

# Intro to Android: Getting going with development

Assoc. Professor Donald J. Patterson  
INF 241 Winter 2012



# Intro to Android



<http://developer.android.com/guide/index.html>

# Intro to Android

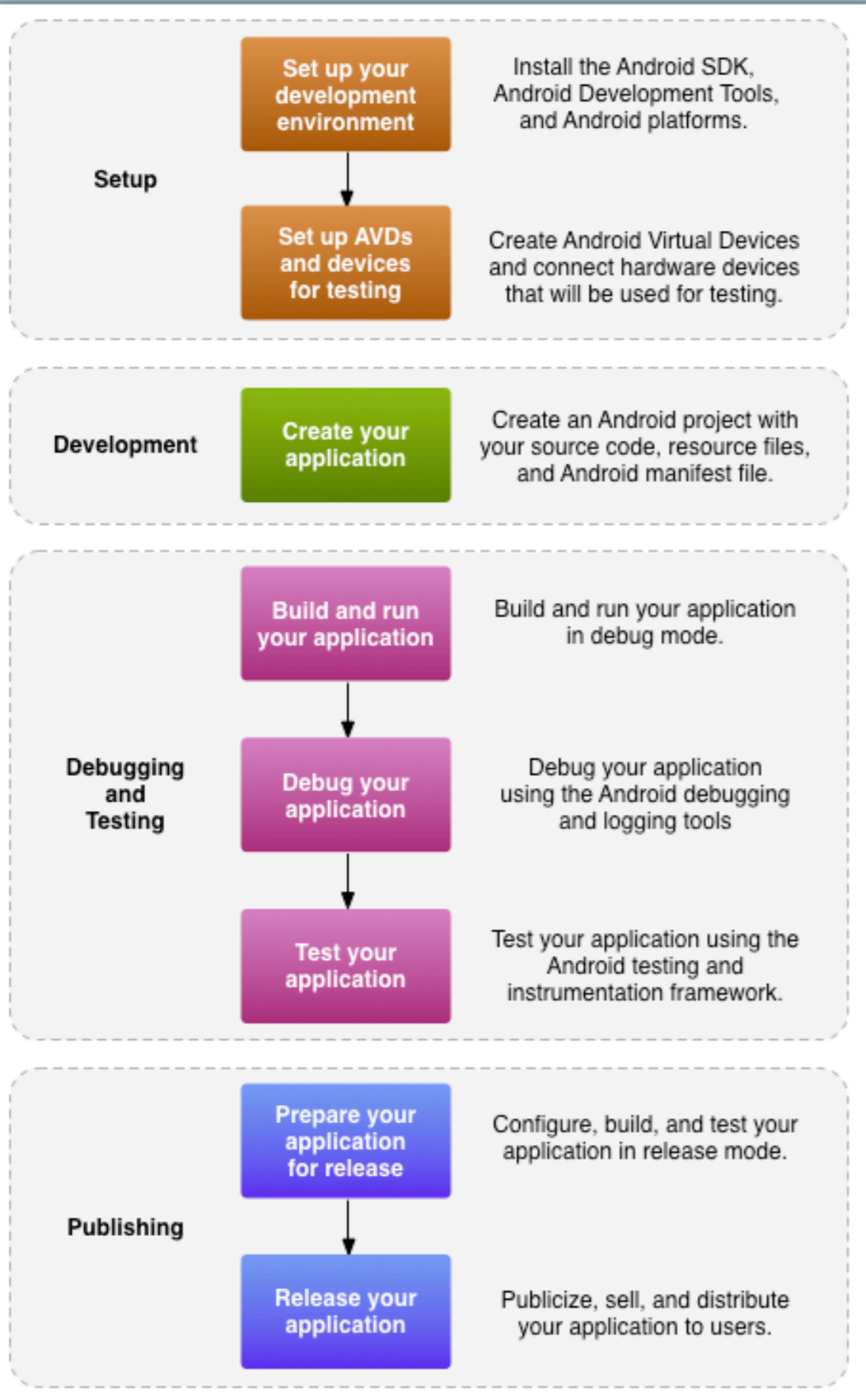
The screenshot shows a browser window titled "Android Developers" with the URL "developer.android.com/index.html". The browser's address bar includes search engines like Google and Bing, and a search box with the text "developing for android". The browser's bookmark bar shows folders like "Class Develop...", "UCI", "DR", "quub", "Blogs and Wikis", "960 Gridder", and "Rails". The website's header features the "Android Developers" logo, a search box for developer docs, and a navigation menu with tabs for "Home", "SDK", "Dev Guide", "Reference", "Resources", "Videos", and "Blog".

The main content area is divided into several sections:

- Developer Announcements:** A section with a shopping bag icon containing an Android robot. The text reads: "We've completely redesigned Android Market for phones to make it easier to explore Android apps, games, and other content. Look for the new version coming to your Android phone!" with a "Learn more »" link.
- Ice Cream Sandwich!** A section featuring a 3D illustration of an Android robot shaped like an ice cream sandwich. The text reads: "Android 4.0 is here, delivering a unified UI for phones and tablets and innovative features for users and developers. Check out the [Platform Highlights](#) for an overview of the new features in Android 4.0." Below this, it says: "For more information about API changes, read the [platform notes](#) and [diff report](#). If you're new to Android, get started with the [SDK starter package](#)."
- Download:** A section with a download icon. The text reads: "The Android SDK has the tools, sample code, and docs you need to create great apps." with a "Learn more »" link.
- Publish:** A section with a shopping bag icon. The text reads: "Android Market is an open service that lets you distribute your apps to handsets." with a "Learn more »" link.
- Contribute:** A section with an Android robot icon. The text reads: "Android Open Source Project gives you access to the entire platform source." with a "Learn more »" link.
- Target Devices:** A section with a pie chart icon. The text reads: "The Device Dashboard provides information about deployed Android devices to help you target suitable device configurations as you build and update your apps." with a "Learn more »" link.

At the bottom of the main content area, there is a navigation bar with three buttons: "Android 4.0" (with an Android robot icon), "Google TV" (with a TV icon), and "Maps API Key" (with a map icon). The page footer contains the text: "Except as noted, this content is licensed under Creative Commons Attribution 2.5. For details and restrictions, see the Content License."

index.html



- SDK (Software development kit)
- AVD (Android Virtual Device)
- .apk (Android application package file)
- “platform”: Gingerbread v2.3.6 / v2.3.4 is our target
- ADT (Android Development Tools) Eclipse plug-ins plus command line tools

<http://developer.android.com/guide/index.html>

# Actually Developing for Android 2.3.6

- Requirements
- SDK/AVD
- Eclipse Plug-in
- Hello World



<http://developer.android.com/guide/index.html>

# Actually Developing for Android 2.3.6

- Requirements
- SDK/AVD
- Eclipse Plug-in
- Hello World



<http://developer.android.com/guide/index.html>

# Actually Developing for Android 2.3.6

- Requirements
- SDK/AVD
- Eclipse Plug-in
- Hello World

<http://developer.android.com/guide/index.html>

# Intro to Android: Requirements

- OS
  - Windows XP (32), Vista (32/64) or Windows 7 (32/64)
  - Mac OS 10.5.8 or later (Leopard, x86 only)
  - Linux (e.g., Ubuntu Lucid Lynx (32))
- IDE
  - Eclipse 3.5 or greater
    - “Eclipse IDE for Java Developers” (for example)
- Java
  - JDK 5 or 6 (1.5 or 1.6 (not just JRE!))

