CS 193A

Creating our first Android app

What is CS 193A?

- **CS 193A**:
  Intro to Android Development
  - 1 unit, pass/fail
  - no TAs/SLs; rely on Piazza and peer help
  - sharing of ideas and code encouraged
  - fewer rigid HW requirements;
    creativity and exploration encouraged
  - peer grading component
  - "pass" if you submit reasonable work on most HW

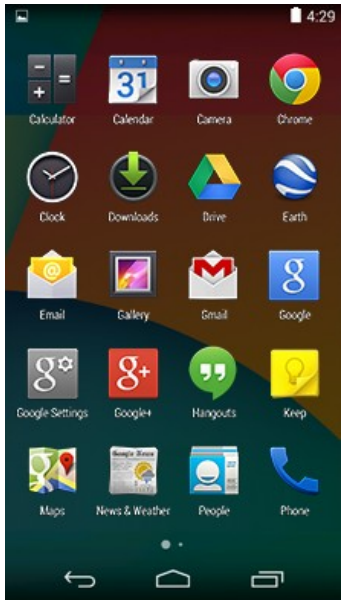
Stanford | ENGINEERING
Computer Science

- Prerequisite: CS 106B or equivalent
  - basic Java programming knowledge
  - object-oriented programming
  - data structures (`ArrayList`, `HashMap`, etc.)

What is Android?

- mobile operating system maintained by **Google**
  - originally purchased from Android, Inc. in 2005
- runs on phones, tablets, 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



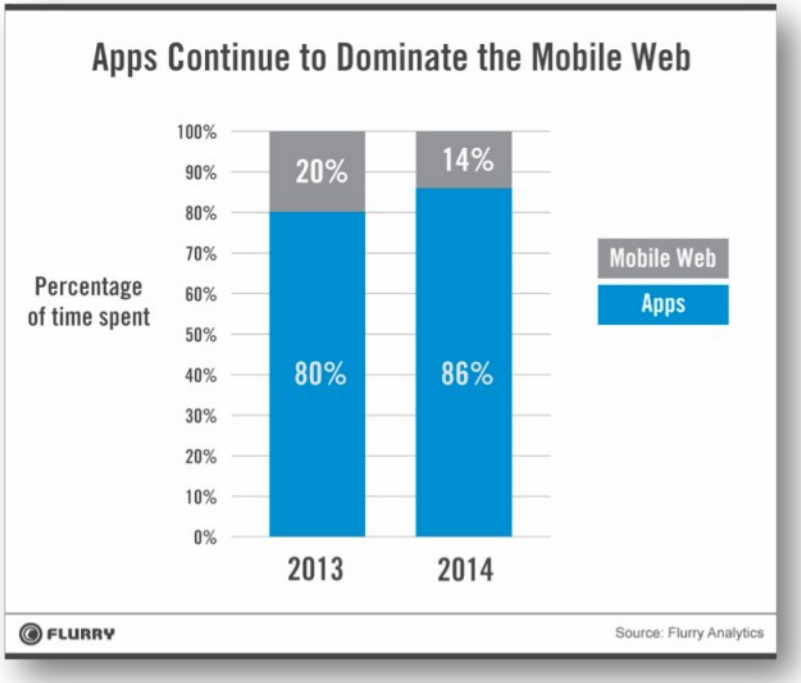
- code is released as open source (periodically)
  - easier to customize, license, pirate, etc. than iOS

Why develop for Android?

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

Mobile Web App

Native App on iOS



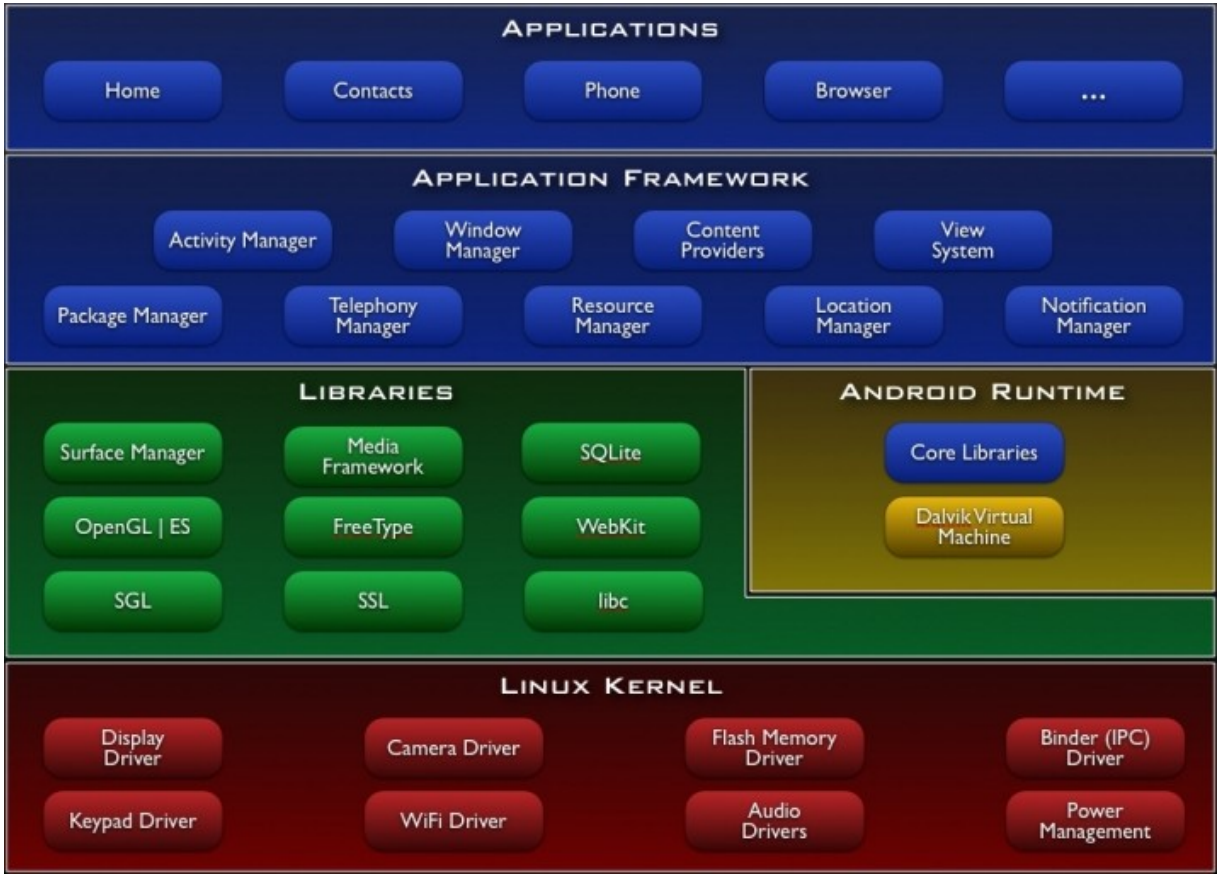
Why not iOS?

