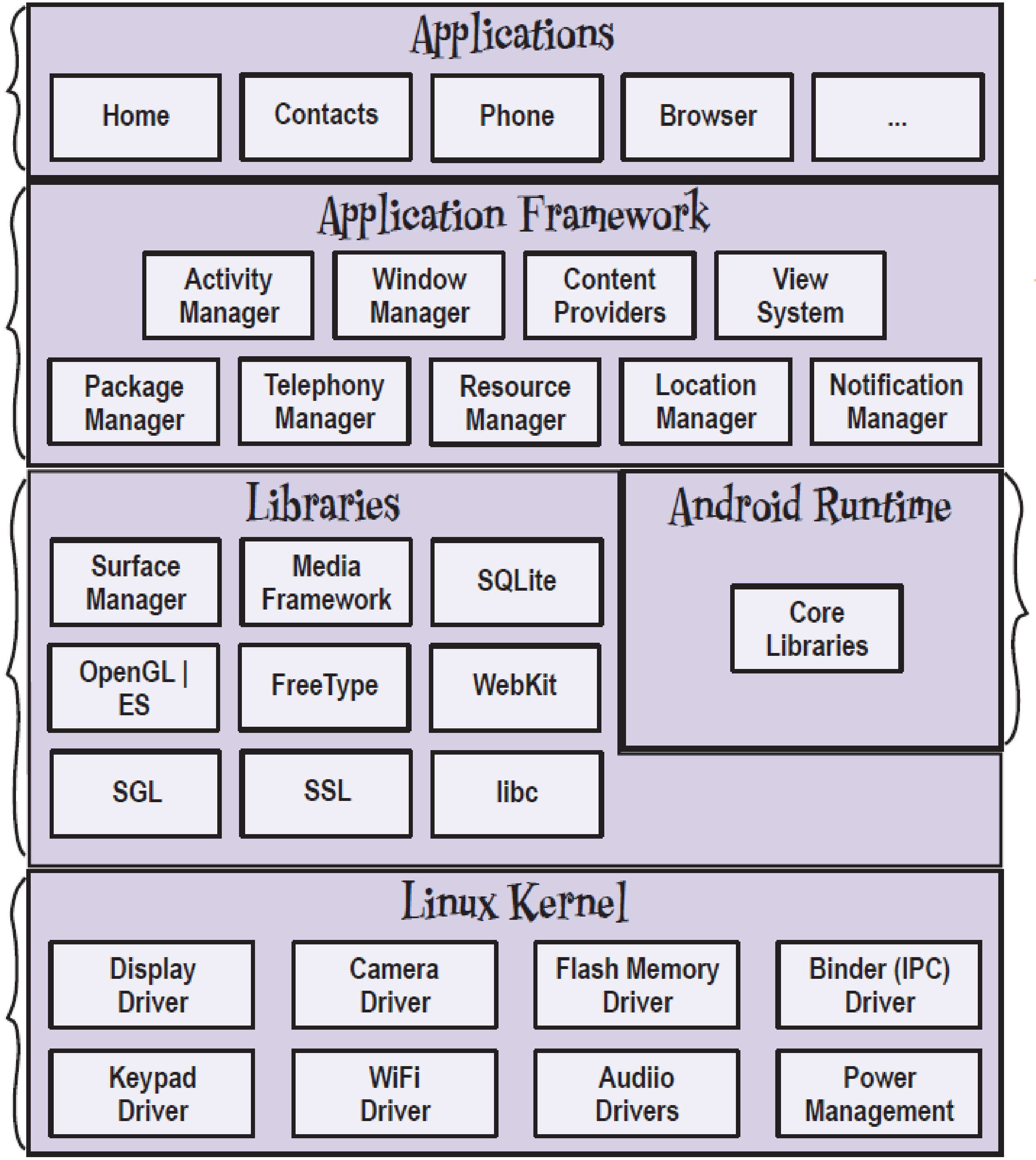
Android Platform Dissected

Android comes with a set of core applications such as Contacts, Calendar, Maps, and a browser.

When you build your apps, you have access to the same APIs used by the core applications. You use these APIs to control what your app looks like and how it behaves.

Underneath the application framework lies a set of C and C++ libraries. These libraries get exposed to you through the framework APIs.

Underneath everything else lies the Linux kernel. Android relies on the kernel for drivers, and also core services such as security and memory management.



The Android runtime comes with a set of core libraries that implement most of the Java programming language. Each Android app runs in its own process.

Android devices in the market!

- **Android devices come in all shapes and sizes including, but not limited to, the following types of devices:**
- **Smartphones**
- **Tablets**
- **E-reader devices**
- **Internet TVs**
- **Automobiles**
- **Smartwatches**
- **Tablets**

Android community

- **Stack Overflow** → most likely your Android issue has already been solved there!
- **Google Android Training** → developer.android.com
- **Android Discuss** → monitored closely by Google Android Team

Obtaining the required tools for development

- When developing for Android, you can use a Mac, a Windows PC, or a Linux machine.
- All necessary tools are freely available!.
- First download, **Java JDK 8**.
- Then download and install **Android Studio (IntelliJ IDEA)**
- Then through Android Studio, use **SDK Manager** Tool to download multiple versions of Android SDK.
- Multiple SDK versions enable you to write Android app that targets several Android OS versions.
- Finally you configure your **Emulators**

SDK Platform
There's one of these for each version of Android.

