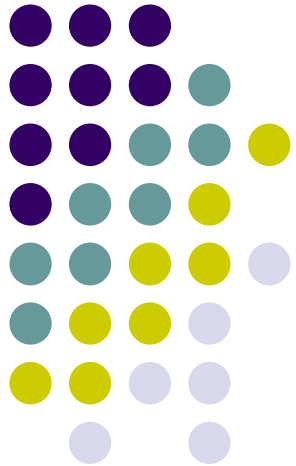
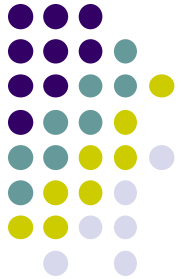
CS 528 Mobile and Ubiquitous Computing
Lecture 1b: Introduction to Android

Emmanuel Agu



What is Android?

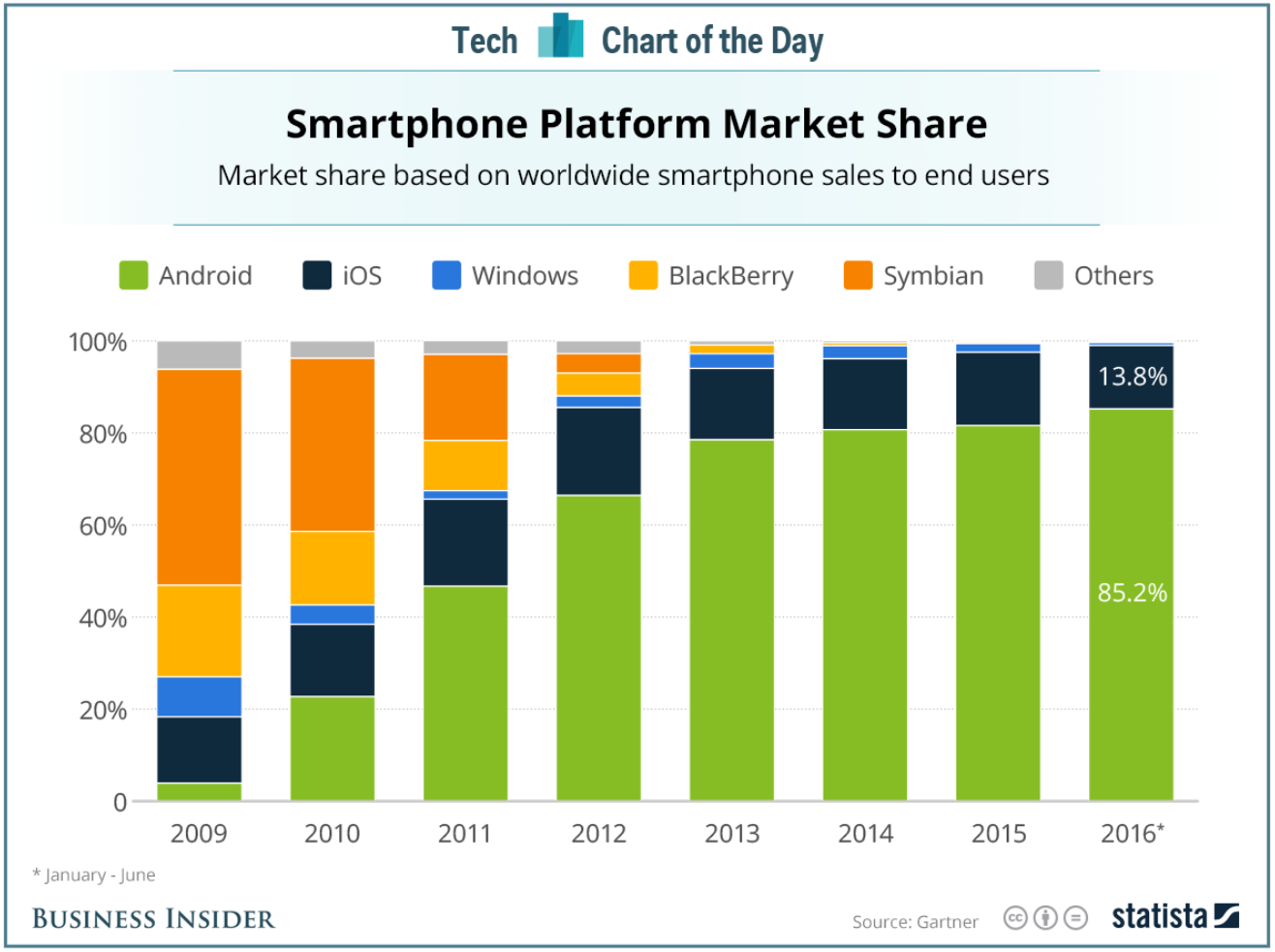


- Android is world's leading mobile operating system
  - Open source (https://source.android.com/setup/)

- **Google:**
  - Owns Android, maintains it, extends it
  - Distributes Android OS, developer tools, free to use
  - Runs Android app market

SmartPhone OS

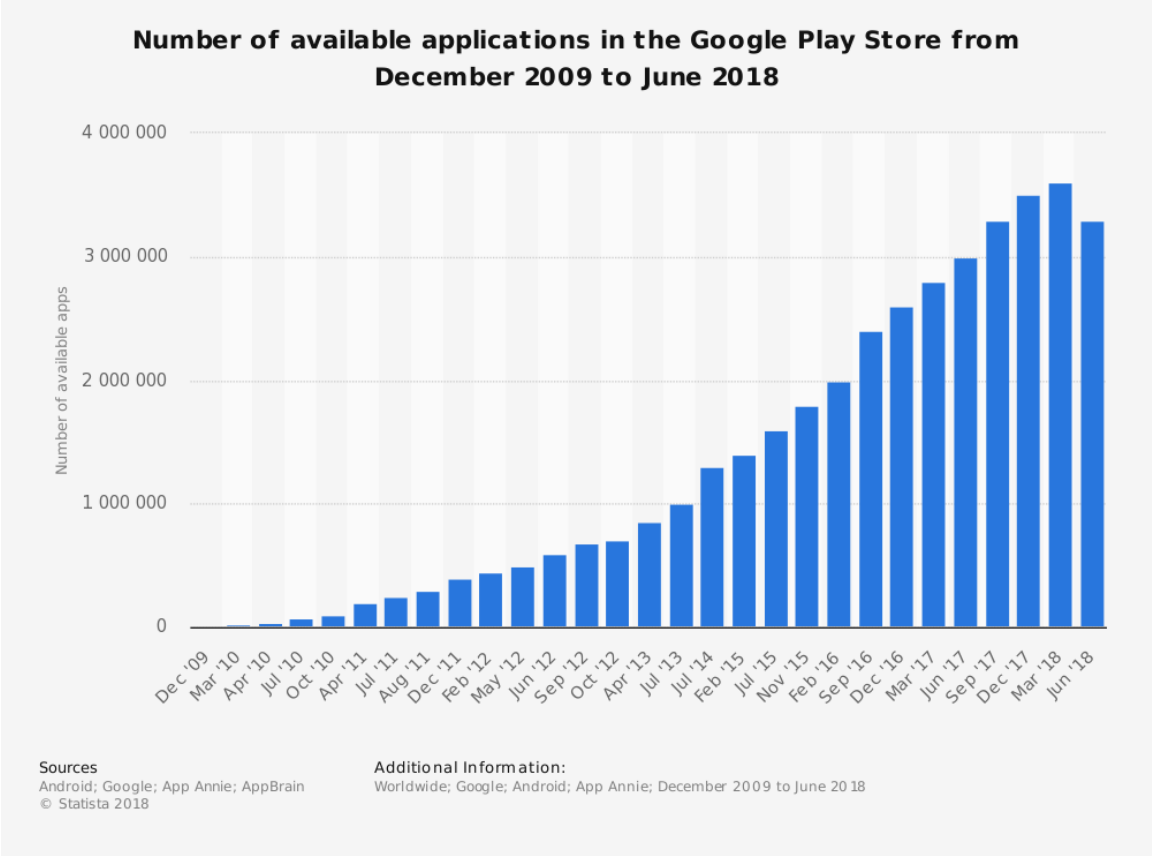
- Over 80% of all phones sold are smartphones
- Android share 86% worldwide



Source: Statista

Android Growth

- Over 2 billion Android users, March 2017 (ref: the verge)
- 3.3 million apps on the Android app market (ref: statista.com)
  - Games, organizers, banking, entertainment, etc



Android is Multi-Platform

Google Glass
(being redone)

In-car console

Smartwatch

Android runs on
all these devices

Tablet

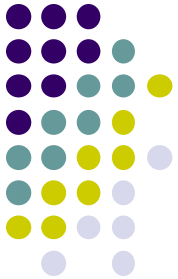
Devices/Things
(e.g. Raspberry Pi)

Television

Smartphone

This Class: Focuses
Mostly on Smartphones!

