ECE 1778:
Creative Applications for Mobile Devices

Lecture 4

January 29, 2014

Today

1. Logistics – Plan, Assignments
   - Description of Assignment P3/A3
2. Notes on Group Work
   - How to create block diagram
   - Working in Groups
3. External Devices
   - Recon Instruments Goggles
   - TI Sensor TAG
   - Node Sensor
4. Some Notes on Programming Android
   - Life cycle; Sensors; Debugging
5. Proposal Discussions

Logistics

Assignments

- A2 and P2 were due yesterday
- A3 and P3 are ready, and posted
  - But are **not due** for 2 weeks,
  - To give you time to work on your plans, due next week.

- A1 and P1 have been graded
  - Some comments from TA on P1

Assignments P3 and A3

A3 Part 1 Due One week from Friday

A3 Part 2 and P3 Due in 2 weeks [Tuesday]

Assignment P3 – for Programmers

Location, Motion Sensors and Image Capture

- Learning how
  - to determine out where the phone is, geographically (GPS)
  - how it is moving (accelerometer)
  - how to use the camera to capture images

1. Read the relevant parts of
   - **Murphy** & Android Development site (Android)
     - Murphy doesn't have good coverage of sensors
     - Previous lecture has notes on sensors
   - or **Mark, Nutting, LaMarche & Olsson** (iPhone)

Assignment P3 – for Programmers

- In response to being shaken,
  - phone takes a picture 1 second after the shaking stops, and
  - records the GPS location at the same time.
- Each location should be stored in a growing list;
  - when the user views the list item, your application should display the picture taken at that location.
  - The list should be maintained over separate invocations of the app (i.e. stored in a file)
  - it should be possible to delete a list item, which would remove the corresponding image in the file system.
- See note on using the camera preview properly
- Due in 2 weeks,
  - February 11th, 6pm, penalty for being late

Assignment A3 – for Appers

- Recall: one goal of course is to give experience in reaching across disciplines
- Anticipate that Appers will be teaching Programmers the **language** of their discipline, and the basic concepts
  - Please, programmers, ask questions – get jargon explained
  - **AND** vice-verse.

- Assignment A3 is an attempt to bringing Appers a little into the world of programming & engineering
  - To give you practice talking to each other

Basic Idea of A3

- Apper: Choose from one of 5 technical areas listed
  - **that you are not already familiar with,**
  - **And that your programming partners are familiar with:**

1. Fast Searching
2. Databases
3. Digital Signal Processing
4. Optimization
5. Internet Communication

Then – Part 1

- Spend an hour with your partners, learning about this area, and take notes.
  - Don’t use any other sources of information
- Write up those notes & submit this Friday (Feb 7, 6pm)
  - 500 words + pictures

Then – Part 2

- Pursue a deeper understanding of the topic, via Internet
- Write another 500 words, due Tuesday February 11th, 6pm
  - Do a better job of describing the topic; add some nuance.
- Offer some additional commentary on your view of this learning process
  - how it went
  - how much you learned from Part 1 vs. Part 2,
  - what would have made it better

Project

Project Stages

1. **Forming Groups**
   - Must be formed now.
2. **One-Page Proposal**
   - I will respond today to all sent yesterday; remainder as in
3. **Project Plan**
   - Due Next week Feb 5th
4. **Proposal/Plan Presentations**
   - February 12 & 13
   - **NOTE EXTRA LECTURE Thursday Feb 13, 6-8pm, MP 103**
5. **Spiral 2 & Spiral 4 Presentations**
   - 2: March 5/12 4: March 19/26
6. **Final Presentations**
   - Weeks of April 2 & 9
7. **Final Report Due April 10th**

Plan Due Following Week: Feb 5 @ 6pm

1. Goal & Motivation, made more precise
   - What & Why
2. Rough design of what the user sees
   - Mock-ups of screens
   - https://gomockingbird.com or https://moqups.com
     - Any drawing package will do
3. Block Diagram overview of planned code
   - The large (5 or so) pieces of they system
     - With short prose description of each
   - Should be linked to the screens
   - In moment will discuss creation of block diagrams

Plan, continued

4. Statement of Risks/Issues
   - What roadblocks/issues/challenges do you foresee?
   - App-wise, programming-wise, hardware-wise, ethics-wise
5. What do you need to learn that you don't know
   - all members
6. **Important:** Appers
   - Submit a separate essay on how App relates to field of Apper, **and how the Apper will contribute to project**
   - Want clear thought about the activities of the Apper
   - 500 words

- Document must have these sections; will lose marks if missing

Plan Document

- Plan document length: 1500 words max
  - Not including 500 word Apper essay (item #6)
  - Should include pictures
  - Include word count, penalty for overage.

