Android Development (301)

•••

Introduction to the course and Android development

Prerequisite

202 - Java Development

Basic Java Programming knowledge

Object-oriented Programming

Data structures (ArrayList, HashMap, etc)

What is Android?

- Mobile OS maintained by Google
- Originally purchased from Android, Inc. in 2005
- Runs on phones, tables, watches, TVs, ...
- Based on Java (dev language) and Linux(kernel)
- The #1 mobile OS worldwide, and now #1 overall OS worldwide!
- Has over 1 million apps published in Play Store
- Code is released as open source (periodically)
  - Easier to customize, license, pirate, etc. than iOS

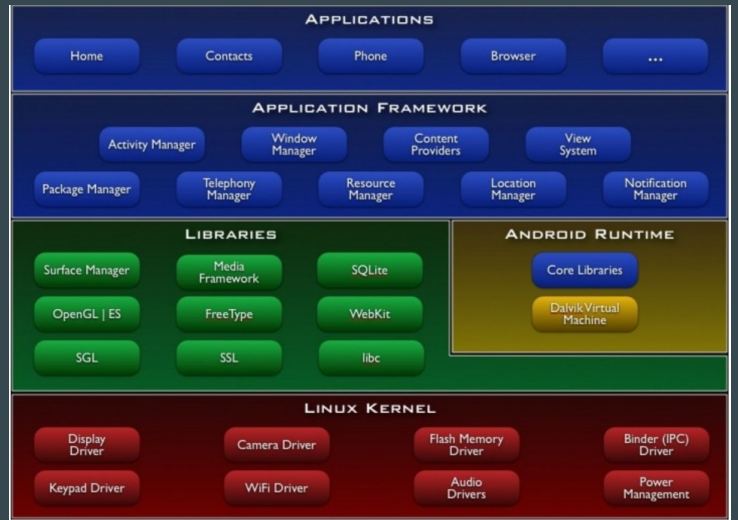
Why develop for Android?

- Why not just write a website? Android has a browser...
- Better, snappier UI with a more consistent user experience
- Able to use different kinds of widgets/controls than a web page.
- More direct access to the device's hardware (camera, GPS, etc.)
- Users highly prefer apps over mobile web browsing

iOS vs Android?

Android Architecture

- Android OS provides libraries for many system features like contacts, phone dialing, notifications, 2D/3D graphics, database access, security / encryption, camera, audio, input/output ...
- Android Java code is compiled into a special **Dalvik** binary format.