- Why not write apps for **iOS**, which runs on iPhones and iPads?
  - familiar programming language
    (Java instead of Obj-C or Swift)
  - free developer tools
    (Apple charges $$$ for theirs)

  - more liberated app store
    (can make an app and put on your phone or others')
  - Android has a larger install base
  - there is already a **CS 193P** class for building iOS apps! Take it!

Android architecture

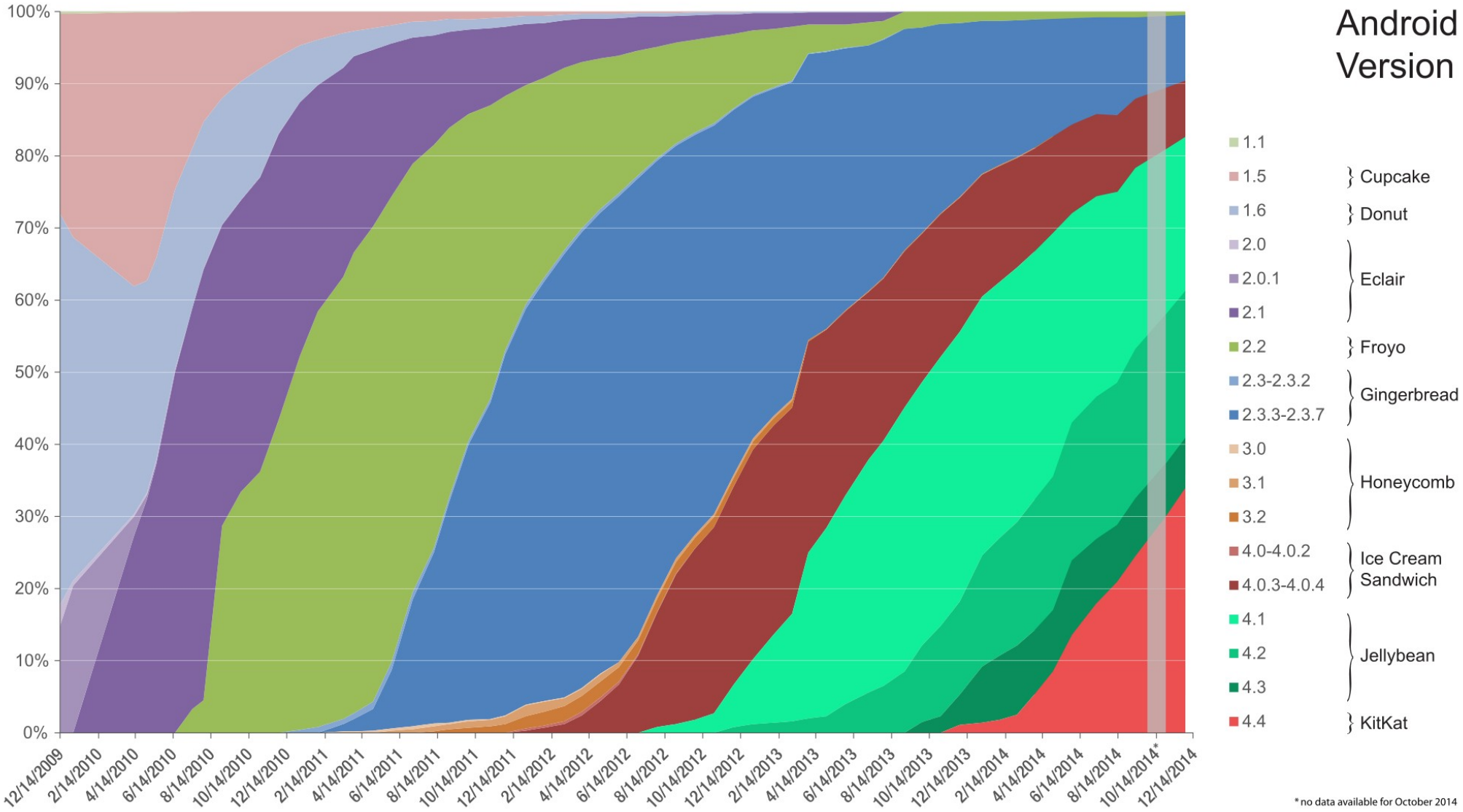
- Android OS provides libraries for many system features like contacts, phone dialing, notifications, 2D/3D graphics, database access, security / encryption, camera, audio, input/output, ...
  - Android Java code is compiled into a special **Dalvik** binary format



Android version history (link)

