MOBILE APPLICATION DEVELOPMENT

ANDROID OVERVIEW

MOBILE COMPUTING BACKGROUND

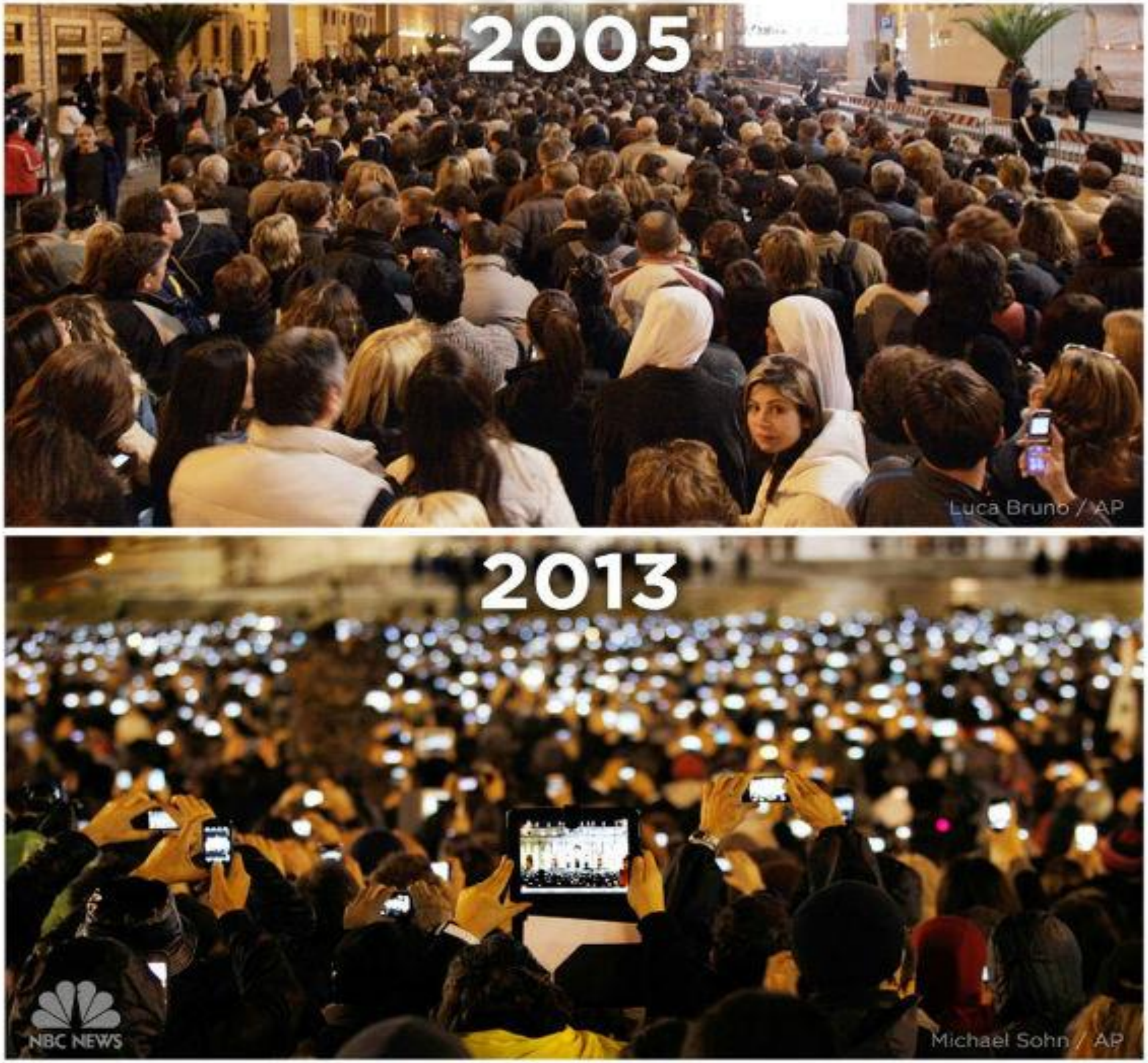
MOBILE BY THE NUMBERS

7.7 billion people on the planet

8.9 billion mobile connections

5 billion unique mobile phone users

Ref: https://www.bankmycell.com/blog/how-many-phones-are-in-the-world



2005

Luca Bruno / AP

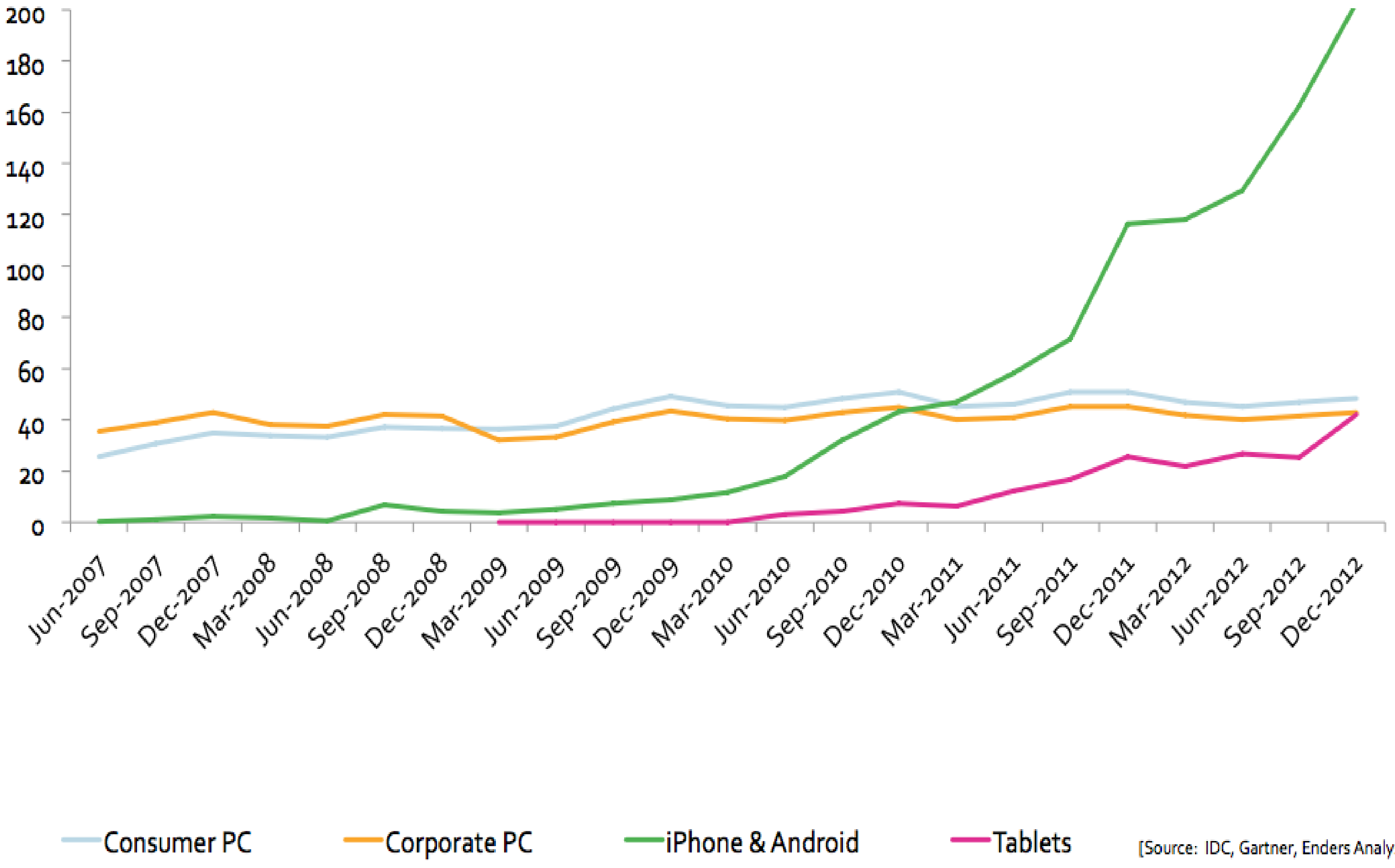
2013

NBC NEWS

Michael Sohn / AP

MOBILE DEVICES VS PC SALES

Quarterly unit sales (m)

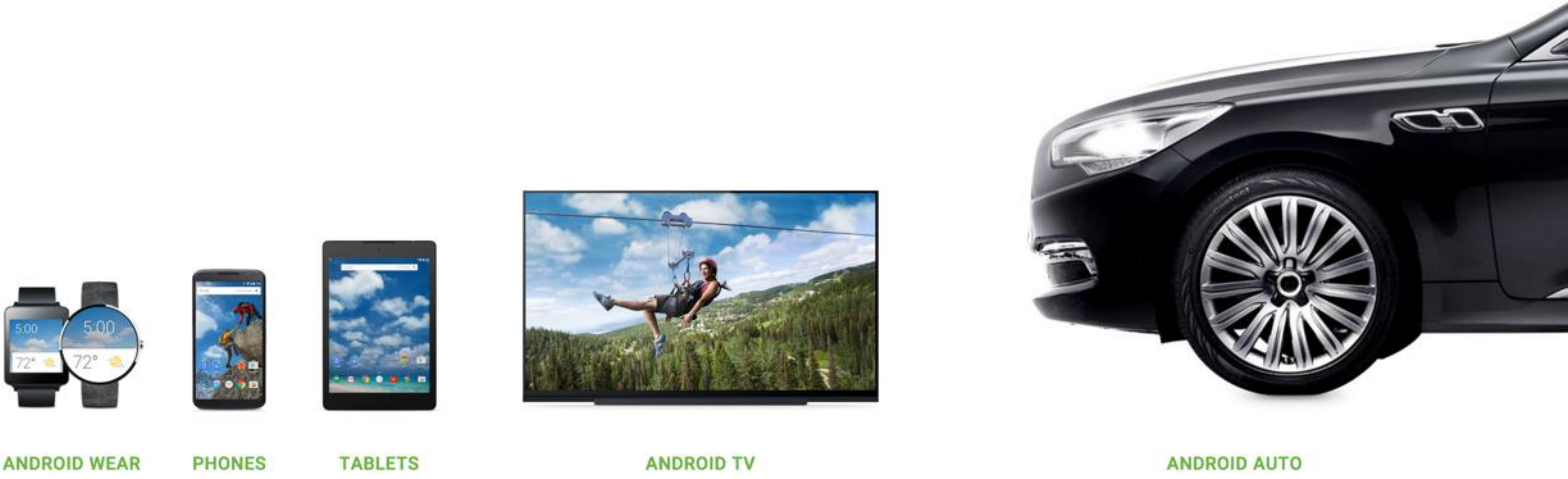


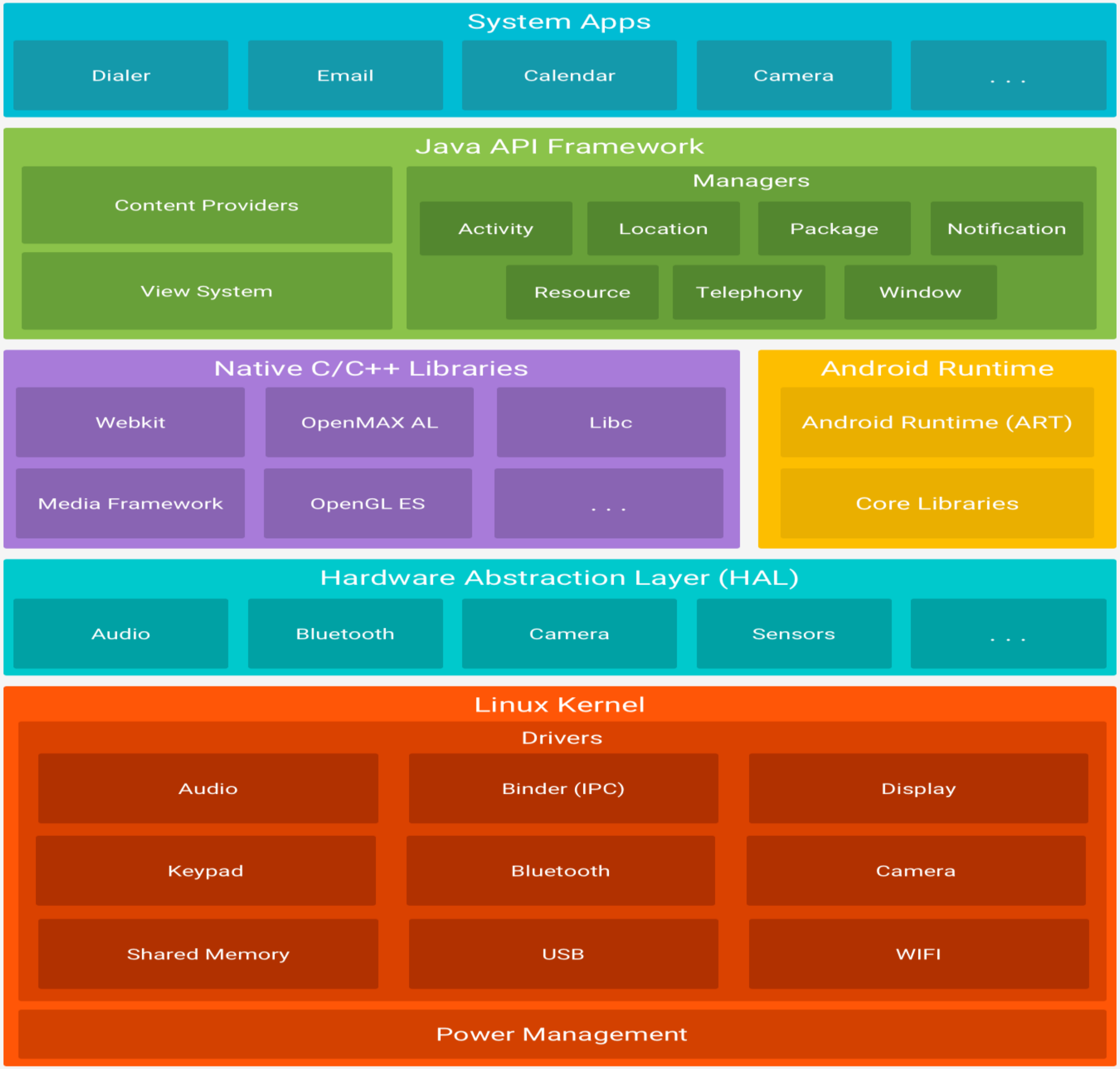
[Source: IDC, Gartner, Enders Analysis]

WHAT IS ANDROID?

- A software stack for mobile devices that includes
  - A free, open-source OS
  - An open-source development platform for creating apps
  - Key Applications
- Uses Linux to provide core system services
  - Security
  - Memory management
  - Process management
  - Power management
  - Hardware drivers

ANDROID FOR DIFFERENT DEVICES





System Apps

Dialer | Email | Calendar | Camera | . . .

Java API Framework

Content Providers

View System

Managers

Activity | Location | Package | Notification

Resource | Telephony | Window

Native C/C++ Libraries

Webkit | OpenMAX AL | Libc

Media Framework | OpenGL ES | . . .

Android Runtime

Android Runtime (ART)

Core Libraries

Hardware Abstraction Layer (HAL)

Audio | Bluetooth | Camera | Sensors | . . .

