COMP-304
Summer 2018

Introduction to Android Development

Objectives:

- Define Android Platform
- Explain the differences between leading mobile operating systems
- Explain Android Development Environment
- Write a simple Android application using Android Studio

What is Android

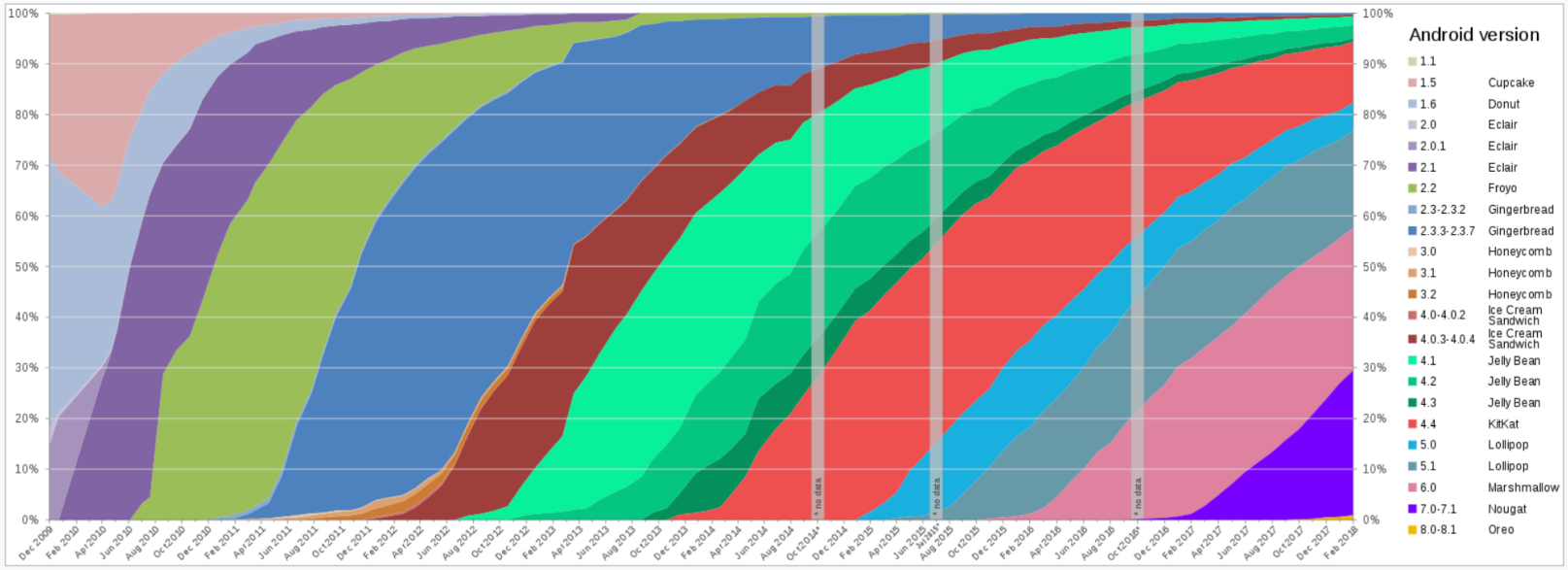
- Android is an open source software stack for a wide range of mobile devices and a corresponding open source project led by Google..
- **Andy Rubin** has been credited as the father of the Android platform.
  - His company, Android Inc., was acquired by Google in 2005
- Android is an **open source** platform
  - No need to pay royalties or license fees to develop for the platform.
- Google and the Open Handset Alliance announced in 2008 the availability of the Android platform source code to everyone, for free, under the new Android Open Source Project.

https://source.android.com/index.html

What is Android

- Google hosts the Android open source project and provides online Android documentation, tools, forums, and the Software Development Kit (SDK) for developers.
- All major Android news originates at Google.
- The company has also hosted a number of events at conferences and the Android Developer Challenge (ADC), a contest to encourage developers to write killer Android applications—for $10 million dollars in prizes to spur development on the platform.
- The winners and their apps are listed on the Android website.

Android Versions



Global Android version distribution since December 2009, as of May 2018. Android Marshmallow the oldest supported version has running on 26.0% of all Android devices accessing Google Play; Android Nougat the most popular versions on 30.8% (while Nougat 7.0 only, is a little less popular at 23.0%), and all supported including Oreo on 61.4%.