Version	API level	Date	Name
1.0-1.1	1,2	Sep 2008	none
1.5	3	Apr 2009	Cupcake
1.6	4	Sep 2009	Donut
2.0-2.1	5,6,7	Oct 2009	Eclair
2.2	8	May 2010	Froyo
2.3	9,10	Dec 2010	Gingerbread
3.0	11,12,13	Feb 2011	Honeycomb
4.0	14,15	Oct 2011	Ice Cream Sandwich
4.1-4.3	16,17,18	Jun 2012	Jelly Bean
4.4	19,20	Sep 2013	Kit Kat
5.0	21	Jun 2014	Lollipop

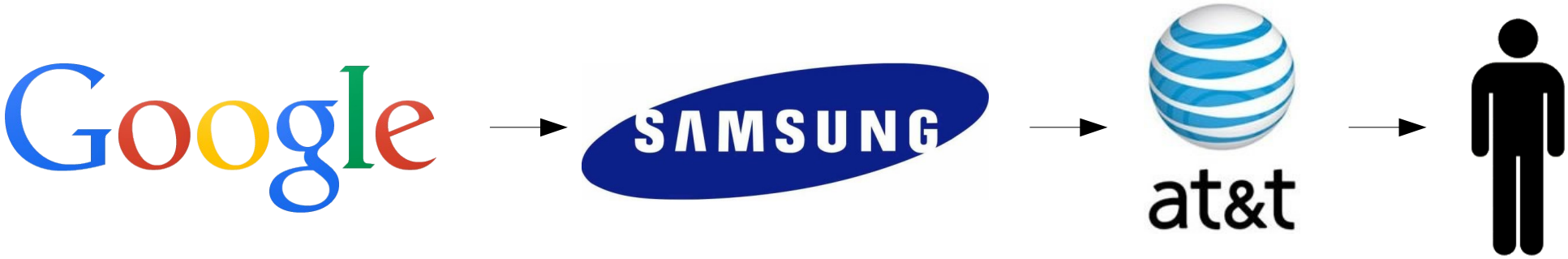
Android version distribution



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?"

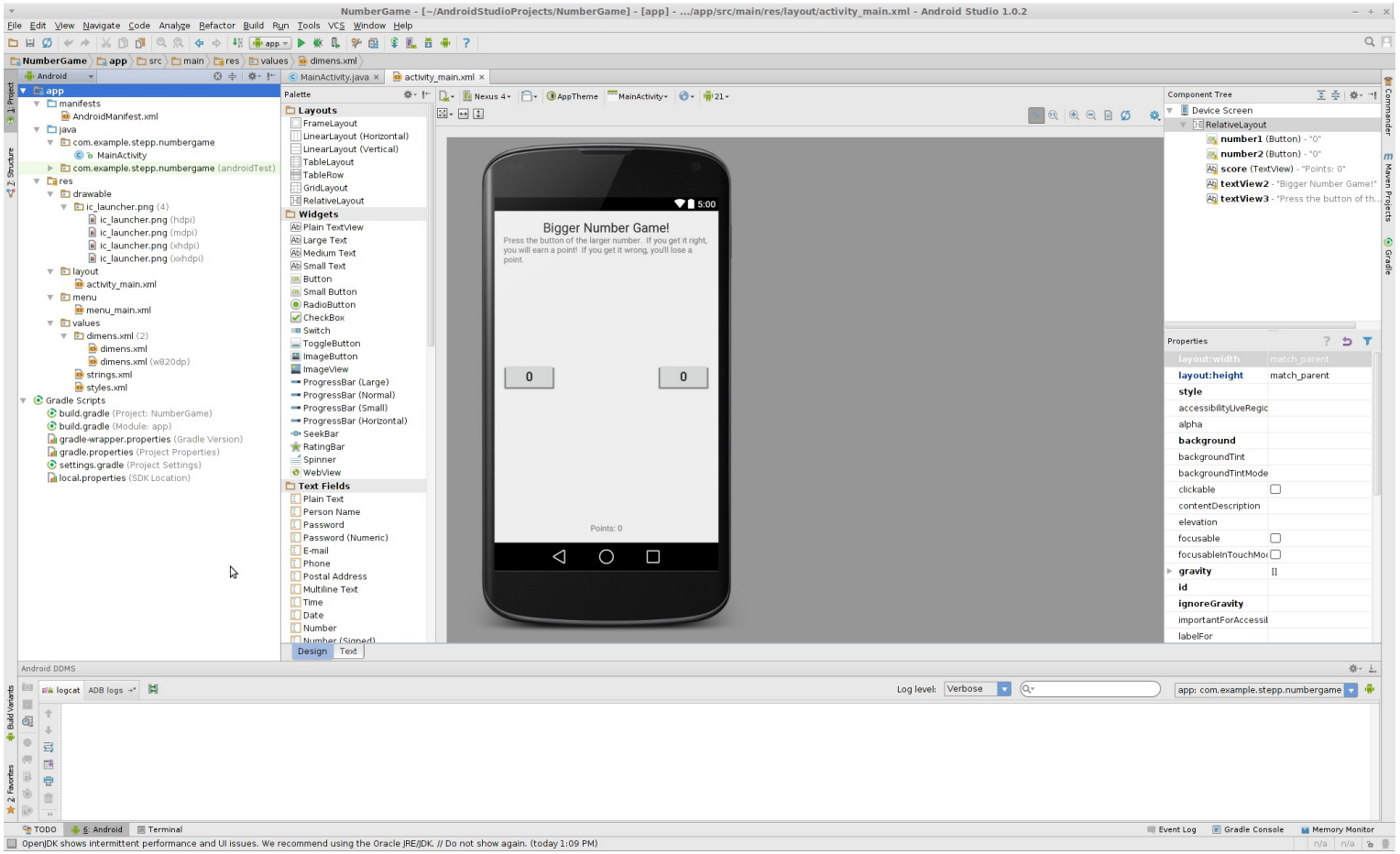
- Several companies affect whether your device is up-to-date:
  - Google; phone manufacturer; service provider; ...