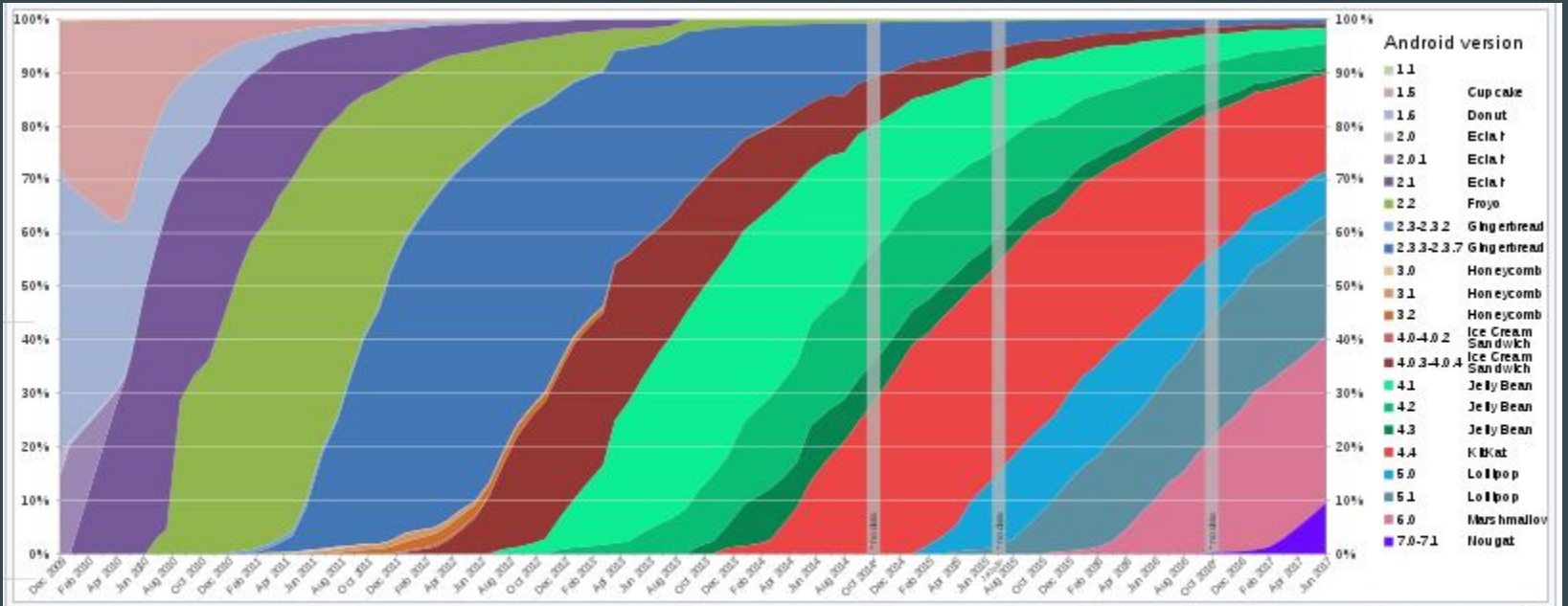
Android Version History

Code name	Version number	Initial release date	API level	Security patches[1]
(No codename)[2]	1.0	September 23, 2008	1	Unsupported
(Internally known as "Petit Four")[2]	1.1	February 9, 2009	2	Unsupported
Cupcake	1.5	April 27, 2009	3	Unsupported
Donut[3]	1.6	September 15, 2009	4	Unsupported
Eclair[4]	2.0 – 2.1	October 26, 2009	5 – 7	Unsupported
Froyo[5]	2.2 – 2.2.3	May 20, 2010	8	Unsupported
Gingerbread[6]	2.3 – 2.3.7	December 6, 2010	9 – 10	Unsupported
Honeycomb[7]	3.0 – 3.2.6	February 22, 2011	11 – 13	Unsupported
Ice Cream Sandwich[8]	4.0 – 4.0.4	October 18, 2011	14 – 15	Unsupported
Jelly Bean[9]	4.1 – 4.3.1	July 9, 2012	16 – 18	Unsupported
KitKat[10]	4.4 – 4.4.4	October 31, 2013	19 – 20	Unsupported[11]
Lollipop[12]	5.0 – 5.1.1	November 12, 2014	21 – 22	Supported
Marshmallow[13]	6.0 – 6.0.1	October 5, 2015	23	ported
Nougat[14]	7.0 – 7.1.2	August 22, 2016	24 – 25	Supported
Oreo[15]	**8.0 – 8.1**	August 21, 2017	26 – 27	Supported

Sort ascending

Legend: Old version | Older version, still supported | **Latest version**

Android version distribution



Global Android version distribution since December 2009, as of June 2017. As of December, Android Marshmallow is the most widely used version of Android, running on 29.7% of all Android devices accessing Google Play, while Android Lollipop runs on 26.3% of devices (79.8% on it or newer).

Version issues

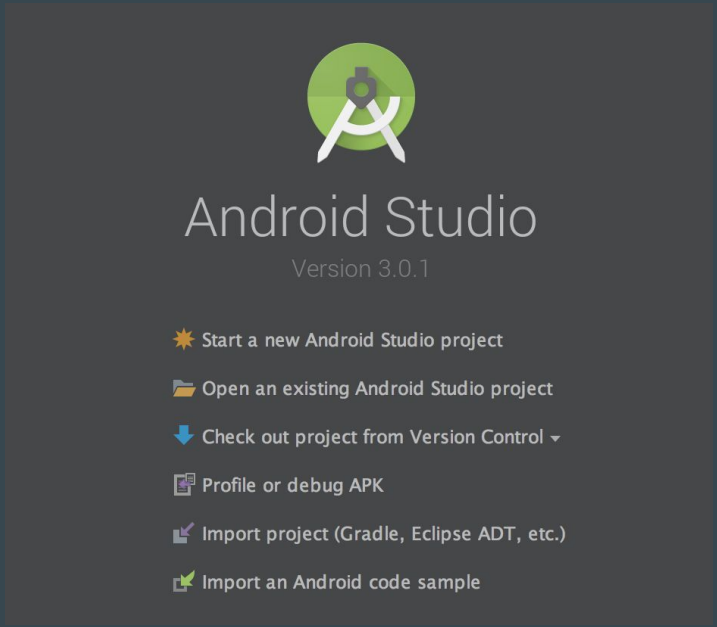
- Check your phone's version of Android:
  - Settings → System → About Device → Android version
  - "Why wouldn't my phone have the newest Android version? Can't I just update it?"