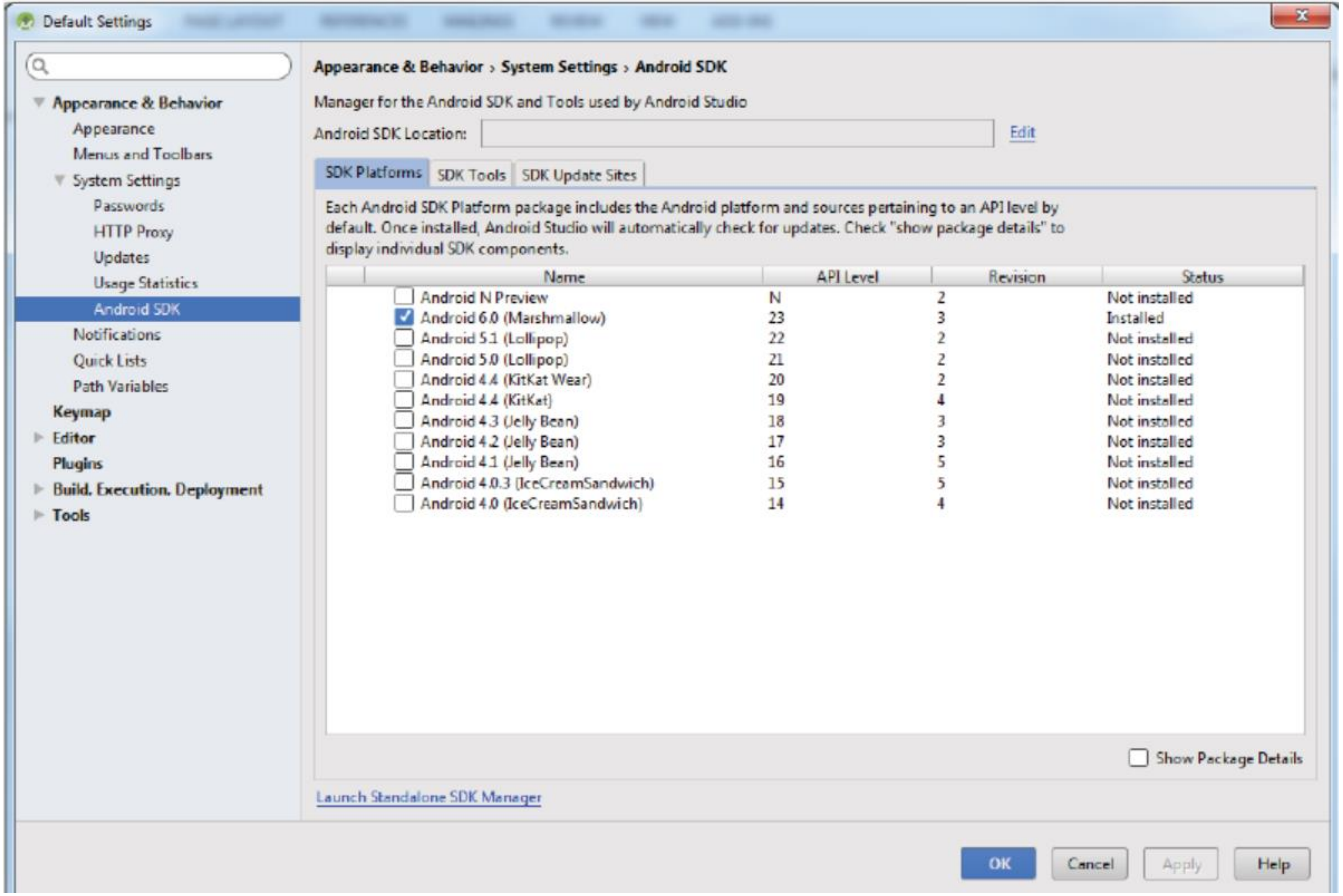
SDK Tools
Tools for debugging and testing, plus other useful utilities. The SDK also features a set of platform dependent tools.

Sample apps
If you want practical code examples to help you understand how to use some of the APIs, the sample apps might help you.

Documentation
So you can access the latest API documentation offline.

Android support
Extra APIs that aren't available in the standard platform.

Google Play Billing
Allows you to integrate billing services in your app.



Appearance & Behavior › System Settings › Android SDK

Manager for the Android SDK and Tools used by Android Studio

Android SDK Location:

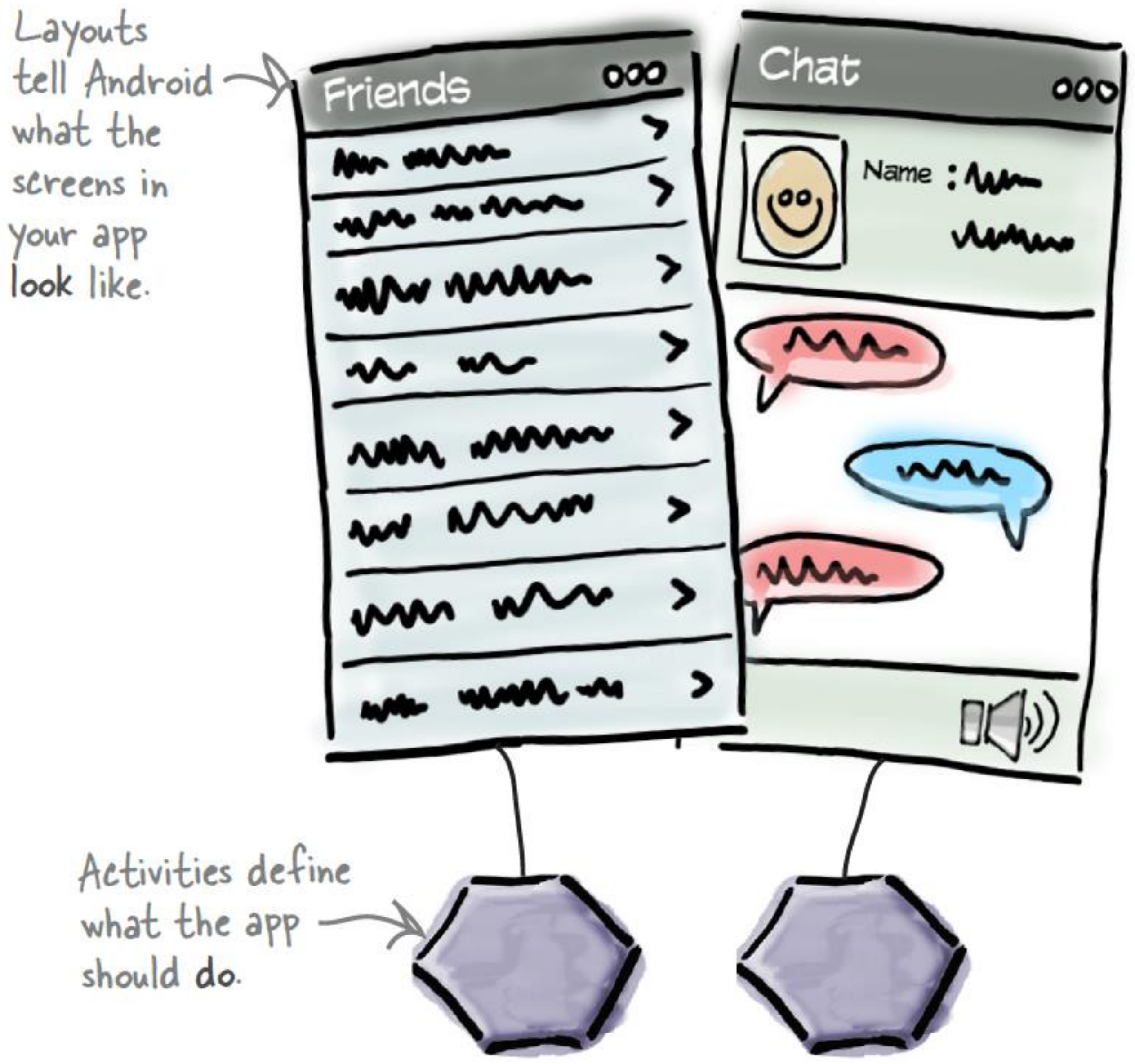
Each Android SDK Platform package includes the Android platform and sources pertaining to an API level by default. Once installed, Android Studio will automatically check for updates. Check "show package details" to display individual SDK components.