Linux Kernel

Drivers

Audio | Binder (IPC) | Display

Keypad | Bluetooth | Camera

Shared Memory | USB | WIFI

Power Management

See: An Overview of the Android Architecture (Android Studio).pdf and https://developer.android.com/guide/platform/

ANDROID FEATURES

- **Application framework** enabling reuse and replacement of components
- **Integrated browser** based on the open source WebKit engine
- **Optimized graphics** powered by a custom 2D graphics library; 3D graphics based on the OpenGL ES 1.0 specification (hardware acceleration optional)
- **SQLite** for structured data storage
- **Media support** for common audio, video, and still image formats (MPEG4, H.264, MP3, AAC, AMR, JPG, PNG, GIF)
- **GSM Telephony** (hardware dependent)
- **Bluetooth, EDGE, 3G, and WiFi** (hardware dependent)
- **Camera, GPS, compass, and accelerometer** (hardware dependent)
- **Rich development environment** including a device emulator, tools for debugging, memory and performance profiling, and a plugin for the Eclipse IDE

ANDROID VERSIONING

- On the order of 29 versions in 11 years.
- Slowing down, current pace is one large, major release a year
  - will this slow down more?
- Android releases have a code name, version number, and API level
- Most recent:
  - Pie, Version 9, API level 28
  - Android Q, Version 10, API level 29
- https://developer.android.com/preview/

A SHORT HISTORY OF ANDROID

- 2001 Palm Kyocera 6035, combing PDA and phone
  - PDA = personal data assistant, PalmPilot
- 2003 - Blackberry smartphone released
- 2005
  - Google acquires startup Android Inc. to start Android platform.
  - Work on Dalvik VM begins
- 2007
  - Open Handset Alliance announced
  - Early look at SDK
  - June, iPhone released
- 2008
  - Google sponsors 1st Android Developer Challenge
  - T-Mobile G1 announced, released fall
  - SDK 1.0 released
  - Android released open source (Apache License)
  - Android Dev Phone 1 released

SHORT HISTORY CONT.

- 2009
  - SDK 1.5 (Cupcake) after Alpha and Beta
    - New soft keyboard with “autocomplete” feature
  - SDK 1.6 (Donut)
    - Support Wide VGA
  - SDK 2.0/2.0.1/2.1 (Eclair)
    - Revamped UI, browser
- 2010
  - Nexus One released to the public
  - SDK 2.2 (Froyo)
    - Flash support, tethering
  - SDK 2.3 (Gingerbread)
    - UI update, system-wide copy-paste

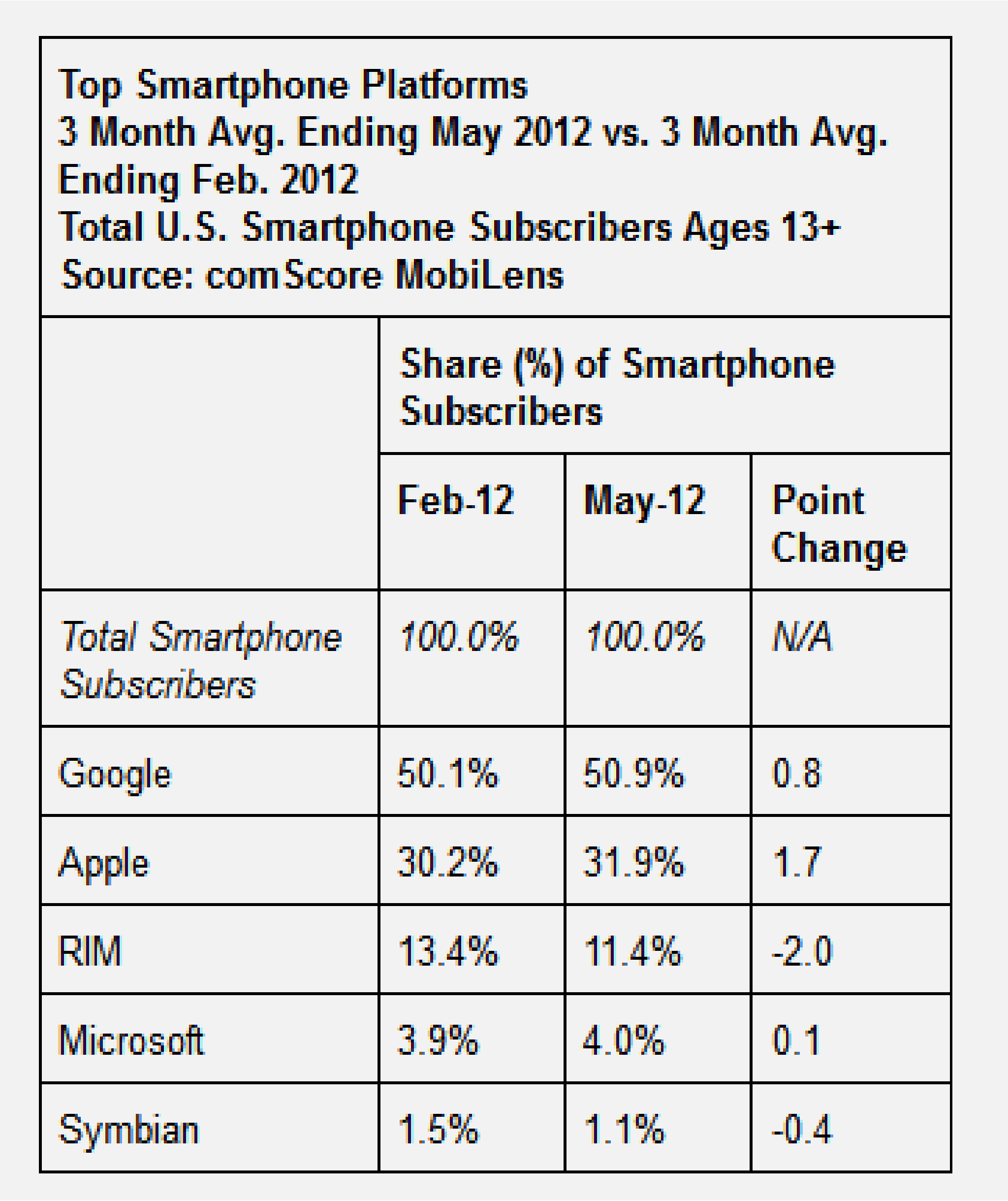
https://en.wikipedia.org/wiki/Android_version_history

SHORT HISTORY CONT.

- 2011
  - SDK 3.0 (Honeycomb) for tablets only
    - New UI for tablets, support multi-core processors, fragments
  - SDK 3.1 and 3.2
    - Hardware support and UI improvements
  - SDK 4.0 (Ice Cream Sandwich)
    - For Q4, combination of Gingerbread and Honeycomb

SHORT HISTORY CONT.

- 2012
  - Android 4.1, "Jelly Bean" released in July
- 2013
  - Android 4.4, KitKat released October 31, 2013



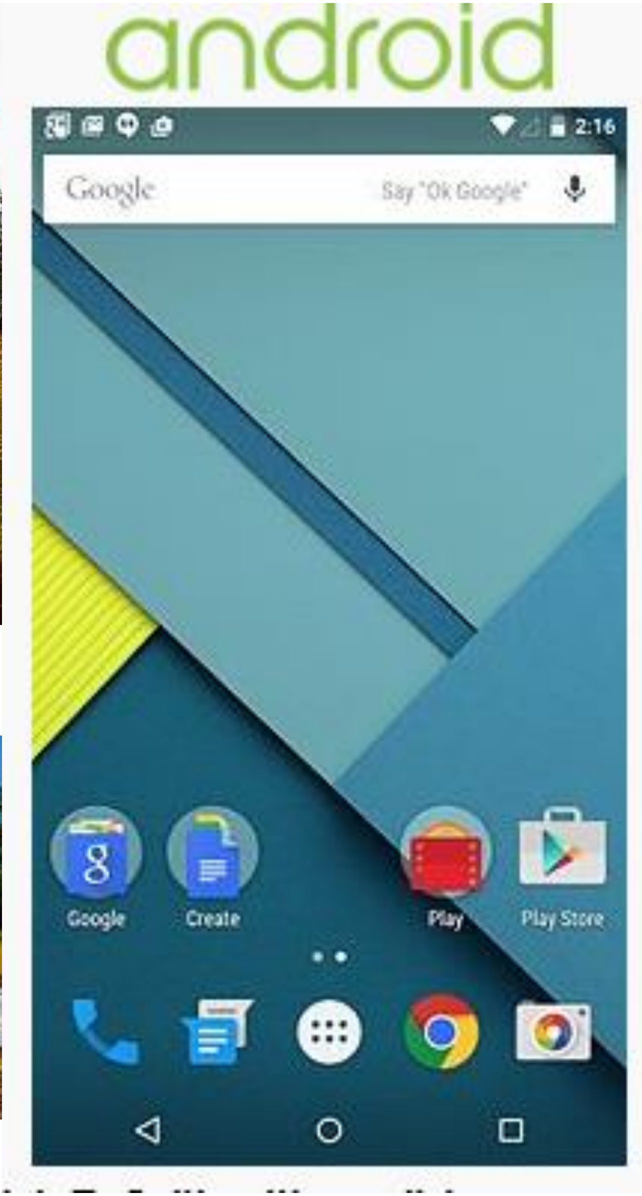
Top Smartphone Platforms
3 Month Avg. Ending May 2012 vs. 3 Month Avg. Ending Feb. 2012
Total U.S. Smartphone Subscribers Ages 13+
Source: comScore MobiLens

	Share (%) of Smartphone Subscribers		
	Feb-12	**May-12**	**Point Change**
Total Smartphone Subscribers	*100.0%*	*100.0%*	*N/A*
Google	50.1%	50.9%	0.8
Apple	30.2%	31.9%	1.7
RIM	13.4%	11.4%	-2.0
Microsoft	3.9%	4.0%	0.1
Symbian	1.5%	1.1%	-0.4

SHORT HISTORY (GETTING LONGER)

- November, 2014
  Android 5.0 Lollipop released.
  API level 21
  "Material Design"

- October, 2015
  Android 6.0 Marshmallow
  API level 23



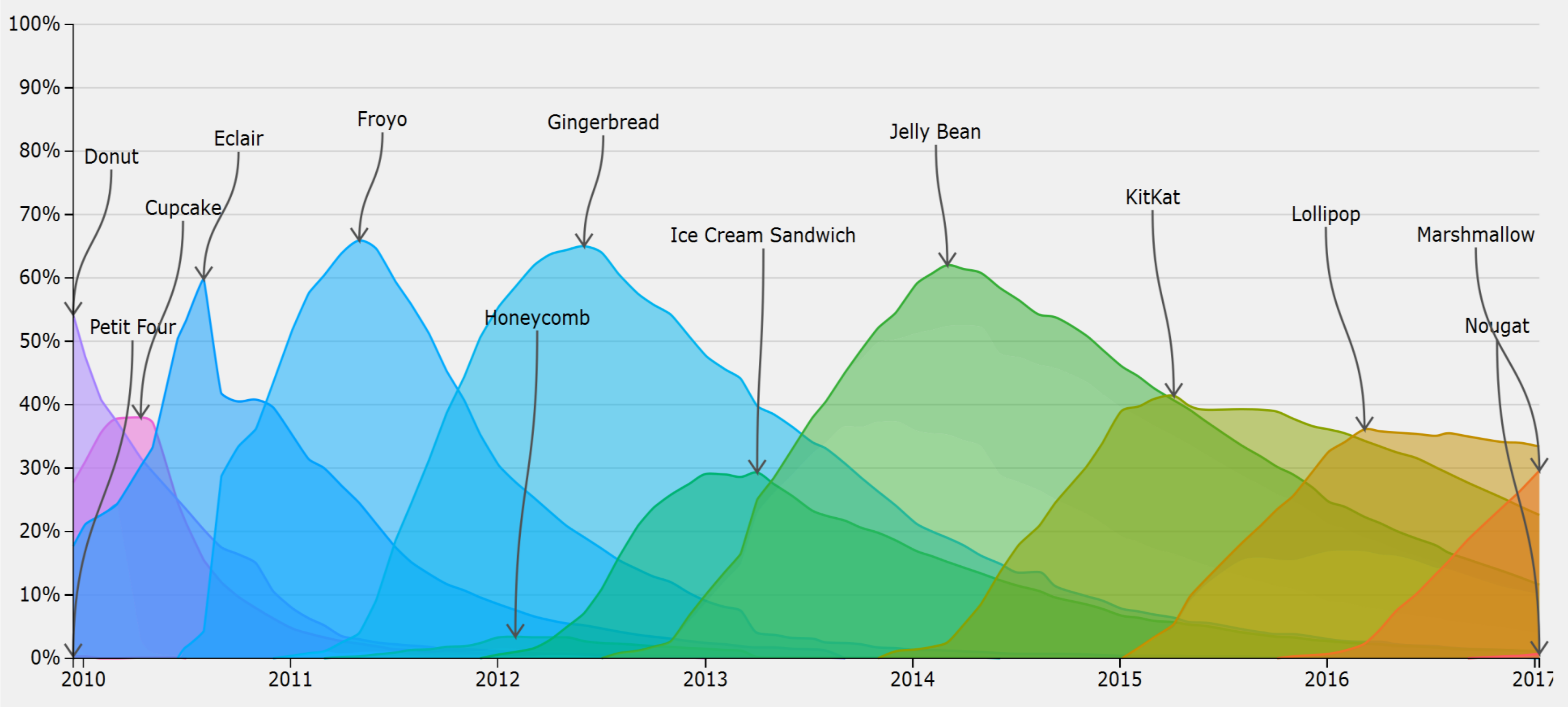
STILL MORE

- August 2016
  - Nougat
  - Daydream Virtual Reality Interface
  - Doze functionality to improve battery life
- August 2017
  - Oreo
  - Jetpack, tools for building apps, common libraries and frameworks