Why Android for Mobile Computing and Ubicomp?



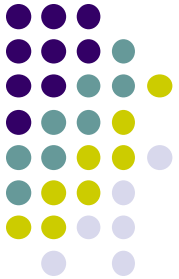
- Android for Mobile programmable modules
  - Audio/video playback, taking pictures, database, location detection, maps
- Android for Ubicomp programmable modules
  - Sensors (temperature, humidity, light, etc), proximity
  - Face detection, activity recognition, place detection, speech recognition, speech-to-text, gesture detection, place type understanding, etc
  - Machine learning, deep learning

Android Versions

- Class will use Android 7 (“Nougat”)
- Officially released December 5, 2016
- Latest version is Android 9 (Pie), released August 2018
- Below is Android version distribution as at July 23, 2018

Version	Codename	API	Distribution
2.3.3 - 2.3.7	Gingerbread	10	0.2%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	0.3%
4.1.x	Jelly Bean	16	1.2%
4.2.x		17	1.9%
4.3		18	0.5%
4.4	KitKat	19	9.1%
5.0	Lollipop	21	4.2%
5.1		22	16.2%
6.0	Marshmallow	23	23.5%
7.0	Nougat	24	21.2%
7.1		25	9.6%
8.0	Oreo	26	10.1%
8.1		27	2.0%

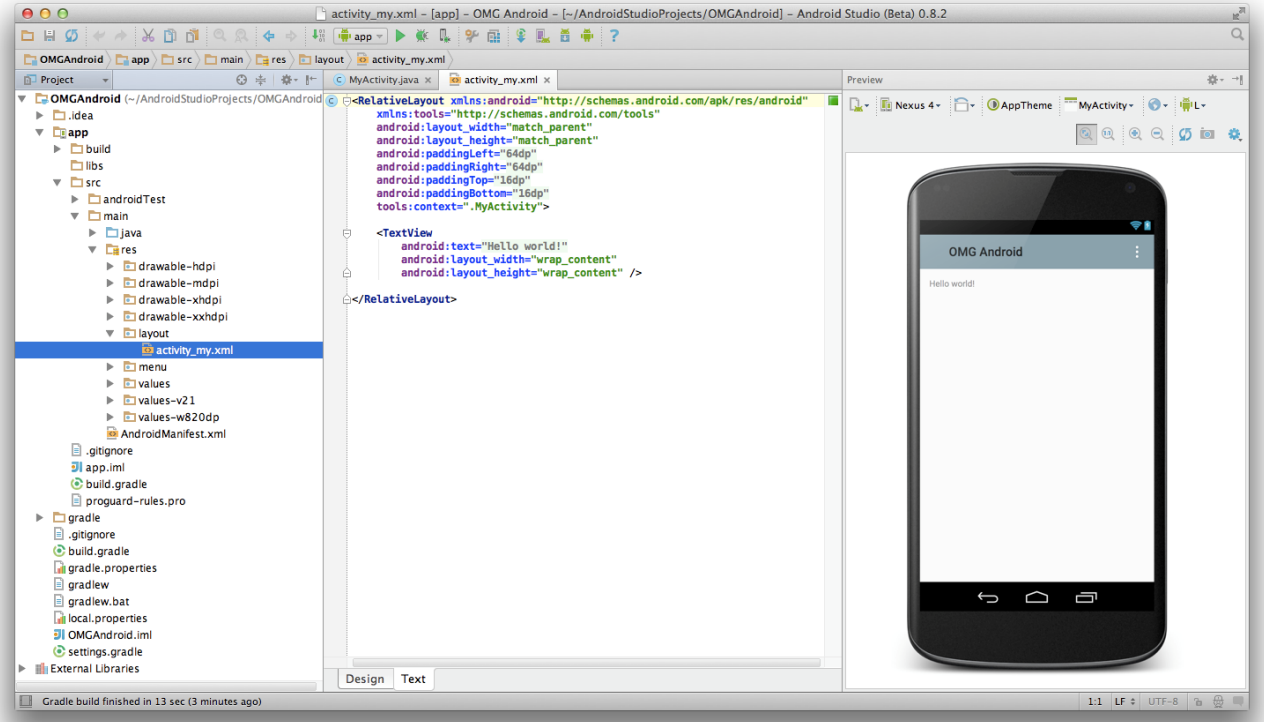
Source: http://developer.android.com/about/dashboards/index.html