Android Version 1.1 Release Date: February 9, 2009

In 2016, Google released Android 7.0:
- split-screen multi-window mode
- redesigned notification shade
- redefined "doze" feature
- switched from JRE to OpenJDK

NOTE: each Android versions has its own features and APIs, not always backwards compatible. need to choose version that you anticipate your target audience will be using.

https://en.wikipedia.org/wiki/Android_version_history#/media/File:Android_historical_version_distribution_-_vector.svg

Android Versions

- First there was the Cupcake, Donut, Éclair, Froyo, Gingerbread, Honeycomb, Ice Cream Sandwich**,** Jelly Bean, **KitKat**, Lollipop, Marshmallow, Nougat, and **Oreo. Android Pie** was released in August 2018.
- They actually **name versions alphabetically** as a way to take note

Android Versions

Version	Code name	Release date	API level	ART/DVM	Distribution	First devices to run version
8.1	Oreo	December 5, 2017	27	ART	0.5%	Pixel, Pixel XL, Nexus 6P, Nexus 5X
8.0		August 21, 2017	26	ART	4.1%	N/A
7.1	Nougat	October 4, 2016	25	ART	7.8%	Pixel, Pixel XL
7.0		August 22, 2016	24	ART	23.0%	Nexus 5X, Nexus 6P
6.0	Marshmallow	October 5, 2015	23	ART	26.0%	
5.1	Lollipop	March 9, 2015	22	ART	18.0%	Android One
5.0		November 3, 2014	21	ART 2.1.0	4.9%	Nexus 6, Nexus 9
4.4	KitKat	October 31, 2013	19	DVM (and ART 1.6.0)	10.5%	Nexus 5
4.3	Jelly Bean	July 24, 2013	18	DVM	0.6%	Nexus 7 2013
4.2		November 13, 2012	17	DVM	2.2%	Nexus 4, Nexus 10
4.1		July 9, 2012	16	DVM	1.7%	Nexus 7
4.0	Ice Cream Sandwich	October 19, 2011	15	DVM	0.4%	Galaxy Nexus
2.3	Gingerbread	February 9, 2011	10	DVM 1.4.0	0.3%	Nexus S

Bc android is open source and freely avail to manufacturers for customization, there are no fixed hardware or software configurations.

Base OS supports many features, including:

Storage--SQLite
Connectivity
Messaging
Media Support
Hardware support
Multi-touch
Multi-tasking
Tethering

Android's web browser is based on the open source WebKit and Chrome's V8 JavaScript engine

http://en.wikipedia.org/wiki/Android_(operating_system)

Applications and downloads

Year	Month	Applications available	Downloads to date
2009	March	2,300[109]	
	December	16,000[110]	
2010	March	30,000[111]	
	April	38,000[112]	
	July	70,000[113]	
	September	80,000[114]	
	October	100,000[115]	
2011	April		3 billion[116]
	May	200,000[117]	4,5 billion[117]
	July	250,000[118]	6 billion[118]
	October	500,000[119][120]	
	December		10 billion[121]
2012	April		15 billion[122]
	June	600,000[123]	20 billion[123]
	September	675,000[124]	25 billion[124]
	October	700,000[125]	
2013	May		48 billion[126]
	July	1 million[127]	50 billion[127]
2016			82 billion[128]
2017	February	2.7 million[1]	

http://en.wikipedia.org/wiki/Android_Market

Comparisons to competitors

Worldwide Smartphone Sales to End Users by Operating System in 2017 (Thousands of Units)

Operating System	2017 Units	2017 Market Share (%)	2016 Units	2016 Market Share (%)
Android	1,320,118.1	85.9	1,268,562.7	84.8
iOS	214,924.4	14.0	216,064.0	14.4
Other OS	1,493.0	0.1	11,332.2	0.8
Total	**1,536,535.5**	**100.0**	**1,495,959.0**	**100.0**

Source: Gartner (February 2018)

https://www.gartner.com/newsroom/id/3859963

Android platform

- Android is an operating system and a software platform upon which applications are developed
- The Android platform is best described as a *stack because it is a collection of* components, including:

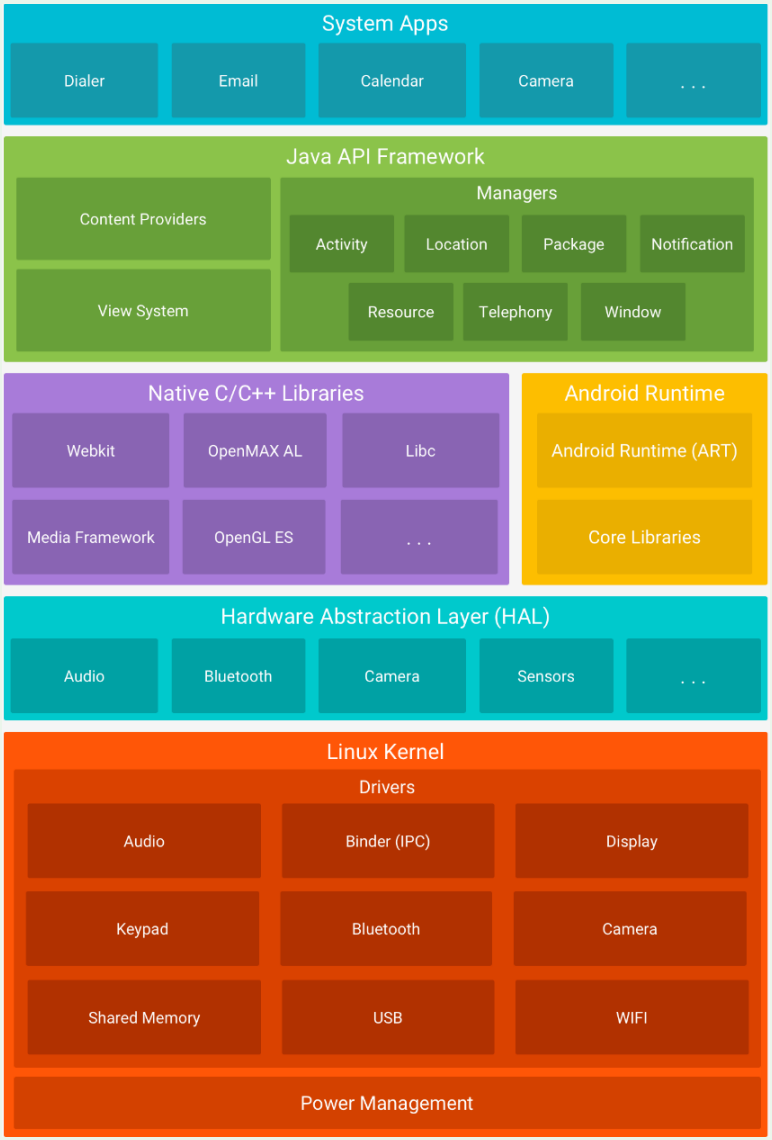
Architecture of Android:

  - **Linux** kernel-based operating system — this layer contains all the low-level device drivers
  - **Hardware Abstraction Layer** (HAL)
  - **Android Runtime** (ART) — ART located in same layer with the libs, provides set core of libs enable devs to write Android apps using Java; includes Dalvik virtual machine, enables every Android app run in its own process with its own VM. compiled into Dalvik executables, optimized for battery-powered moble devices w/limited memory and CPU power
  - **Native C/C++ Libraries** — contains the code that provides the main features of an Android OS
  - **Java API** Framework
  - **Systems Apps** — app framework exposes capabilities of Android OS to app devs so they can make use of them in their apps

- See the picture in next page for details

Android platform – Software Stack

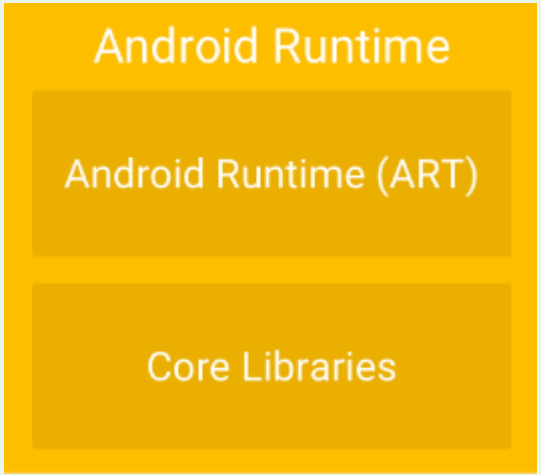
Android devices include but not limited to:
- Smartphones
- tablets
- e-reader devices
- internet tvs
- automobiles
- smartwatches



https://developer.android.com/guide/platform/

Android Runtime

- ART is written to **run multiple virtual machines on low-memory devices by executing DEX files**, a bytecode format designed specially for Android that's optimized for minimal memory footprint.



- Build toolchains, such as Jack, compile Java sources into **DEX bytecode**, which can run on the Android platform.

- Some of the major features of ART include the following:
  - **Ahead-of-time** (AOT) and **just-in-time** (JIT) compilation
  - Optimized **garbage collection** (GC)
  - Better **debugging** support

https://developer.android.com/guide/platform/

JIT versus AOT

- **Dalvik Just In Time (JIT) compiler**, every time that the app is running:
  - **dynamically translates a portion of the Dalvik bytecode into machine code and caches it**
    - uses less physical space on the device.
  - Then it **takes the next portion** and so on

- **ART Ahead Of Time (AOT) compiler**, when the app is installed on the device:
  - **statically translates the DEX bytecode into machine code** and stores in the device's storage.
    - the code executes **much faster**
    - **less battery drain** because uses native execution uses CPU less than JIT