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

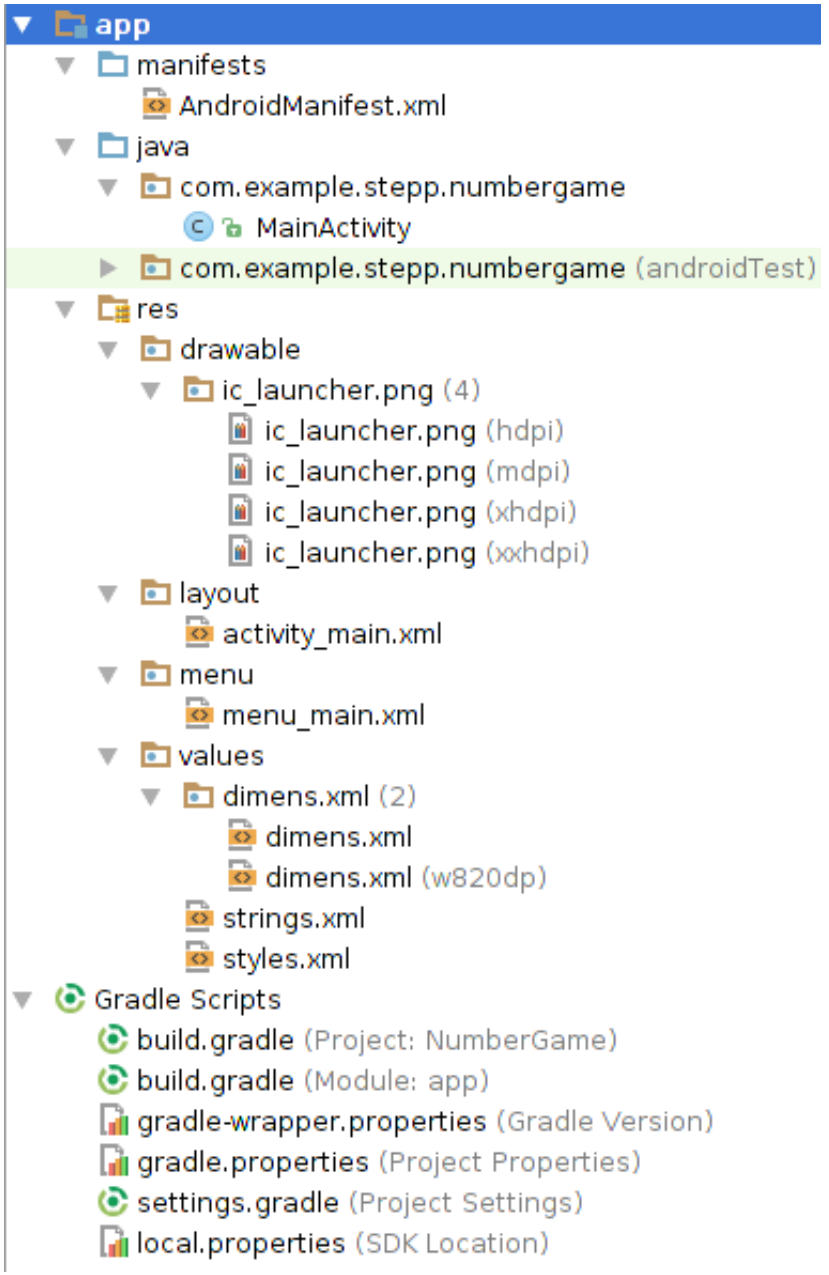
Android Studio

- Google's official Android IDE, in v1.0 as of November 2014
  - replaces previous Eclipse-based environment
  - based on IntelliJ IDEA editor; free to download and use