Android Developer Environment

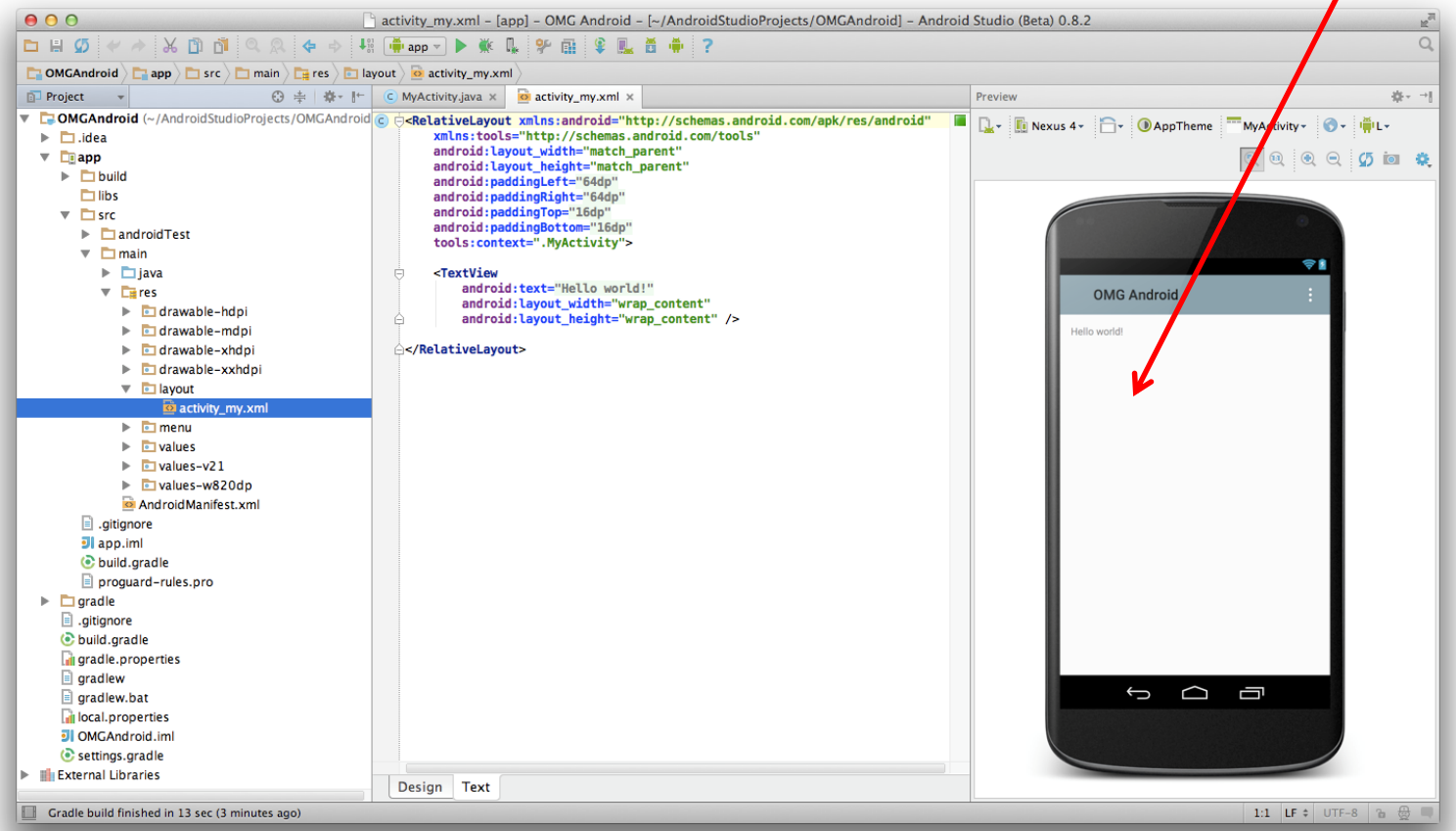
New Android Environment: Android Studio

- Old Android dev environment used **Eclipse + plugins**
- Google developed it's own IDE called **Android Studio**
- Integrated development environment, cleaner interface, specifically for Android Development (e.g. drag and drop app design)
- In December 2014, Google announced it will stop supporting Eclipse IDE



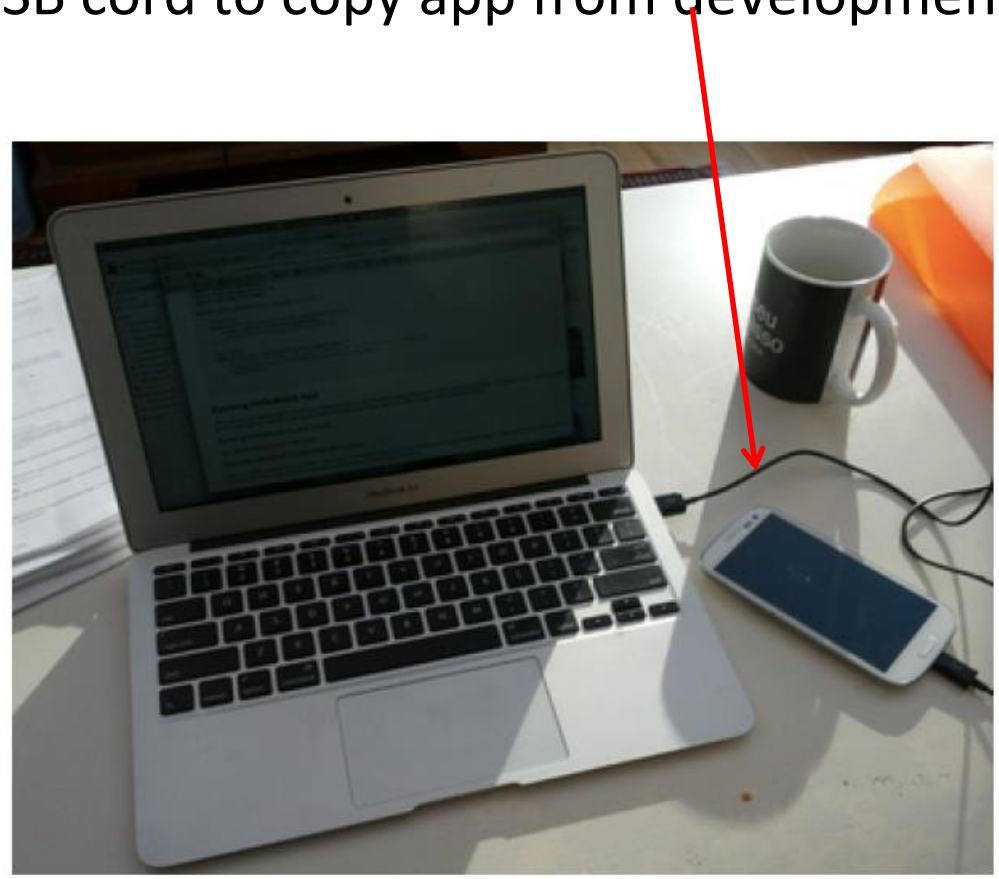
Where to Run Android App

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

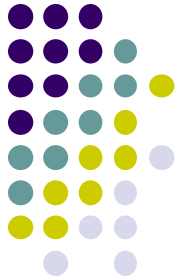


Running Android App on Real Phone

- Need USB cord to copy app from development PC to phone



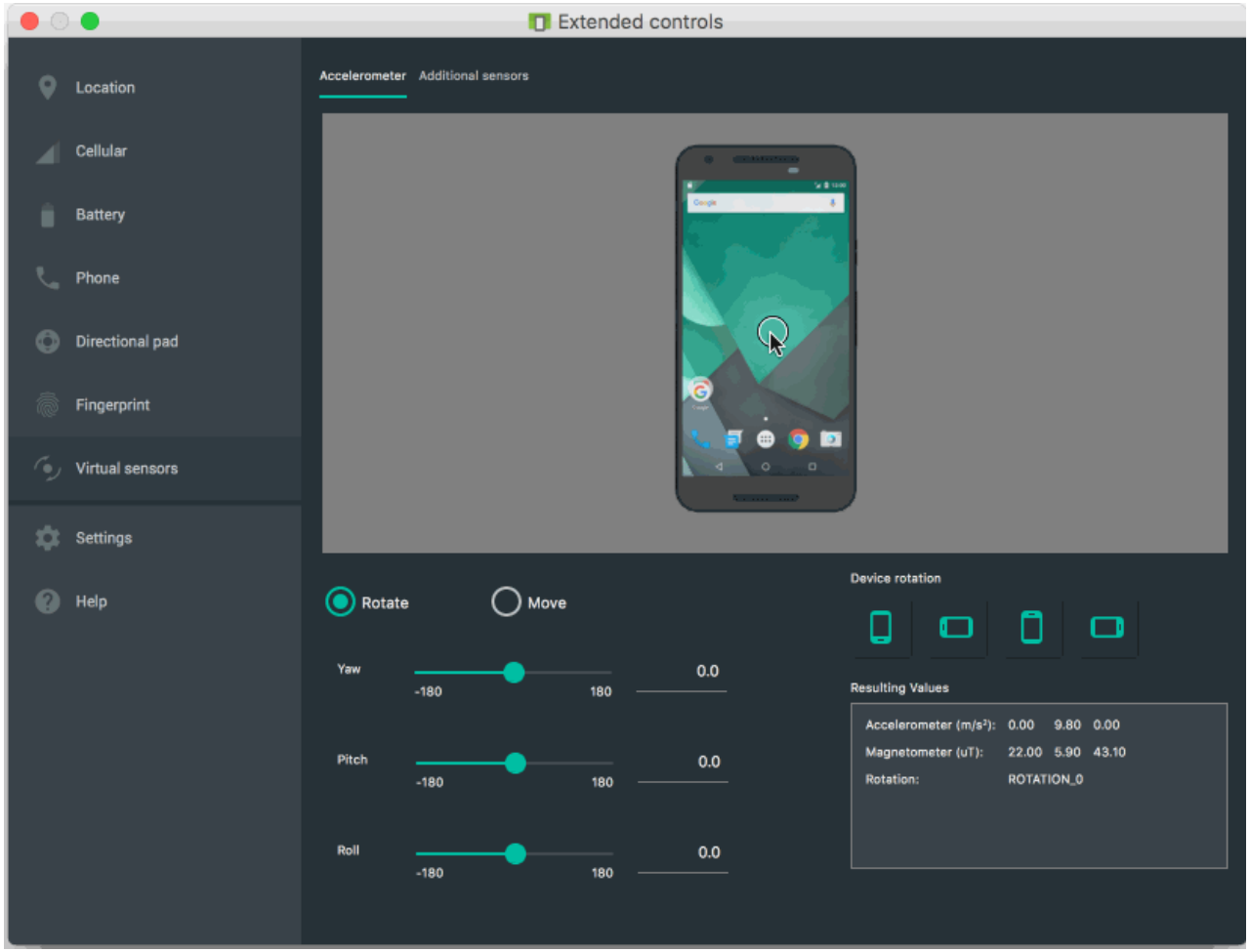
Emulator Pros and Cons (Vs Real Phone)