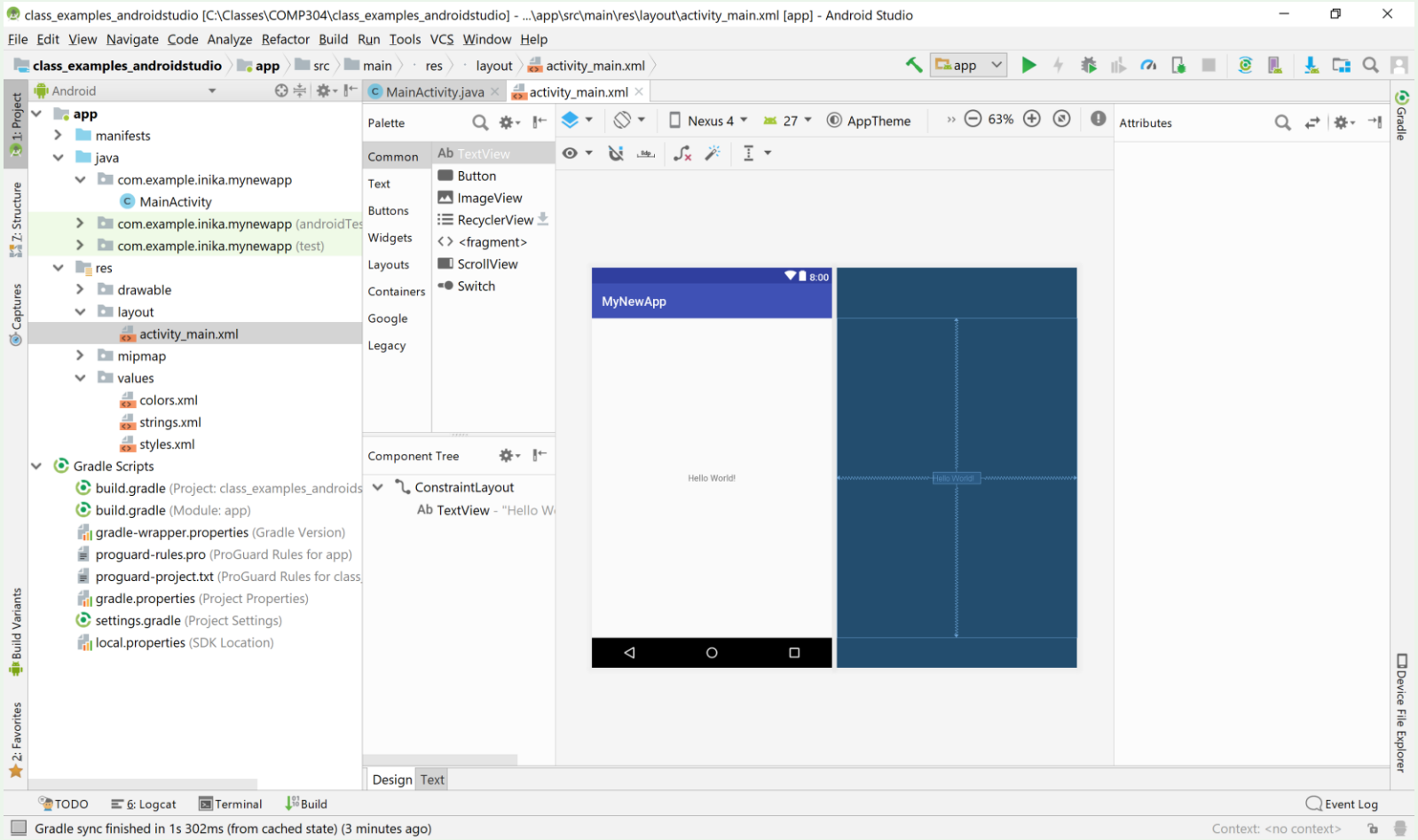
Application Fundamentals

- Android apps can be written using Kotlin, Java, and C++ languages.
- The Android SDK tools compile your code along with any data and resource files into an APK, an **Android package**, which is an archive file with an **.apk** suffix.
- One APK file contains all the contents of an Android app and is **the file that Android-powered devices use to install the app**.
- By default, **every app runs in its own Linux process**.
- Each process has its own virtual machine (VM), so an **app's code runs in isolation from other apps**.