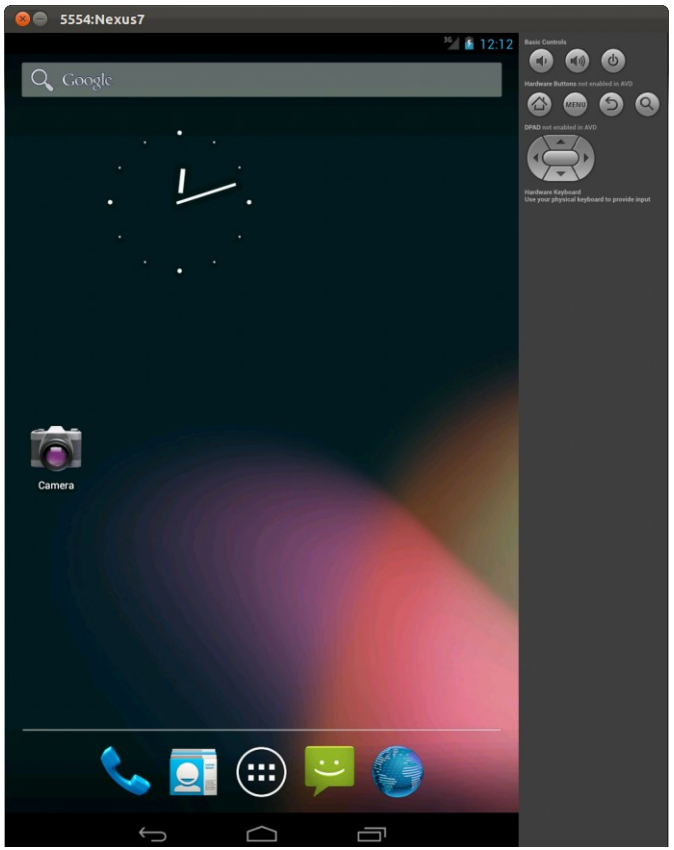
Project structure

- **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

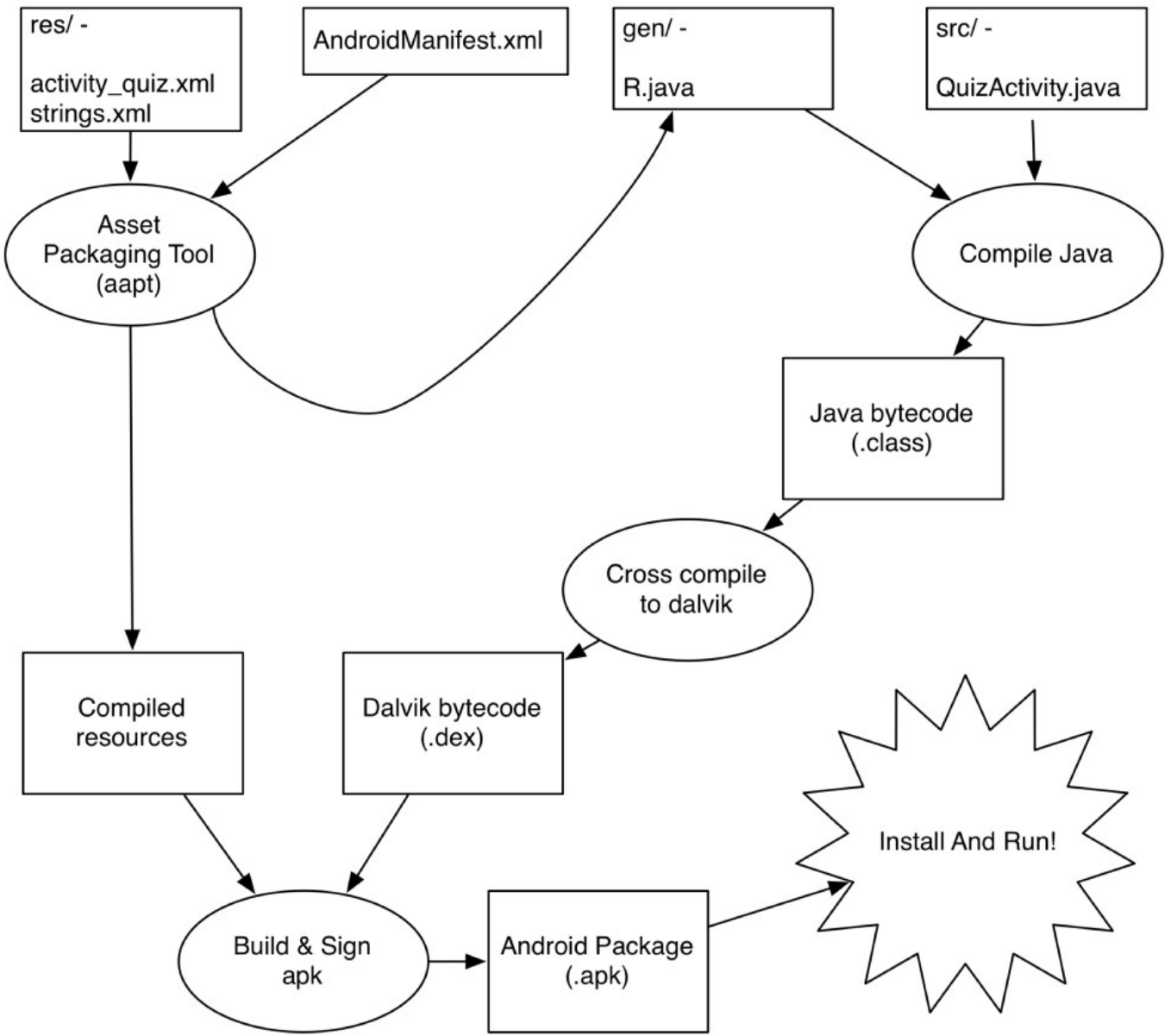


Virtual Devices (AVDs)