https://blog.xamarin.com/understanding-androids-doze-functionality/

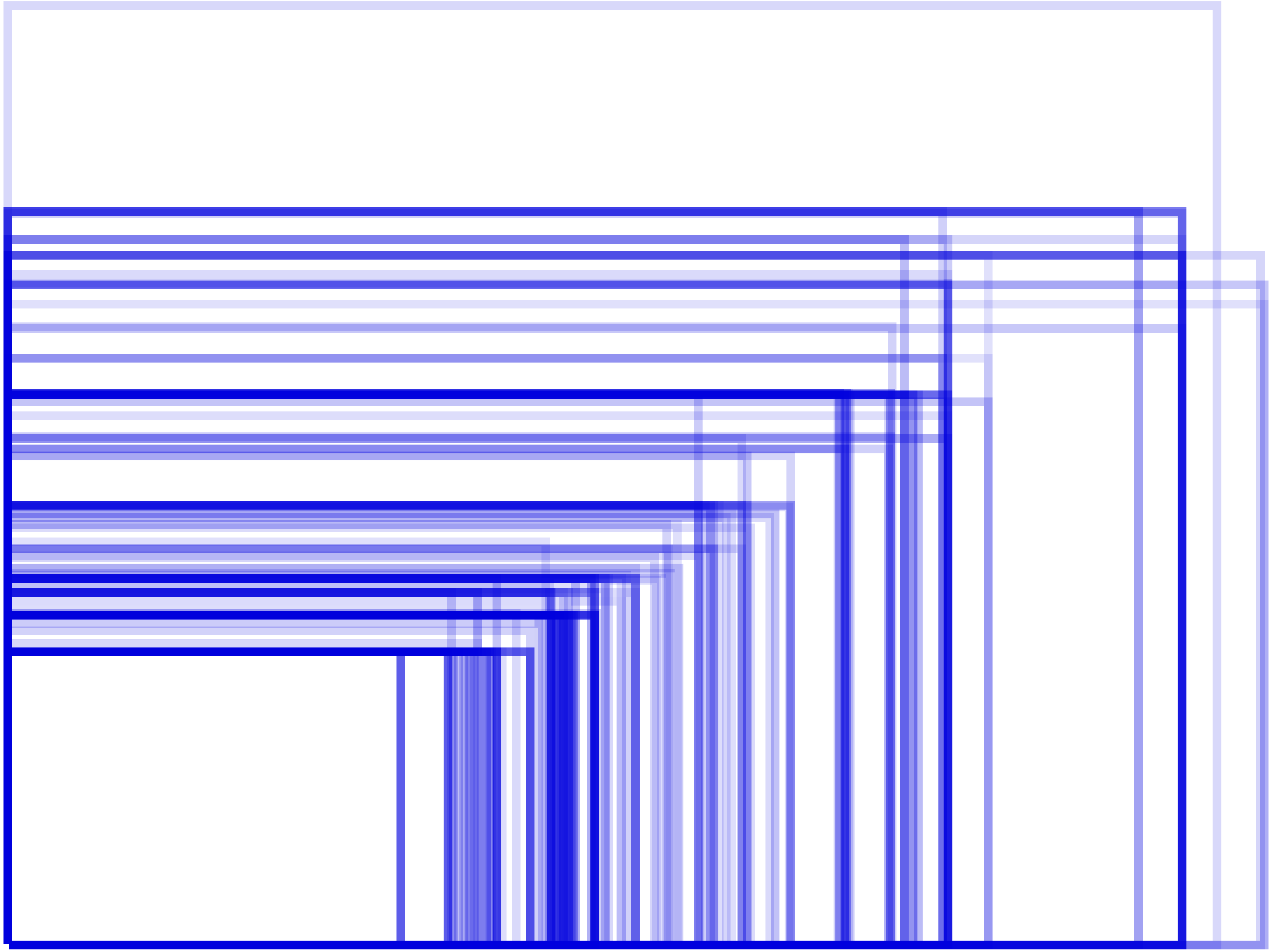
https://developer.android.com/jetpack/

DOMINANT VERSION

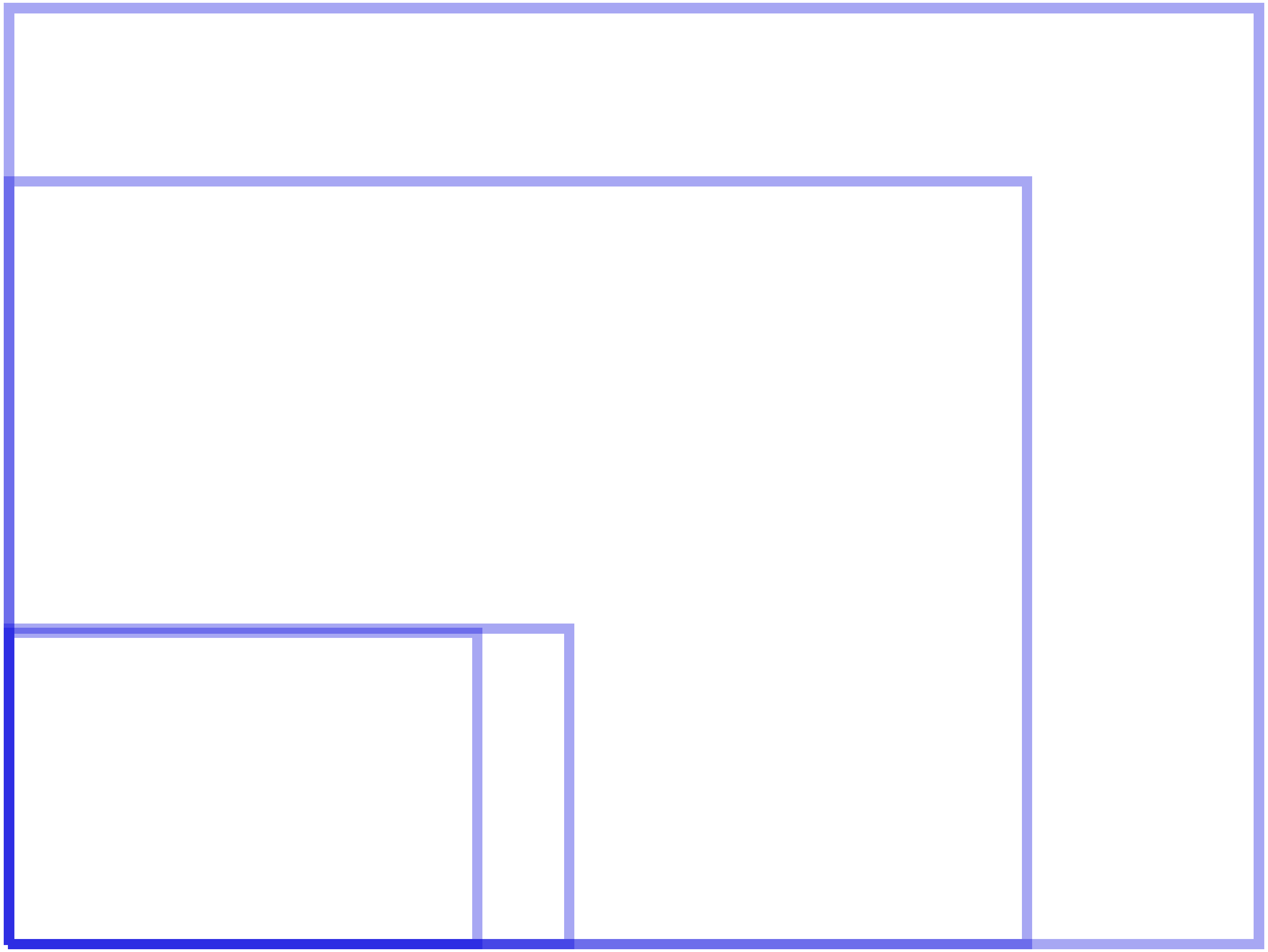


- http://www.bidouille.org/misc/androidcharts

ANDROID SCREEN SIZES - AUGUST 2014

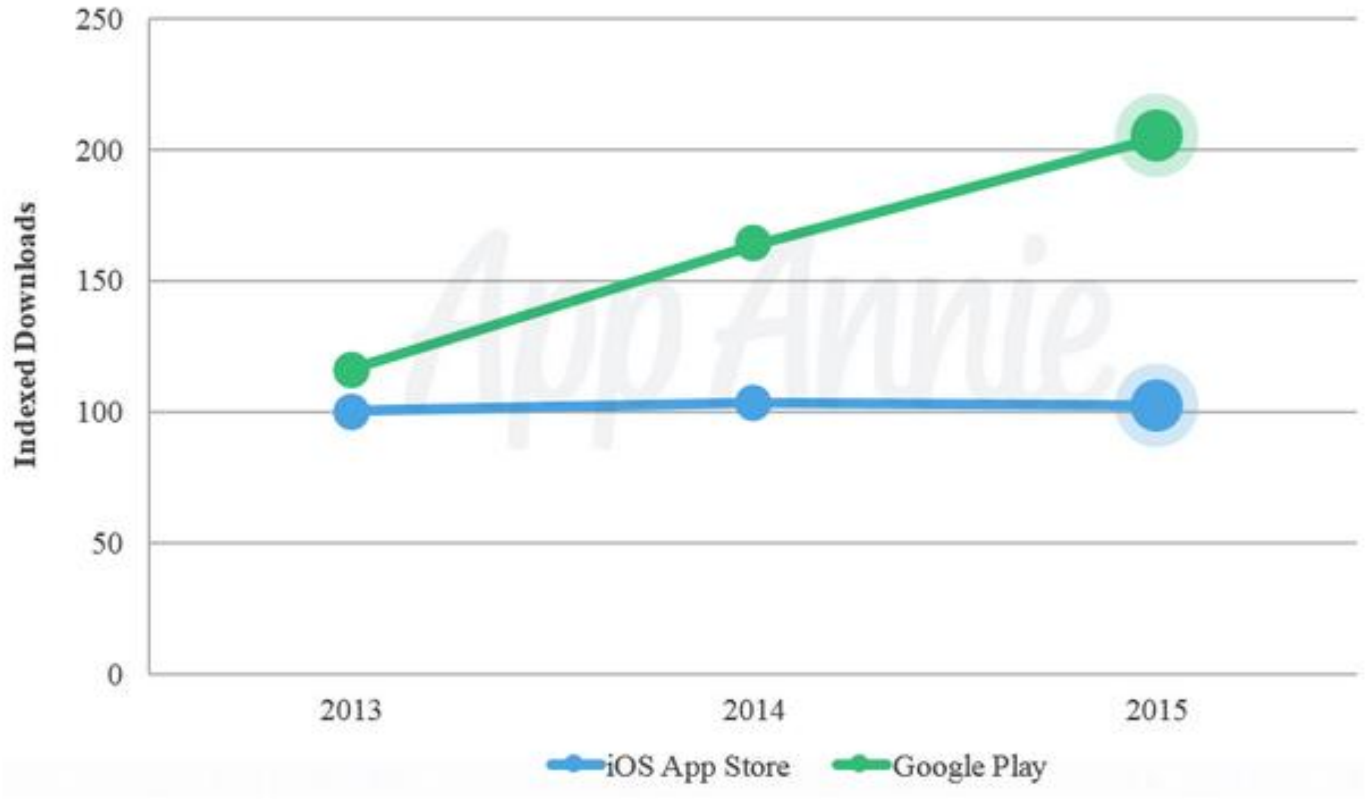


IOS SCREEN SIZES - AUGUST 2014



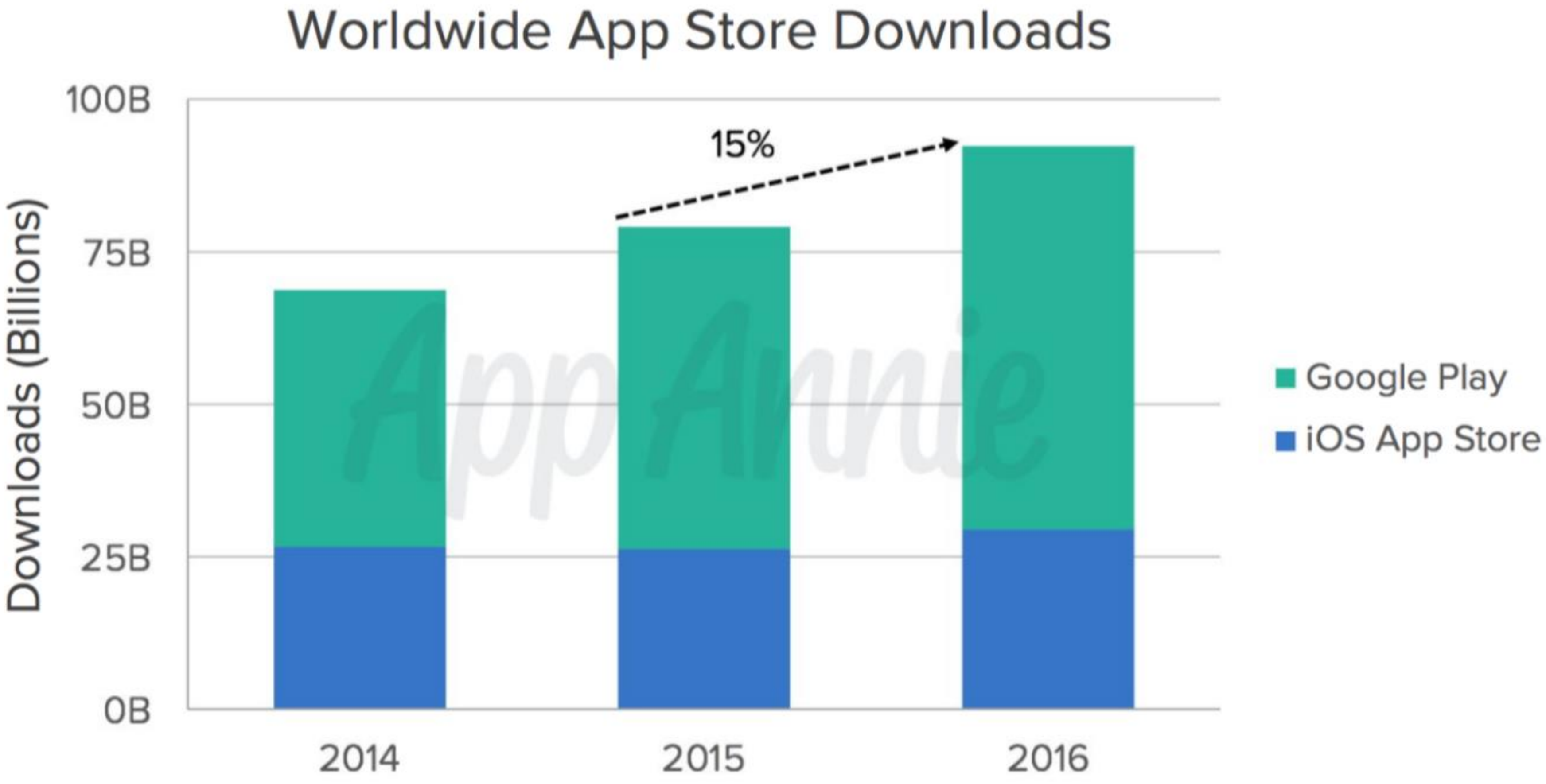
IPHONE VS. ANDROID

2015 APP DOWNLOADS

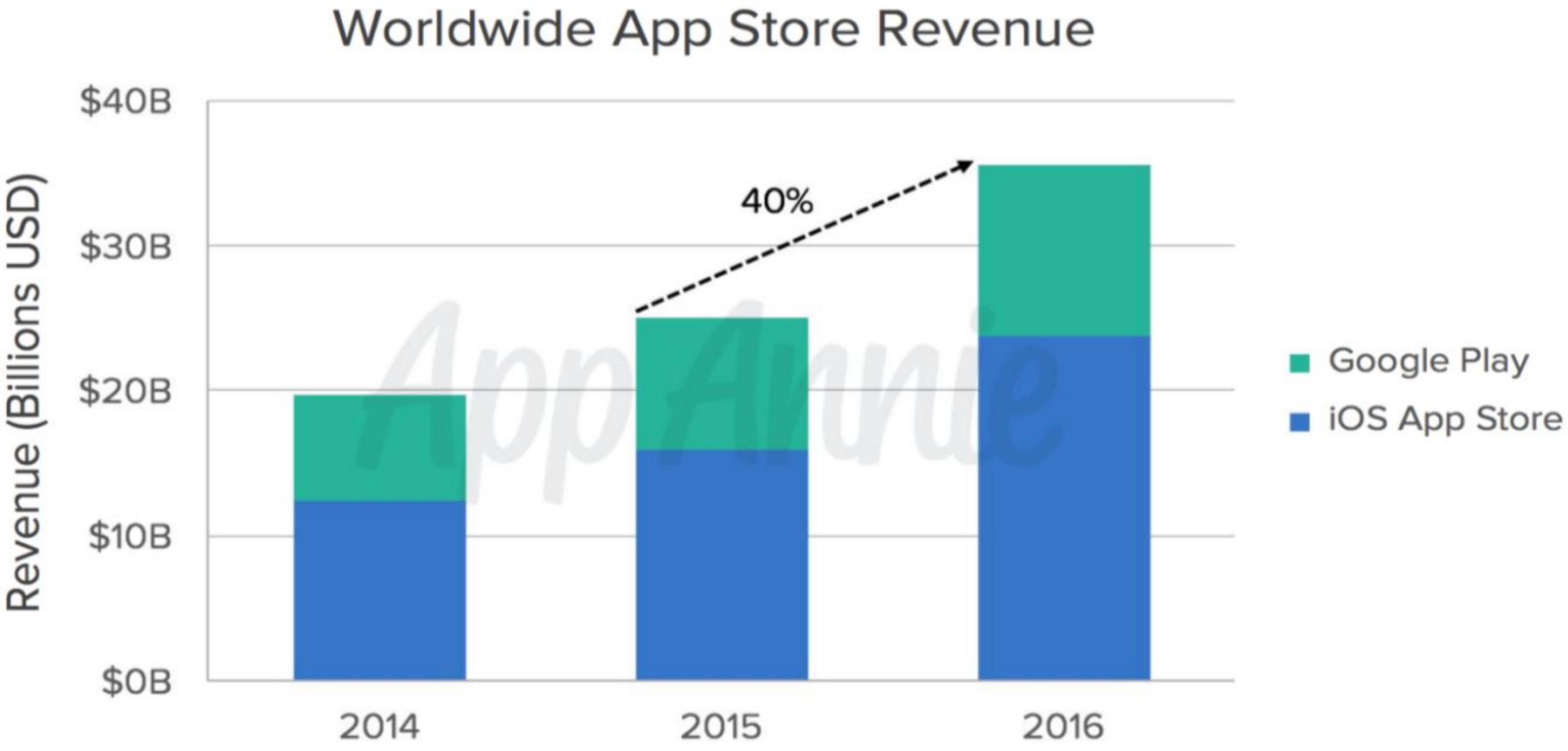


https://www.appannie.com/

ANDROID VS IOS



REVENUE



- Strategy: attract developers with comparison of revenue generated by applications, average revenue per user, etc.