- Seeking clarity, not quantity of words
  - ‘Omit needless words’

- Submit PDF doc to Blackboard under ‘Assignment’ Plan
  - Due Tuesday February 4th at 6pm
  - Worth 10% of grade (includes presentation done following week)

Block Diagrams and Enhancements

Block Diagrams

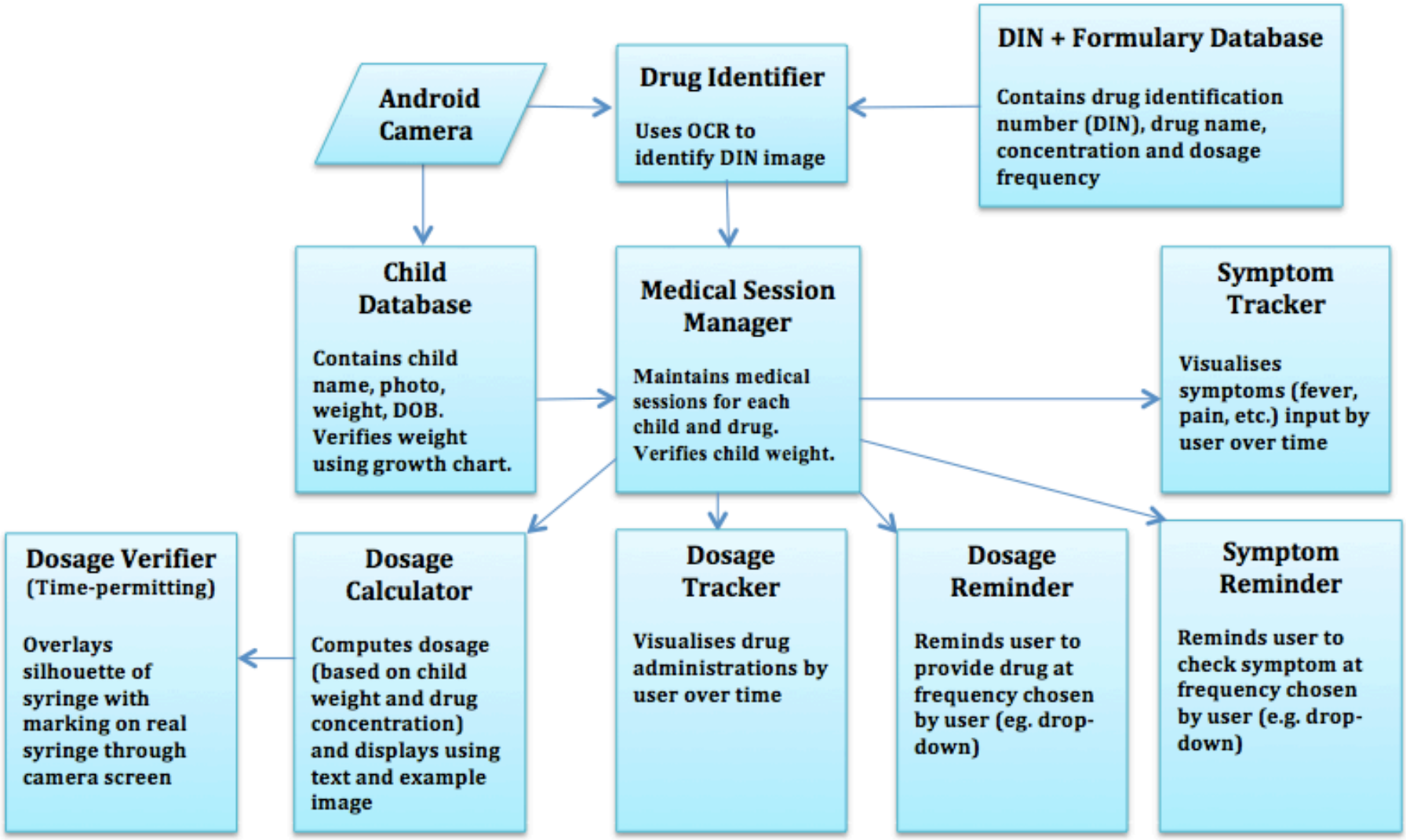
- Block diagram gives the five major pieces of an App's functionality
  - Is the first step in the standard divide & conquer approach
- Draw blocks, give each block a name
  - That gives a sense of what it does
  - Associate words say in more detail what function each block has
- How each block communicates with others

- May need to break blocks themselves down
- Following Example is Snap N' Dose Block Diagram

Snap ‘n Dose

- Goal: To design a mobile application that will increase caregivers’ ability to appropriately dose common over-the-counter liquid medications to children by allowing caregivers to:
  - record child **profiles**
  - add and maintain a drug **inventory**
  - calculate and administer the appropriate **dose** of medication
  - **track** & set **reminders** for medication administration & symptoms