- 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



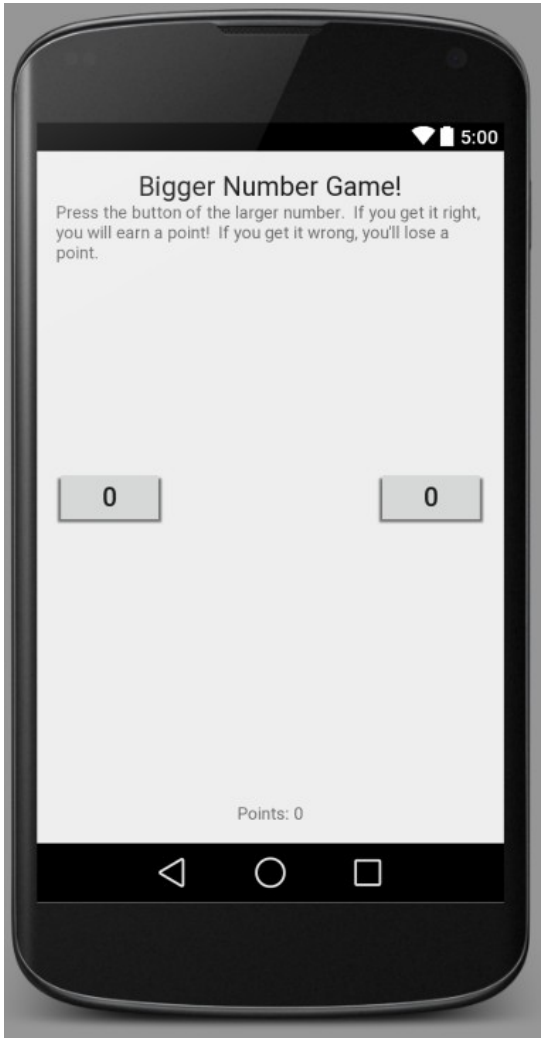
App build process



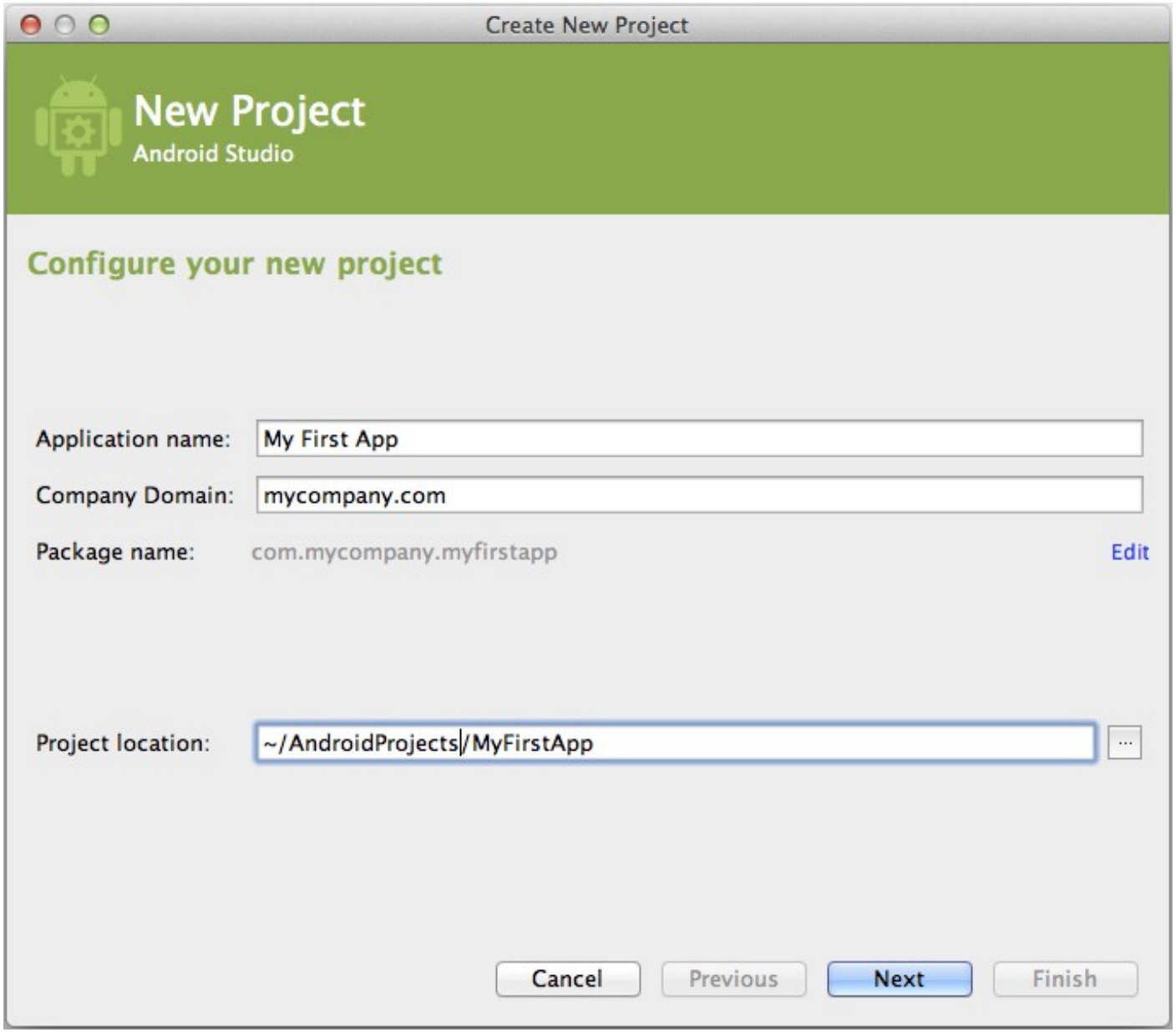
Top-down design

- Let's start from a design of an app that we want to create and then learn the necessary skills to build that app.

- "Bigger Number" game (really dumb)
  - user is shown two numbers
  - must choose which one is bigger by clicking on the appropriate button
  - game pops up brief "correct" / "incorrect" message after each guess
  - get points for each correct answer (lose points for incorrect answers)

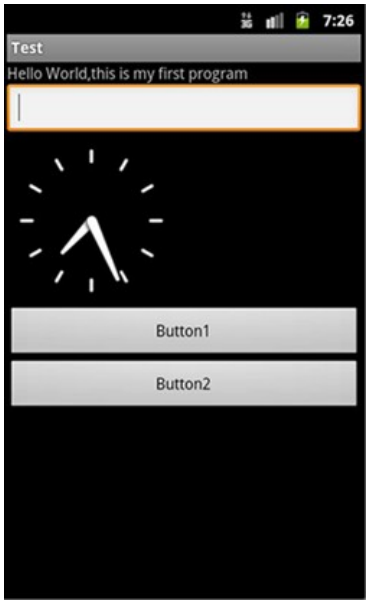


Creating a new project



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

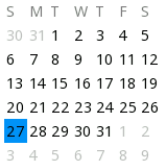
- **action bar**: a menu of common actions at top of app
- **notification area**: topmost system menu and icons