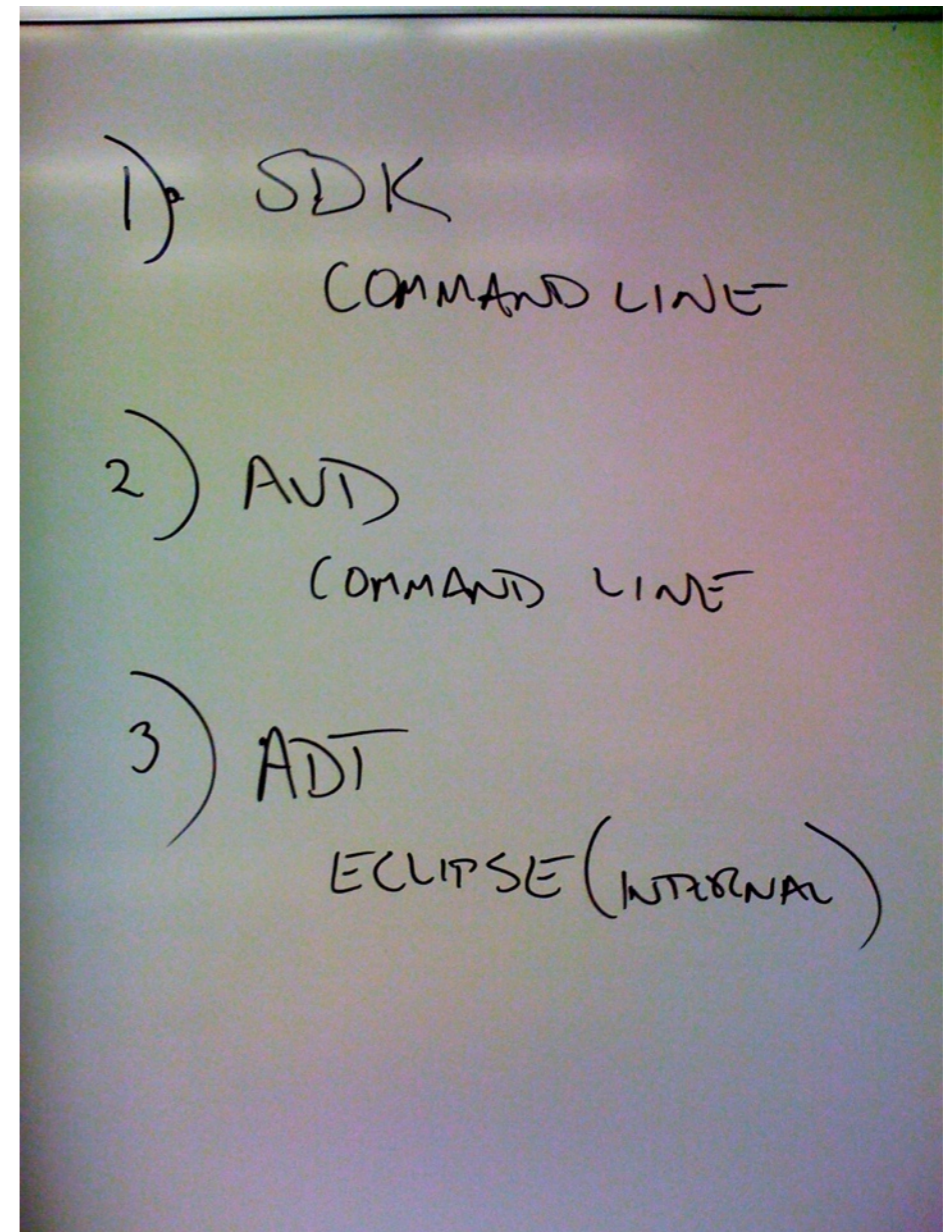


<http://developer.android.com/guide/index.html>



# Actually Developing for Android

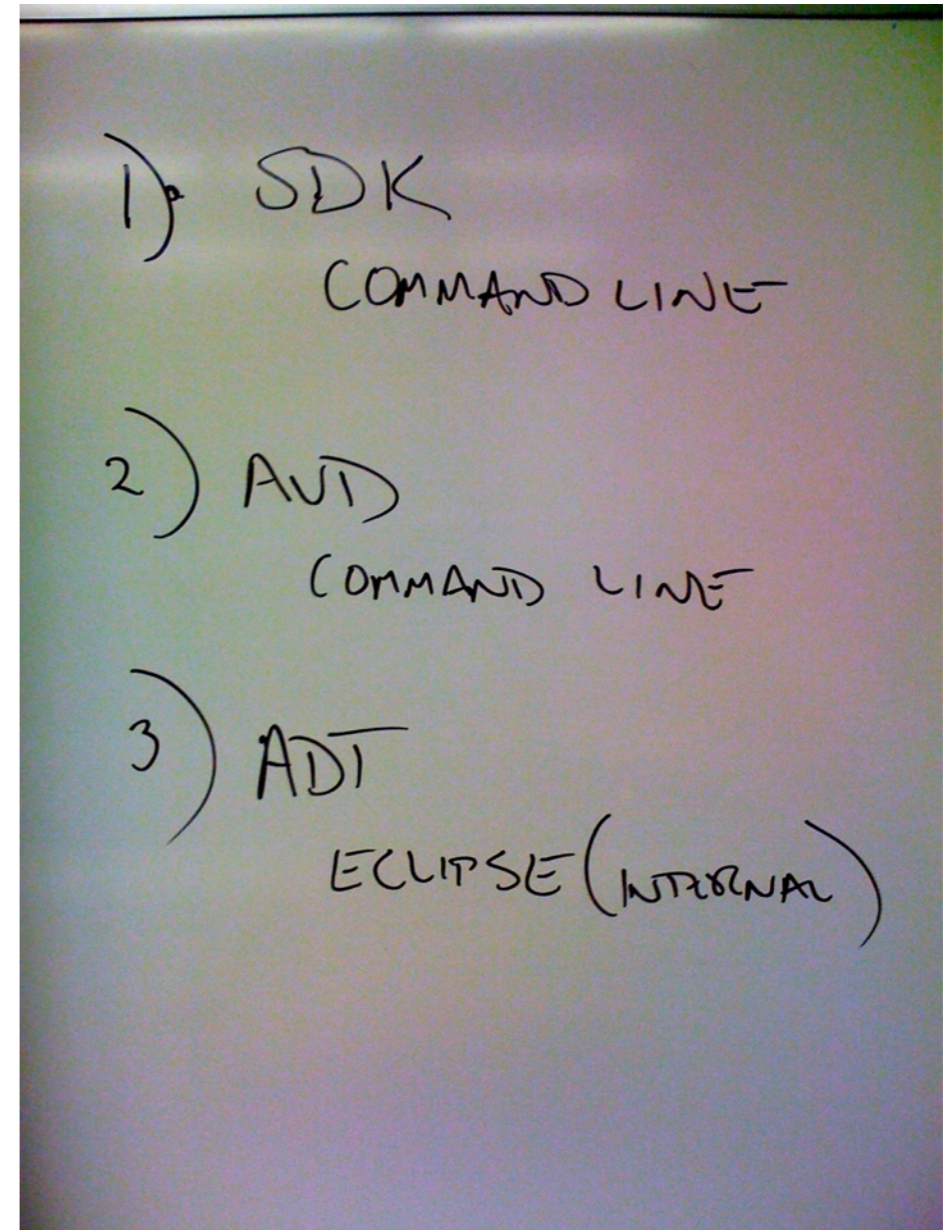
- Requirements
- SDK/AVD
- Eclipse Plug-in
- Hello World



<http://developer.android.com/guide/index.html>

## Actually Developing for Android

- Requirements
- SDK/AVD
- Eclipse Plug-in
- Hello World



<http://developer.android.com/guide/index.html>

# Intro to Android: SDK

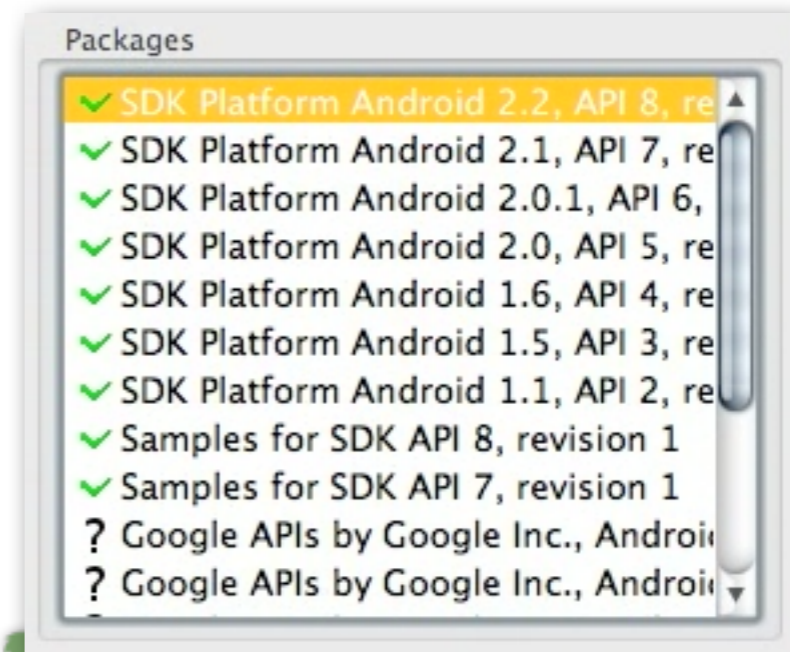
- Download and unpack the appropriate “Android SDK and AVD manager”

Platform	Package	Size	MD5 Checksum
Windows	<a href="#">android-sdk_r15-windows.zip</a>	33895447 bytes	cc2aadf7120d12b574981461736a96e9
	<a href="#">installer_r15-windows.exe</a> (Recommended)	33902520 bytes	ee8481cb86a6646a4d963d5142902c5c
Mac OS X (intel)	<a href="#">android-sdk_r15-macosx.zip</a>	30469921 bytes	03d2cdd3565771e8c7a438f1c40cc8a5
Linux (i386)	<a href="#">android-sdk_r15-linux.tgz</a>	26124434 bytes	f529681fd1eda11c6e1e1d44b42c1432

- This is a program that downloads and manages the installation of libraries for you
- Load the correct “Platform SDK”
- Create an Android Virtual Device

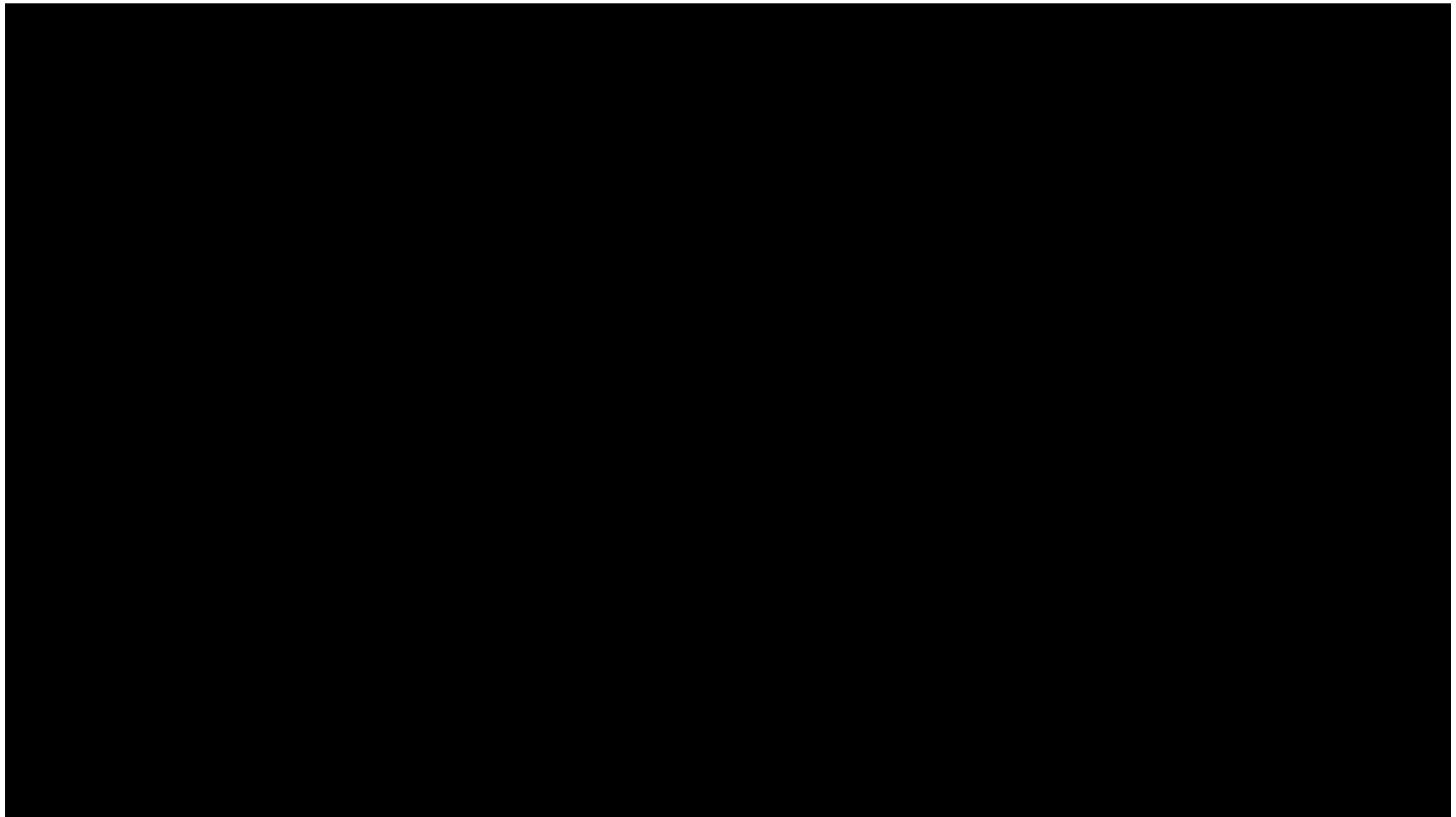
List of existing Android Virtual Devices located at /Users/djp3/.android/avd

AVD Name	Target Name	Platform	API Level
✓ Fall_2010_INF_1	Android 2.2	2.2	8



<http://developer.android.com/sdk/adding-components.html>

# Intro to Android: SDK



# Intro to Android: AVD

```
djp3 — djp3@dhcp-v019-018.mobile.uci.edu: /Users/djp3/Downloads — bash — 95x29
[512]djp3@dhcp-v019-018: ~/Downloads
$
```

I



# Actually Developing for Android

- Requirements
- SDK/AVD
- Eclipse Plug-in
- Hello World



<http://developer.android.com/guide/index.html>

# Actually Developing for Android

- Requirements
- SDK/AVD
- Eclipse Plug-in
- Hello World



<http://developer.android.com/guide/index.html>

## Intro to Android: Eclipse Plug-in

- The Eclipse Plug-in is called the “ADT”
  - Android Development Tools
  - Support for menu options in Eclipse which support
    - Automatically building Android projects
    - User-Interface building for Android
    - Debugging support for Android
    - Packaging files for the Android Market (.apk files)
  - The ADT is installed from within Eclipse