WHY ANDROID?

- Powerful and open SDK
- No licensing fees
- Thriving developer community
- Low barrier to entry
- Huge potential market of users

ANDROID DEVELOPMENT TOOLS

SETUP DEVELOPMENT ENVIRONMENT

- Install JDK 8 or 10
- Install Android Studio
  - includes API level 28
- Use SDK manager to download lower API levels
- Detailed install instructions available on Android site http://developer.android.com/sdk/installing.html

ELEMENTS OF ANDROID PROJECTS

- *Application Name*
  - seen by users on app chooser, app list, store
- *Project Name*
  - in IDE, can be different, often directory
- *Package Name*
  - Java package name, not using default package
- *Minimum SDK version*
  - how far back of API level do you support, ~16 as of Jan 2017
- *Compile SDK version*
  - SDK version (PI level) where your app has been complied. it is strongly recommended that you always compile with the latest SDK.
- *Target SDK version*
  - Level of API you had in mind for app, most recent?
- *Theme*

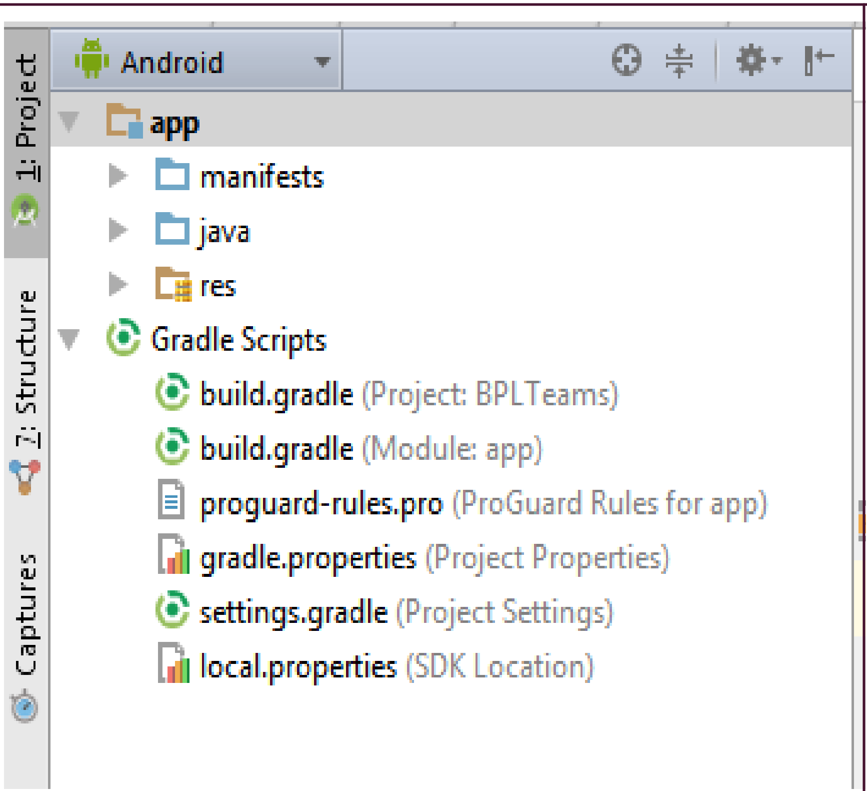
SDK VERSIONS RELASIONSHIP

minSdkVersion <= targetSdkVersion <= compileSdkVersion

Ideally, the relationship would look more like this in the steady state:

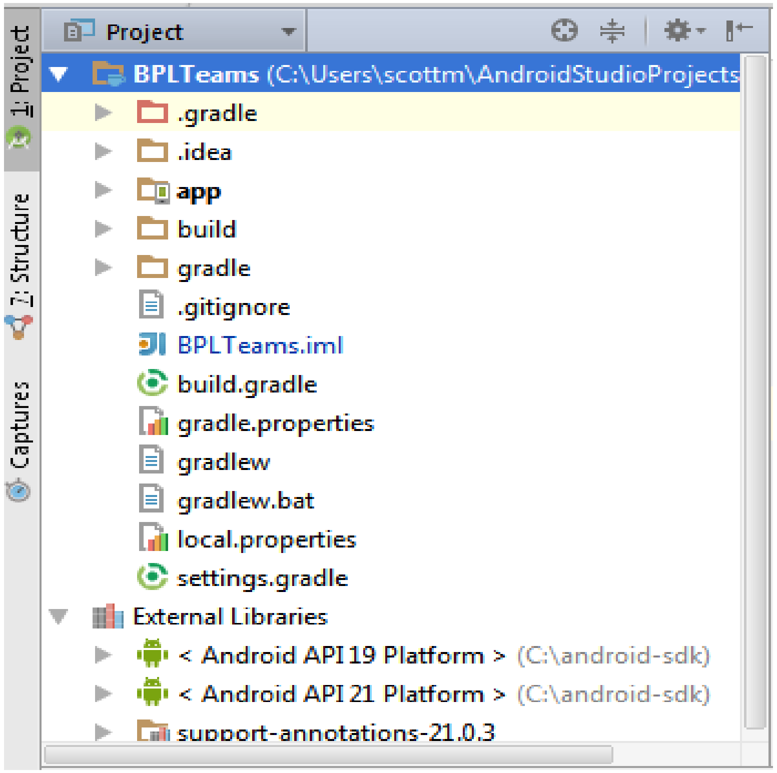
minSdkVersion (lowest possible) <=
    targetSdkVersion == compileSdkVersion (latest SDK)

ANDROID PROJECTS



```
app
    manifests
    java
    res
Gradle Scripts
    build.gradle (Project: BPLTeams)
    build.gradle (Module: app)
    proguard-rules.pro (ProGuard Rules for app)
    gradle.properties (Project Properties)
    settings.gradle (Project Settings)
    local.properties (SDK Location)
```

Android Project View



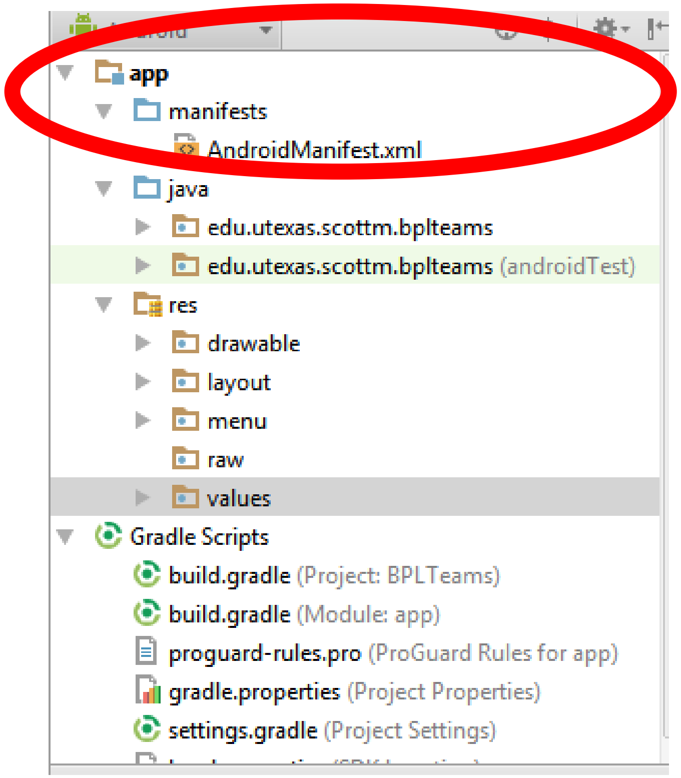
```
BPLTeams (C:\Users\scottm\AndroidStudioProjects
    .gradle
    .idea
    app
    build
    gradle
    .gitignore
    BPLTeams.iml
    build.gradle
    gradle.properties
    gradlew
    gradlew.bat
    local.properties
    settings.gradle
External Libraries
    < Android API 19 Platform > (C:\android-sdk)
    < Android API 21 Platform > (C:\android-sdk)
    support-annotations-21.0.3
```

Classic Project View

ANDROID PROJECT COMPONENTS

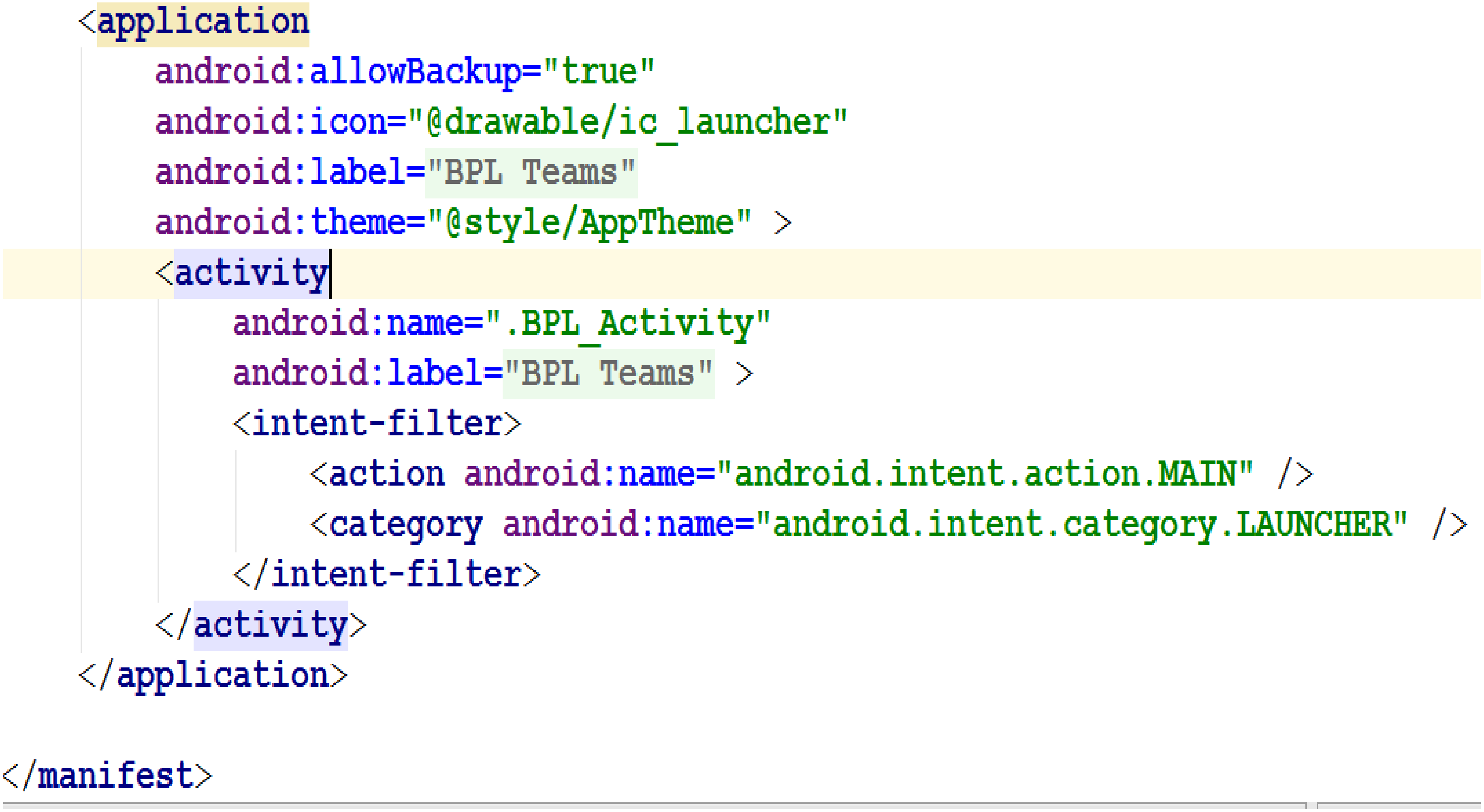
ANDROID PROJECT COMPONENTS - MANIFESTS

- AndroidManifest.xml
- Like a table of contents for your app
- Main activity
- Target and min SDK
- Declare all the parts of your apps:
  - activities, services
- Request permissions
  - network, location, …



- app
  - manifests
    - AndroidManifest.xml
  - java
    - edu.utexas.scottm.bplteams
    - edu.utexas.scottm.bplteams (androidTest)
  - res
    - drawable
    - layout
    - menu
    - raw
    - values
- Gradle Scripts
  - build.gradle (Project: BPLTeams)
  - build.gradle (Module: app)
  - proguard-rules.pro (ProGuard Rules for app)
  - gradle.properties (Project Properties)
  - settings.gradle (Project Settings)