- Several companies affect whether your device is up-to-date:
  - Google; phone manufacturer; service provider; ...

- If any company in the chain doesn't want to push out an update for your device, it can become out of date.

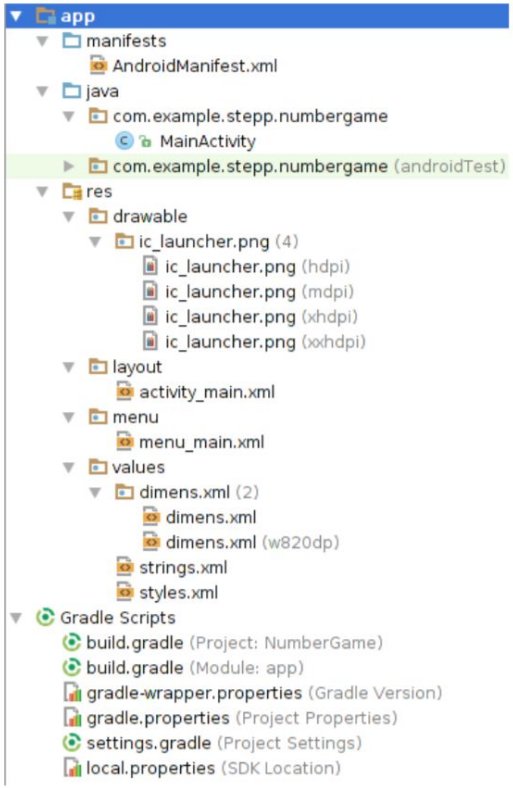
Android Studio

- Google's official Android IDE, in v3.0.1 as of January of 2018
  - Based on IntelliJ IDEA editor; free to download and use



Project structure

- **AndroidManifest.xml**
  - overall project config and settings
- **src/java/...**
  - source code for your Java classes
- **res/...** = resource files *(many are **XML**)*
  - drawable/ = images
  - layout/ = descriptions of GUI layout
  - menu/ = overall app menu options
  - values/ = constant values and arrays
  - strings = localization data
  - styles = general appearance styling
- **Gradle**
  - a build/compile management system
  - **build.gradle** = main build config file



Virtual Devices (AVDs)

