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
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)

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

* no data available for October 2014
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
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

Application name: My First App
Company Domain: mycompany.com
Package name: com.mycompany.myfirstapp
Project location: ~/AndroidProjects/MyFirstApp
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
<table>
<thead>
<tr>
<th>Analog/DigitalClock</th>
<th>Button</th>
<th>Checkbox</th>
<th>Date/TimePicker</th>
</tr>
</thead>
<tbody>
<tr>
<td>EditText 1&lt;br&gt;(206)555-1212&lt;br&gt;..............</td>
<td>Gallery</td>
<td>ImageView/Button</td>
<td>ProgressBar</td>
</tr>
<tr>
<td>RadioButton &lt;br&gt;Vertically aligned text&lt;br&gt;<strong>Bold &amp; Italic</strong></td>
<td>Spinner</td>
<td>TextView</td>
<td>MapView, WebView</td>
</tr>
</tbody>
</table>
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
package com.example.stepp.numbergame;

import ...

public class MainActivity extends ActionBarActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        setContentView(R.layout.activity_main);
        super.onCreate(savedInstanceState);
    }

    public void button1_click(View view) {
        // your code goes here
    }
}
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:

  ```java
  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.)

```java
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)
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.