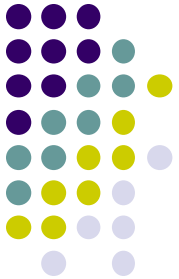


- Pros:
  - Conveniently test app on basic hardware by clicking in software
  - Easy to test app on various emulated devices (phones, tablets, TVs, etc), various screen sizes
- Cons:
  - Limited support, access to hardware, communications, sensors
  - E.g. GPS, camera, video recording, making/receiving phone calls, Bluetooth devices, USB devices, battery level, sensors, etc
  - Slower than real phone

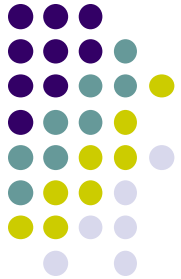
New Support for Sensors

- Can now emulate some sensors (e.g. location, accelerometer), but still limited





Demo: Android Studio



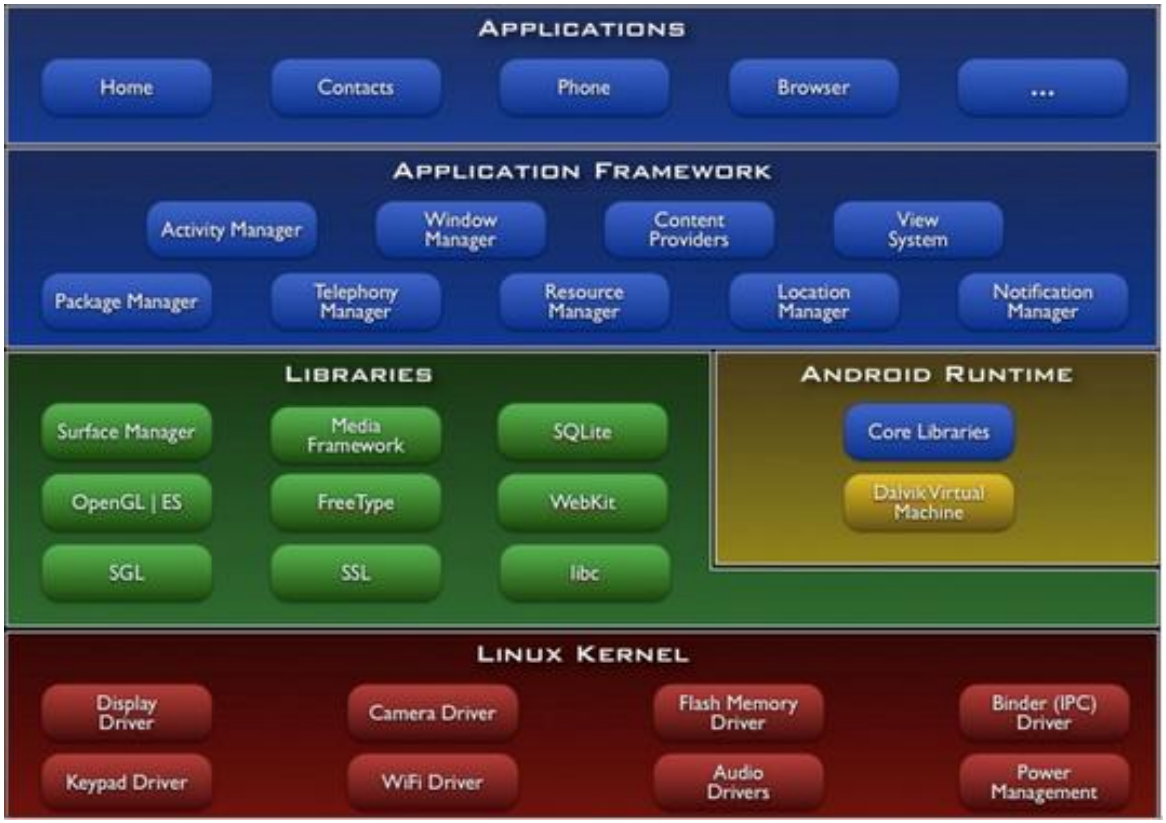
Android Software Framework

Android Functionality as Apps

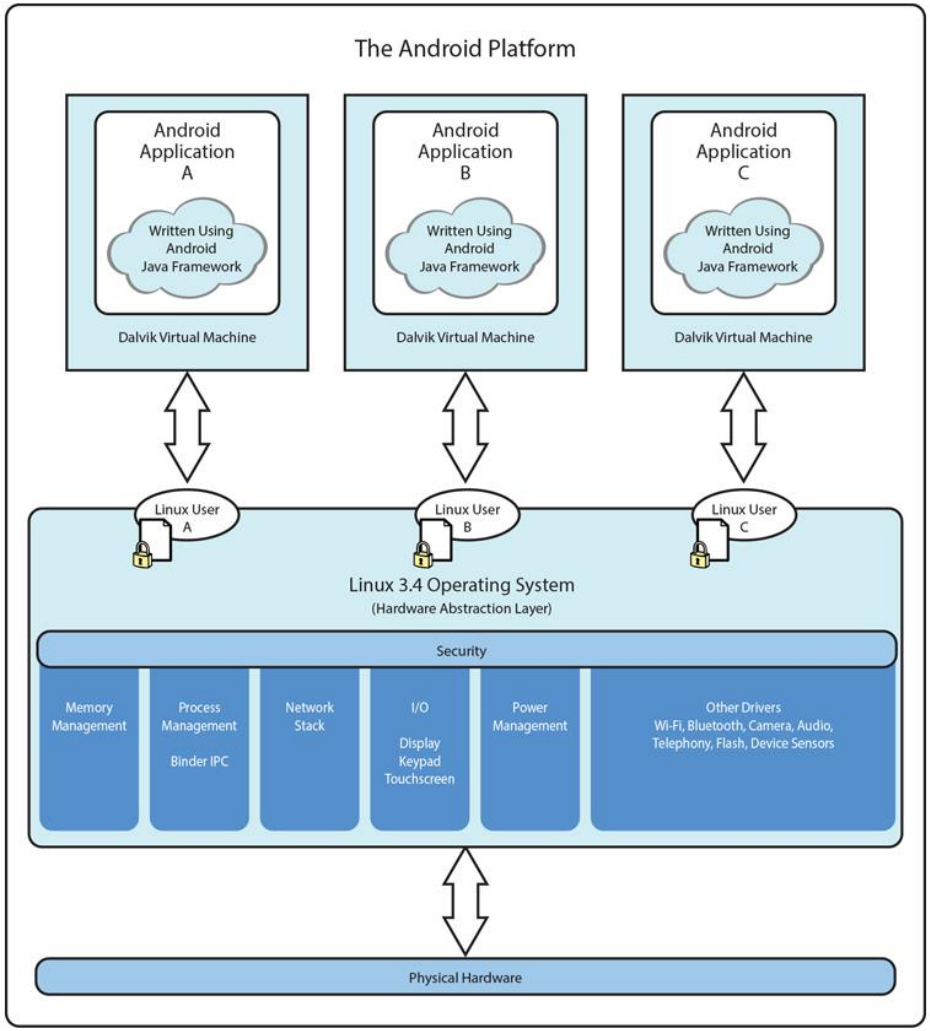
- Android functionality: collection of mini-applications (apps)
- Even dialer, keyboard, etc