Name	API Level	Revision	Status
☐ Android N Preview	N	2	Not installed
✓ Android 6.0 (Marshmallow)	23	3	Installed
☐ Android 5.1 (Lollipop)	22	2	Not installed
☐ Android 5.0 (Lollipop)	21	2	Not installed
☐ Android 4.4 (KitKat Wear)	20	2	Not installed
☐ Android 4.4 (KitKat)	19	4	Not installed
☐ Android 4.3 (Jelly Bean)	18	3	Not installed
☐ Android 4.2 (Jelly Bean)	17	3	Not installed
☐ Android 4.1 (Jelly Bean)	16	5	Not installed
☐ Android 4.0.3 (IceCreamSandwich)	15	5	Not installed
☐ Android 4.0 (IceCreamSandwich)	14	4	Not installed

☐ Show Package Details

So what makes an Android App?

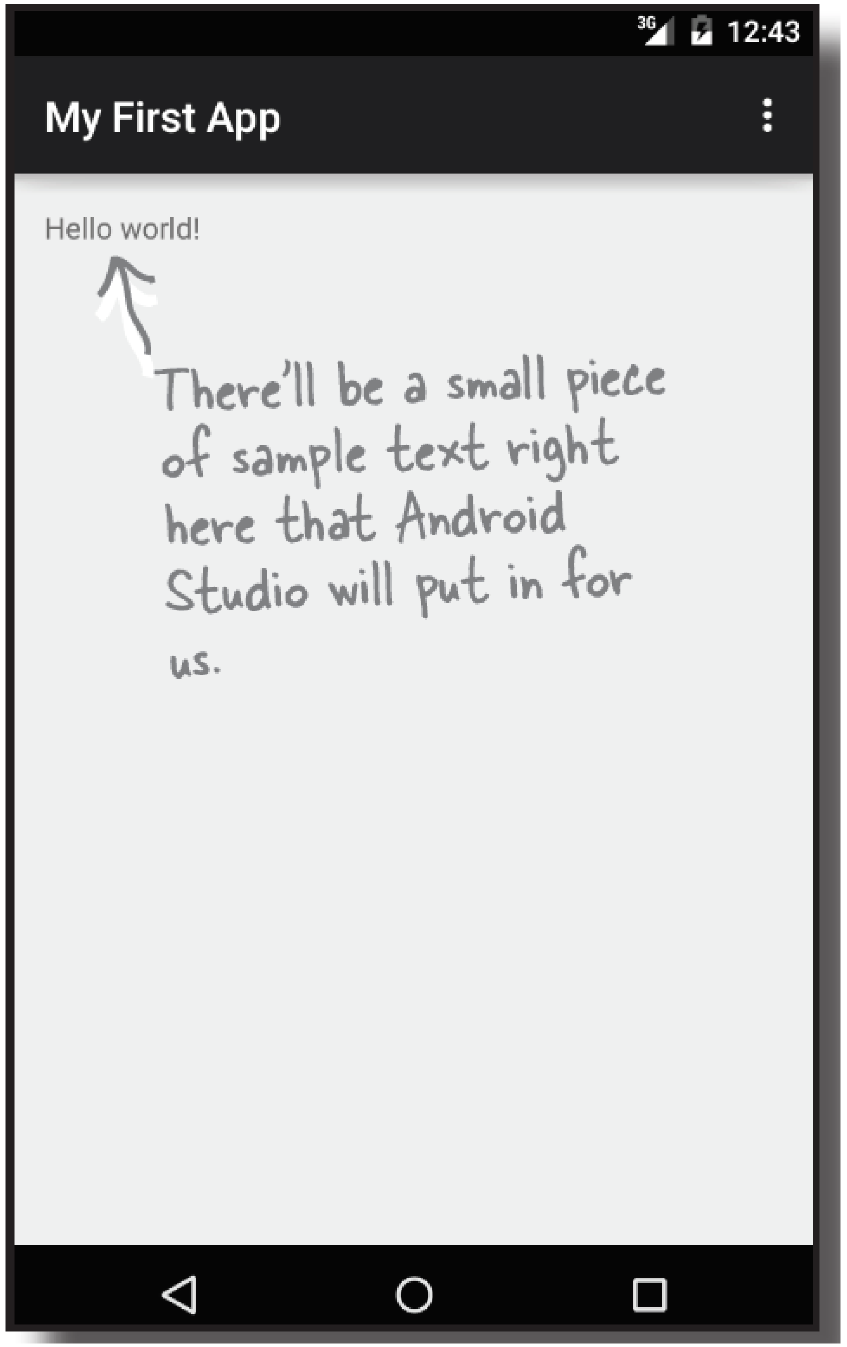
- A typical Android app is comprised of one or more **screens (Activities)**.
- You define what each screen looks like using a **layout** to define its appearance.
- Layouts are usually defined using XML, and can include GUI components such as buttons, text fields, and labels.
- **Java** on the other hand defines what each activity should do.
- For example, if some activity has a button, Java will respond to that button click.
- Extra **resources** (images, data) can be added too.
- Your App is just a bunch of **files** and **directories**!!


Layouts tell Android what the screens in your app look like.
Friends
Chat
Name :
Activities define what the app should do.

(Griffiths, 2015)

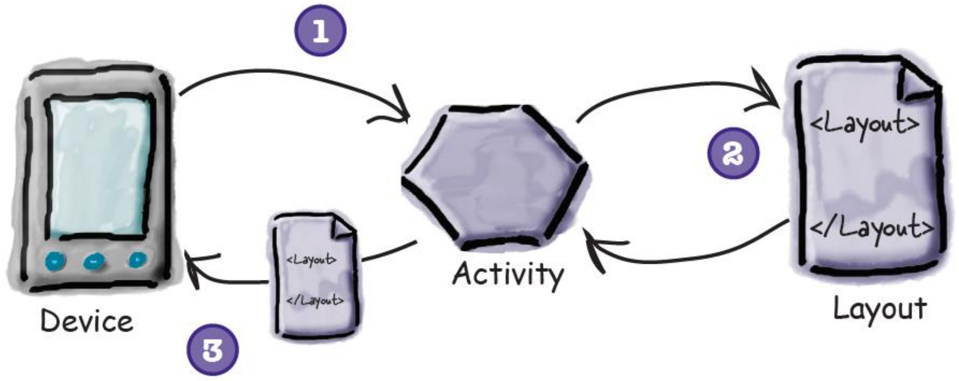
Example App I: Building a basic app Demo

OBJECTIVES:
1 – ANDROID STUDIO TEST DRIVE
2 – SETUP AVD
3 – CLOSE LOOK AT ACTIVITIES AND LAYOUTS
4 – INTRODUCE PROJECT STRUCTURE


12:43
My First App
Hello world!
There'll be a small piece of sample text right here that Android Studio will put in for us.