https://developer.android.com/guide/components/fundamentals

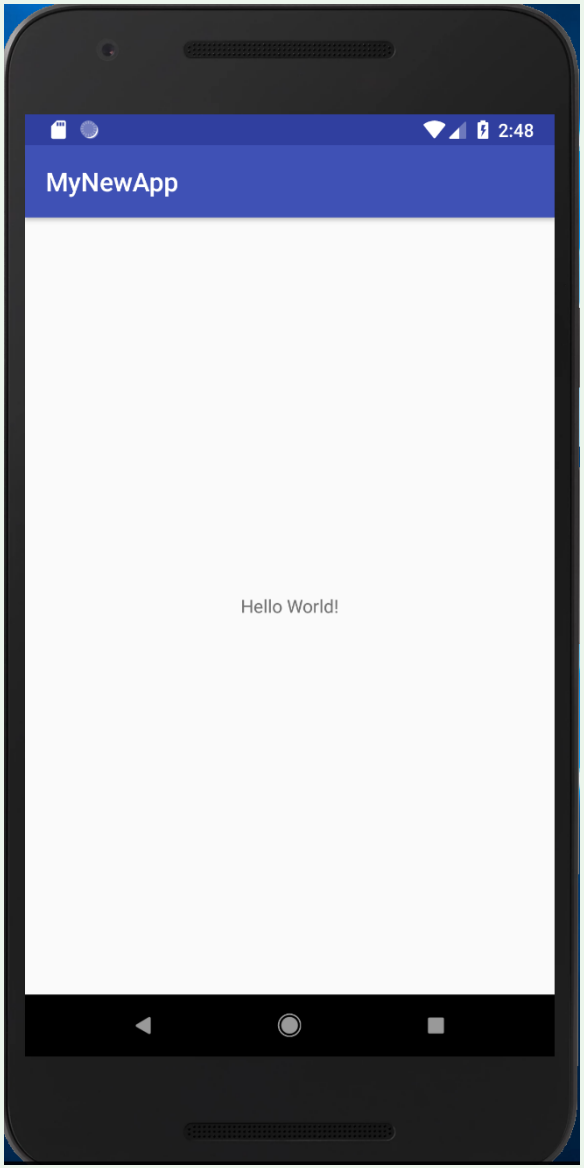
Development Environment

- **Android Studio** is the official Android IDE, based on IntelliJ IDEA.



Android emulator

- The Android **emulator**, is one of the most important tools provided with the Android SDK.
- You will use this tool frequently when designing and developing Android applications.
- The emulator runs on your computer and behaves much as a **mobile device** would.
- You can **load** Android applications into the emulator, **test**, and **debug** them.

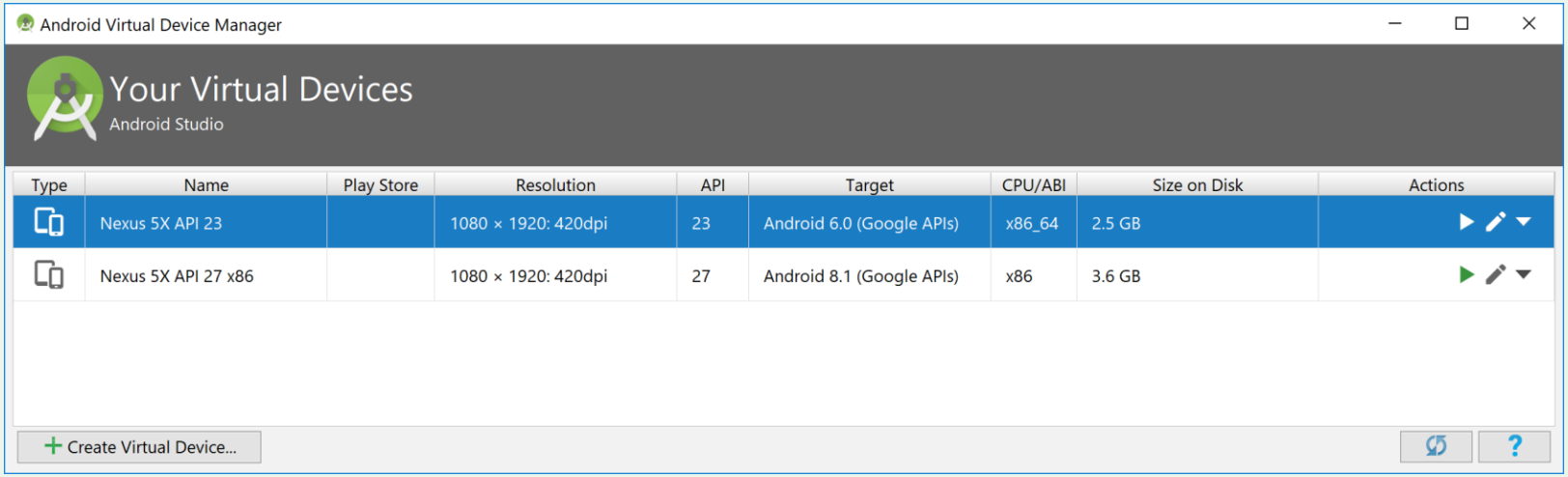


AVD Manager

- The AVD Manager is a tool you can use to create and manage Android virtual devices (AVDs), which define device configurations for the Android Emulator.

- To launch the AVD Manager:
  - In Android Studio, select Tools > Android > AVD Manager, or click the AVD Manager icon in the toolbar.