ANDROID MANIFEST - SAMPLE

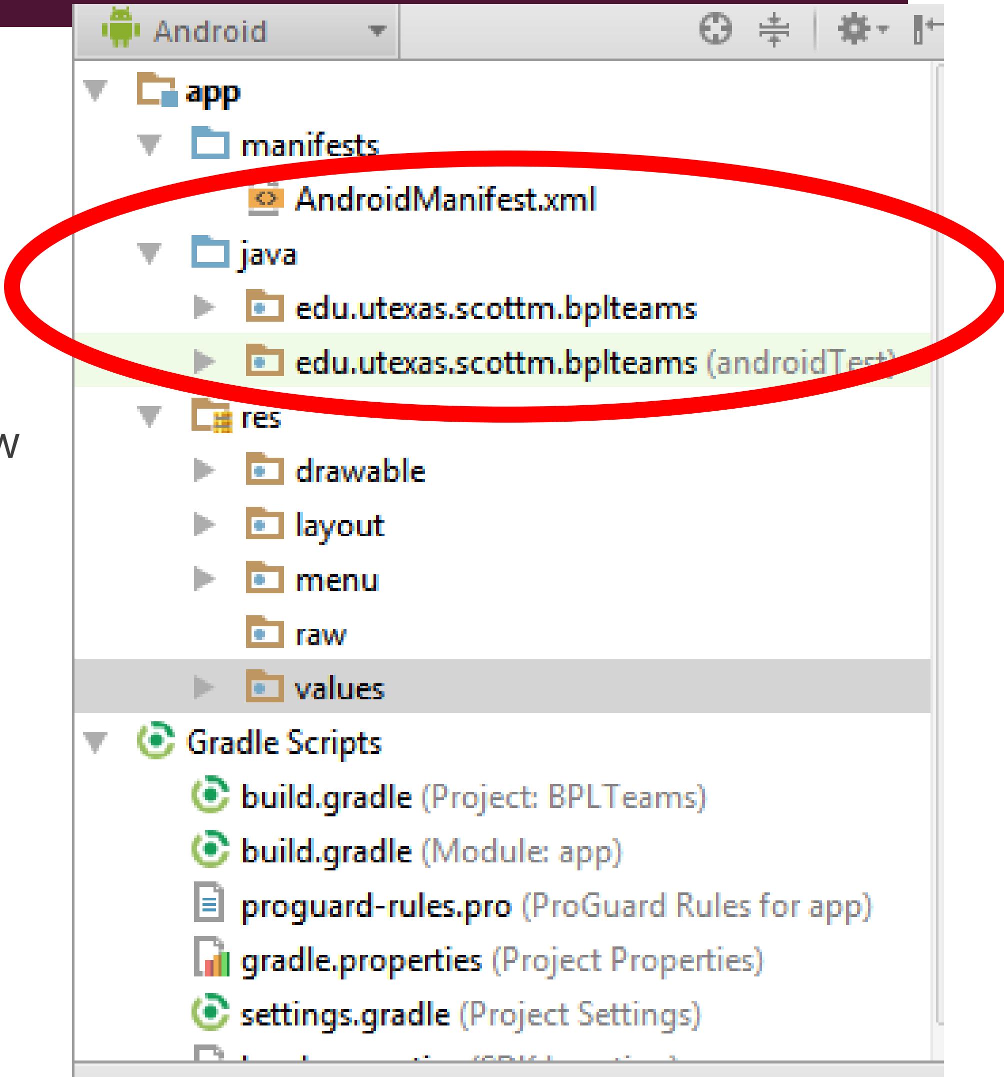


```xml
    <application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="BPL Teams"
        android:theme="@style/AppTheme" >
        <activity
            android:name=".BPL_Activity"
            android:label="BPL Teams" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

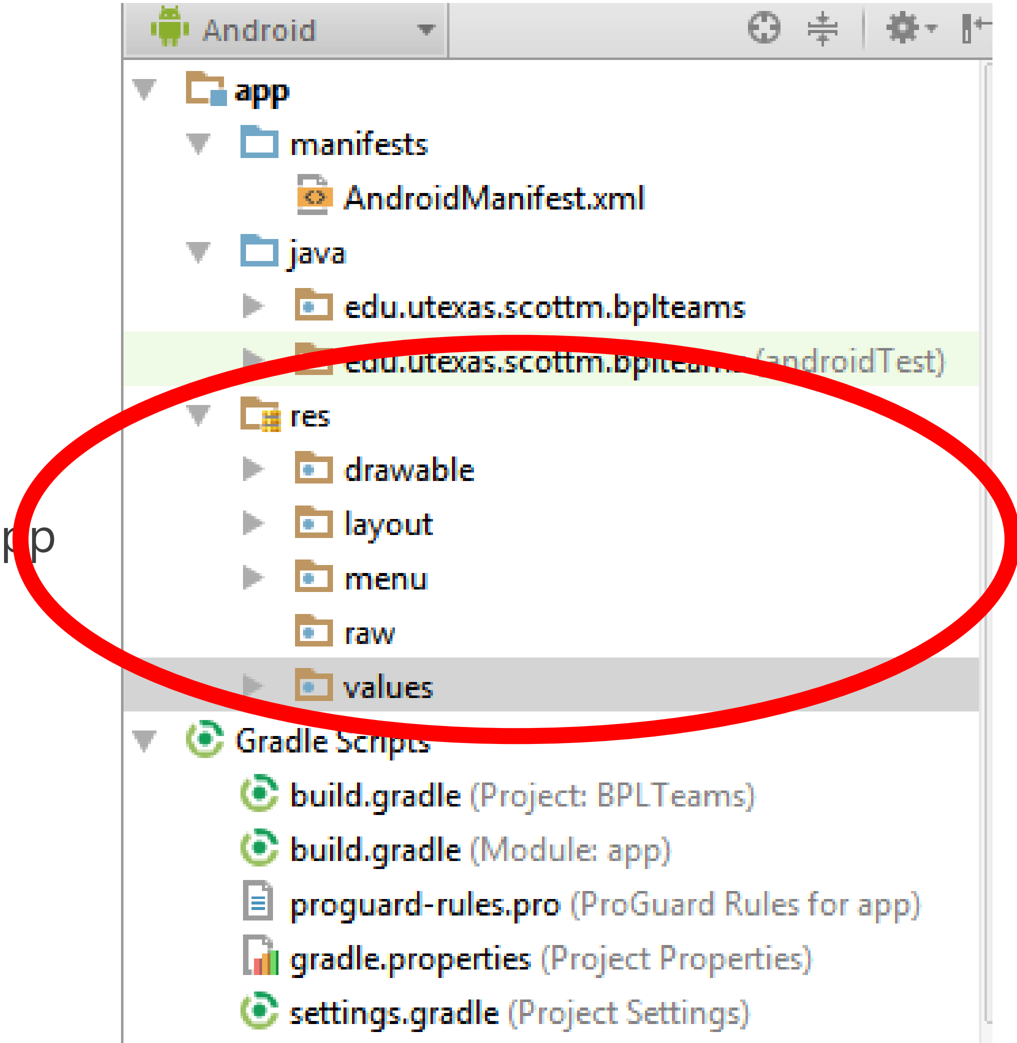
ANDROID PROJECT COMPONENTS - JAVA SOURCE CODE

- Source Code:
- In java directory in Android Project View
- Actually in src directory on system


Android
app
manifests
AndroidManifest.xml
java
edu.utexas.scottm.bplteams
edu.utexas.scottm.bplteams (androidTest)
res
drawable
layout
menu
raw
values
Gradle Scripts
build.gradle (Project: BPLTeams)
build.gradle (Module: app)
proguard-rules.pro (ProGuard Rules for app)
gradle.properties (Project Properties)
settings.gradle (Project Settings)

ANDROID PROJECT COMPONENTS - RESOURCES

- Resources or the res directory
- non source code resources for the app
- packaged up with app
- large role and use in development of app



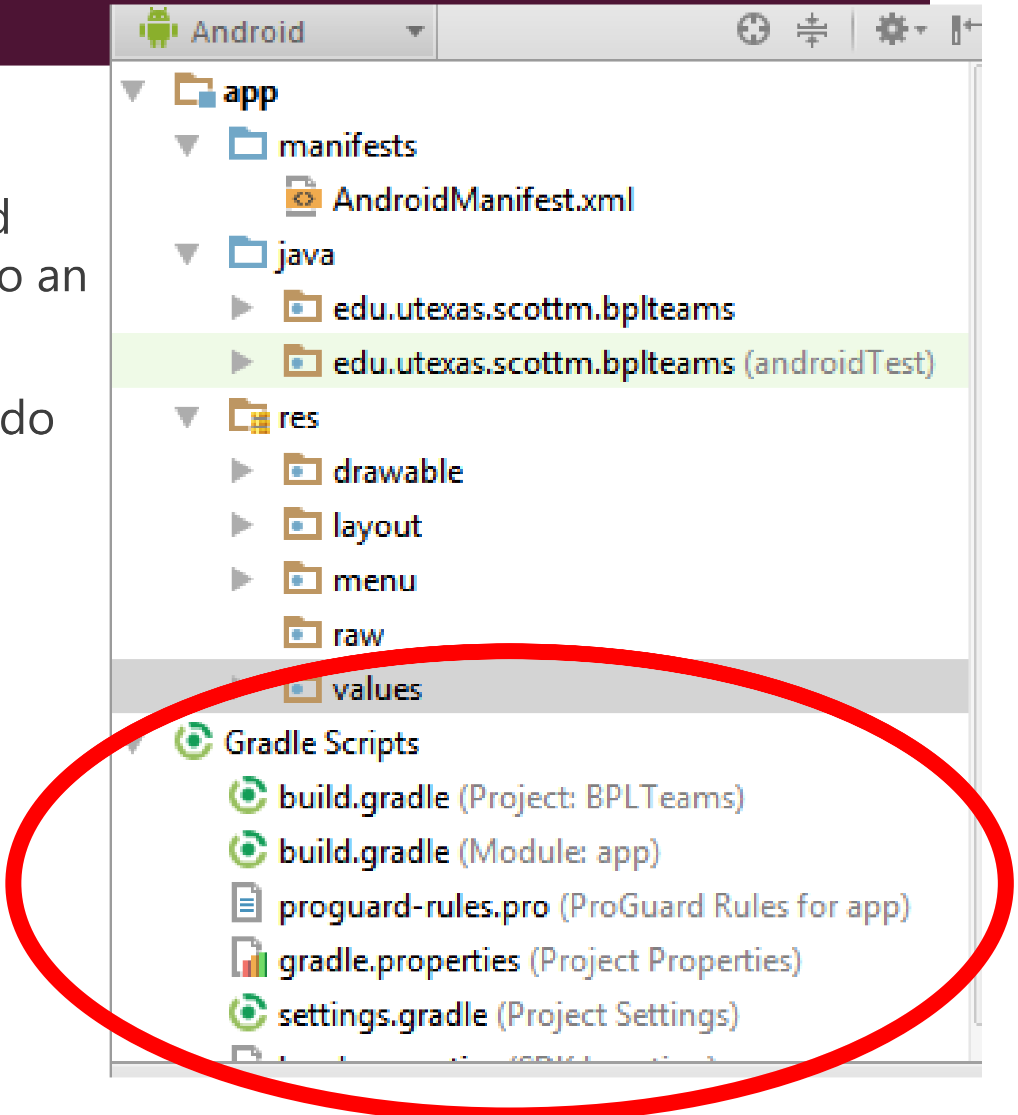
```
Android
app
    manifests
        AndroidManifest.xml
    java
        edu.utexas.scottm.bplteams
        edu.utexas.scottm.bplteams (androidTest)
    res
        drawable
        layout
        menu
        raw
        values
Gradle Scripts
    build.gradle (Project: BPLTeams)
    build.gradle (Module: app)
    proguard-rules.pro (ProGuard Rules for app)
    gradle.properties (Project Properties)
    settings.gradle (Project Settings)
```

RESOURCE DIRECTORIES

- **res/drawable** for graphic images
  such as png, jpeg
- **res/layout** for xml files that define the layout of user interfaces inside the app
- **res/menu** for xml based menu specifications
- **res/values** for lists of strings, dimensions, colors, lists of data
- **res/raw** for other kinds of files such as audio clips, video clips, csv files, raw text
- **res/xml** for other general purpose xml files

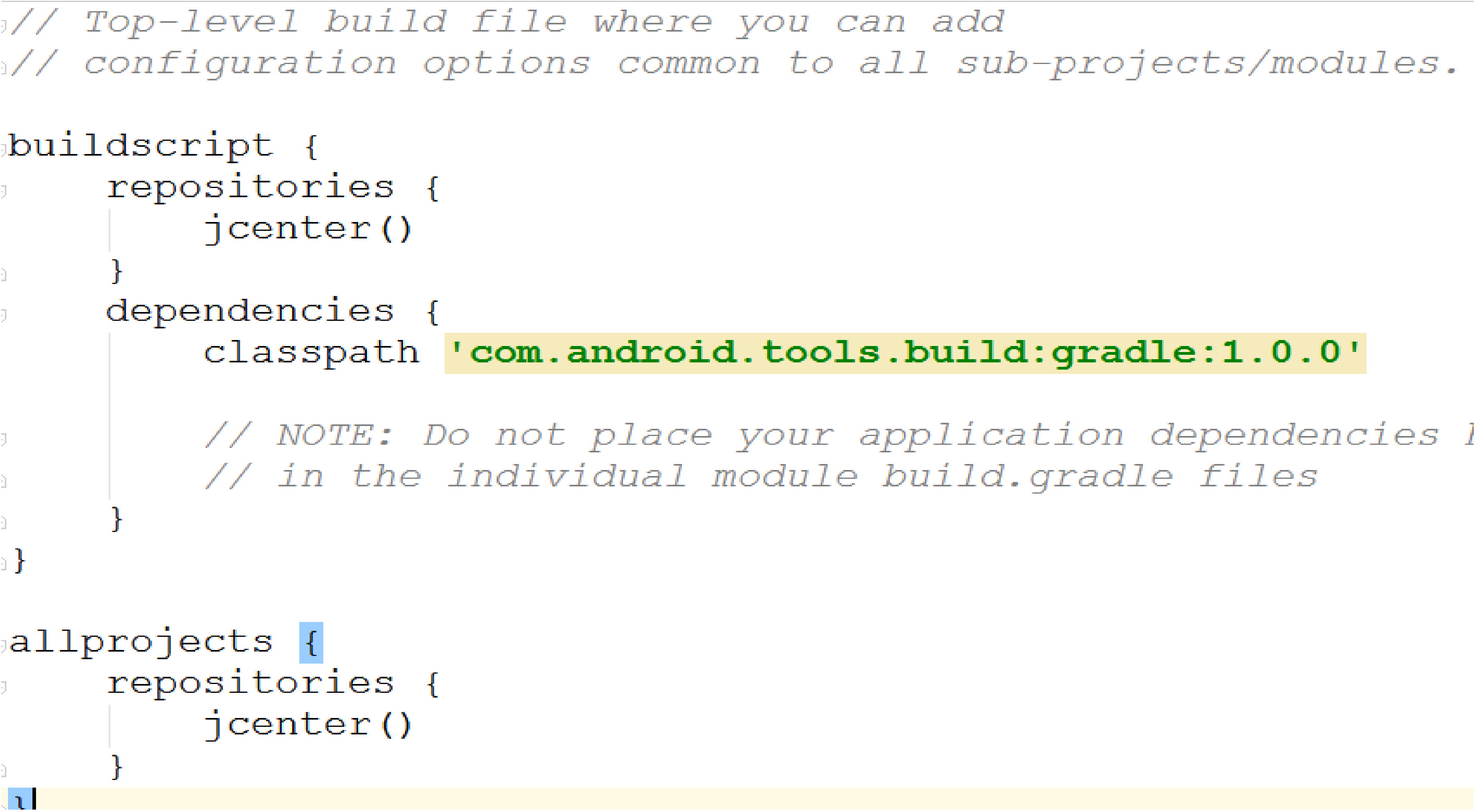
GRADLE

- Gradle is the build engine that Android Studio uses to convert your project into an APK
- What needs to be created and how to do it
- Like
  - make for C/C++
  - Ant/Maven for Java
- build.gradle file



```
app
  manifests
    AndroidManifest.xml
  java
    edu.utexas.scottm.bplteams
    edu.utexas.scottm.bplteams (androidTest)
  res
    drawable
    layout
    menu
    raw
    values
Gradle Scripts
  build.gradle (Project: BPLTeams)
  build.gradle (Module: app)
  proguard-rules.pro (ProGuard Rules for app)
  gradle.properties (Project Properties)
  settings.gradle (Project Settings)
```

SAMPLE BUILD.GRADLE FILE - PROJECT



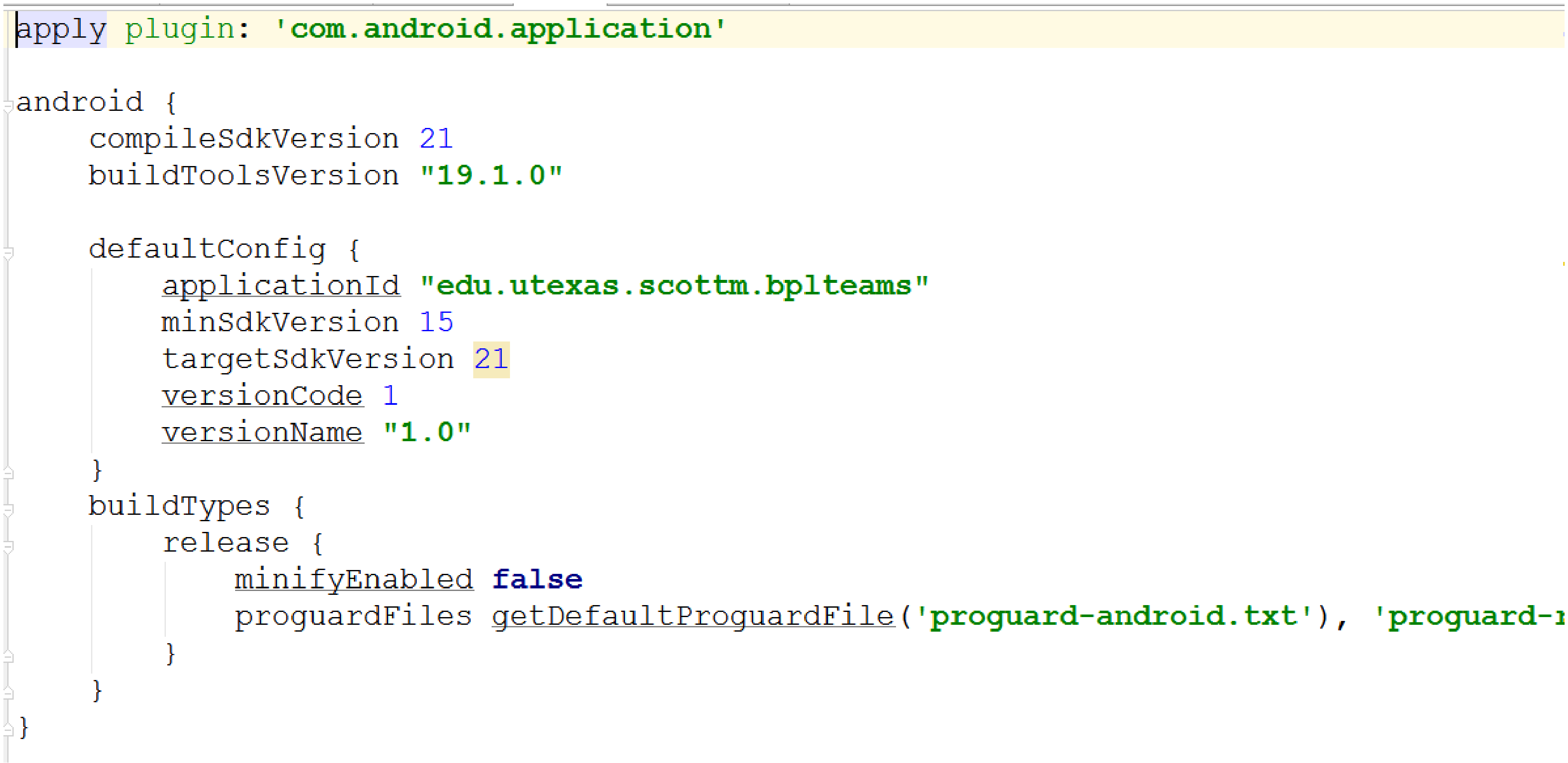
```
// Top-level build file where you can add
// configuration options common to all sub-projects/modules.