<http://developer.android.com/sdk/eclipse-adt.html>



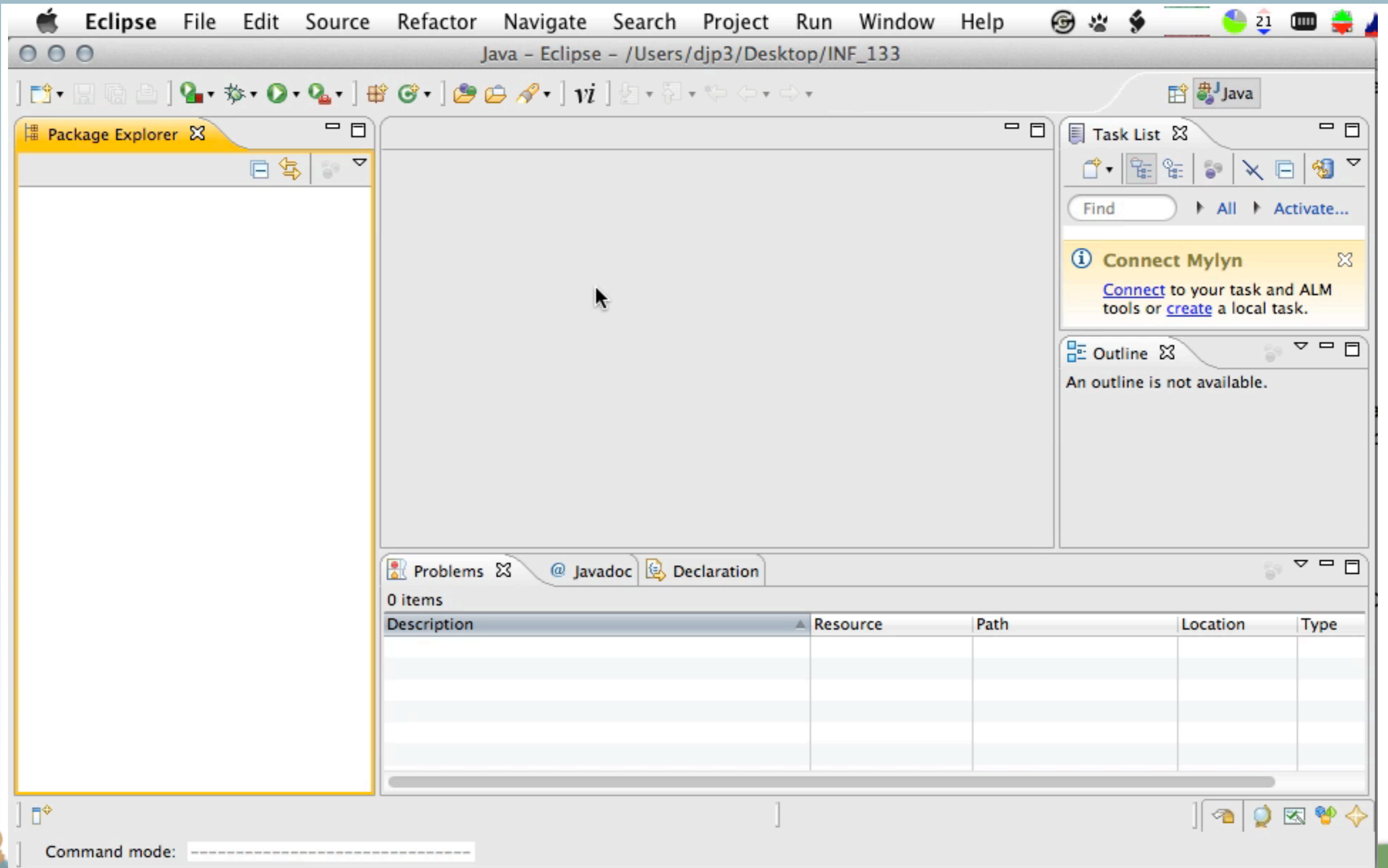
# Intro to Android: Eclipse Plug-in

- Install the plug-in
- Restart Eclipse
- Configure the plug-in



<http://developer.android.com/sdk/eclipse-adt.html>

# Intro to Android: Eclipse Plug-in



<http://developer.android.com/sdk/eclipse-adt.html>

# Actually Developing for Android

- Requirements
- SDK/AVD
- Eclipse Plug-in
- Hello World



<http://developer.android.com/guide/index.html>

# Actually Developing for Android

- Requirements
- SDK/AVD
- Eclipse Plug-in
- Hello World

<http://developer.android.com/guide/index.html>

# Actually Developing for Android

A stylized illustration of a city skyline at the bottom of the slide. It features various buildings in shades of orange, brown, and blue, interspersed with green palm trees and other foliage. The entire scene is set against a light green background.

<http://developer.android.com/guide/index.html>

# Hello World

- Create a project
- Build a basic U/I
- Run the Application
- Improve the U/I
- Debug the Application

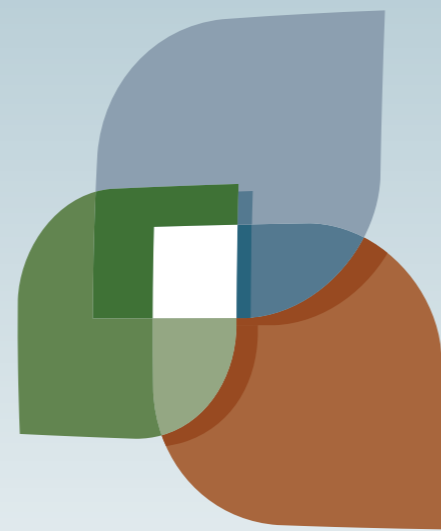


<http://developer.android.com/guide/index.html>

# Hello World

- Create a project
- Build a basic U/I
- Run the Application
- Improve the U/I
- Debug the Application