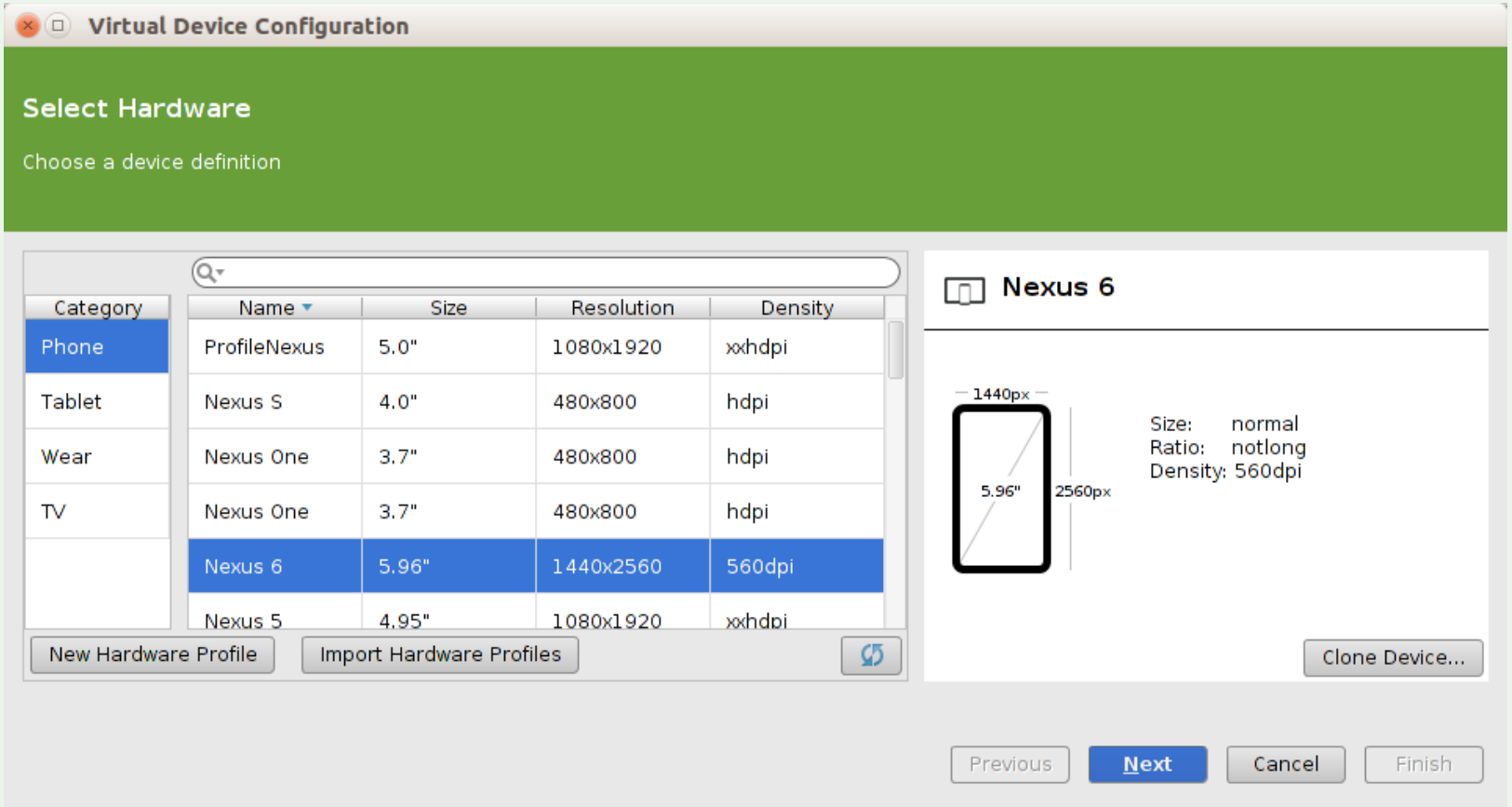
AVD Manager


Android Virtual Device Manager
Your Virtual Devices
Android Studio
Type
Name
Play Store
Resolution
API
Target
CPU/ABI
Size on Disk
Actions
Nexus 5X API 23
1080 × 1920: 420dpi
23
Android 6.0 (Google APIs)
x86_64
2.5 GB
Nexus 5X API 27 x86
1080 × 1920: 420dpi
27
Android 8.1 (Google APIs)
x86
3.6 GB
Create Virtual Device...