buildscript {
    repositories {
        jcenter()
    }
    dependencies {
        classpath 'com.android.tools.build:gradle:1.0.0'

        // NOTE: Do not place your application dependencies
        // in the individual module build.gradle files
    }
}

allprojects {
    repositories {
        jcenter()
    }
}
```

SAMPLE BUILD.GRADLE FILE - MODULE / APP



```
apply plugin: 'com.android.application'

android {
    compileSdkVersion 21
    buildToolsVersion "19.1.0"

    defaultConfig {
        applicationId "edu.utexas.scottm.bplteams"
        minSdkVersion 15
        targetSdkVersion 21
        versionCode 1
        versionName "1.0"
    }
    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-r
        }
    }
}
```

WHY GRADLE?

Large commercial developers need:

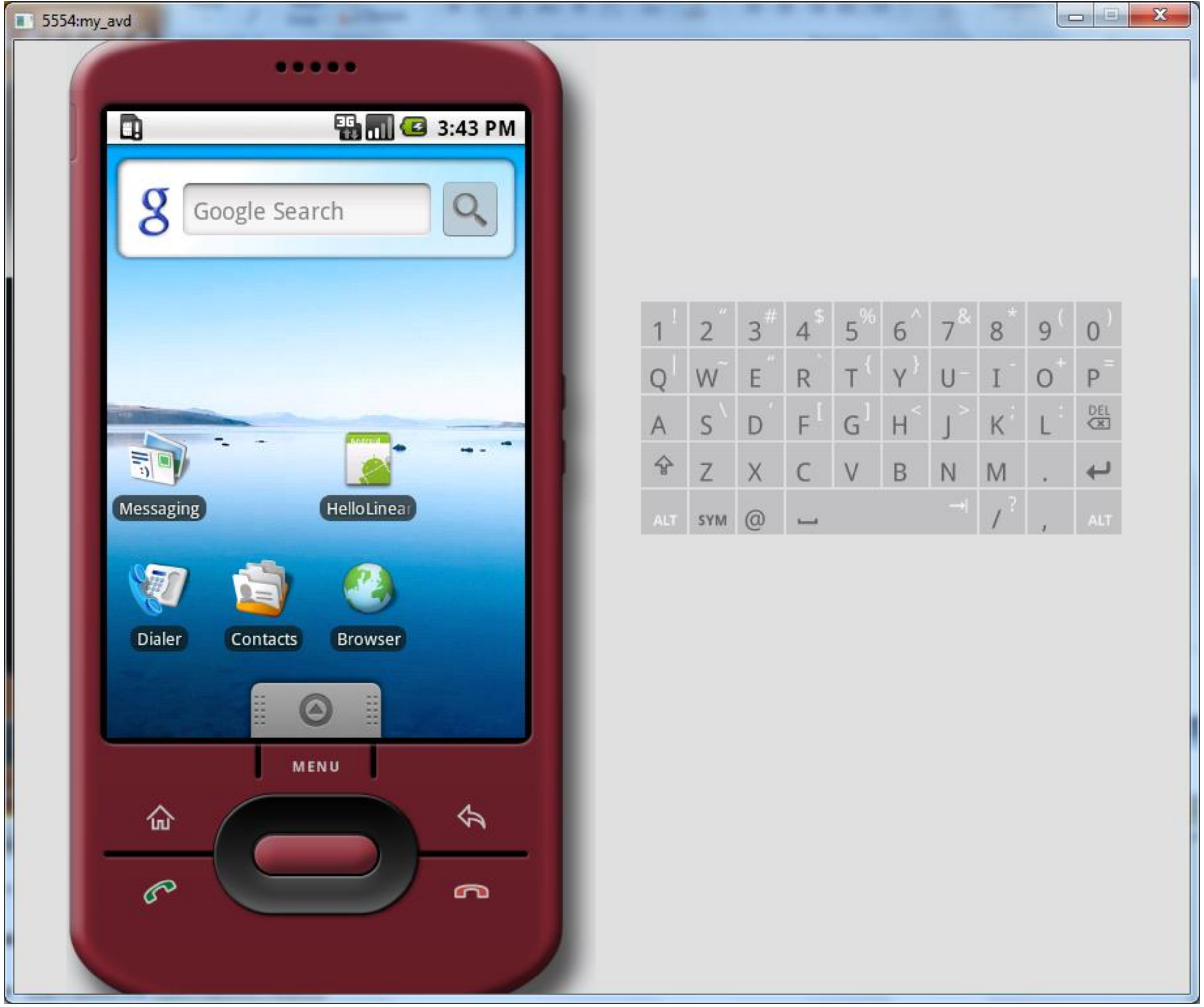
- Scripting
  - Run tests after compiling
  - APK signing
- Create versions of APKs for different:
  - Pricing tiers (free vs. paid)
  - Form factors (e.g., phone vs. tablet)
  - OS versions (e.g., Lollipop vs. pre-Lollipop)

EMULATORS

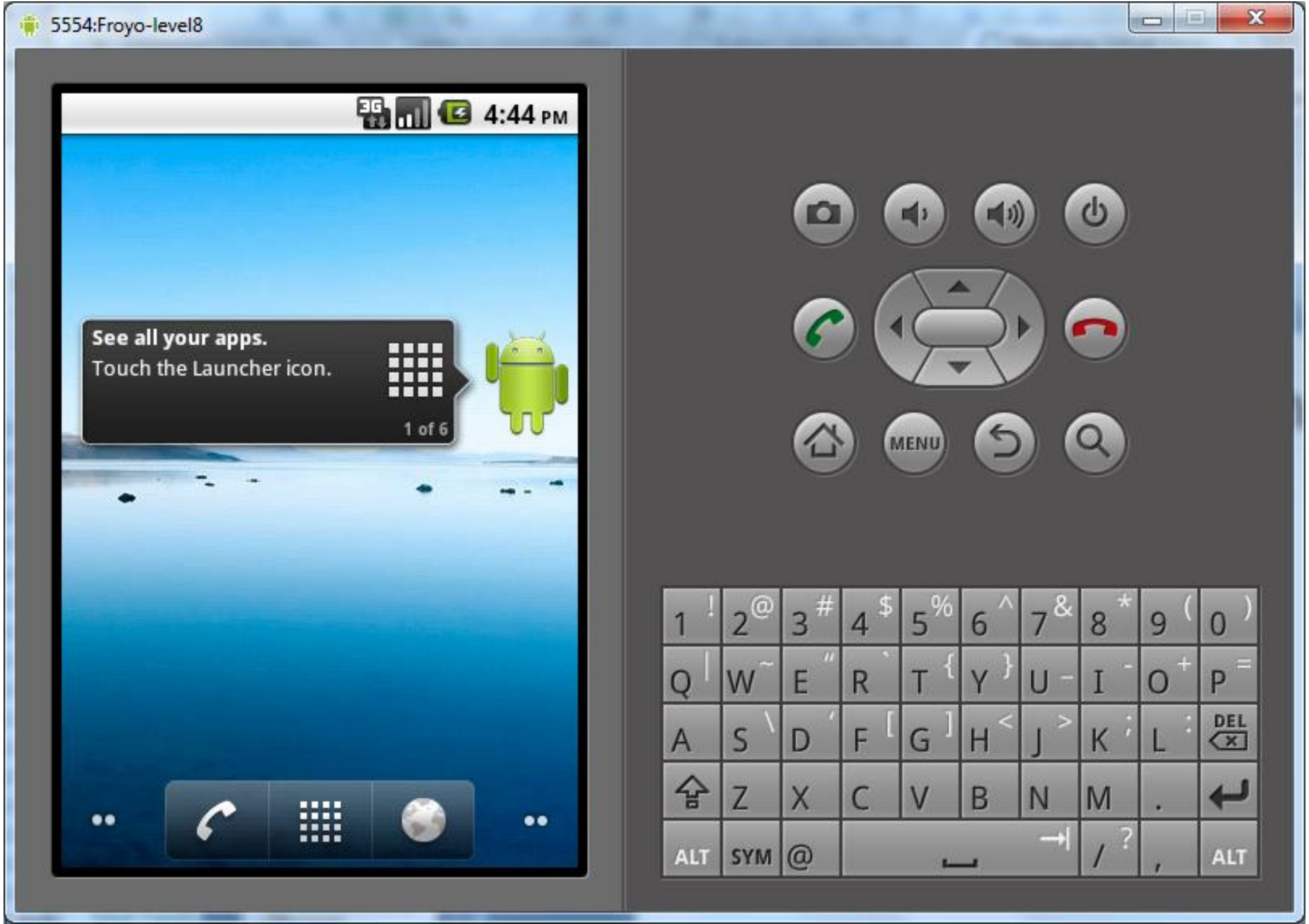
ANDROID EMULATOR OR ANDROID VIRTUAL DEVICE (AVD)

- Emulator is useful for testing apps but is not a substitute of a real device
- Emulators are called **Android Virtual Devices** (AVDs)
- Android SDK and AVD Manager allows you to create AVDs that target any Android API level
- AVD have configurable resolutions, RAM, SD cards, skins, and other hardware

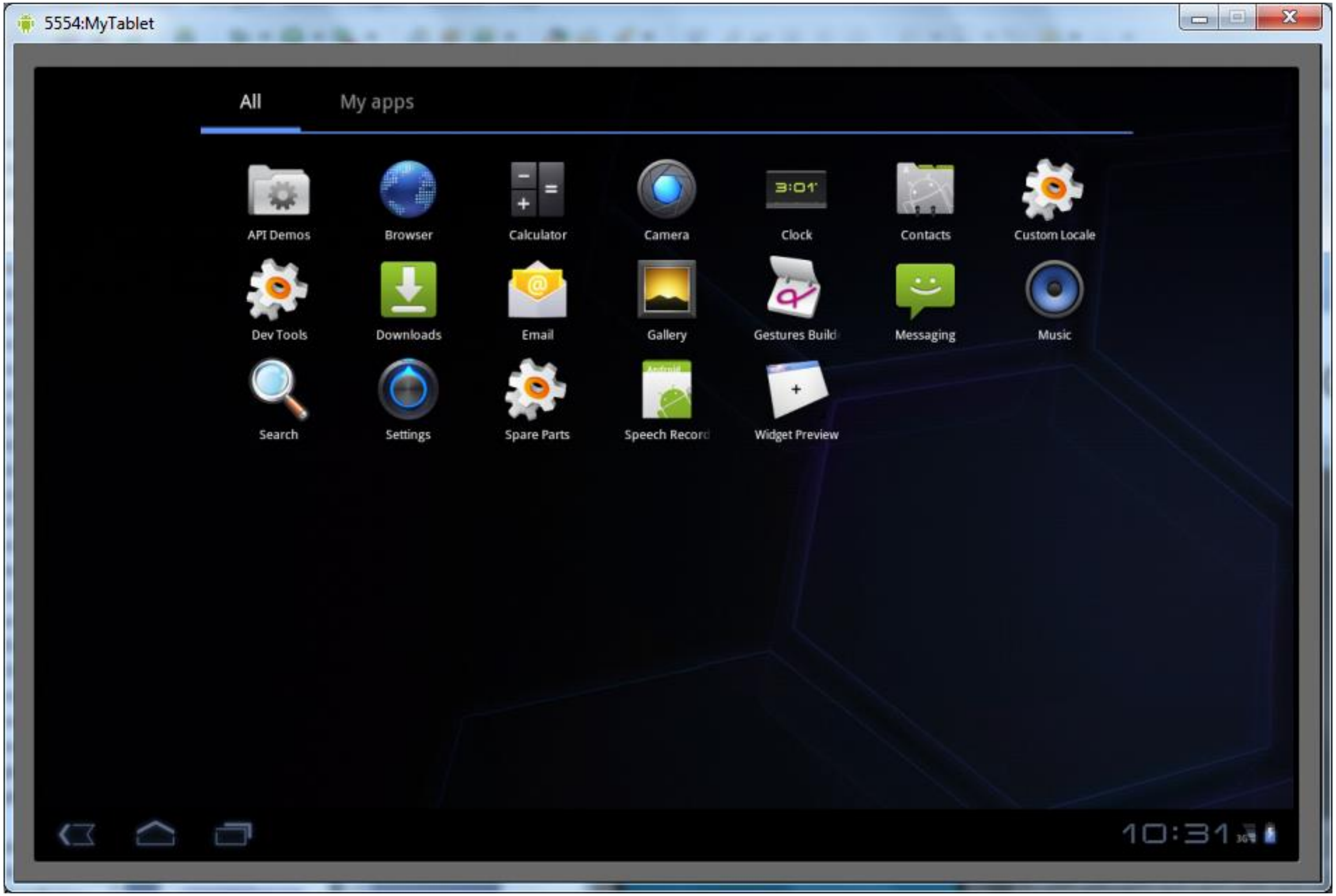
ANDROID EMULATOR: 1.6



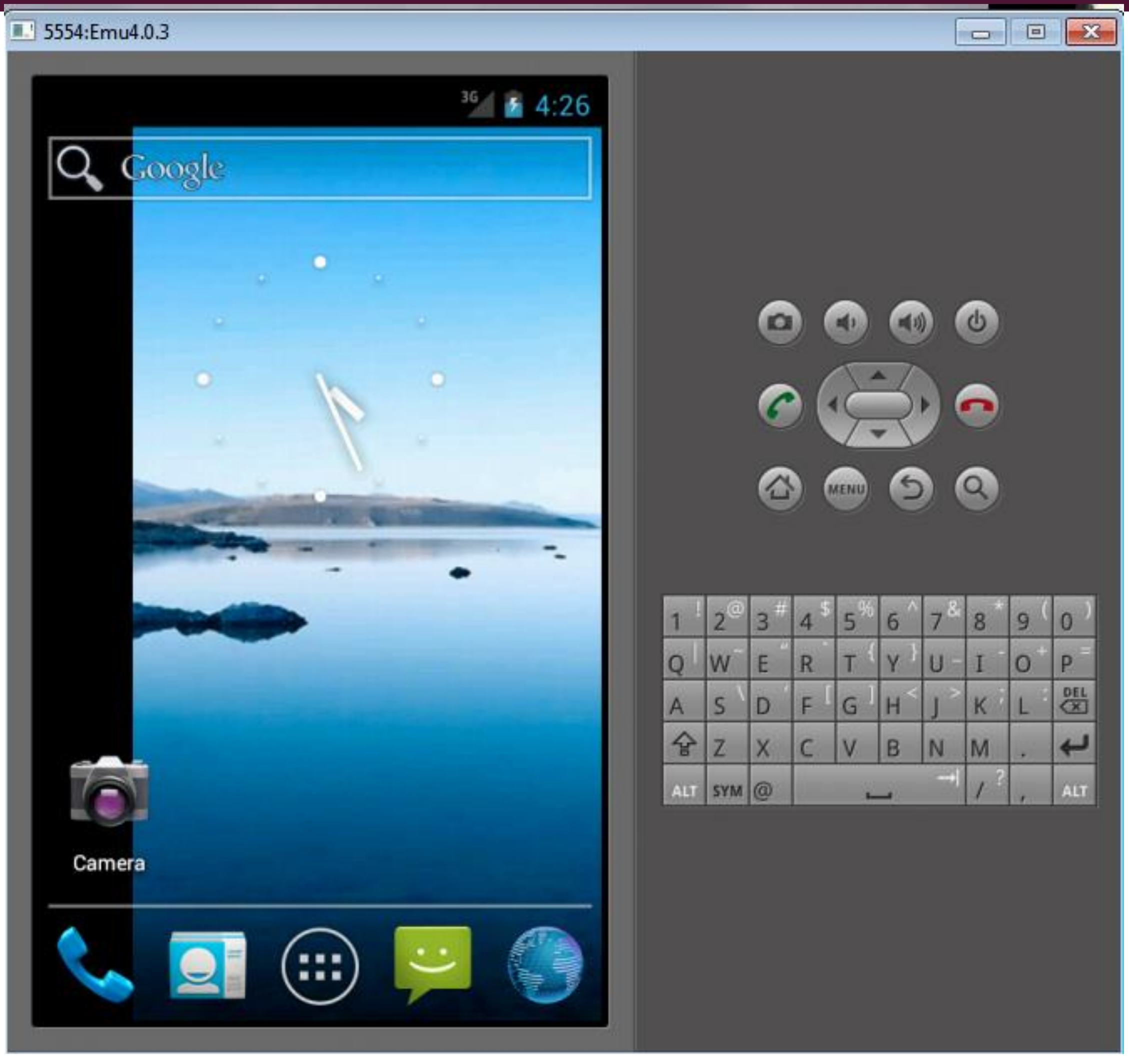
ANDROID EMULATOR: 2.2



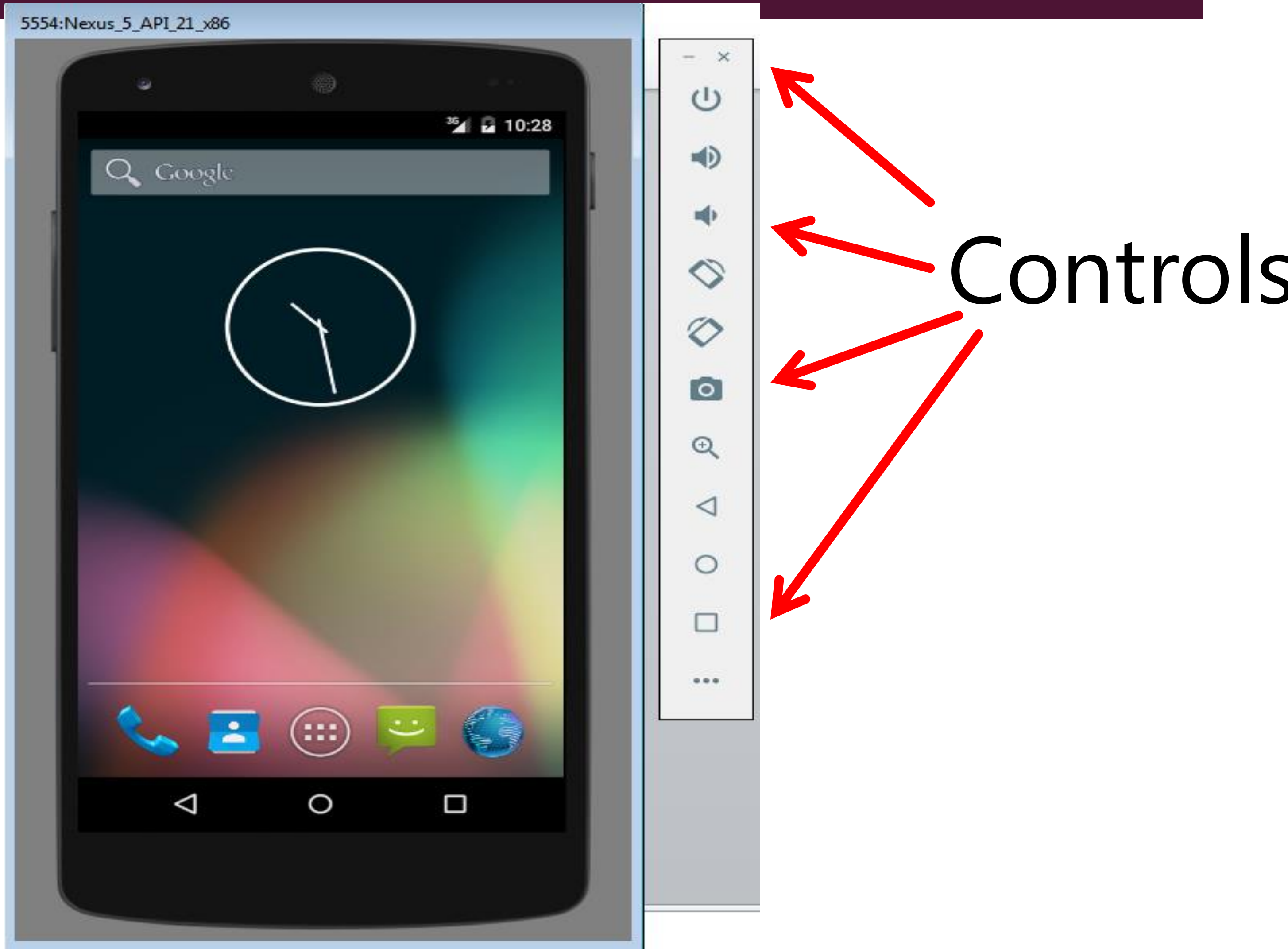
ANDROID EMULATOR: 3.0



ANDROID EMULATOR: 4.0



ANDROID EMULATOR: 5.0



EMULATOR BASICS

- Host computer’s keyboard can be used
- Host’s mouse acts as finger
- Uses host’s Internet connection
- Other buttons work: Home, Back, Search, volume up and down, etc.
- More info at

  https://developer.android.com/studio/run/emulator

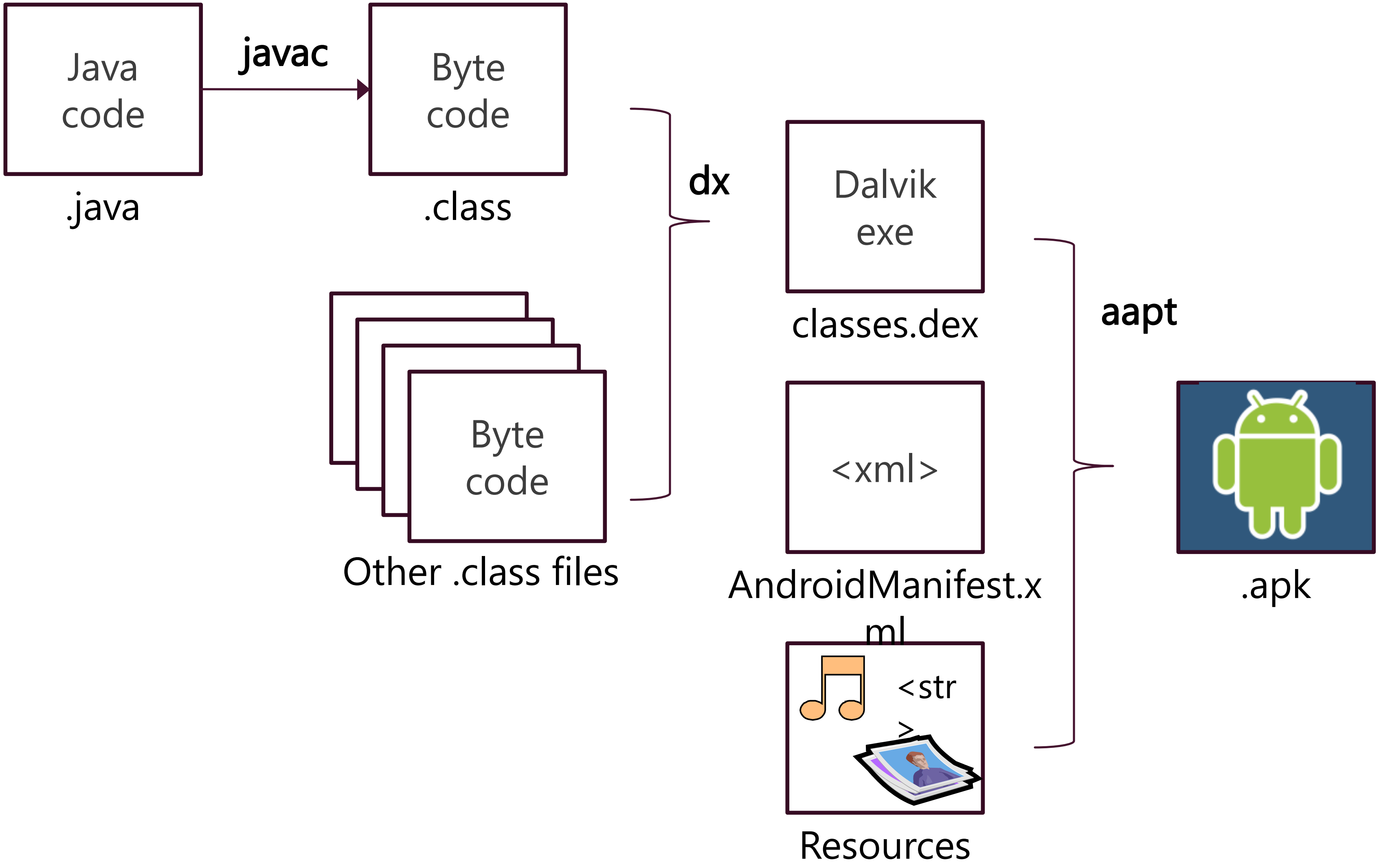
EMULATOR LIMITATIONS

- No support for USB connections
- No support for device-attached headphones
- No support for determining connected state
- No support for determining SD card insert/eject
- No support for Bluetooth
- No support for NFC

ANDROID RUNTIME: DALVIK VM