Android Software Framework

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



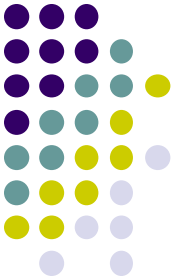
Android Software Framework



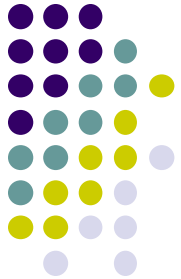
- Each Android app runs in its own security sandbox (VM, minimizes complete system crashes)
- Android OS multi-user Linux system
- Each app is a different user (assigned unique Linux ID)
- Access control: only process with the app's user ID can access its files

Ref: Introduction to Android Programming, Annuzzi, Darcey & Conder

Android Programming Languages



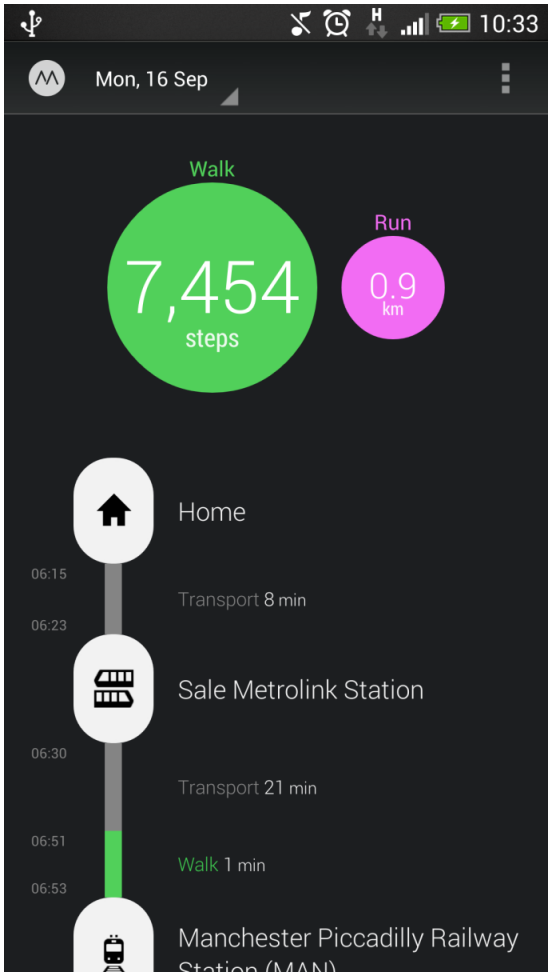
- Two main languages to program Android
  1. Java-based (Native) programming + XML:
     - We will focus on that in this class
  2. Kotlin:
     - New alternative way, Higher level, easier?
     - We will give overview of Kotlin later in class
     - Google is encouraging developers to switch to kotlin



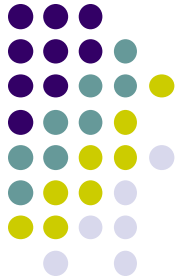
Android Apps: Big Picture

UI Design using XML

- UI design code (XML) separate from the program (Java)
- Why? Can modify UI without changing Java program
- **Example:** Shapes, colors can be changed in XML file without changing Java program
- UI designed using either:
  - Drag-and drop graphical (WYSIWYG) tool or
  - Programming Extensible Markup Language (XML)
- **XML:** Markup language, both human-readable and machine-readable''



Android App Compilation



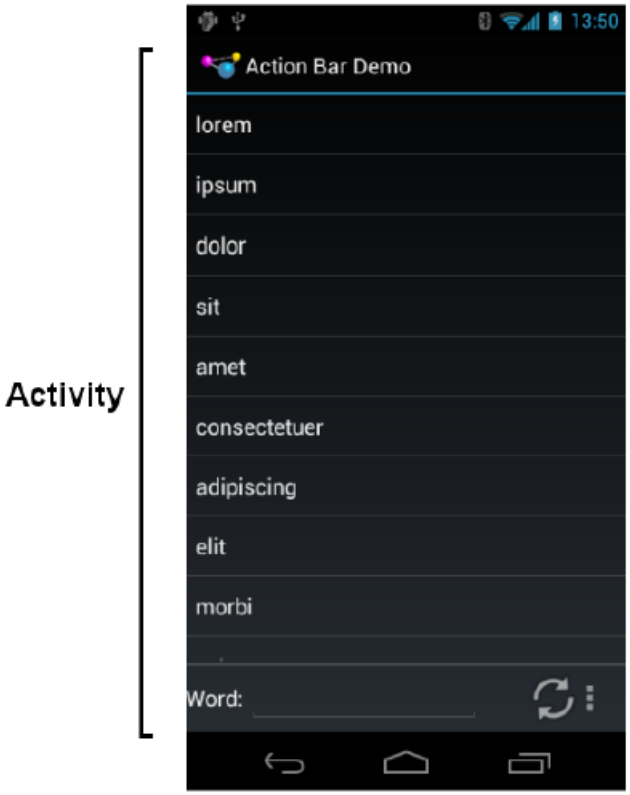
- Android Studio compiles code, data and resource files into **Android PacKage (filename.apk)**.
  - .apk is similar to .exe on Windows
- Apps download from Google Play, or copied to device as **filename.apk**
- Installation = installing **apk file**

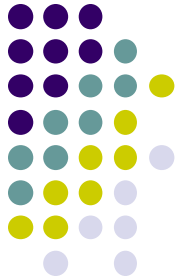
Activities

- Activity? 1 Android screen or dialog box
- Apps
  - Have at least 1 activity that deals with UI
  - Entry point, similar to **main( )** in C
  - Typically have multiple activities

- Example: A camera app
  - **Activity 1:** to focus, take photo, launch activity 2
  - **Activity 2:** to view photo, save it

- Activities
  - independent of each other
  - E.g. Activity 1 can write data, read by activity 2
  - App Activities derived from Android's **Activity** class





Our First Android App

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

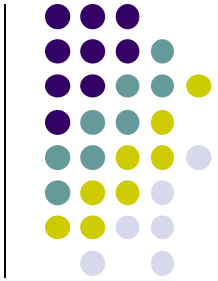
Next: Samples of AndroidManifest.xml
Hello World program

Start in AndroidManifest.xml
Read list of activities (screens)
Start execution from Activity tagged Launcher

Create/execute activities (declared in java files)
E.g. MainActivity.Java

Format each activity using layout In XML file (e.g. Activity_my.xml)

Inside “Hello World” AndroidManifest.xml



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

Your package name

Android version

List of activities (screens) in your app

```
<?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

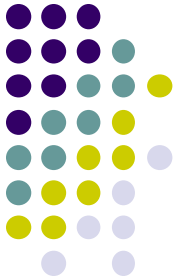
Execution Order

Start in AndroidManifest.xml
Read list of activities (screens)
Start execution from Activity tagged Launcher

Next → Create/execute activities (declared in java files)
E.g. MainActivity.Java

Format each activity using layout In XML file (e.g. Activity_my.xml)

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



Package declaration →

Import needed classes →

My class inherits from Android activity class →

Initialize by calling onCreate( ) method of base Activity class →

```
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);
  }
}
```

Note: Android calls your Activity's onCreate method once it is created

Use screen layout (design) declared in file main.xml

Execution Order

Start in AndroidManifest.xml
Read list of activities (screens)
Start execution from Activity tagged Launcher

Create/execute activities (declared in java files)
E.g. MainActivity.Java

Format each activity using layout In XML file (e.g. Activity_my.xml)

Next

Simple XML file Designing UI

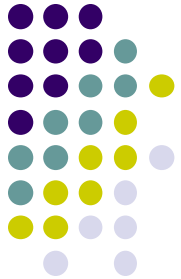
- After choosing the layout, then widgets added to design UI
- XML Layout files consist of:
  - UI components (boxes) called **Views**
  - Different types of views. E.g
    - **TextView:** contains text,
    - **ImageView:** picture,
    - **WebView:** web page
  - **Views** arranged into layouts or **ViewGroups**