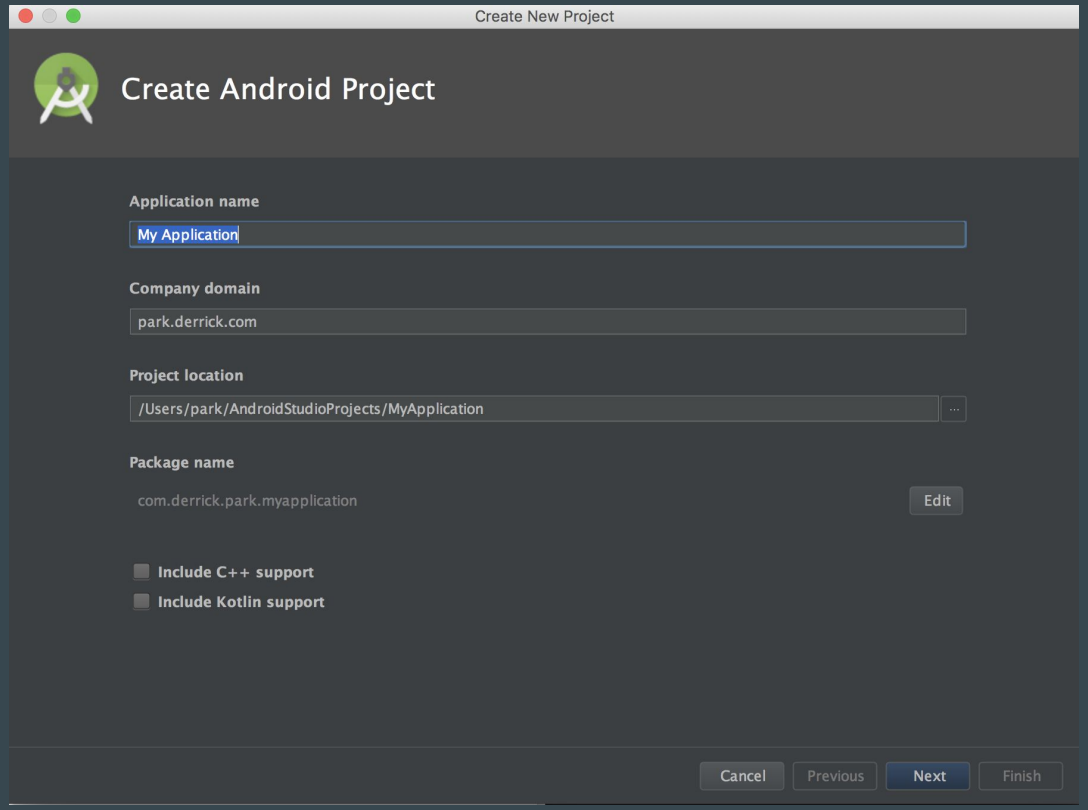
- allows you to run your project in an emulator
  - a software simulation of an entire Android tablet, phone, watch
  - when you click the "Run" button in Android Studio,
    it builds your app, installs it on the virtual device, and loads it
- must set up virtual device first
  in Android Studio
- alternative: install your app
  on your actual Android device!
  - pro: app will run faster,
    better test of real execution
  - con: requires Android device,
    must be plugged into dev PC

Creating Our First Android App

Top-down Design

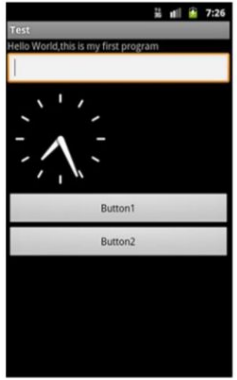
- Let's start from a design of an app that we want to create and then learn the necessary skills to build that app.
- "The Bigger The Better" game.
  - User is shown two numbers
  - Must choose which one is bigger by clicking on the right button
  - Game pops up brief "correct" / "incorrect" message after each guess
  - Get points for each correct answer
- We need to learn:
  - How to create and position graphical widgets
  - How to respond to user events