<http://developer.android.com/guide/index.html>



L U C I





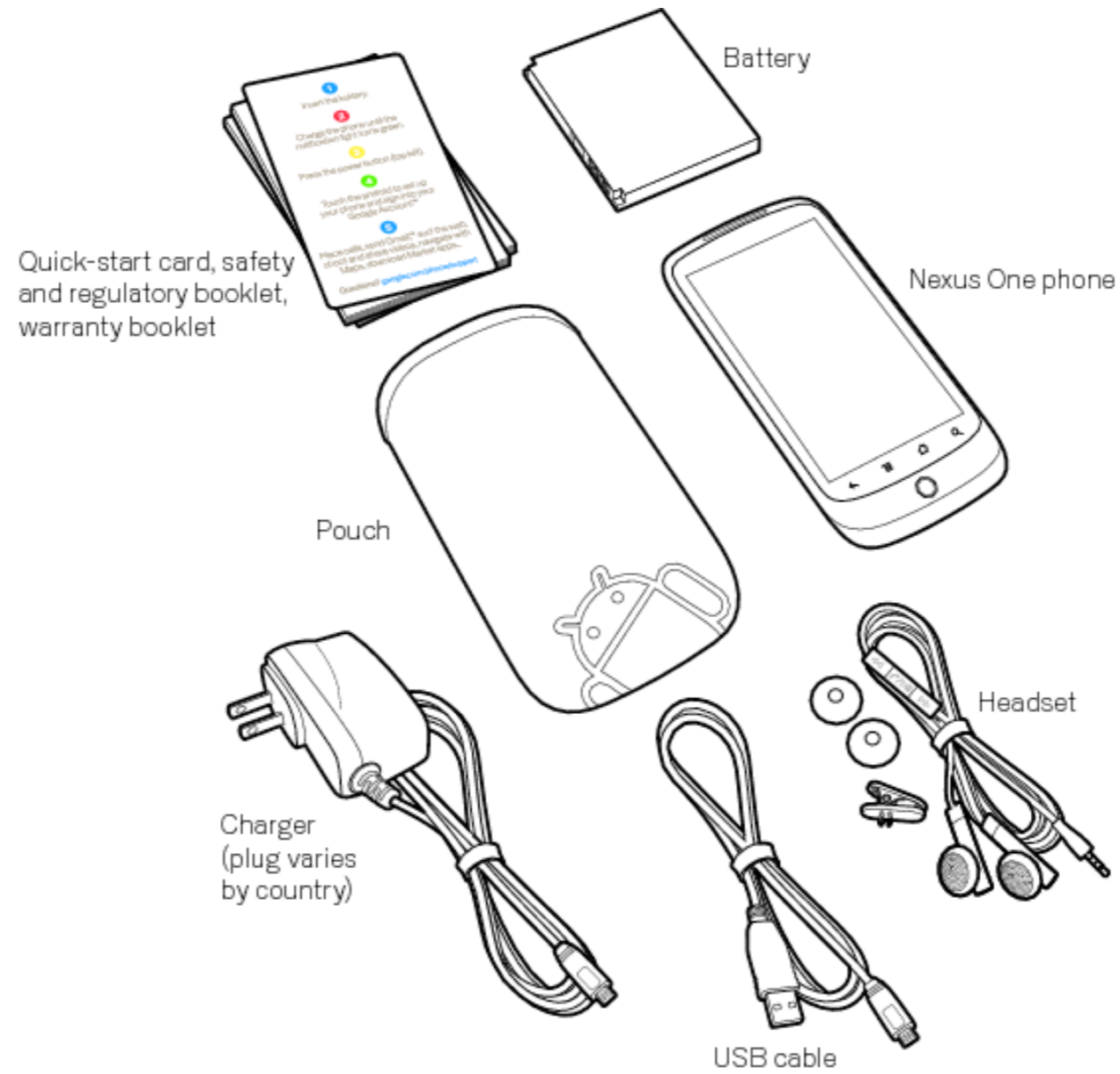
# User Interaction: Intro to Android

Assoc. Professor Donald J. Patterson  
INF 241 Winter 2012



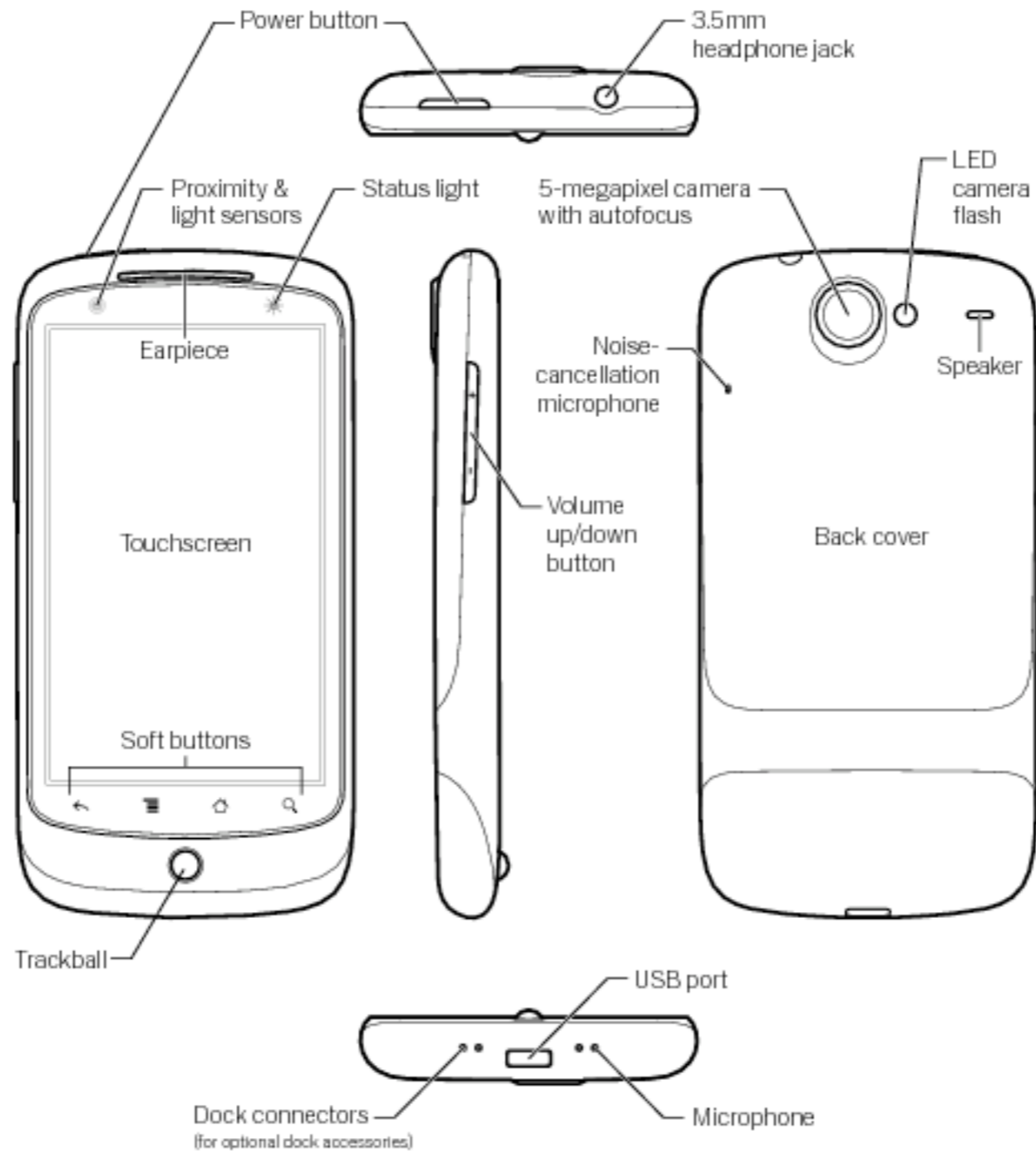
# Checking out the phone

- Unpack the phone



# Checking out the phone

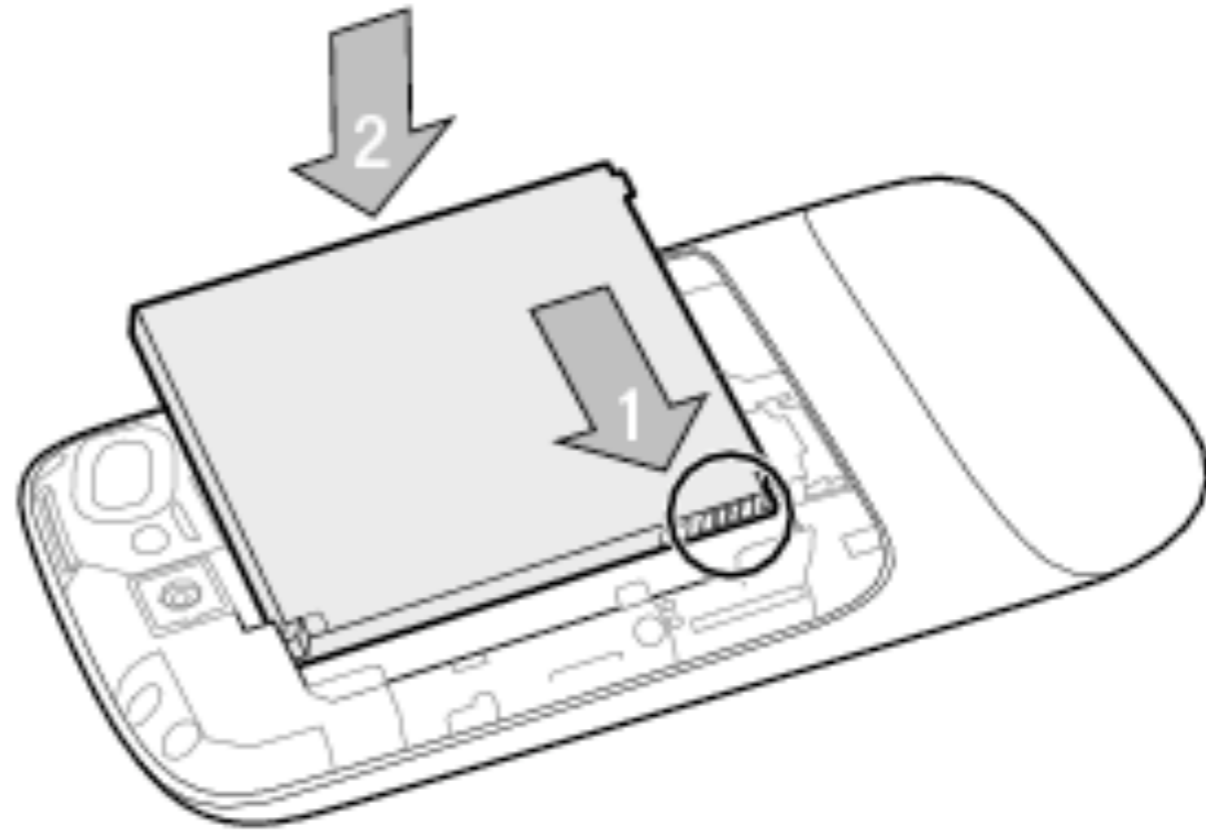
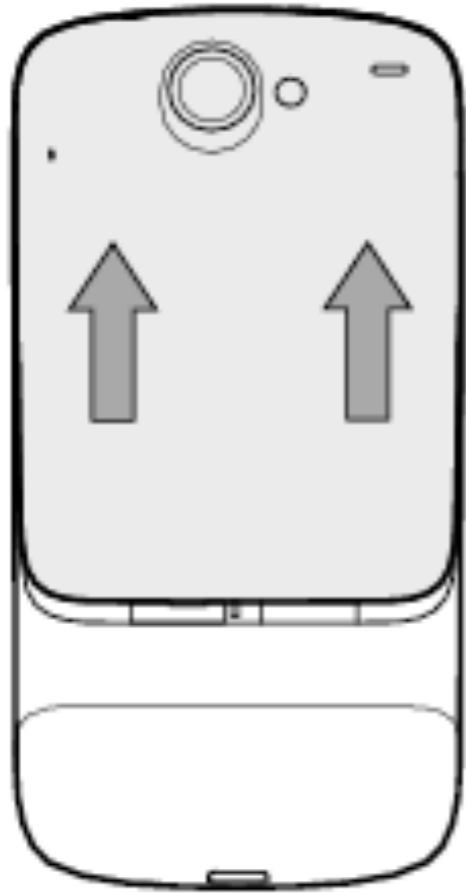
- Take a look at the sensors



<http://www.google.com/support/android/bin/topic.py?hl=en&topic=28930>

## Checking out the phone

- Install the battery - Do Not Damage My Phones!



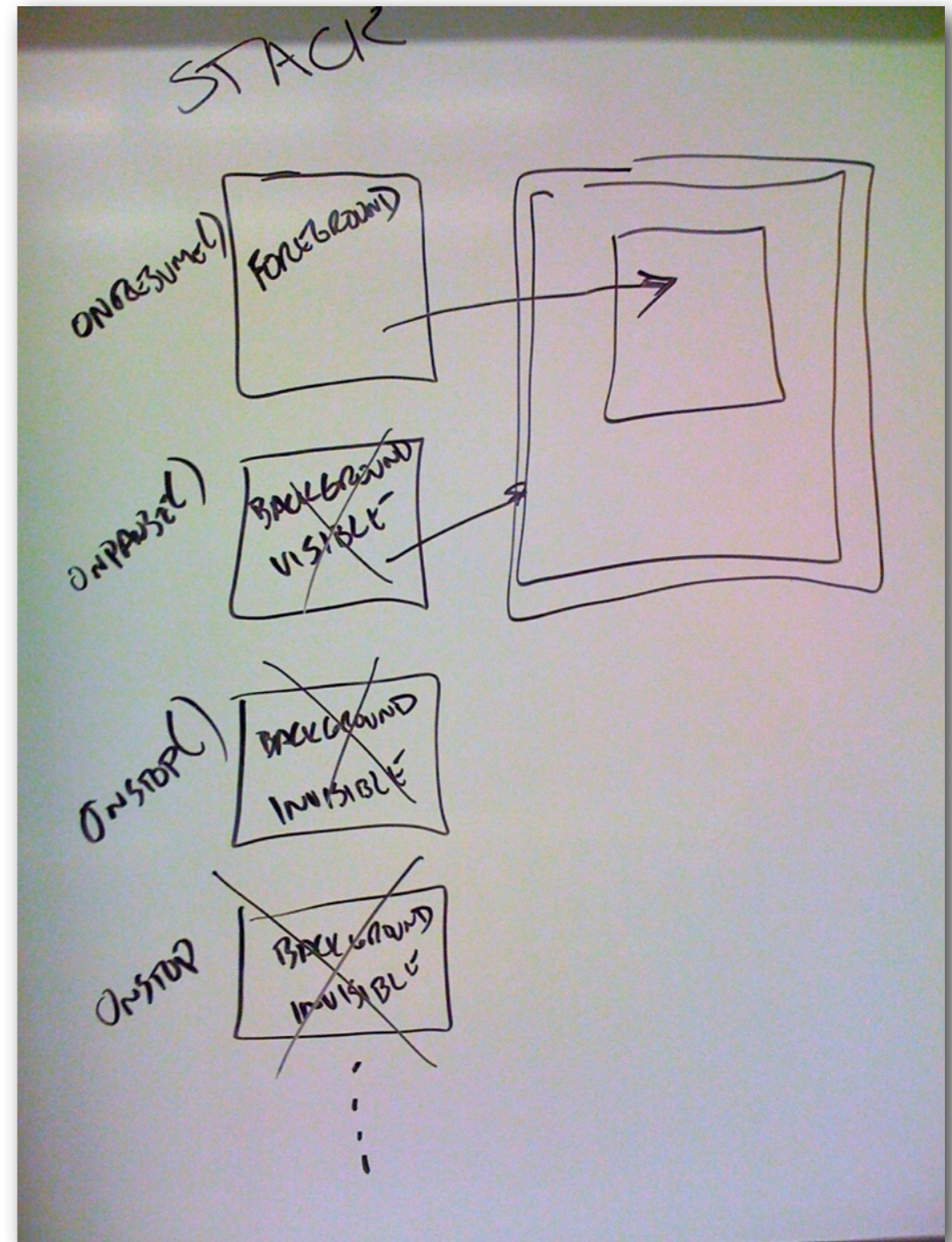
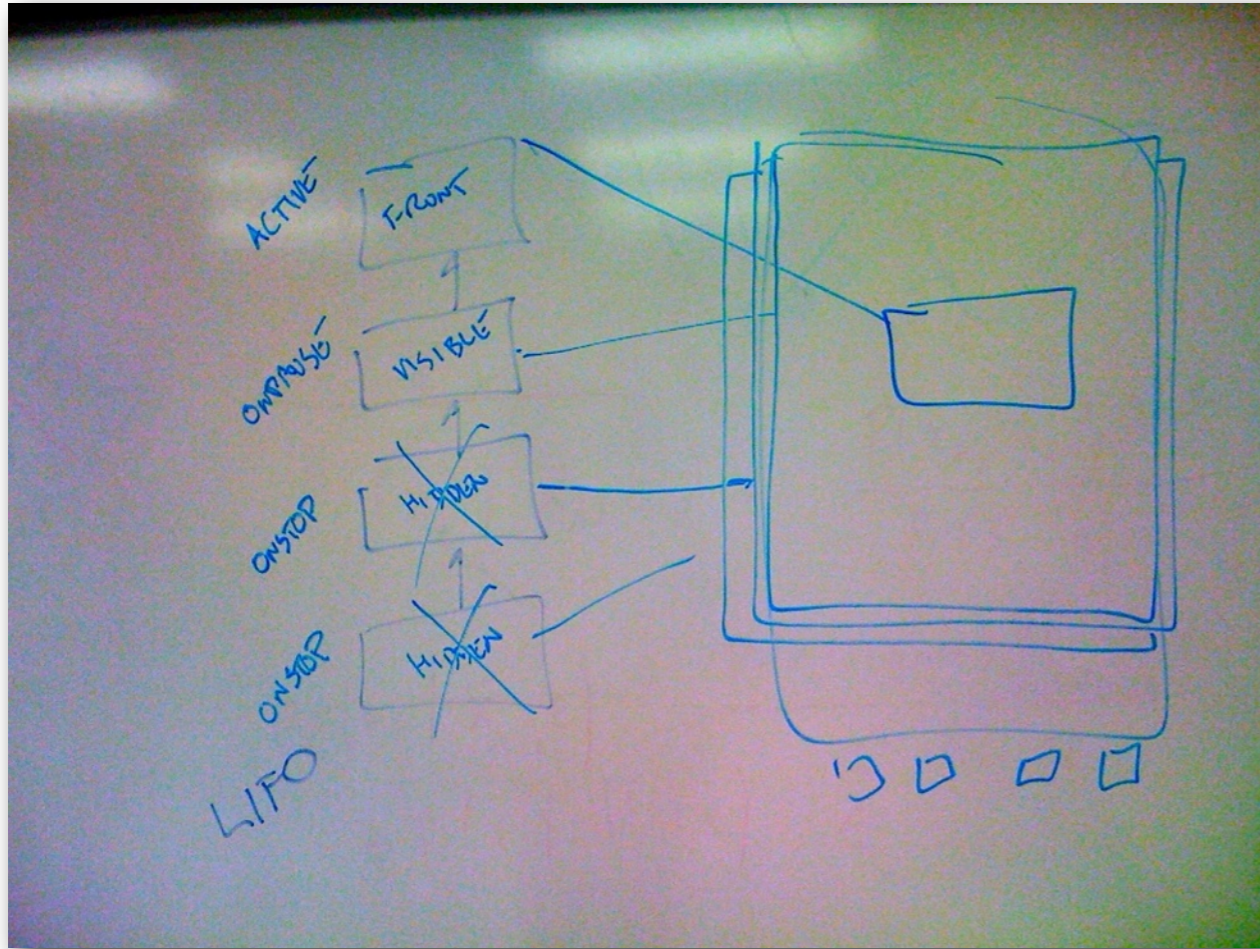
## Checking out the phone

- Charge the phone to 100%
  - USB to computer
  - USB to wall plug
- While charging, go through on-phone tutorial
- Do not sync to your Google
- Enable Location reporting
- Set Date and Time
- Register your device with OIT (or send me the MAC address)
  - Home -> Menu -> Settings -> About Phone -> Status

<http://www.google.com/support/android/bin/topic.py?hl=en&topic=28930>



# Activity Stack



# Intents

- Requests made by an Activity to have the phone do something
  - e.g., take a picture, look up a contact etc.
- Specified abstractly so that new software can answer the Intent
- Calling an Intent requires an Action and some data
  - e.g., ACTION\_DIAL, tel:555-1212 = use phone to call 555-1212
- An Activity can start an Intent
- A Broadcast Receiver can respond to an Intent
  