- Subset of Java developed by Google
- Optimized for mobile devices (better memory management, battery utilization, etc.)
- Dalvik runs .dex files that are compiled from .class files
- Introduces new libraries
- Does not support some Java libraries like AWT, Swing

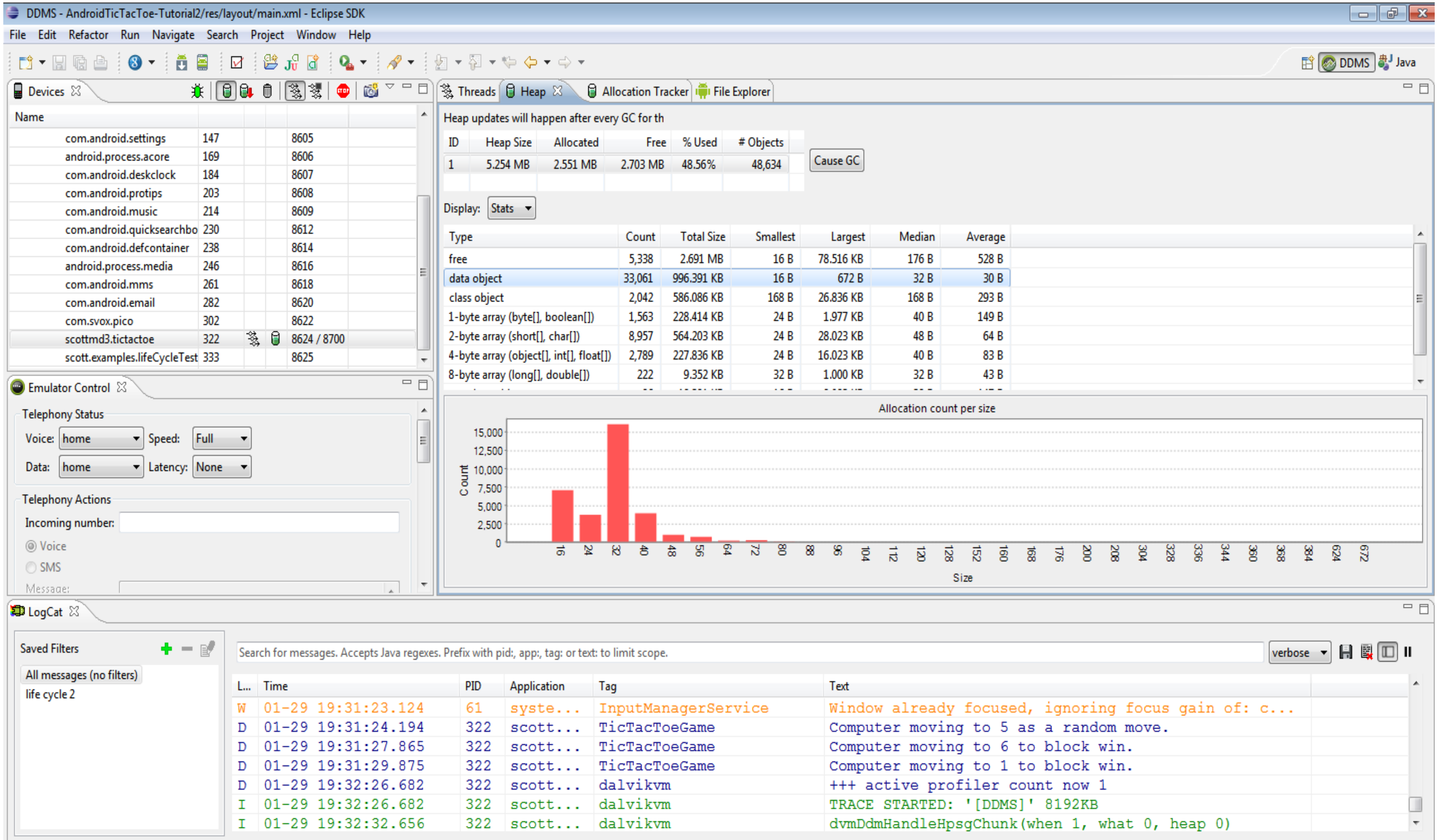
PRODUCING AN ANDROID APP



DALVIK DEBUG MONITOR SERVER

- DDMS
- debugging tool
- "provides, screen capture on the device, thread and heap information on the device, logcat, process, and radio state information, incoming call and SMS spoofing, location data spoofing, and more."
- can interact with DDMS via Android Studio

DDMS



DDMS - AndroidTicTacToe-Tutorial2/res/layout/main.xml - Eclipse SDK
File Edit Refactor Run Navigate Search Project Window Help
DDMS
Java
Devices
Threads
Heap
Allocation Tracker
File Explorer
Emulator Control
LogCat
Cause GC

Name			
com.android.settings	147		8605
android.process.acore	169		8606
com.android.deskclock	184		8607
com.android.protips	203		8608
com.android.music	214		8609
com.android.quicksearchbo	230		8612
com.android.defcontainer	238		8614
android.process.media	246		8616
com.android.mms	261		8618
com.android.email	282		8620
com.svox.pico	302		8622
scottmd3.tictactoe	322		8624 / 8700
scott.examples.lifeCycleTest	333		8625

Heap updates will happen after every GC for th

ID	Heap Size	Allocated	Free	% Used	# Objects
1	5.254 MB	2.551 MB	2.703 MB	48.56%	48,634

Display: Stats

Type	Count	Total Size	Smallest	Largest	Median	Average
free	5,338	2.691 MB	16 B	78.516 KB	176 B	528 B
data object	33,061	996.391 KB	16 B	672 B	32 B	30 B
class object	2,042	586.086 KB	168 B	26.836 KB	168 B	293 B
1-byte array (byte[], boolean[])	1,563	228.414 KB	24 B	1.977 KB	40 B	149 B
2-byte array (short[], char[])	8,957	564.203 KB	24 B	28.023 KB	48 B	64 B
4-byte array (object[], int[], float[])	2,789	227.836 KB	24 B	16.023 KB	40 B	83 B
8-byte array (long[], double[])	222	9.352 KB	32 B	1.000 KB	32 B	43 B

Allocation count per size

Telephony Status

Voice: home Speed: Full

Data: home Latency: None

Telephony Actions

Incoming number:

● Voice

○ SMS

Message:

Saved Filters

All messages (no filters)

life cycle 2

Search for messages. Accepts Java regexes. Prefix with pid:, app:, tag: or text: to limit scope.

verbose