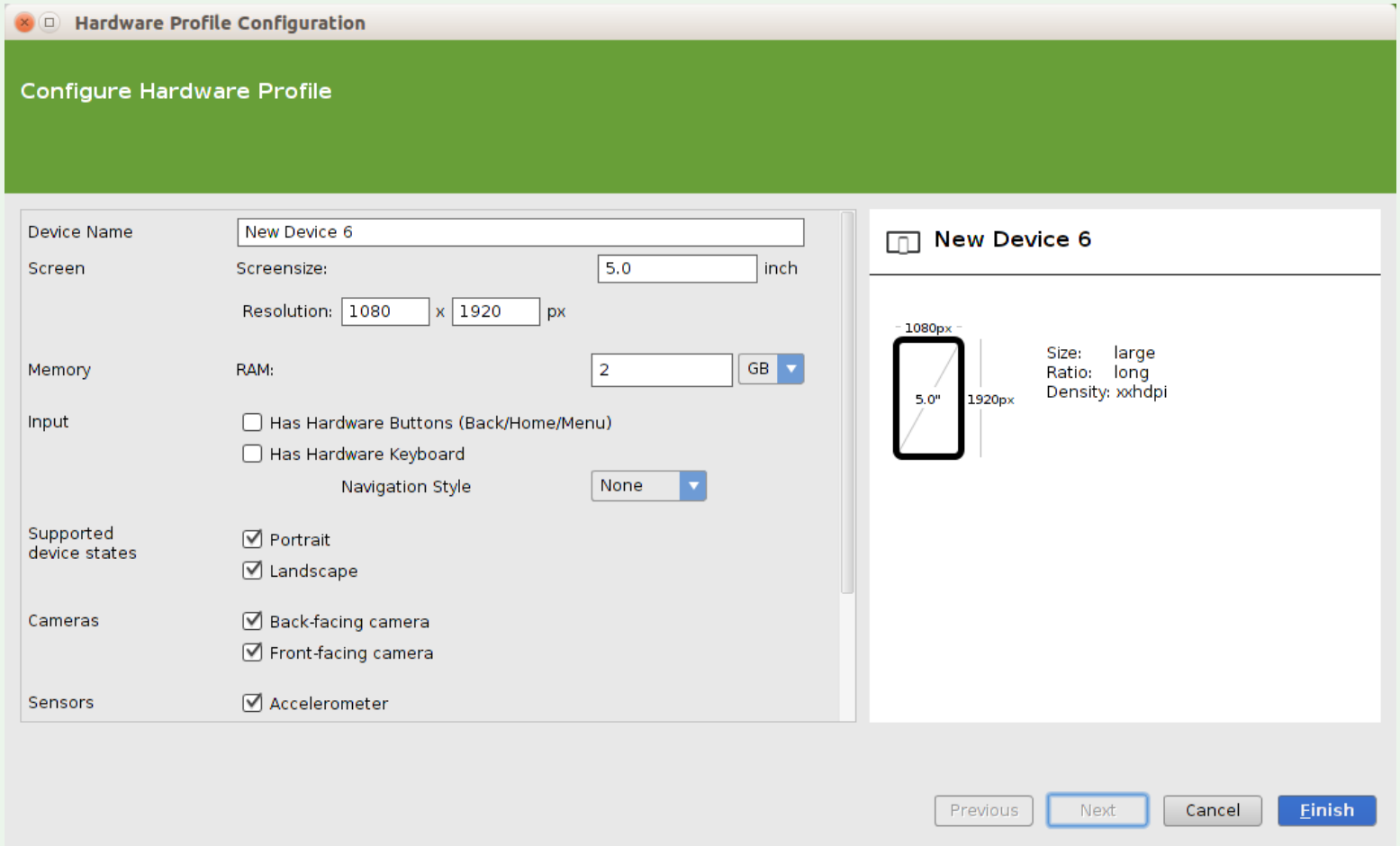
Creating an AVD

- You can create as many AVDs as you would like to use with the Android Emulator.
- To effectively test your app, you should create an AVD that models each device type for which you have designed your app to support.
  - For instance, you should create an AVD for each API level equal to and higher than the minimum version you've specified in your manifest <uses-sdk> tag.
- To create an AVD based on an existing device definition:
  - From the main screen (figure 1), click Create Virtual Device.
  - In the Select Hardware window, select a device configuration, such as Nexus 6, then click Next.