- An Intent is a message sent from one app to another

<http://developer.android.com/reference/android/content/Intent.html>



# Simulating Sensors

- On the emulator



# Sensors in the emulator



Emulator

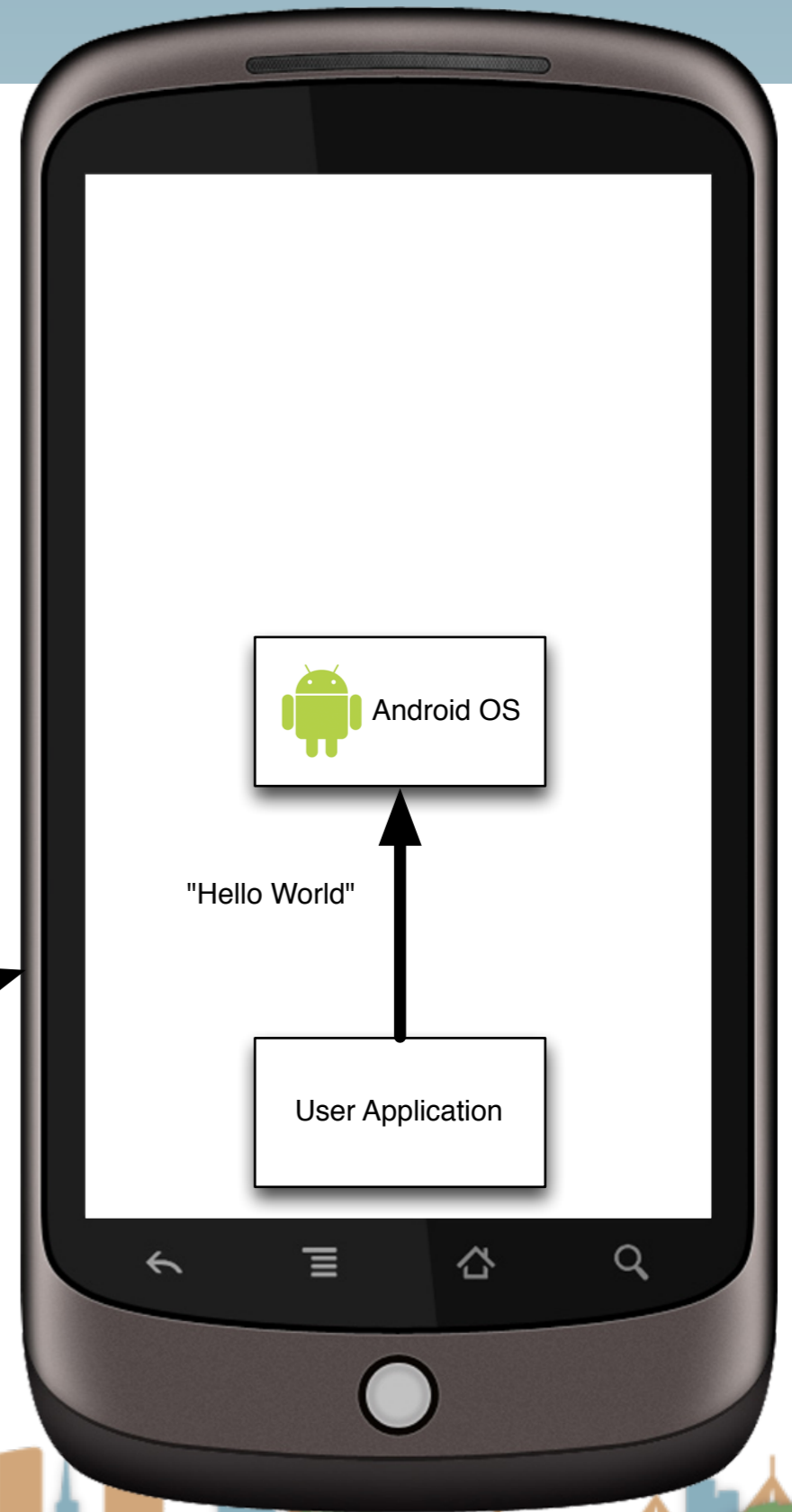
# Sensors in the emulator

- Simulating Sensors on the Emulator requires:
  - A 3rd-party program on the emulator
  - A 3-rd party program on your computer
  - configuration of the two
  - small changes to your code



# How to handle the assignment

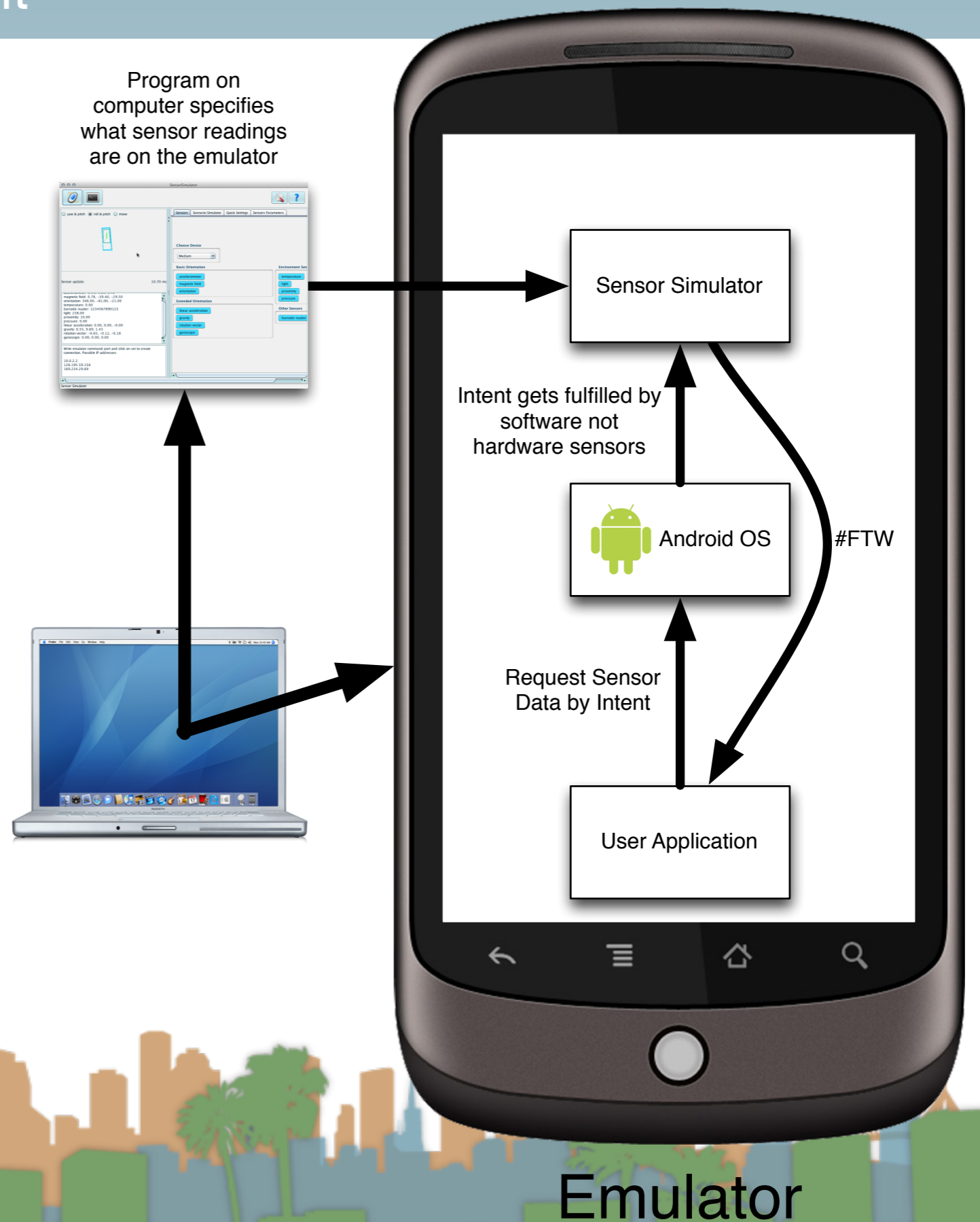
- Your approach: Stage 1
  - get your environment working



Emulator

# How to handle the assignment

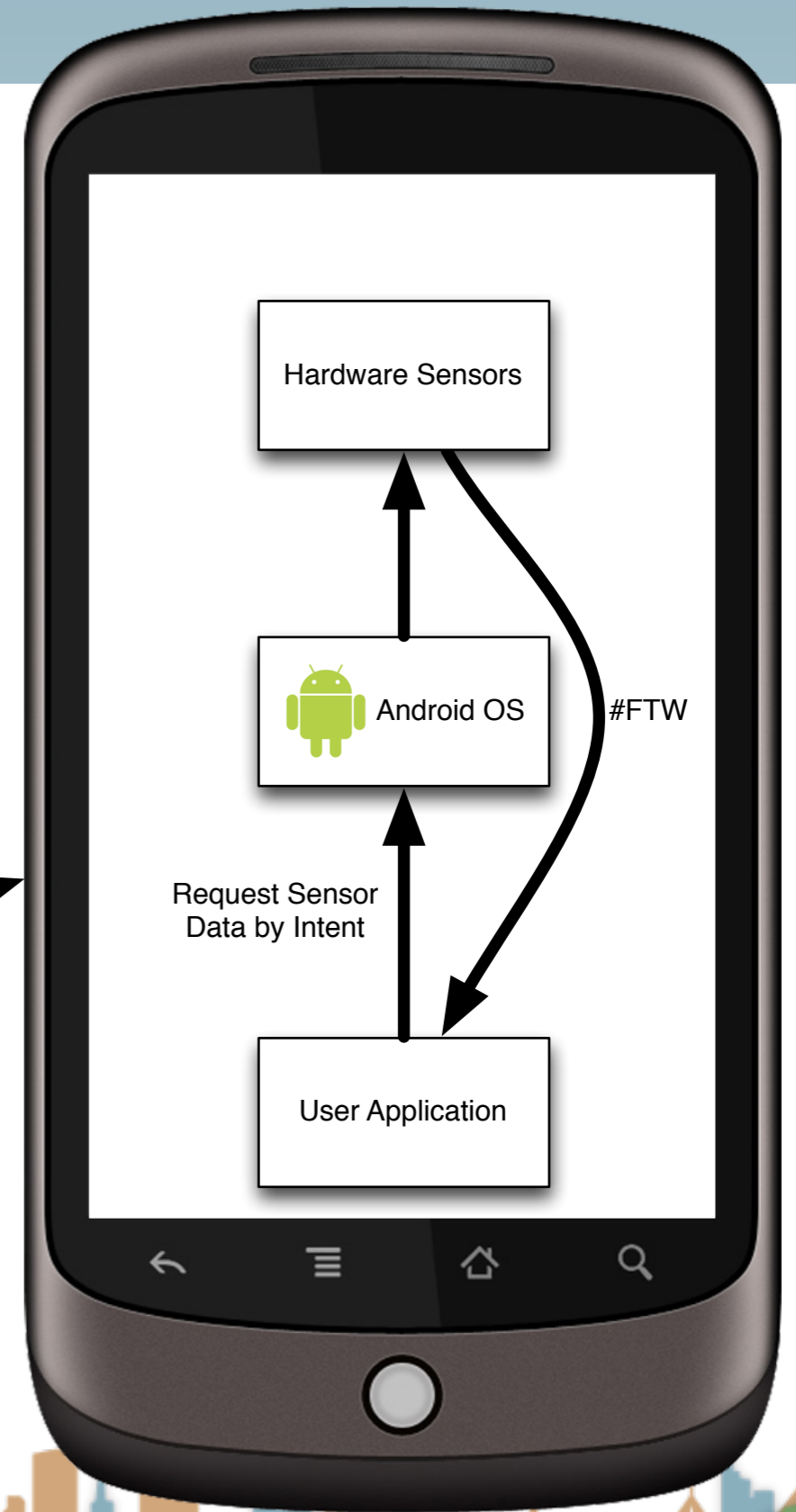
- Your approach: Stage 2
  - get your program working with simulated sensors
  - fast and easy to iterate