```
<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>
```

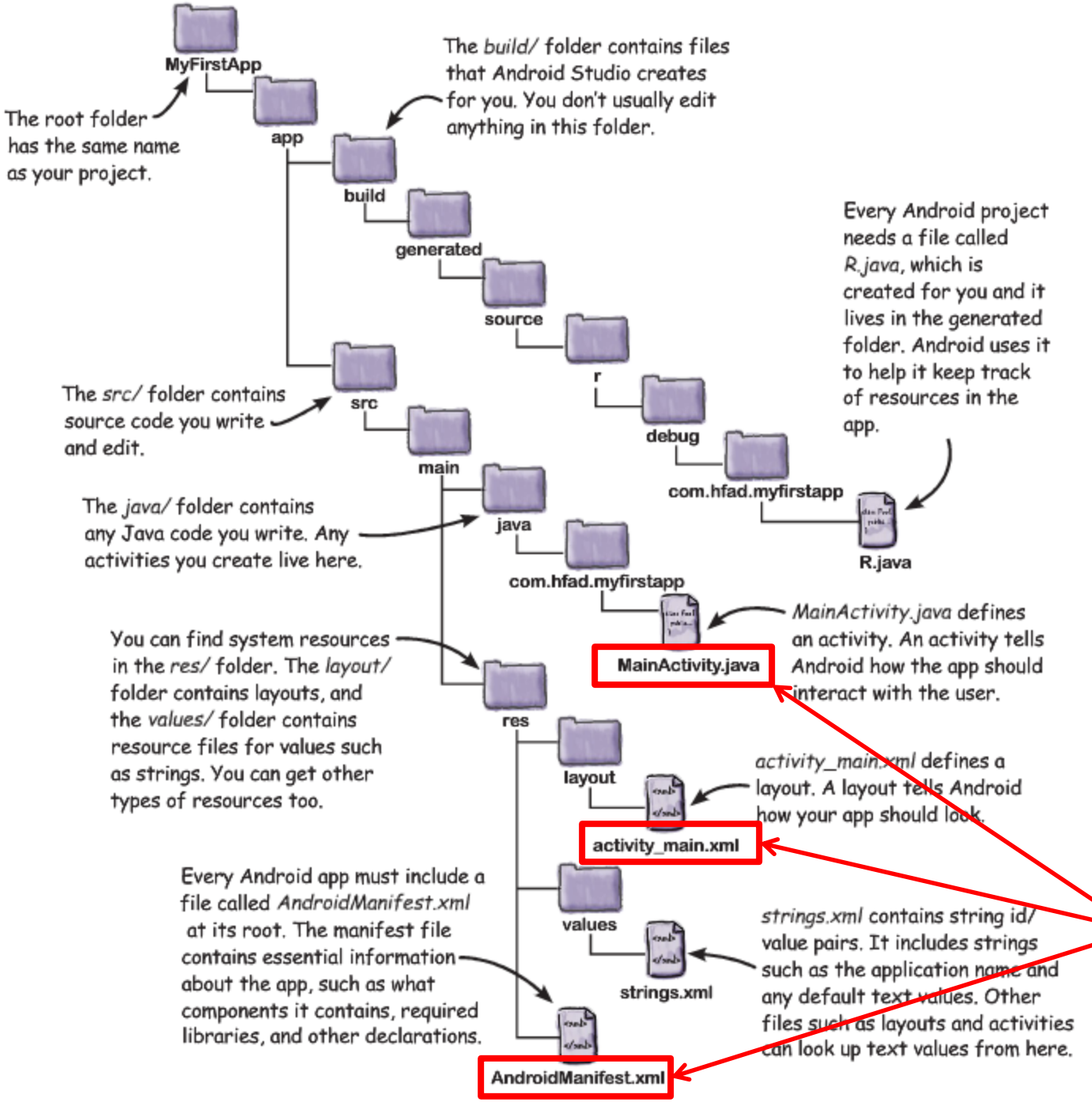
Declare Layout

Add widgets

Widget properties
(e.g. center contents
horizontally and vertically)



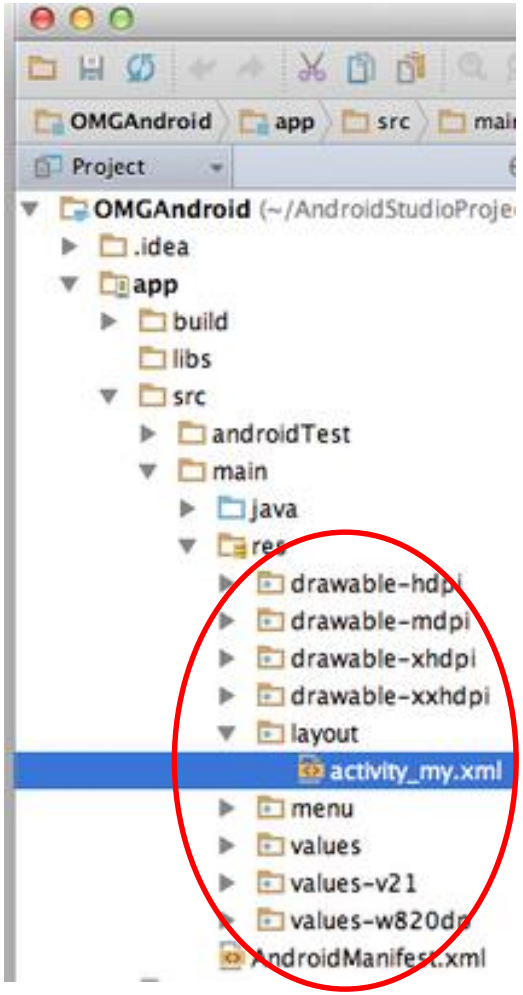
Android Files



Android Project File Structure

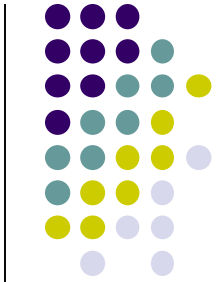
Files in an Android Project

- **res/** (resources) folder contains static resources you can embed in Android screen (e.g. pictures, string declarations, etc)
- **res/menu/:** XML files for menu specs
- **res/drawable-xyz/:** images (PNG, JPEG, etc) at various resolutions
- **res/raw:** general-purpose files (e.g. audio clips, mpeg, video files, CSV files
- **res/values/:** strings, dimensions, etc



Concrete Example: Files in an Android Project

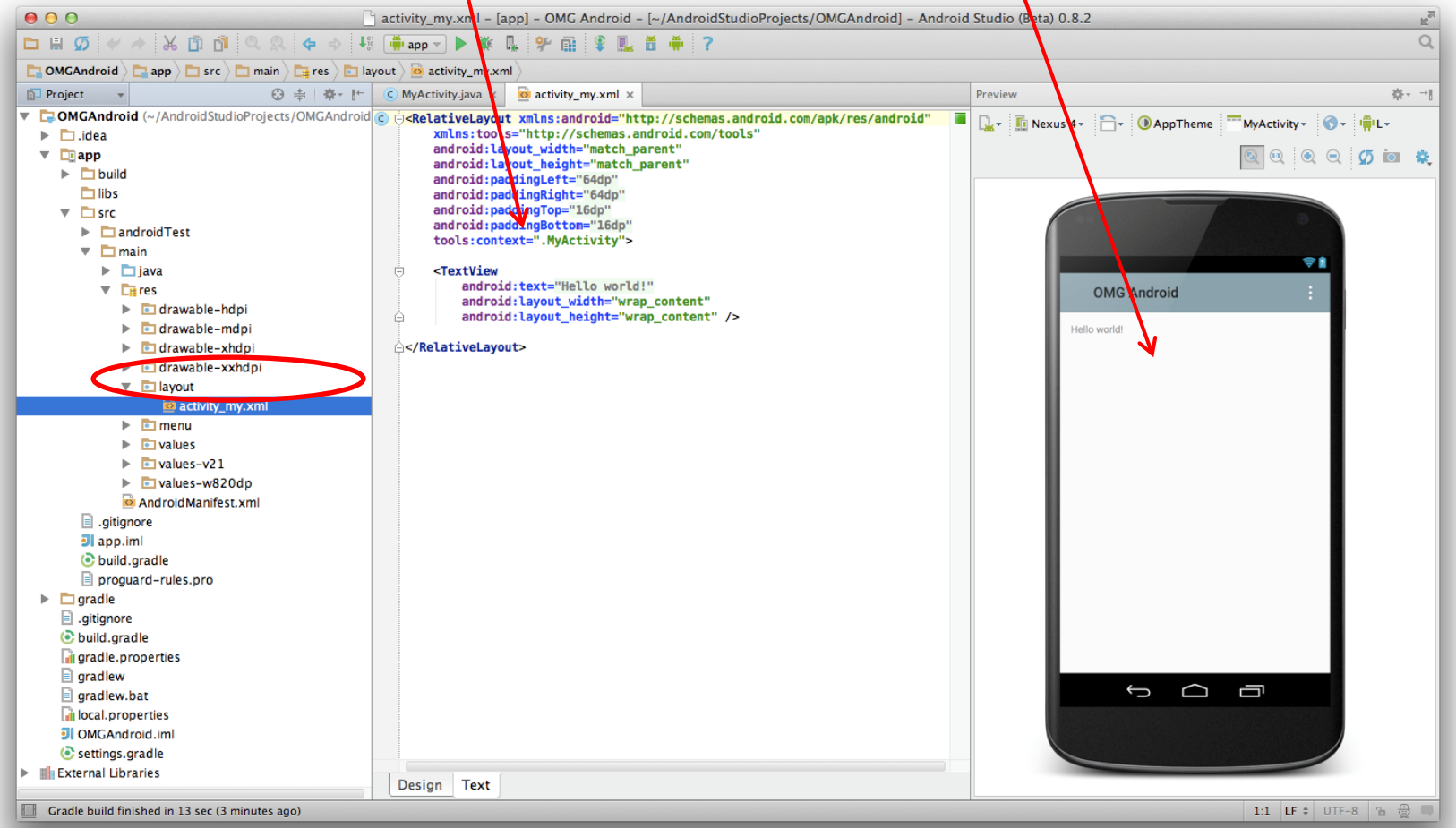
- **res/layout:** layout, dimensions (width, height) of screen cells are specified in XML file here
- **res/drawable-xyz/:** The images stored in jpg or other format here
- **java/:** App's response when user clicks on a selection is specified in java file here