Creating a new project



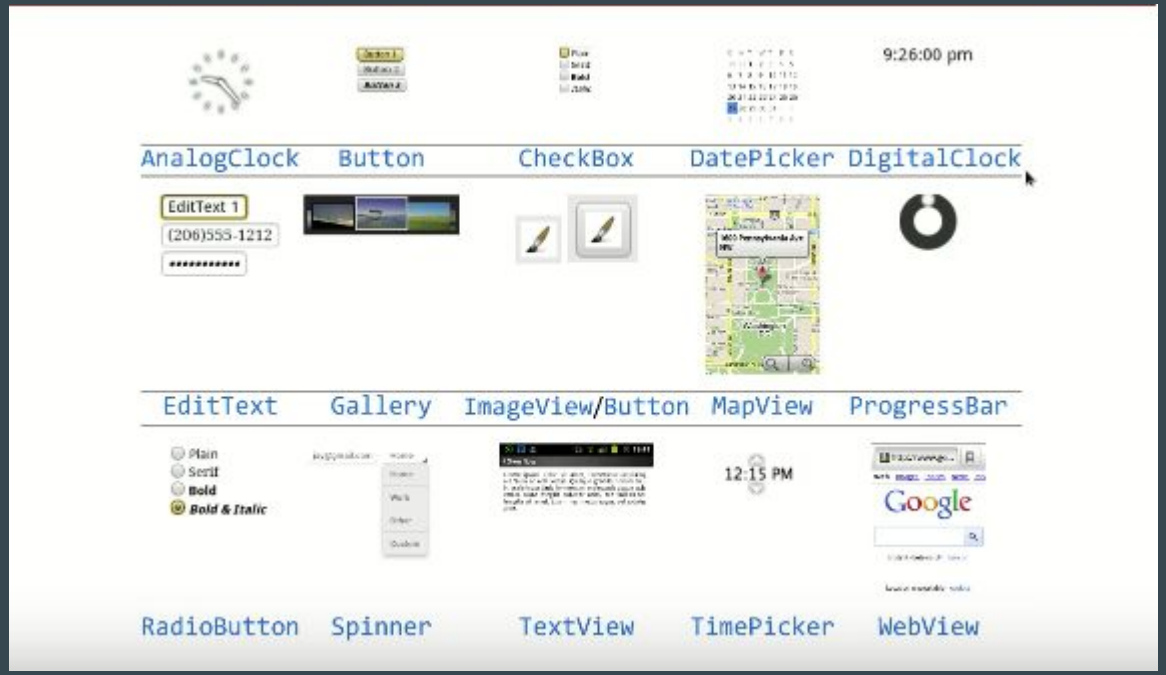
Android terminology

- **activity**: a single screen of UI that appears in your app
  - the fundamental units of GUI in an Android app



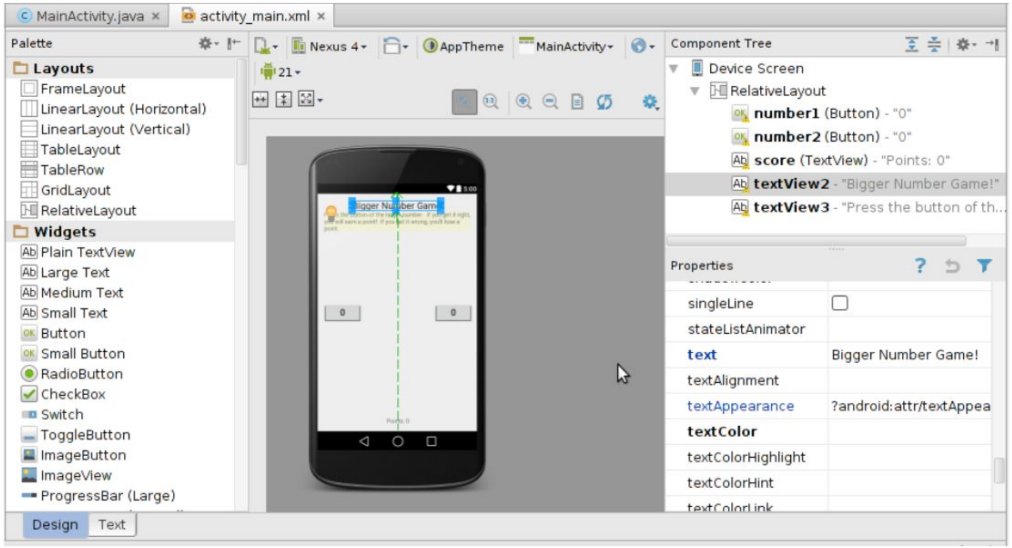
- **view**: items that appear onscreen in an activity
  - **widget**: GUI control such as a button or text field
  - **layout**: invisible container that manages positions/sizes of widgets
- **event**: action that occurs when user interacts with widgets
  - e.g. clicks, typing, scrolling
- **action bar**: a menu of common actions at top of app
- **notification area**: topmost system menu and icons

Android widgets



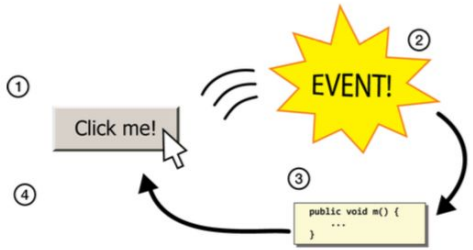
Designing a UI

- open XML file for your layout (e.g. `activity_main.xml`)
- drag widgets from left **Palette** to the preview image
- set their properties in lower-right **Properties** panel