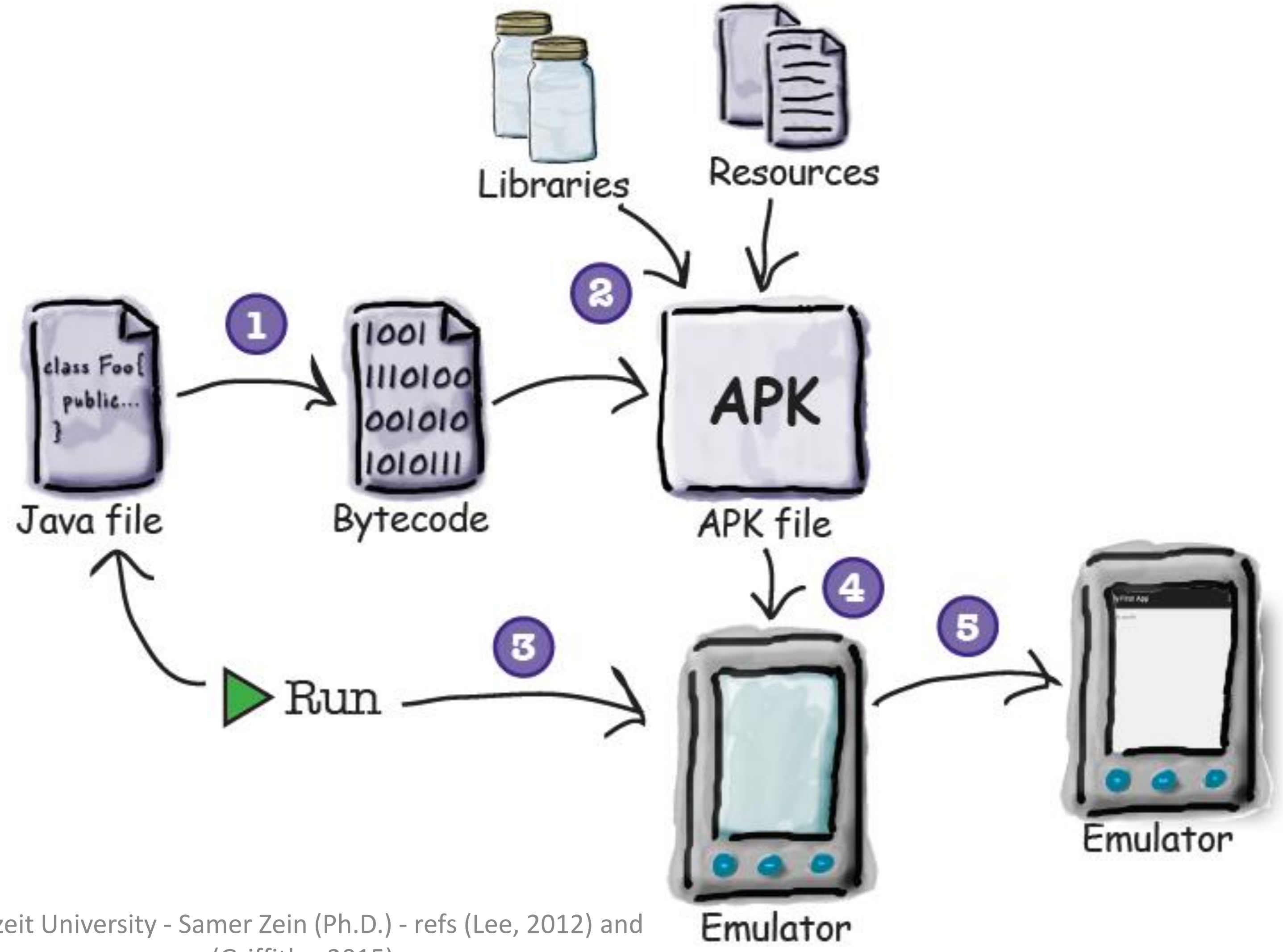
Activities and Layouts Intro

1. The device launches your app and creates an activity object.
2. The activity object specifies a layout.
3. The activity tells Android to display the layout on screen.
4. The user interacts with the layout that's displayed on the device.
5. The activity responds to these interactions by running application code.



Compile, package, deploy and run

Choosing the Run option doesn't just run your app. It also deals with all the preliminary tasks that are needed for the app to run:



Specifying Minimum and Target API Levels

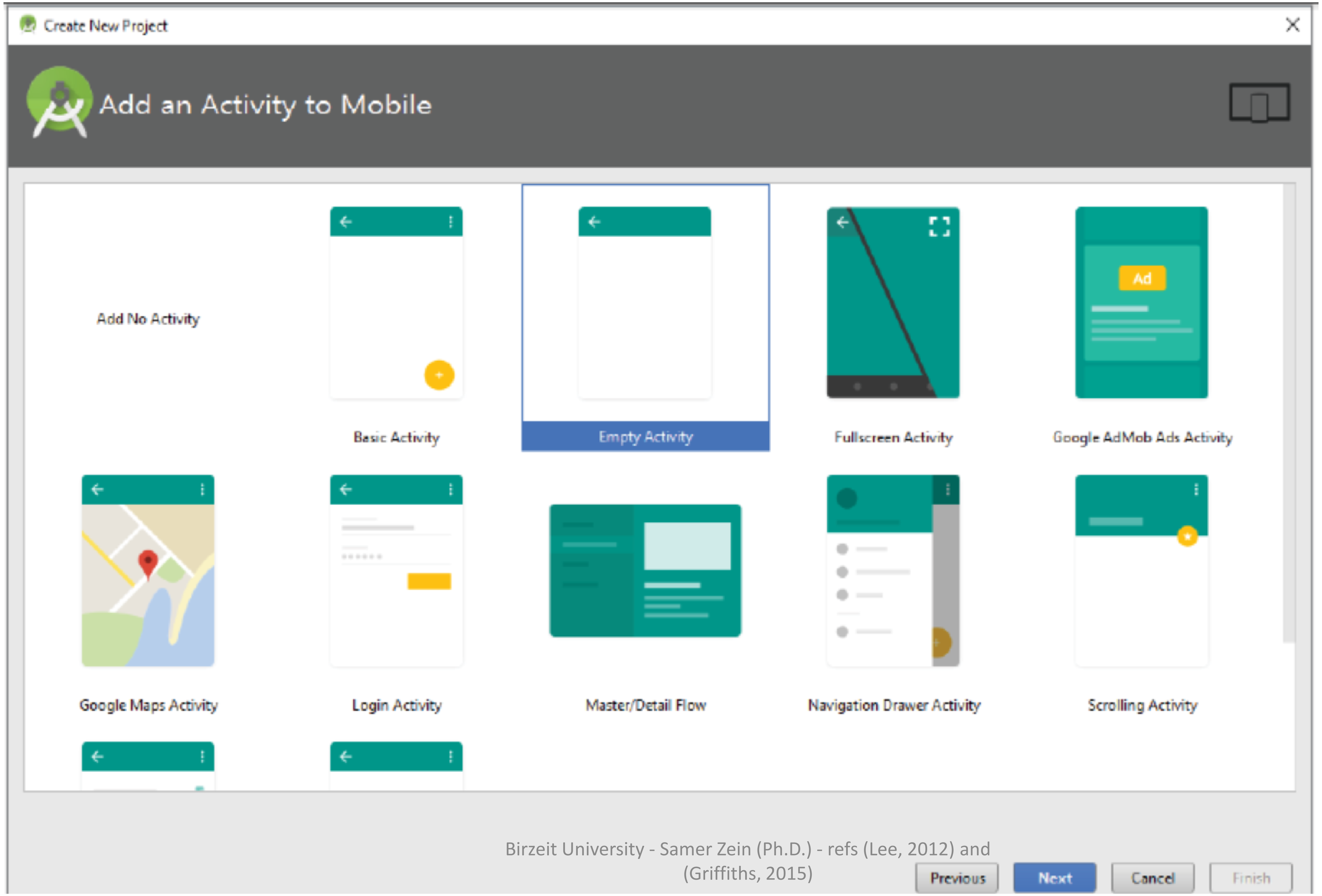
- As we will see progressively, **AndroidManifest.xml** is the main configuration file for your project. It specifies details about your App.
- Though **AndroidManifest.xml**, you need to specify which versions of Android it supports.
- Specifically, the **minSdkVersion** and **targetSdkVersion** attributes for the **<uses-sdk>** element identify the **lowest API level** with which your app is compatible
- and the **highest API level** against which you've designed and tested your app.

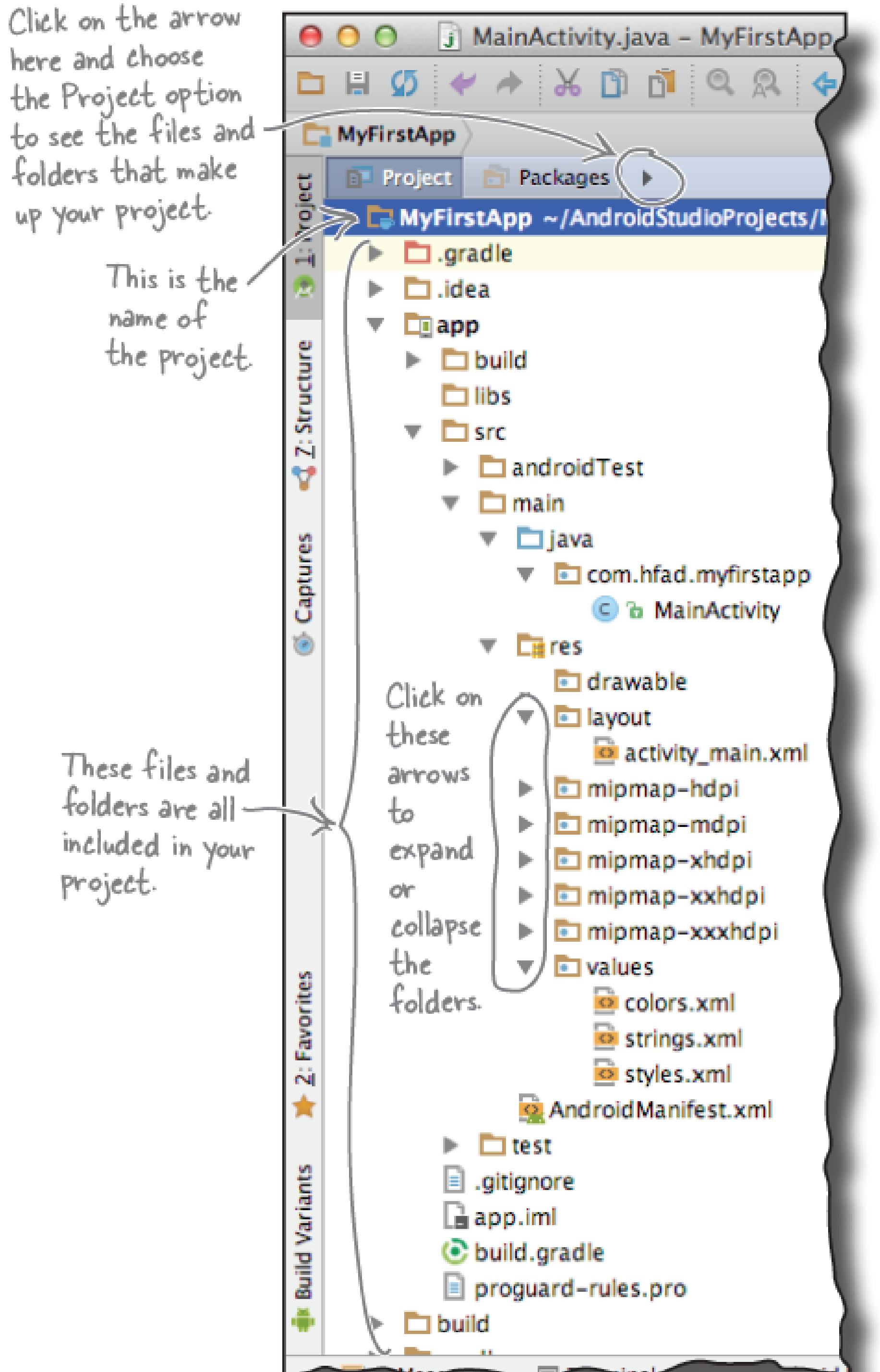
Specifying Minimum and Target API Levels..2

- As new versions of Android are released, some style and behaviors may change.
- To allow your app to take advantage of these changes and ensure that your app fits the style of each user's device,
- you should set the **targetSdkVersion** value to match the latest Android version available

More on Activities

- One of the **Main** building blocks of Android Apps
- Android apps can have zero or more **Activities**.
- Activity is like a GUI, but has to serve one action.
- The purpose of **Activity** is to interact with users.
- From the moment an activity appears on the screen to the moment it is hidden, it goes through a number of stages.
- These stages are known as an activity's life cycle.
- Understanding the life cycle of an activity is vital to ensuring that your application works correctly.
- Additionally, there are Fragments
- **Fragments** can be seen as "miniature" activities that can be groups together.