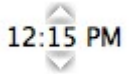
Android widgets

Analog/DigitalClock

Button

Checkbox





Date/TimePicker

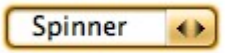
EditText

Gallery

ImageView/Button

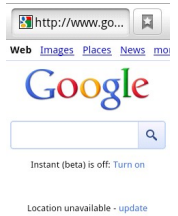
ProgressBar

RadioButton



Spinner

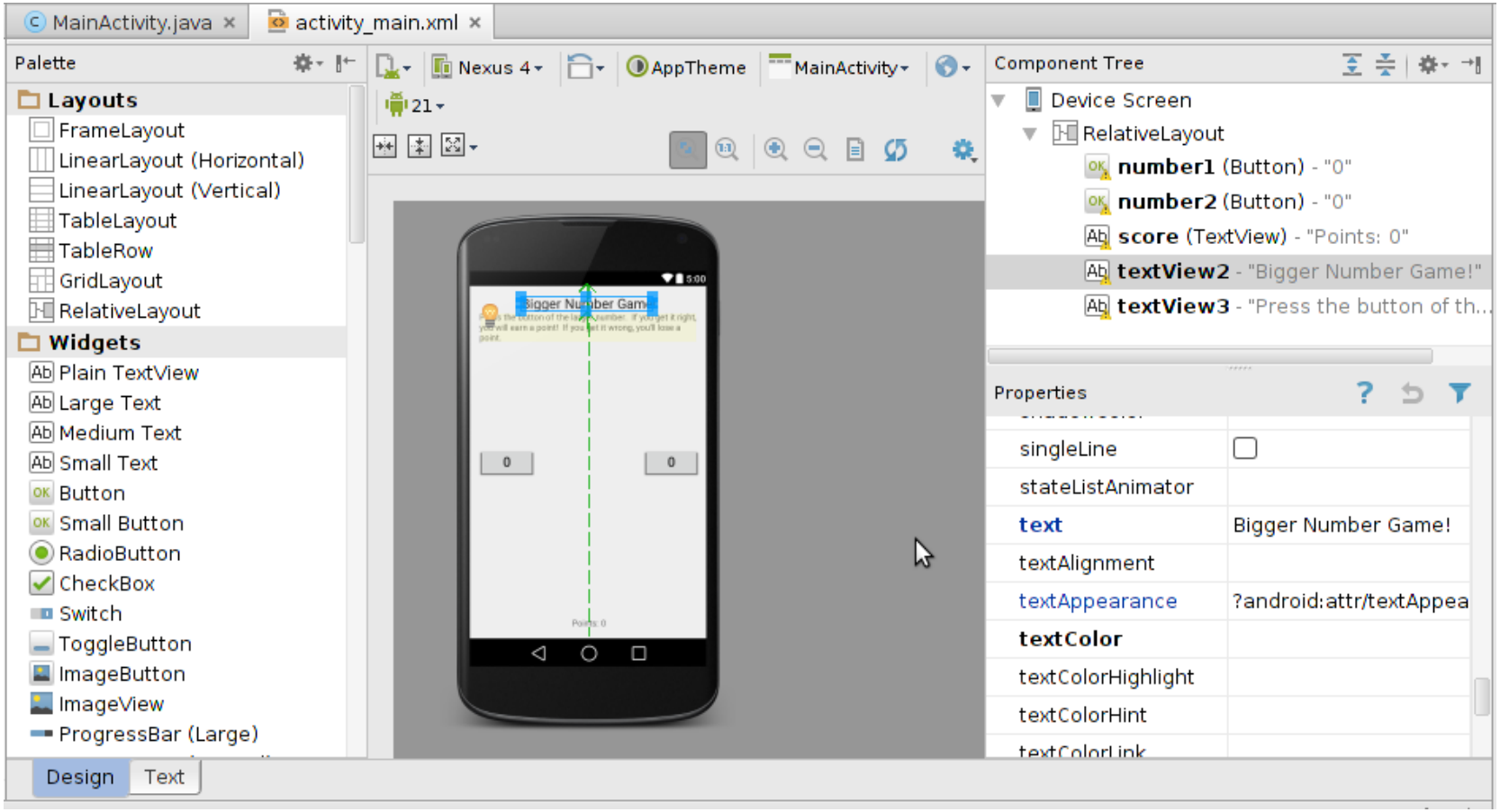
TextView



MapView, WebView

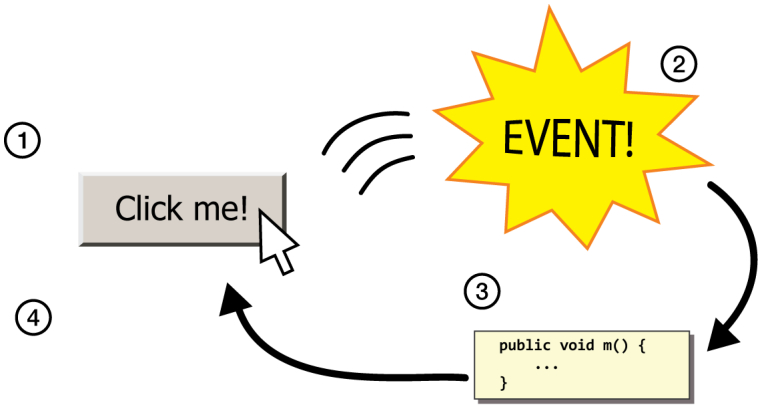
Designing a user interface

- 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

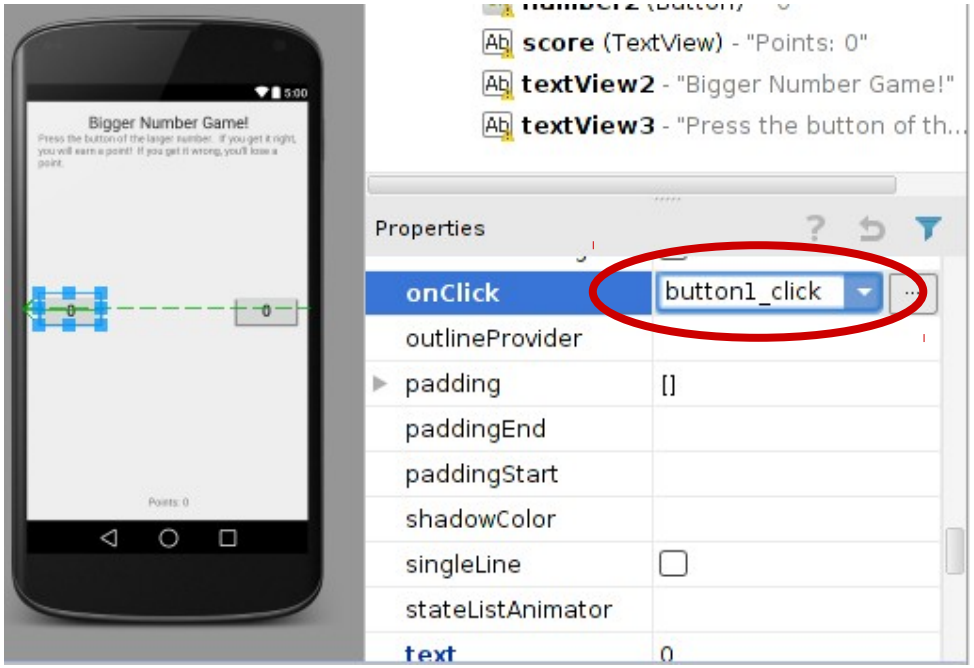


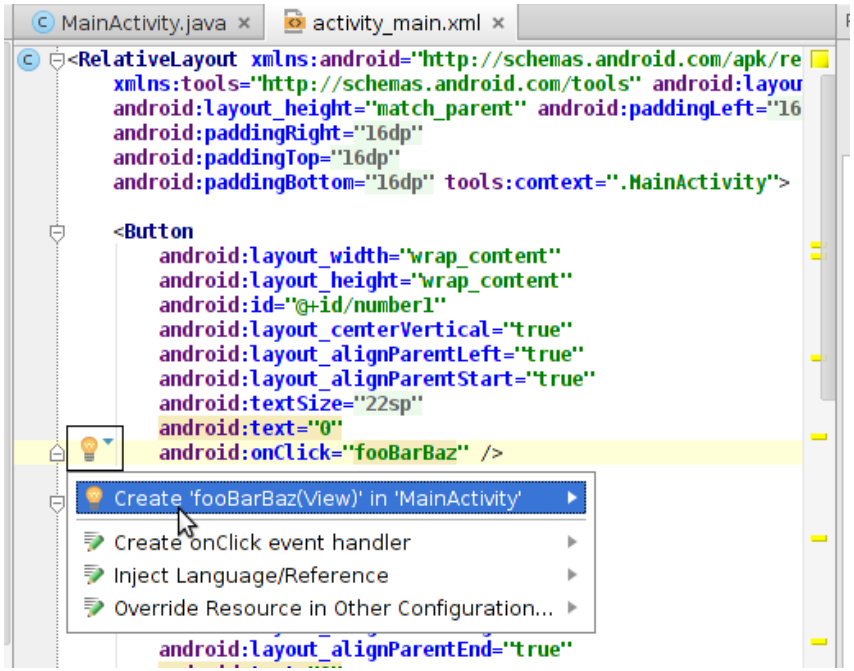
- **event-driven programming**: Overall execution of your program is largely dictated by user events.
  - Commonly used in graphical programs.