Events

- **event**: An external stimulus your program can respond to.
- Common kinds of events include:
  - Mouse motion / tapping, Keys pressed,
  - Timers expiring, Network data available



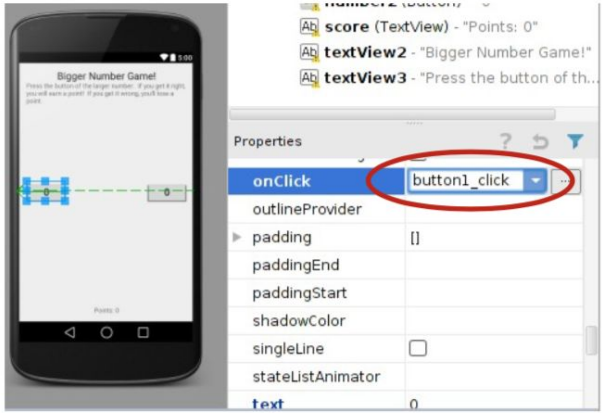
- **event-driven programming**: Overall execution of your program is largely dictated by user events.
  - Commonly used in graphical programs.

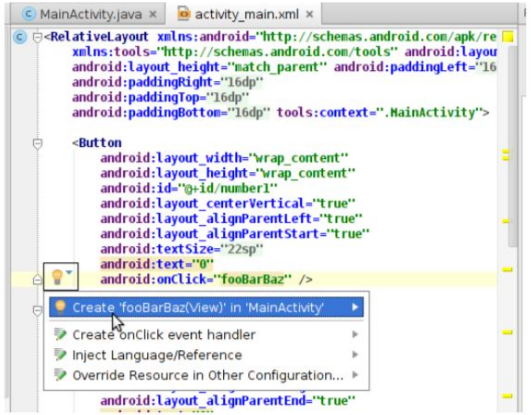
- To respond to events in a program, you must:
  - Write methods to handle each kind of event ("listener" methods).
  - Attach those methods to particular GUI widgets.

No main()

Setting an event listener

- select the widget in the **Design** view
- scroll down its **Properties** until you find `onClick`
- type the name of a method you'll write to handle the click
- switch to the **Text view** and find the XML for that button
- click the "Light Bulb" and choose to "**Create**" the method





Event listener Java code

MainActivity.java × activity_main.xml ×

```java
1   package com.example.stepp.numbergame;
2
3   import ...
8
9   public class MainActivity extends ActionBarActivity {
10      @Override
11      protected void onCreate(Bundle savedInstanceState) {
12          setContentView(R.layout.activity_main);
13          super.onCreate(savedInstanceState);
14      }
15
16      public void button1_click(View view) {
17          // your code goes here
18
19      }
```

View objects

- each widget has an associated Java object you can access
- they are subclasses of parent class **`View`**
  - examples: `Button`, `TextView`, `EditText`, ...

- `View` objects have many `get` and `set` methods that correspond to the properties in the Design view:
  - background, bottom, ID, left, margin, padding, right, text, textAlignment, textSize, top, typeface, visibility, x, y, z, ...
  - example: for a Button's **text** property, there will be methods:

```
public String getText()
public void setText(String text)
```

  - Find list of properties in Design view, or typing ".get" on a button in Java code, or at: https://developer.android.com/reference/

Interacting with widgets

- accessing a widget in the Java code:
  1. in Design view, give that view a unique **ID** property value
  2. in Java code, call `findViewById` to access its View object
     - pass it a parameter of `R.id.`***`your_unique_ID`***
     - cast the returned value to the appropriate type (`Button`, `TextView`, etc.)

```
public void button1_onclick(View view) {
    TextView tv = (TextView) findViewById(R.id.mytextview);
    tv.setText("You clicked it!");
}
```

Finishing our app (Demo)

Displaying Toasts

```
Toast.makeText(this,
               "message",
               duration).show();
```

- where ***duration*** is `Toast.LENGTH_SHORT` or `LENGTH_LONG`

- A "Toast" is a pop-up message that appears for a short time.
- Useful for displaying short updates in response to events.
- Should not be relied upon extensively for important info.