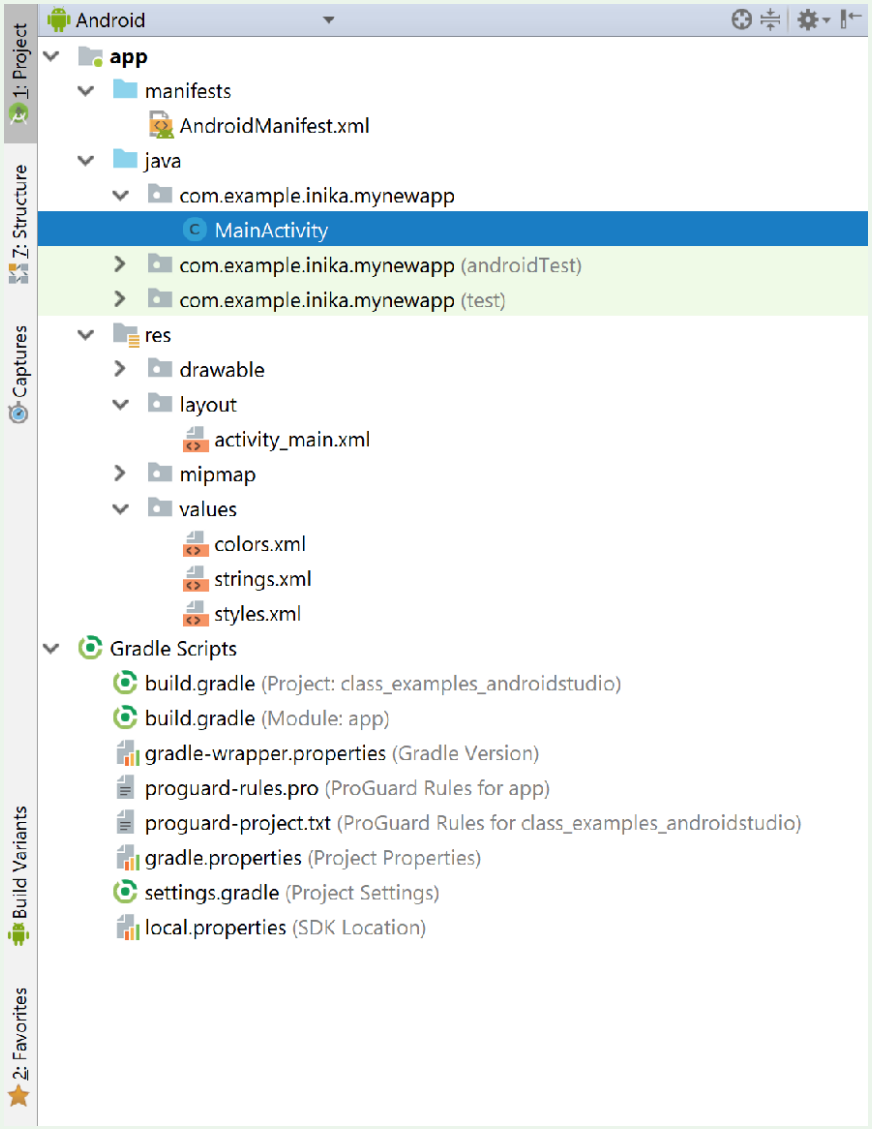
Select Hardware Window



Creating a Custom Device Configuration



Android Studio Project Structure



Developing Android Applications

```
public class FirstAndroidApp extends Activity {
   /** Called when the activity is first created. */
   @Override
   public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);
    }
}
```

Developing Android Applications

- Notice that the class is based on the **Activity** class.
  - An Activity is a single application entity that is used to perform actions.
- An application may have many separate activities, but the user interacts with them **one at a time**.
- The **onCreate()** method will be called by the Android system when your Activity starts — it is where you should perform all **initialization** and **UI setup**.
- An activity is not required to have a user interface, but usually will.

Developing Android Applications

- Running the application:

Developing Android Applications

- **Construct the UI:**

```
import android.app.Activity;
import android.os.Bundle;
import android.widget.TextView;

public class FirstAndroidApp extends Activity {
   /** Called when the activity is first created. */
   @Override
   public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);
      //
      TextView tv = (TextView)findViewById(R.id.text_message);
      tv.setText("Android is Cool!");
   }
}
```

Developing Android Applications

- An Android user interface is composed of hierarchies of objects called Views.
- A **View** is a drawable object used as an element in your UI layout, such as a button, image, or (in this case) a text label.
- Each of these objects is a subclass of the **View** class and the subclass that handles text is **TextView**

Developing Android Applications

- Running the application:

Developing Android Applications

- Google Tutorial: https://developer.android.com/training/basics/firstapp/creating-project.html

Application Architecture

- An Android application consists of one or more of the following classifications:
- **Activities** - An activity represents a single screen with a user interface..
  - When a user selects an application from the home screen or application launcher, an activity is started.
- **Services** - A service is a **component that runs in the background** to perform long-running operations or to perform work for remote processes.

Application architecture

- **Content providers** - A content provider **manages a shared set of app data**.
  - You can store the data in the file system, an SQLite database, on the web, or any other persistent storage location your app can access.
  - Through the content provider, other apps can query or even modify the data
- **Broadcast receivers** - A broadcast receiver is a component that **responds to system-wide broadcast announcements**.
  - Many broadcasts originate from the system - for example, a broadcast announcing that the screen has turned off, the battery is low, or a picture was captured.
  - Apps can also initiate broadcasts.

Application Configuration

- An Android application, along with a file called **AndroidManifest.xml**, is deployed to a device.
  - **AndroidManifest.xml** contains the necessary configuration information to properly install it to the device.
  - It includes the required **class names** and **types of events** the application is able to process, and the **required permissions** the application needs to run.
  - For example, if an application requires access to the network — to download a file, for example — this permission must be explicitly stated in the manifest file.
  - Many applications may have this specific permission enabled.
  - Such **declarative security** helps reduce the likelihood that a rogue application can cause damage on your device.

References

- Textbook
- Android Documentation
  - https://developer.android.com/guide/components/fundamentals.html
  - http://developer.android.com/tools/studio/index.html
  - https://developer.android.com/training/basics/firstapp/creating-project.html