Create New Project
Add an Activity to Mobile
Add No Activity
Basic Activity
Empty Activity
Fullscreen Activity
Google AdMob Ads Activity
Ad
Google Maps Activity
Login Activity
Master/Detail Flow
Navigation Drawer Activity
Scrolling Activity
Previous
Next
Cancel
Finish



Click on the arrow here and choose the Project option to see the files and folders that make up your project.

This is the name of the project.

These files and folders are all included in your project.

Click on these arrows to expand or collapse the folders.

```
MainActivity.java - MyFirstApp
MyFirstApp
Project   Packages
MyFirstApp ~/AndroidStudioProjects/
  ▶ .gradle
  ▶ .idea
  ▼ app
      ▶ build
        libs
      ▼ src
          ▶ androidTest
          ▼ main
              ▼ java
                  ▼ com.hfad.myfirstapp
                        MainActivity
              ▼ res
                    drawable
                  ▼ layout
                        activity_main.xml
                  ▶ mipmap-hdpi
                  ▶ mipmap-mdpi
                  ▶ mipmap-xhdpi
                  ▶ mipmap-xxhdpi
                  ▶ mipmap-xxxhdpi
                  ▼ values
                        colors.xml
                        strings.xml
                        styles.xml
                AndroidManifest.xml
          ▶ test
        .gitignore
        app.iml
        build.gradle
        proguard-rules.pro
  ▶ build
```

The folder structure includes different types of files

If you browse through the folder structure, you'll see that the wizard has created various types of files and folders for you:

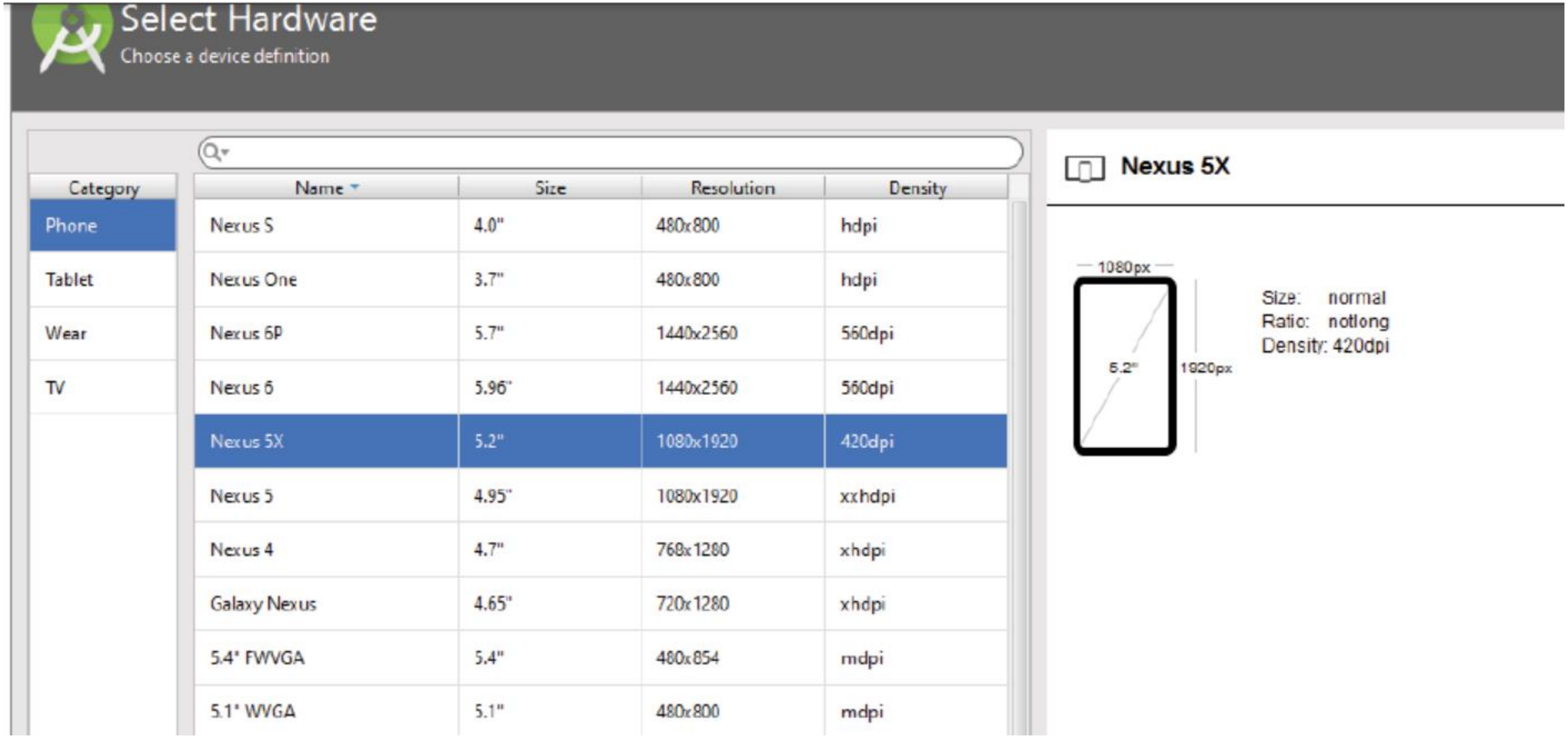
- **Java and XML source files**
  These are the activity and layout files for your app.
- **Android-generated Java files**
  There are some extra Java files you don't need to touch that Android Studio generates for you automatically.
- **Resource files**
  These include default image files for icons, styles your app might use, and any common String values your app might want to look up.
- **Android libraries**
  In the wizard, you specified the minimum SDK version you want your app to be compatible with. Android Studio makes sure your app includes the relevant Android libraries for that version.
- **Configuration files**
  The configuration files tell Android what's actually in the app and how it should run.

Let's take a closer look at some of the key files and folders in Androidville.