```
L... Time                PID  Application  Tag                    Text
W    01-29 19:31:23.124  61   syste...     InputManagerService    Window already focused, ignoring focus gain of: c...
D    01-29 19:31:24.194  322  scott...     TicTacToeGame          Computer moving to 5 as a random move.
D    01-29 19:31:27.865  322  scott...     TicTacToeGame          Computer moving to 6 to block win.
D    01-29 19:31:29.875  322  scott...     TicTacToeGame          Computer moving to 1 to block win.
D    01-29 19:32:26.682  322  scott...     dalvikvm               +++ active profiler count now 1
I    01-29 19:32:26.682  322  scott...     dalvikvm               TRACE STARTED: '[DDMS]' 8192KB
I    01-29 19:32:32.656  322  scott...     dalvikvm               dvmDdmHandleHpsgChunk(when 1, what 0, heap 0)
```

GETTING ACTIVE THROUGH ACTIVITIES

There are 4 types of application components/building blocks:

Activities

1. Activity provides user interface
2. Usually represents a single screen
3. Can contain one or more views
4. Extends the Activity base class

Services

1. No user interface
2. Runs in background
3. Extends the Service base class

BroadcastReceiver

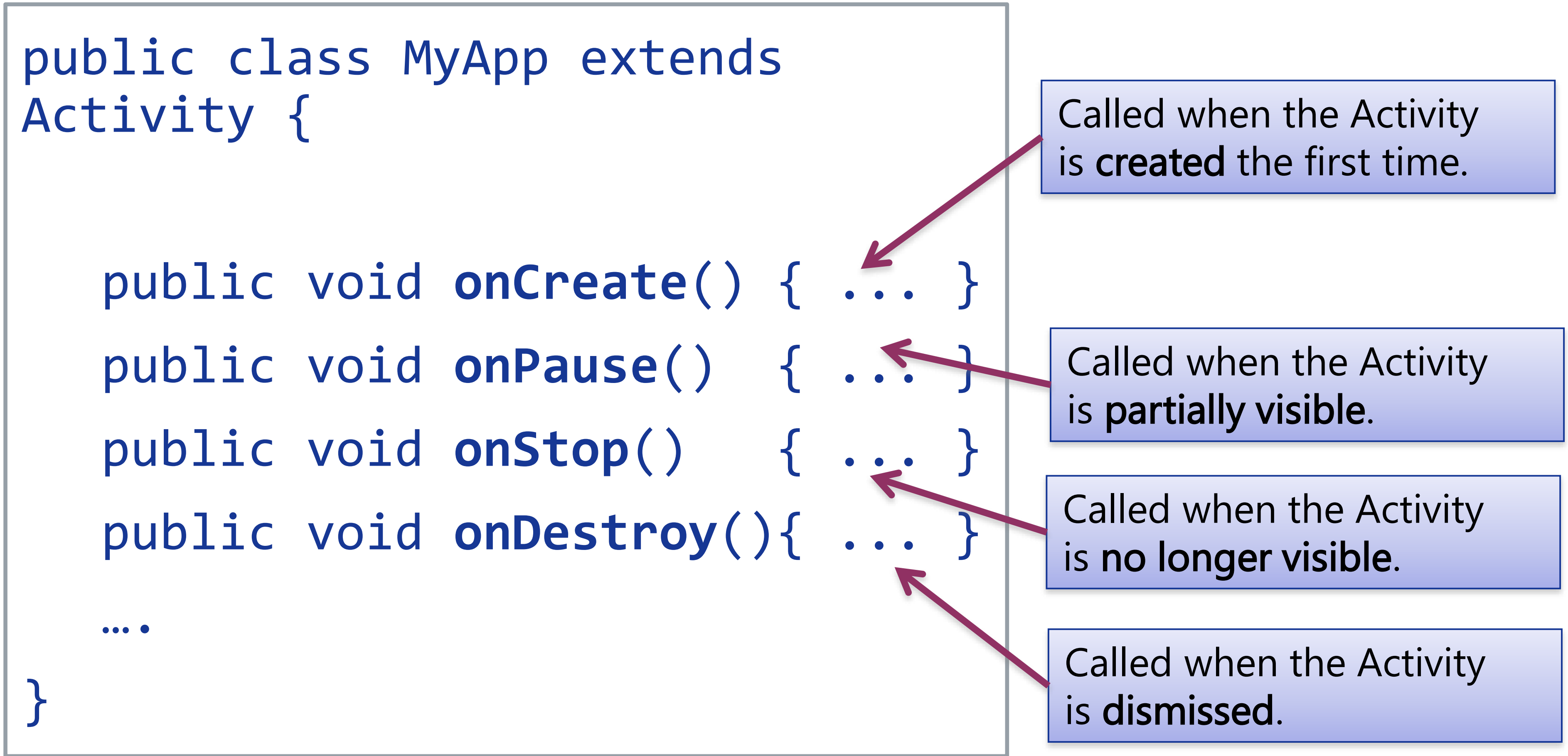
1. Receives and Reacts to broadcast Intents
2. No UI but can start an Activity
3. Extends the BroadcastReceiver base class

ContentProviders

1. Makes application data available to other apps [data sharing]
2. Uses SQLite database as storage
3. Extends the ContentProvider base class

Getting Active Through Activities

Activity



```
public class MyApp extends
Activity {


   public void onCreate() { ... }
   public void onPause()  { ... }
   public void onStop()   { ... }
   public void onDestroy(){ ... }
   ….
}
```

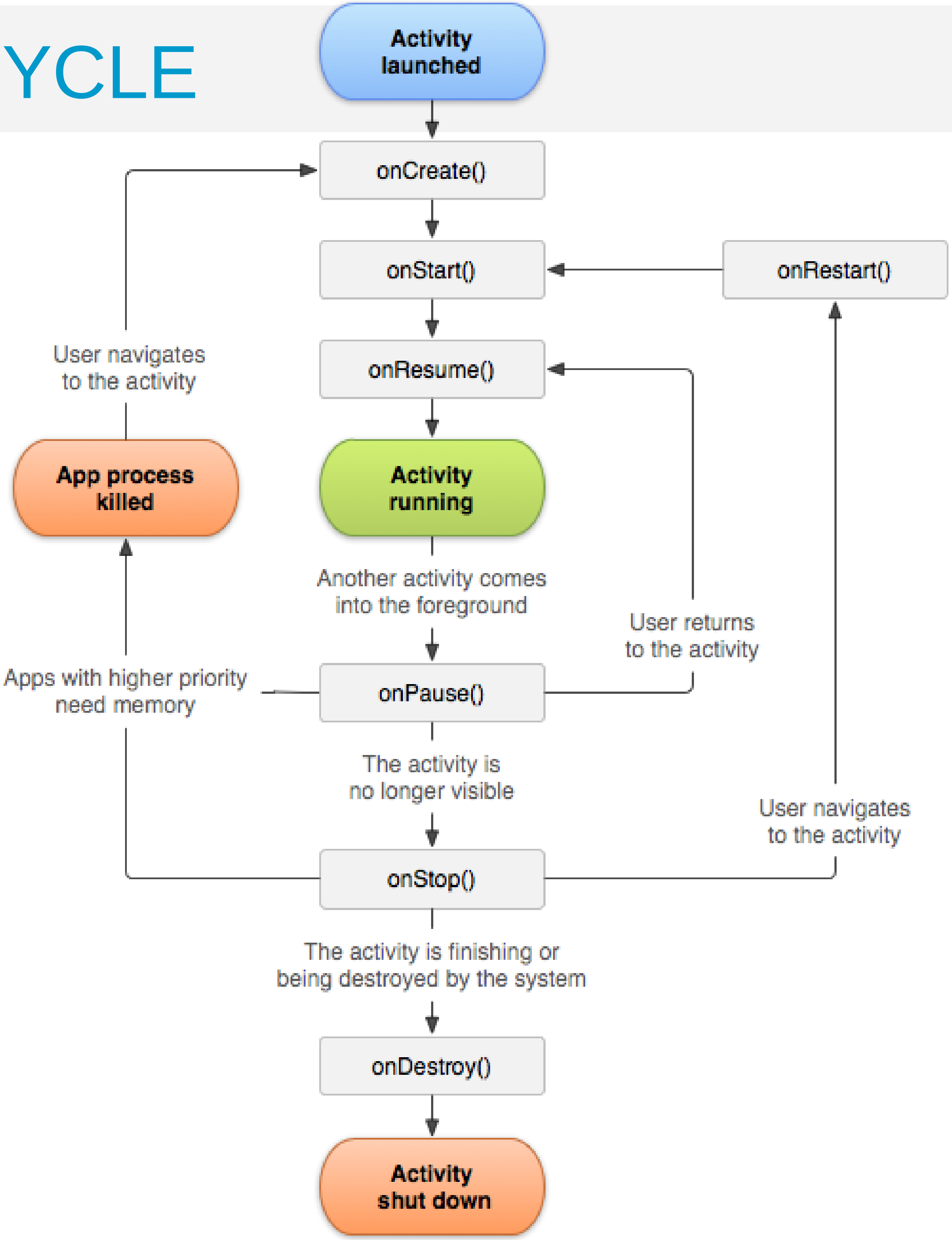
Called when the Activity is **created** the first time.

Called when the Activity is **partially visible**.

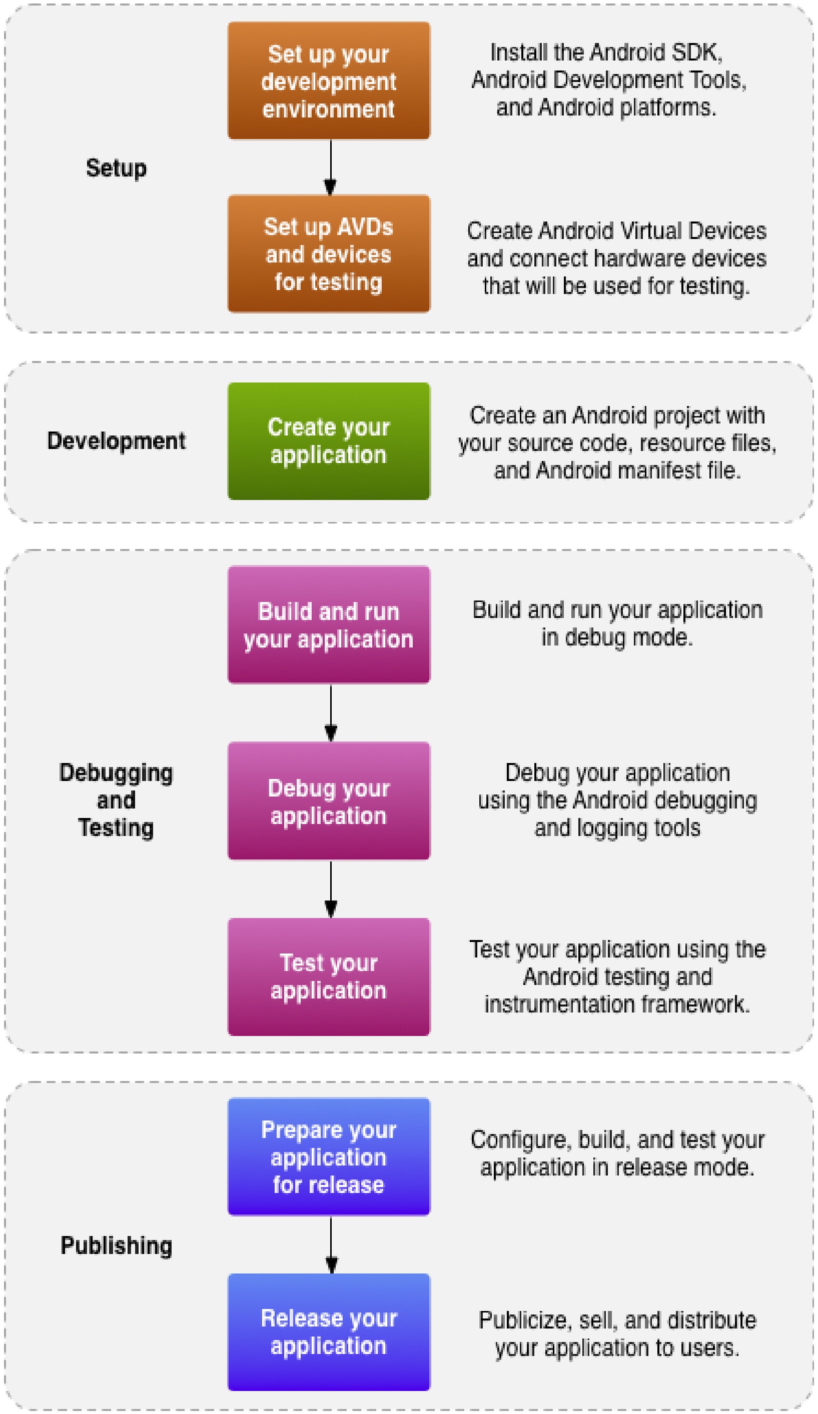
Called when the Activity is **no longer visible**.

Called when the Activity is **dismissed**.

ACTIVITY LIFECYCLE


Activity launched
onCreate()
onStart()
onRestart()
User navigates to the activity
onResume()
App process killed
Activity running
Another activity comes into the foreground
User returns to the activity
Apps with higher priority need memory
onPause()
The activity is no longer visible
User navigates to the activity
onStop()
The activity is finishing or being destroyed by the system
onDestroy()
Activity shut down

DEVELOPER WORKFLOW



Setup

Set up your development environment

Install the Android SDK, Android Development Tools, and Android platforms.

Set up AVDs and devices for testing

Create Android Virtual Devices and connect hardware devices that will be used for testing.

Development

Create your application

Create an Android project with your source code, resource files, and Android manifest file.

Debugging and Testing

Build and run your application

Build and run your application in debug mode.

Debug your application

Debug your application using the Android debugging and logging tools

Test your application

Test your application using the Android testing and instrumentation framework.

Publishing

Prepare your application for release

Configure, build, and test your application in release mode.

Release your application

Publicize, sell, and distribute your application to users.

REFERENCES

- From Android Studio
  - An Overview of the Android Architecture
  - The Anatomy of an Android Studio Android Application
  - Understanding Android Application and Activity Lifecycles
- Read different web links given in various slides