Emulator

# How to handle the assignment

- Your approach: Stage 3
  - get your program working with a real phone



Real Phone

# How to handle the assignment

- Stage 2
  - Start a new project on Android 2.3.3
  - download the sensor simulator from
    - <http://code.google.com/p/openintents/downloads/list?q=sensorsimulator>
  - unzip the code into a lib directory
  - add the sensor-simulator jar to your build path
  - create a basic UI in xml
  - launch the emulator
  - load the sensor simulator into the emulator
    - adb install SensorSimulatorSettings-2.0-rc1.apk

<http://developer.android.com/guide/developing/device.html>

# How to handle the assignment

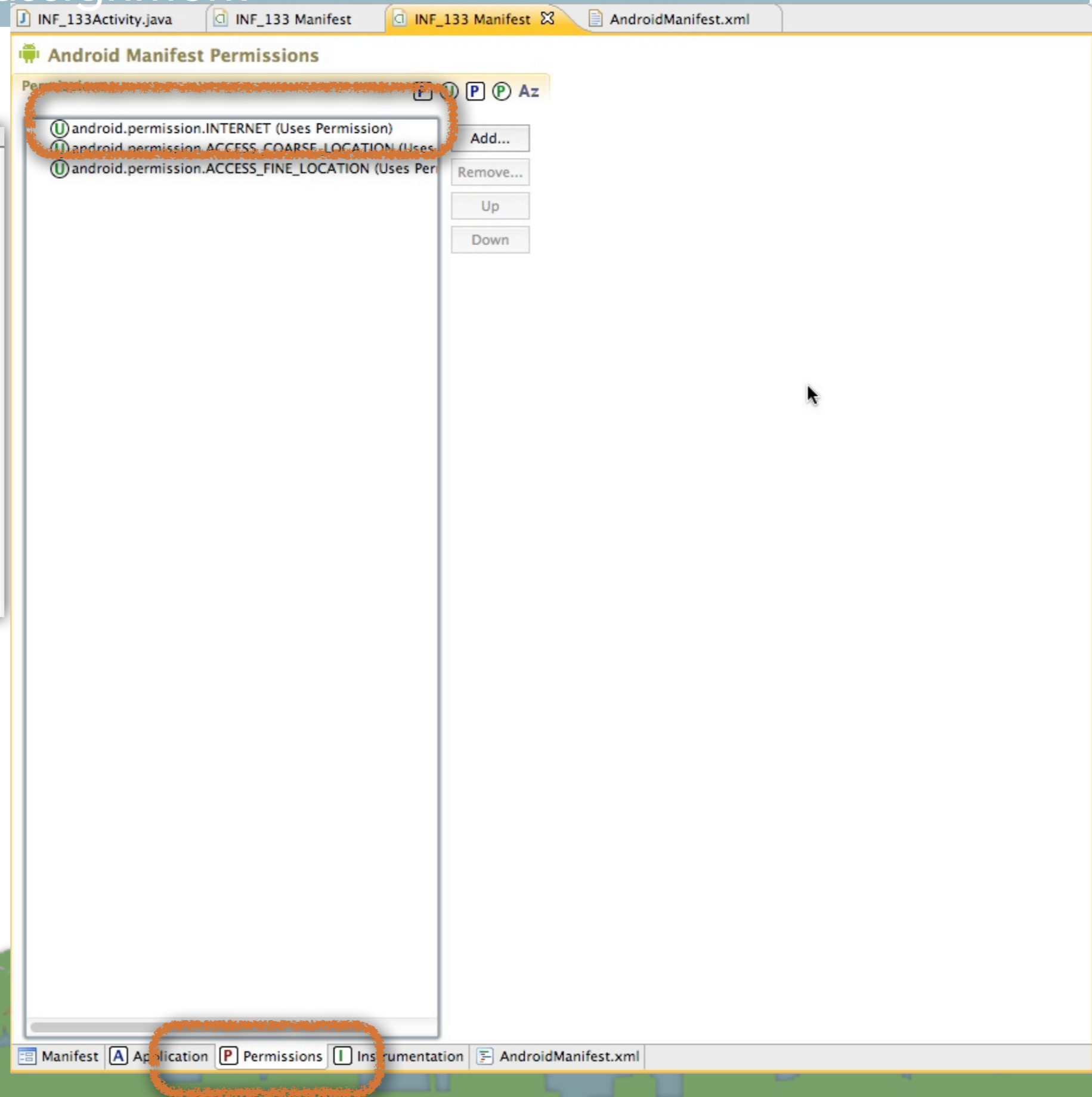
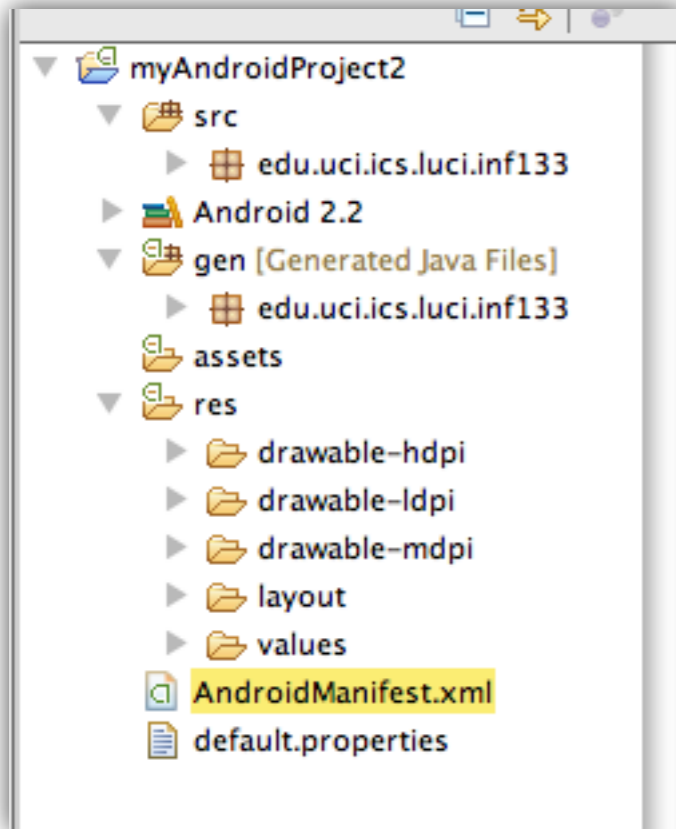
- Stage 2
  - Give your application permission to use the Internet in AndroidManifest.xml
    - `<uses-permission android:name="android.permission.INTERNET" />`
  - Start the sensor simulator on your computer
    - `java -jar ./sensorsimulator-2.0-rc1.jar`
  - Test the sensor software on the phone
  - Make your application debuggable
  - Run your application that requests sensor information

<http://developer.android.com/guide/developing/device.html>



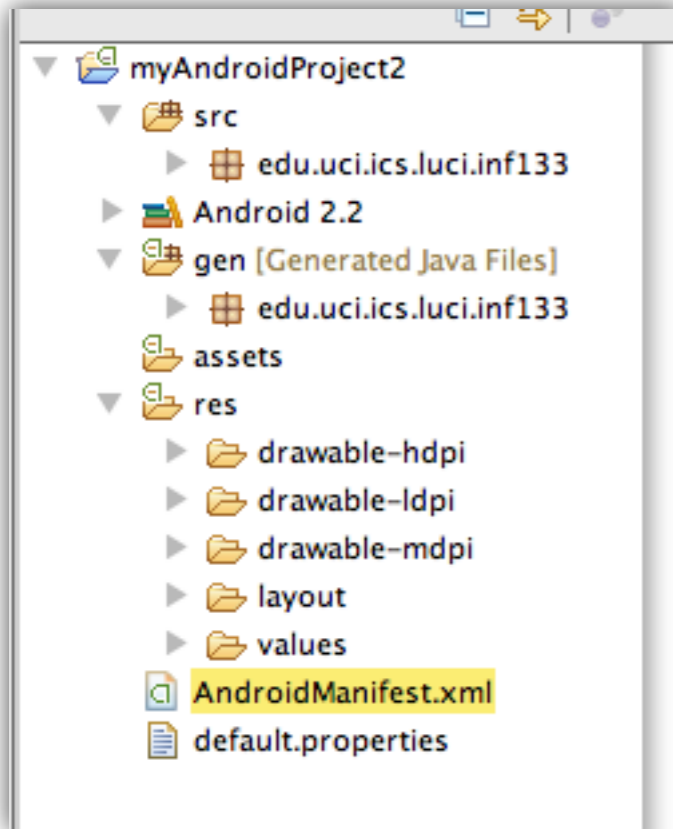
# How to handle the assignment

- Stage 2



# How to handle the assignment

## • Stage 2



### Android Manifest Application

**Application Toggle**

The `application` tag describes application-level components contained in the package, as well as general application attributes.

Define an `<application>` tag in the AndroidManifest.xml

**Application Attributes**

Defines the attributes specific to the application.

Name	<input type="text"/>	Browse...	Debuggable	<input type="text" value="true"/>
Theme	<input type="text"/>	Browse...	Vm safe mode	<input type="text"/>
Label	<input type="text" value="@string/sensorsimulatorsettings"/>	Browse...	Manage space activity	<input type="text"/>
Icon	<input type="text" value="@drawable/mobile_shake_application0"/>	Browse...	Allow clear user data	<input type="text"/>
Description	<input type="text"/>	Browse...	Test only	<input type="text"/>
Permission	<input type="text"/>		Backup agent	<input type="text"/>
Process	<input type="text"/>	Browse...	Allow backup	<input type="text"/>
Task affinity	<input type="text"/>	Browse...	Kill after restore	<input type="text"/>
Allow task reparenting	<input type="text"/>		Restore needs application	<input type="text"/>
Has code	<input type="text"/>		Restore any version	<input type="text"/>
Persistent	<input type="text"/>		Never encrypt	<input type="text"/>
Enabled	<input type="text"/>		Cant save state	<input type="text"/>

**Application Nodes** S P A A R M U Az

- .dbprovider.SensorSimulatorProvider
- .SensorSimulatorSettingsActivity

Buttons: Add..., Remove..., Up, Down

Manifest **A** Application **P** Permissions **I** Instrumentation AndroidManifest.xml

# How to handle the assignment

```
INF_133ActivityTemp. INF_133 Manifest INF_133Activity.java X main.xml main.xml "2
package edu.uci.ics.luci.inf133;

import org.openintents.sensorsimulator.hardware.Sensor;
import org.openintents.sensorsimulator.hardware.SensorEvent;
import org.openintents.sensorsimulator.hardware.SensorEventListener;
import org.openintents.sensorsimulator.hardware.SensorManagerSimulator;

import android.app.Activity;
import android.hardware.SensorManager;
import android.os.Bundle;
import android.widget.TextView;

public class INF_133Activity extends Activity {
    //private SensorManager mSensorManager;
    private SensorManagerSimulator mSensorManager;

    private TextView mTextViewLight;

    private SensorEventListener mEventListenerLight;

    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        mTextViewLight = (TextView) findViewById(R.id.text_light);

        //mSensorManager = (SensorManager) getSystemService(SENSOR_SERVICE);
        mSensorManager = SensorManagerSimulator.getSystemService(this, SENSOR_SERVICE);
        mSensorManager.connectSimulator();

        mEventListenerLight = new SensorEventListener() {

            public void onSensorChanged(SensorEvent event) {
                float[] values = event.values;
                mTextViewLight.setText("Light: " + values[0]);
            }

            public void onAccuracyChanged(Sensor sensor, int accuracy) {
            }

        };
    }
}
```

```
import android.os.Bundle;
import android.widget.TextView;

public class INF_133Activity extends Activity {
    //private SensorManager mSensorManager;
    private SensorManagerSimulator mSensorManager;

    private TextView mTextViewLight;

    private SensorEventListener mEventListenerLight;

    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        mTextViewLight = (TextView) findViewById(R.id.text_light);

        //mSensorManager = (SensorManager) getSystemService(SENSOR_SERVICE);
        mSensorManager = SensorManagerSimulator.getSystemService(this, SENSOR_SERVICE);
        mSensorManager.connectSimulator();

        mEventListenerLight = new SensorEventListener() {

            public void onSensorChanged(SensorEvent event) {
                float[] values = event.values;
                mTextViewLight.setText("Light: " + values[0]);
            }

            public void onAccuracyChanged(Sensor sensor, int accuracy) {
            }

        };
    }

    @Override
    protected void onResume() {
        super.onResume();
        mSensorManager.registerListener(mEventListenerLight,
            mSensorManager.getDefaultSensor(Sensor.TYPE_LIGHT),
            SensorManager.SENSOR_DELAY_FASTEST);
    }

    @Override
    protected void onStop() {
        mSensorManager.unregisterListener(mEventListenerLight);
        super.onStop();
    }
}
```

# What it looks like when it's working (Stage 2)

The screenshot shows the Eclipse IDE interface. The main editor displays the following Java code for `INF_133Activity`:

```
package edu.uci.ics.luci.inf133;

import org.openintents.sensorsimulator.hardware.Sensor;
import org.openintents.sensorsimulator.hardware.SensorEvent;
import org.openintents.sensorsimulator.hardware.SensorEventListener;
import org.openintents.sensorsimulator.hardware.SensorManagerSimulator;

import android.app.Activity;
import android.hardware.SensorManager;
import android.os.Bundle;
import android.widget.TextView;

public class INF_133Activity extends Activity {
    //private SensorManager mSensorManager;
    private SensorManagerSimulator mSensorManager;

    private TextView mTextViewLight;

    private SensorEventListener mEventListenerLight;

    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        mTextViewLight = (TextView) findViewById(R.id.text_light);

        //mSensorManager = (SensorManager) getSystemService(SENSOR_SERVIC
        mSensorManager = SensorManagerSimulator.getSystemService(this...
```

The Package Explorer on the left shows the project structure for `INF_133`, including `src`, `gen`, `Android 2.3.3`, `Referenced Libraries`, `assets`, `bin`, `lib`, and `res` (with subfolders `drawable-hdpi`, `drawable-ldpi`, `drawable-mdpi`, `layout`, and `values`).