- **AndroidManifext.XML:** Contains app name (Pinterest), list of app screens, etc

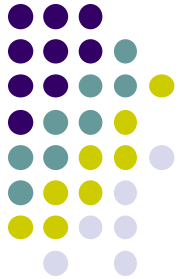


Editting in Android Studio

Editting Android

- Can edit apps in:
  - **Text View:** edit XML directly
  - **Design View:** or drag and drop widgets unto emulated phone





Android UI Design in XML

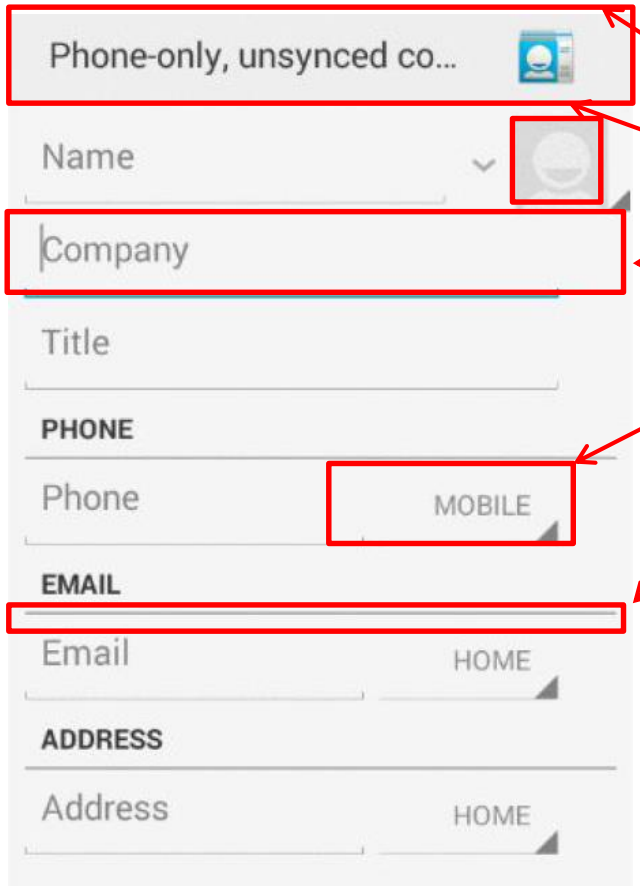
Recall: Files Hello World Android Project

XML file used to design Android UI

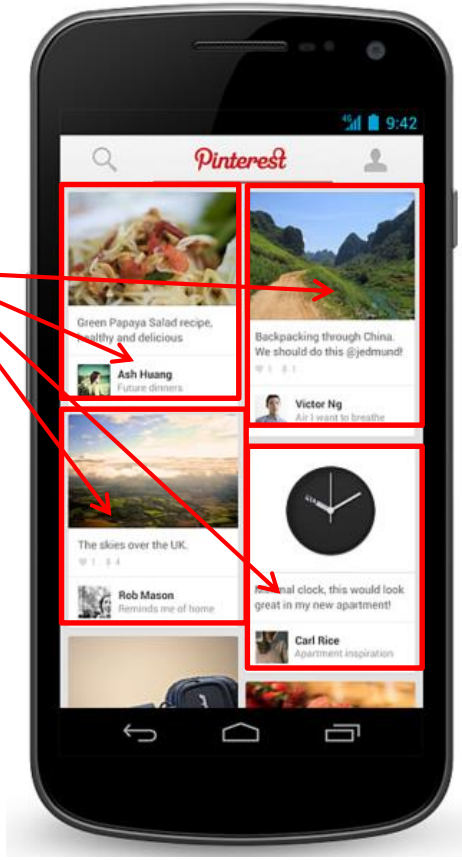
- 3 Files:
  - **Activity_main.xml:** XML file specifying screen layout
  - **MainActivity.Java:** Java code to define behavior, actions taken when button clicked (intelligence)
  - **AndroidManifest.xml:**
    - Lists all app components and screens
    - Like a table of contents for a book
    - E.g. Hello world program has 1 screen, so AndroidManifest.xml has 1 item listed
    - App starts running here (a bit like main( ) in C), launching activity with a tag “LAUNCHER”

Widgets

- ***Android UI design involves arranging widgets on a screen***