- 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.

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

```
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/

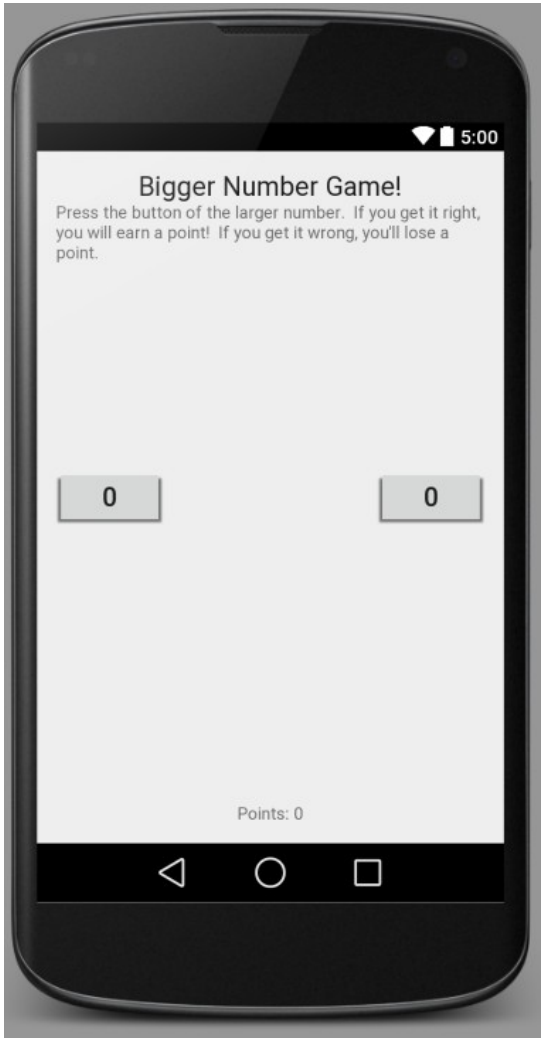
Interacting with widgets

- 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!");
}
```

Exercise: Number game

- New let's build that "Bigger Number" game! Recall:
  - user is shown two numbers
  - must choose which one is bigger by clicking on the appropriate button
  - game pops up brief "correct" / "incorrect" message after each guess
  - get points for each correct answer (lose points for incorrect answers)



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.