The bottom panel shows the LogCat window with the text "Android" and "Launching INF\_133".

# Hints

- Playing a sound
  - The key is the MediaPlayer call
  - Do not instantiate more than one MediaPlayer object

```
static MediaPlayer mp = new MediaPlayer();
public void playSound(String path) {
    if (mp.isPlaying()) {
        return;
    }
    mp.reset();
    try {
        mp.setDataSource(path);
        mp.prepare();
    } catch (Exception ex) {
        Log.d("main thread ex", ex.getStackTrace()[0].toString() + " path: " + path);
    }
    mp.start();
}
```

- <http://developer.android.com/guide/topics/media/index.html>

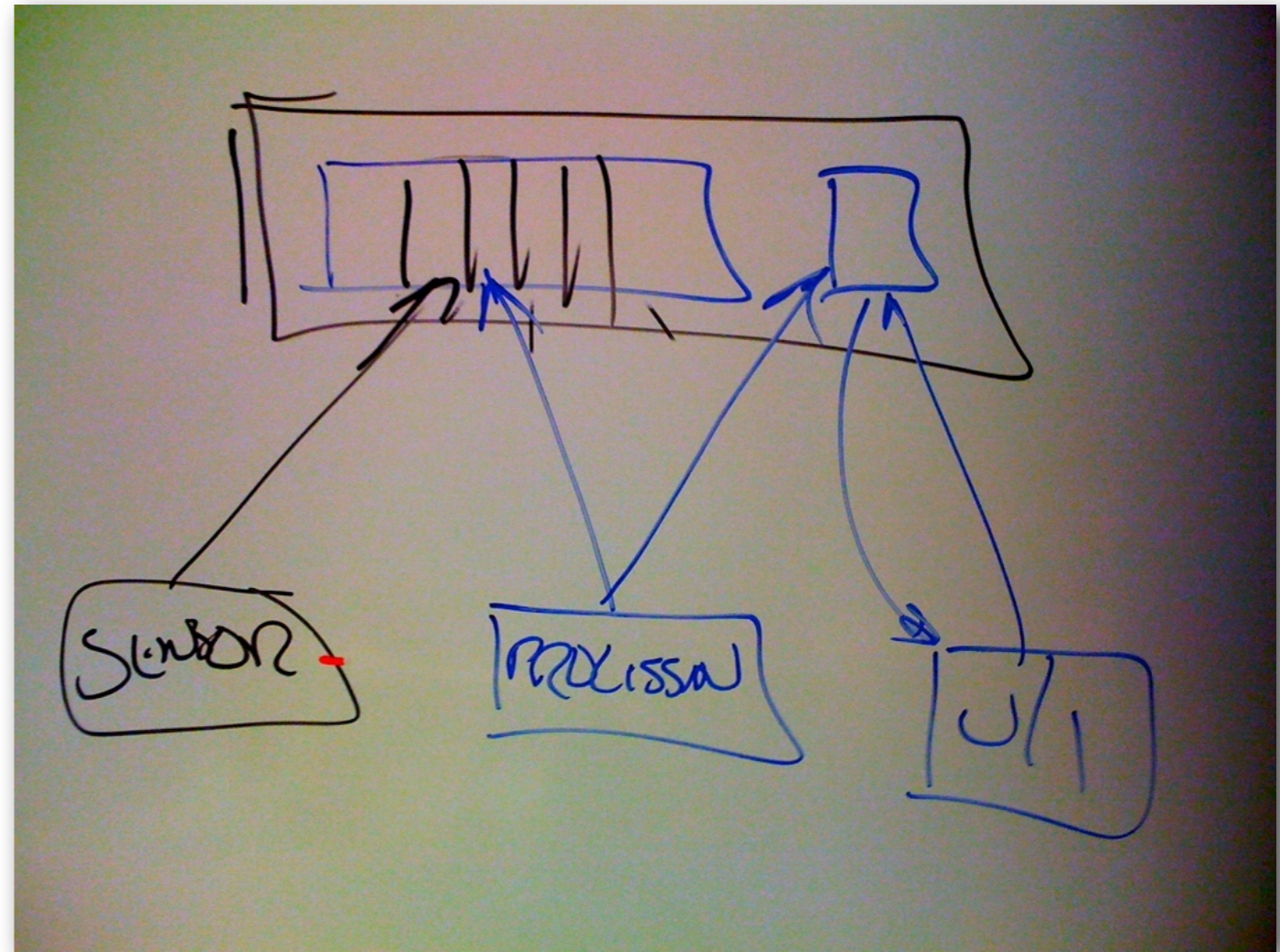
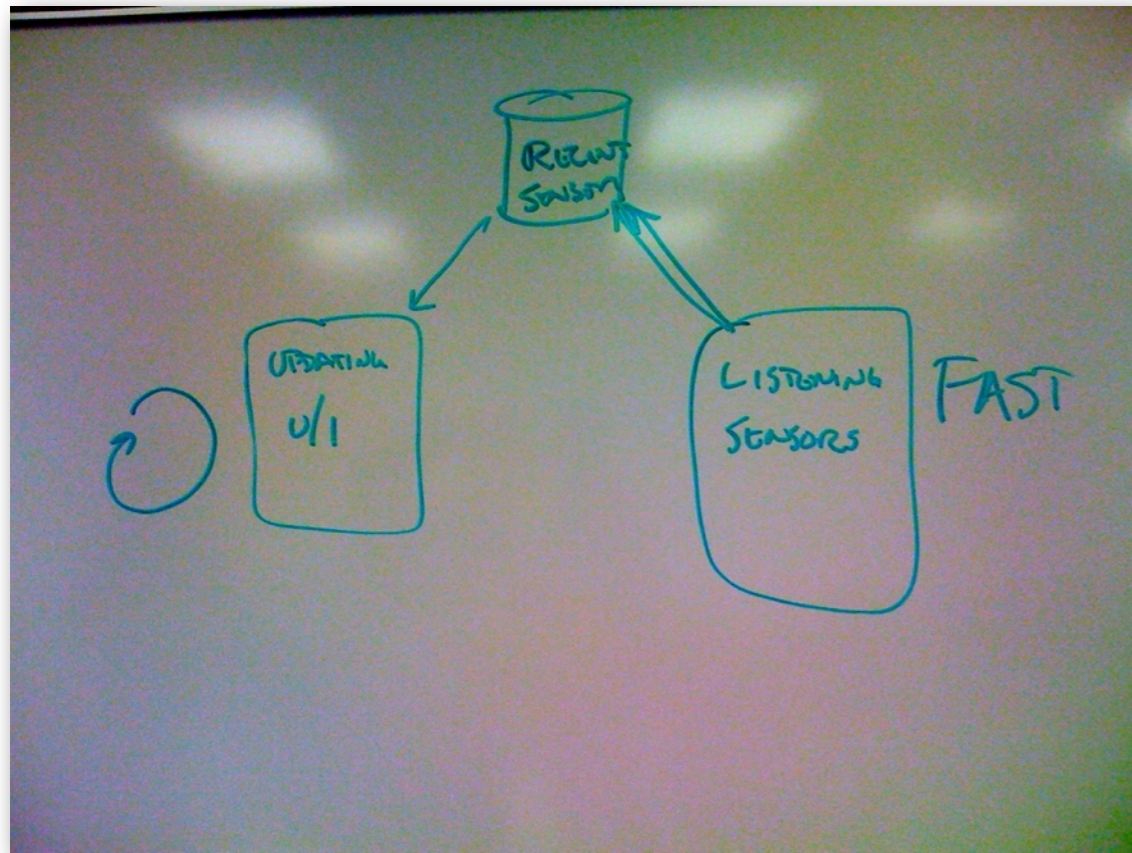