Android Virtual devices

- Android AVDs allow you to test your App.
- Each AVD has a hardware profile; a mapping to a system image; and emulated storage, such as a secure digital (SD) card.
- One important thing to remember about emulators is that they are not perfect.
- There are some applications that are games, or use sensors and GPS, are better tested on real device!
- But it is so easy to deploy your APK on real device using USB connection!
- <u>You need to choose a system image for an API level that's compatible with the app you're building</u>.
- As an example, if you want your app to work on a minimum of API level 19, choose a system image for *at least* API level 19

The Android emulator allows you to run your app on an Android virtual device (AVD), which behaves just like a physical Android device. You can set up numerous AVDs, each emulating a different type of device.



Select Hardware

Choose a device definition

Category
Phone
Tablet
Wear
TV

Name	Size	Resolution	Density
Nexus S	4.0"	480x800	hdpi
Nexus One	3.7"	480x800	hdpi
Nexus 6P	5.7"	1440x2560	560dpi
Nexus 6	5.96"	1440x2560	560dpi
Nexus 5X	5.2"	1080x1920	420dpi
Nexus 5	4.95"	1080x1920	xxhdpi
Nexus 4	4.7"	768x1280	xhdpi
Galaxy Nexus	4.65"	720x1280	xhdpi
5.4" FWVGA	5.4"	480x854	mdpi
5.1" WVGA	5.1"	480x800	mdpi

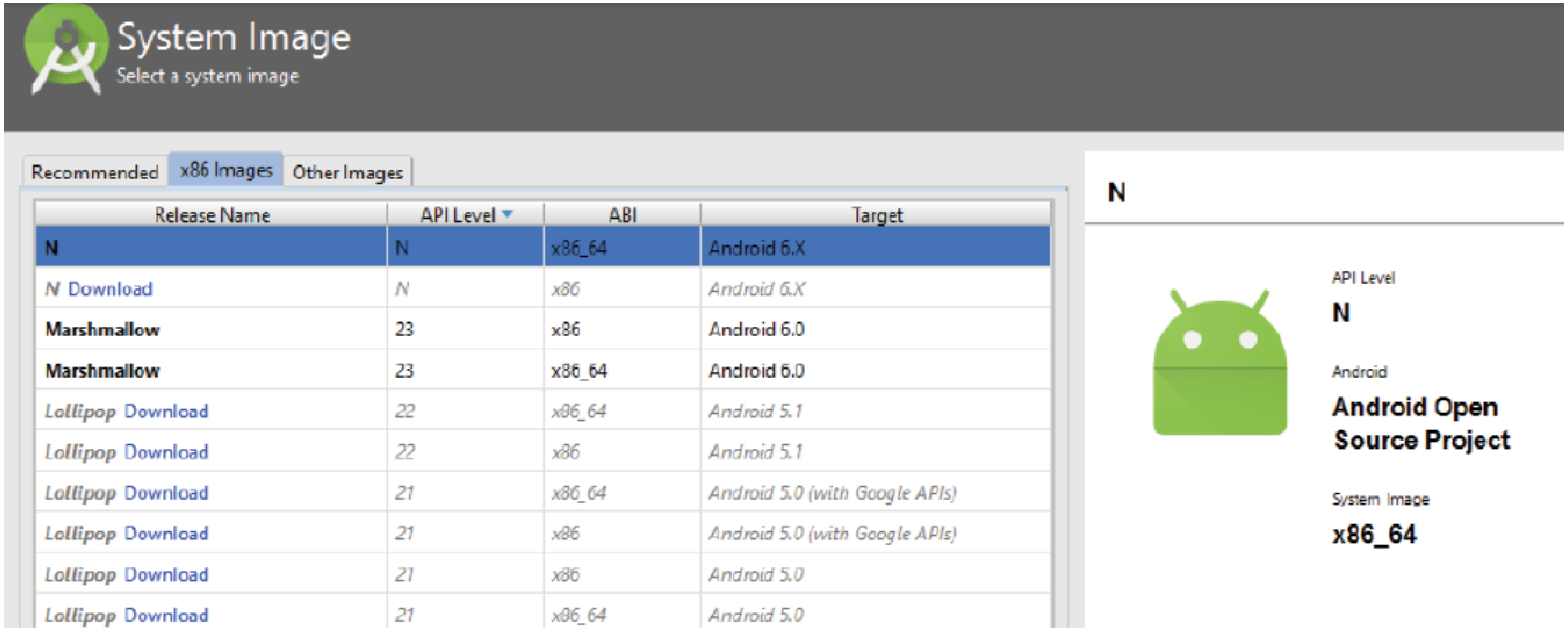
Nexus 5X

1080px

5.2"

1920px

Size: normal
Ratio: notlong
Density: 420dpi



System Image

Select a system image

Release Name	API Level	ABI	Target
N	N	x86_64	Android 6.X
N Download	N	x86	Android 6.X
Marshmallow	23	x86	Android 6.0
Marshmallow	23	x86_64	Android 6.0
Lollipop Download	22	x86_64	Android 5.1
Lollipop Download	22	x86	Android 5.1
Lollipop Download	21	x86_64	Android 5.0 (with Google APIs)
Lollipop Download	21	x86	Android 5.0 (with Google APIs)
Lollipop Download	21	x86	Android 5.0
Lollipop Download	21	x86_64	Android 5.0