- **Widgets?** Rectangles containing texts, image, etc
- **Screen design:** Pick widgets, specify attributes (dimensions, margins, etc)

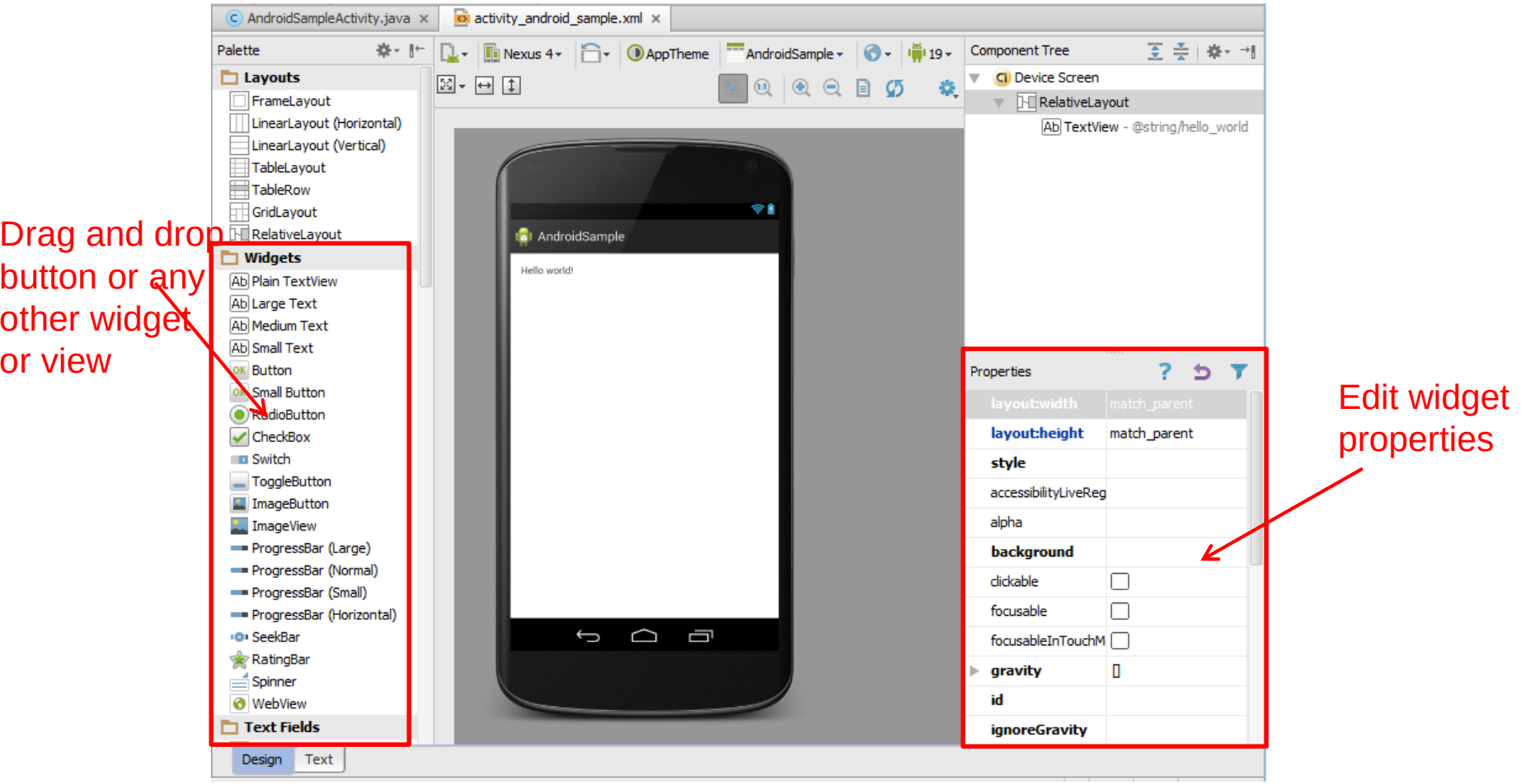


Widgets



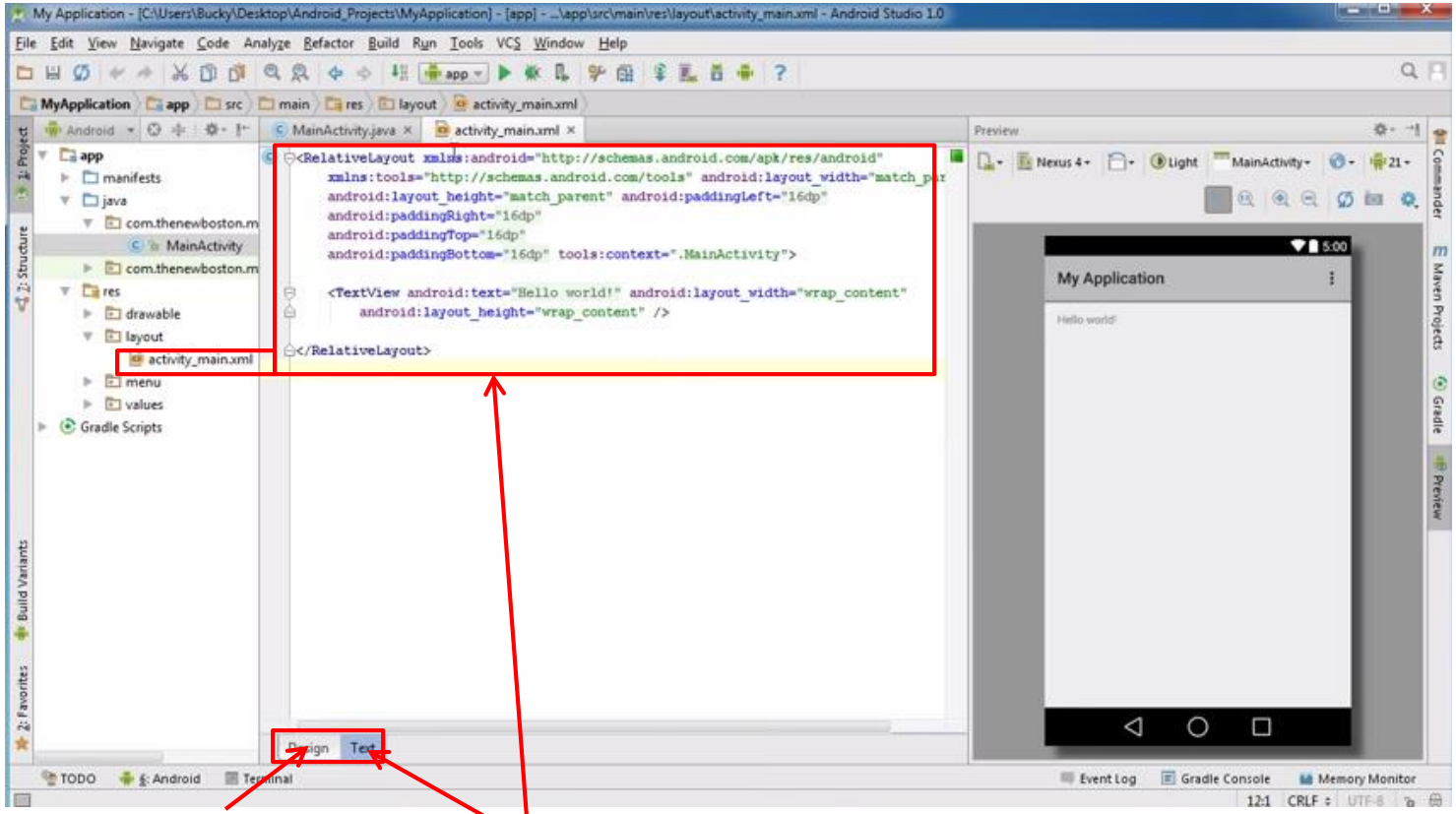
Design Option 1: Drag and Drop Widgets

- Drag and drop widgets in Android Studio Design View
- Edit widget properties (e.g. height, width, color, etc)

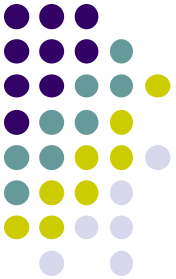


Design Option 2: Edit XML Directly

- **Text view:** Directly edit XML file defining screen (activity_main.xml)
- **Note:** dragging and dropping widgets in design view auto-generates corresponding XML in Text view



References



- Android App Development for Beginners videos by Bucky Roberts (thenewboston)
- Ask A Dev, Android Wear: What Developers Need to Know, https://www.youtube.com/watch?v=zTS2NZpLyQg
- Ask A Dev, Mobile Minute: What to (Android) Wear, https://www.youtube.com/watch?v=n5Yjzn3b_aQ
- Busy Coder's guide to Android version 4.4
- CS 65/165 slides, Dartmouth College, Spring 2014
- CS 371M slides, U of Texas Austin, Spring 2014