# Hints

- Playing a sound
  - You will need to get the audio media onto the phone



# Hints



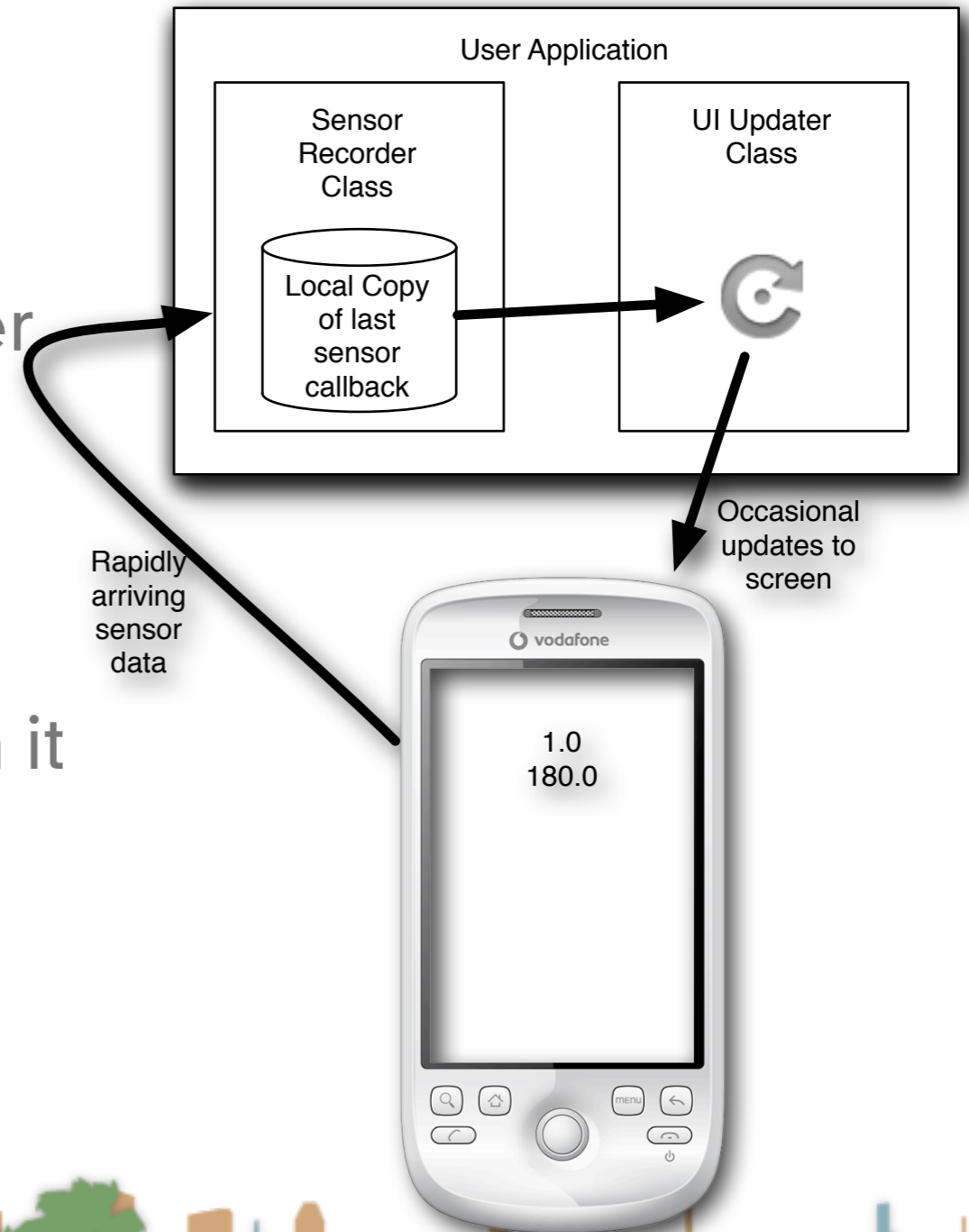


# Hints

- One possible architecture for getting sensor readings

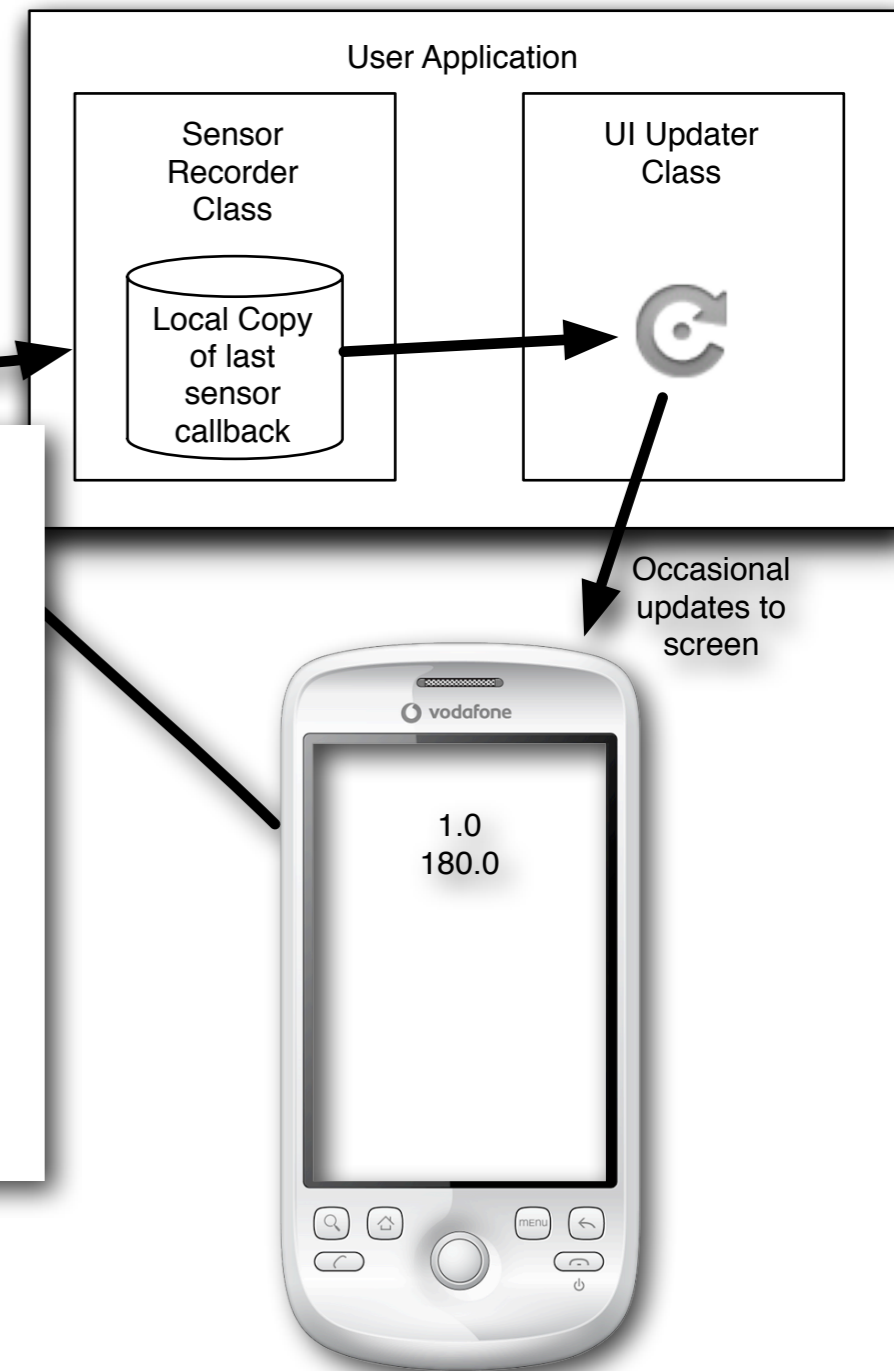
- Steps

- Create a U/I for the data
- Instantiate your Sensor Recorder
- Register for sensor callbacks
- Instantiate your UI Updater
- Have a timer occasionally run it

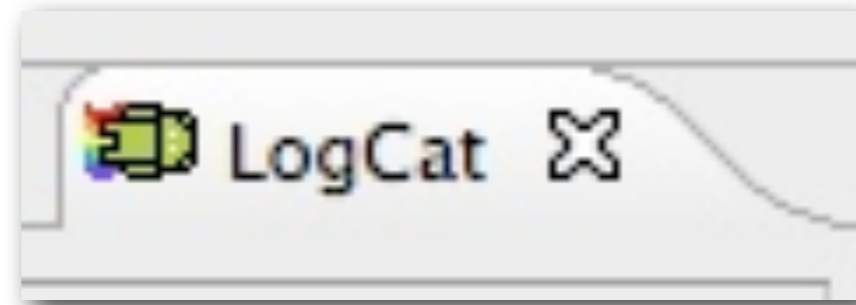
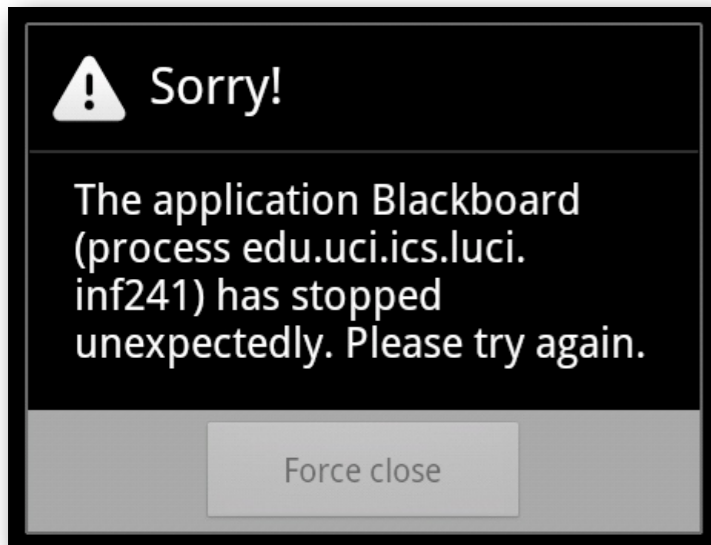


# Hints

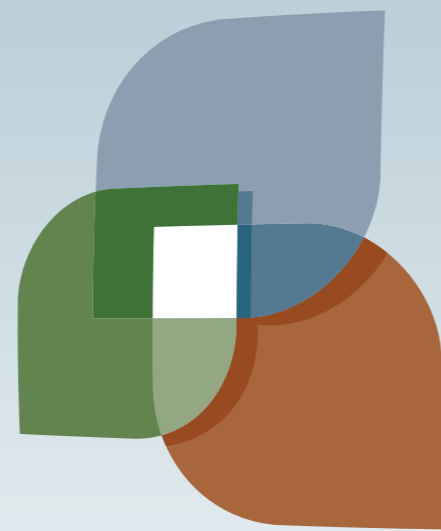
```
public void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.main); //This is from an xml description of the UI  
  
    deviceSensor = new DeviceSensor(); // I implement this class  
  
    /* This is provided by the Android OS */  
    mSensorManager = (SensorManager) getSystemService(SENSOR_SERVICE);  
  
    mSensorManager.registerListener(deviceSensor, mSensorManager  
        .getDefaultSensor(Sensor.TYPE_ORIENTATION),  
        SensorManager.SENSOR_DELAY_FASTEST);  
  
    Timer timerUI = new Timer();  
    UpdateUITask updateValuesTask = new UpdateUITask(this); // I implement this class  
    timerUI.schedule(updateValuesTask, 500, 500);  
}
```



# Troubleshooting



```
W 02-02 14:31:19.308 430 edu.uci.ics.luci.in dalvikvm threadid=1: thread exiting with uncaught exception (group=0x40015560)
E 02-02 14:31:19.318 430 edu.uci.ics.luci.in AndroidRuntime FATAL EXCEPTION: main
E 02-02 14:31:19.318 430 edu.uci.ics.luci.in AndroidRuntime java.lang.RuntimeException: Unable to start activity ComponentInfo{edu.uci.ics
E 02-02 14:31:19.318 430 edu.uci.ics.luci.in AndroidRuntime at android.app.ActivityThread.performLaunchActivity(ActivityThread.java:16
E 02-02 14:31:19.318 430 edu.uci.ics.luci.in AndroidRuntime at android.app.ActivityThread.handleLaunchActivity(ActivityThread.java:166)
E 02-02 14:31:19.318 430 edu.uci.ics.luci.in AndroidRuntime at android.app.ActivityThread.access$1500(ActivityThread.java:117)
E 02-02 14:31:19.318 430 edu.uci.ics.luci.in AndroidRuntime at android.app.ActivityThread$H.handleMessage(ActivityThread.java:931)
E 02-02 14:31:19.318 430 edu.uci.ics.luci.in AndroidRuntime at android.os.Handler.dispatchMessage(Handler.java:99)
E 02-02 14:31:19.318 430 edu.uci.ics.luci.in AndroidRuntime at android.os.Looper.loop(Looper.java:130)
E 02-02 14:31:19.318 430 edu.uci.ics.luci.in AndroidRuntime at android.app.ActivityThread.main(ActivityThread.java:3683)
E 02-02 14:31:19.318 430 edu.uci.ics.luci.in AndroidRuntime at java.lang.reflect.Method.invokeNative(Native Method)
E 02-02 14:31:19.318 430 edu.uci.ics.luci.in AndroidRuntime at java.lang.reflect.Method.invoke(Method.java:507)
E 02-02 14:31:19.318 430 edu.uci.ics.luci.in AndroidRuntime at com.android.internal.os.ZygoteInit$MethodAndArgsCaller.run(ZygoteInit.
E 02-02 14:31:19.318 430 edu.uci.ics.luci.in AndroidRuntime at com.android.internal.os.ZygoteInit.main(ZygoteInit.java:597)
E 02-02 14:31:19.318 430 edu.uci.ics.luci.in AndroidRuntime at dalvik.system.NativeStart.main(Native Method)
E 02-02 14:31:19.318 430 edu.uci.ics.luci.in AndroidRuntime Caused by: java.lang.NumberFormatException: unable to parse 'Preferences table
E 02-02 14:31:19.318 430 edu.uci.ics.luci.in AndroidRuntime at java.lang.Integer.parseInt(Integer.java:383)
E 02-02 14:31:19.318 430 edu.uci.ics.luci.in AndroidRuntime at java.lang.Integer.parseInt(Integer.java:372)
E 02-02 14:31:19.318 430 edu.uci.ics.luci.in AndroidRuntime at java.lang.Integer.parseInt(Integer.java:332)
E 02-02 14:31:19.318 430 edu.uci.ics.luci.in AndroidRuntime at org.openintents.sensorsimulator.hardware.SensorSimulatorClient.connect(
E 02-02 14:31:19.318 430 edu.uci.ics.luci.in AndroidRuntime at org.openintents.sensorsimulator.hardware.SensorManagerSimulator.connect
E 02-02 14:31:19.318 430 edu.uci.ics.luci.in AndroidRuntime at edu.uci.ics.luci.inf241.BlackboardActivity.onCreate(BlackboardActivity.
E 02-02 14:31:19.318 430 edu.uci.ics.luci.in AndroidRuntime at android.app.Instrumentation.callActivityOnCreate(Instrumentation.java:1
E 02-02 14:31:19.318 430 edu.uci.ics.luci.in AndroidRuntime at android.app.ActivityThread.performLaunchActivity(ActivityThread.java:16
E 02-02 14:31:19.318 430 edu.uci.ics.luci.in AndroidRuntime ... 11 more
```



L U C I