N

API Level

N

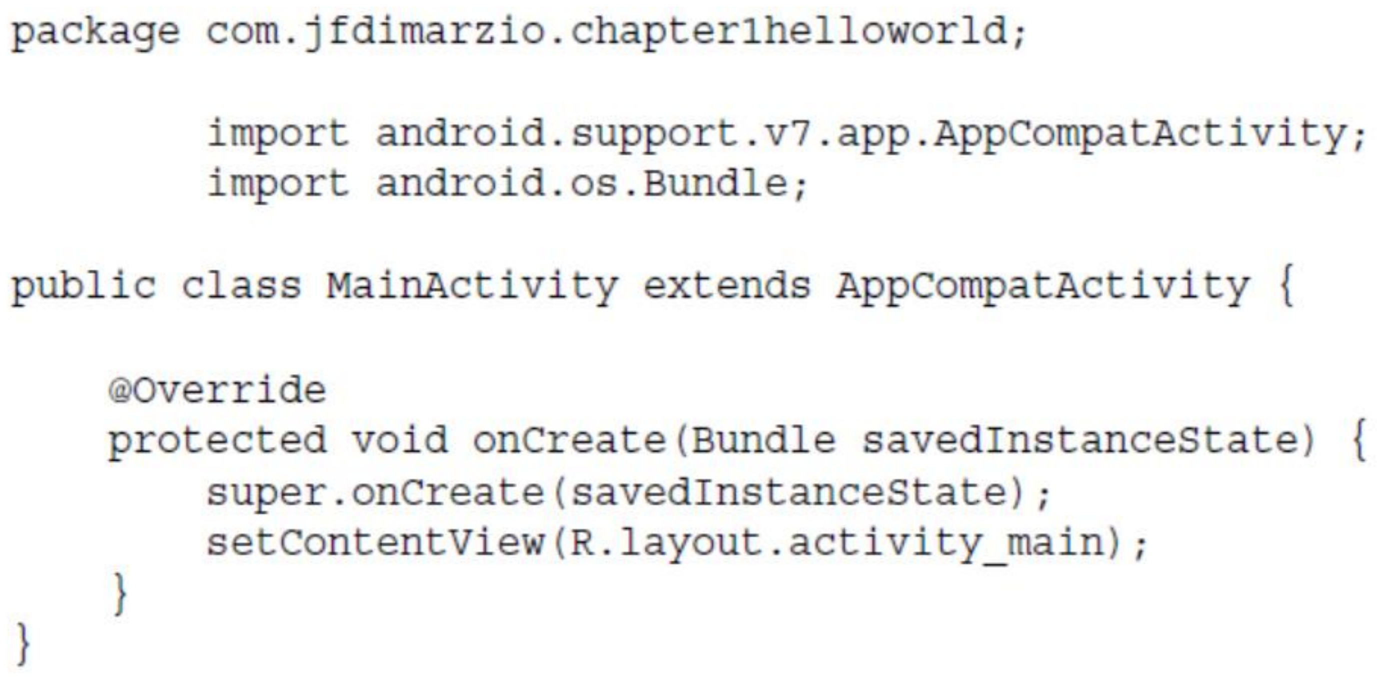
Android

Android Open Source Project

System Image

x86_64

Activities: onCreate()



```
package com.jfdimarzio.chapter1helloworld;

        import android.support.v7.app.AppCompatActivity;
        import android.os.Bundle;

public class MainActivity extends AppCompatActivity {

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

- Your activity class loads its user interface (UI) component using the XML file defined in your res/layout folder.
- Every activity you have in your application must be declared in your AndroidManifest.xml file.

Activities are different

- **Unlike programming paradigms** in which apps are launched with a main() method,
- the Android system initiates code in an Activity instance by invoking **specific callback methods.**
- These callback methods correspond to **specific stages of its lifecycle**.

Activity Purpose

- **The Activity class serves as the entry point for an app's interaction with the user, providing the window in which the app draws its UI.**
- **This window typically fills the screen, but may be smaller than the screen and float on top of other windows.**
- **You implement an activity as a subclass of the Activity class.**
- **Generally, one activity implements one screen in an app.**
- **For instance, one of an app's activities may implement the Preferences screen, while another activity implements the Compose Email screen.**

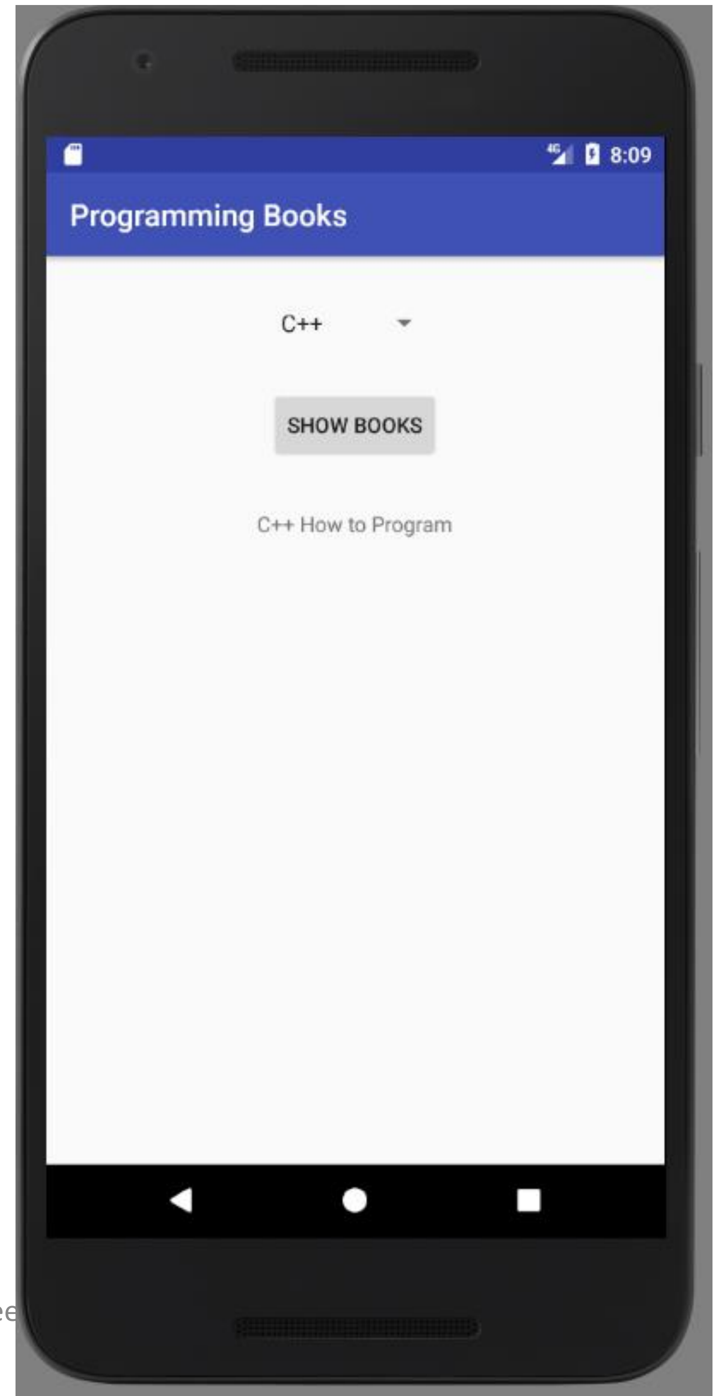
Activities are loosely coupled

- **Although activities work together to form a cohesive user experience in an app,**
- **each activity is only loosely bound to the other activities;**
- **there are usually minimal dependencies among the activities in an app.**
- **In fact, activities often start up activities belonging to other apps.**
- **For example, a browser app might launch the Share activity of a social-media app.**

Example App 2: Interactive App

Objectives:

1. Get the App to do something in response to **user action**.
2. How to get **Activity** and **Layout** talk together.
3. Get deeper on how Android actually works.
4. Introduce **LinearLayout**
5. Introduce **R**, the hidden gem.
6. Introduce **strings.xml**
7. Introduce **Mainifest.xml**


8:09
Programming Books
C++
SHOW BOOKS
C++ How to Program