Snap ‘n Dose: Block Diagram



Example: Sobriety App

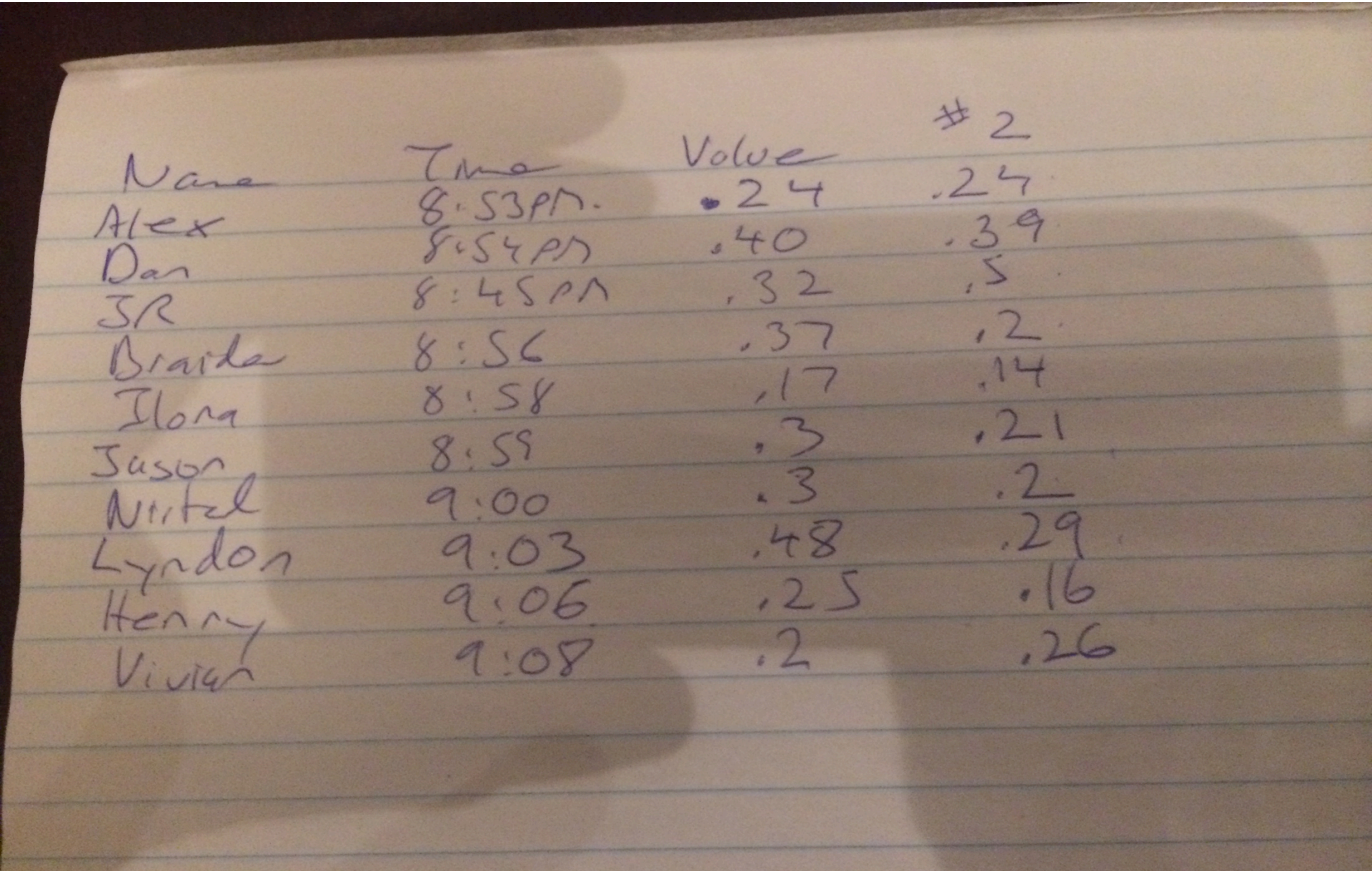
- Based on MyAnkle notion:
  - Use the accelerometer to measure how well you can balance while standing on one foot

- Assume that balance degrades with loss of sobriety:
  - Balance is a function of your brain's ability to coordinate information from three sources: your vision, your sense of body position (derived from subtle changes in muscle tension), and the information sent to the brain by the balance mechanism in your inner ear. When any of the sources do not agree, you feel dizzy and lose your balance (vertigo). Alcohol affects the ability of your brain to interpret that information correctly.
  - Source: http://wiki.answers.com/Q/Why_does_your_balance_go_when_you_are_drunk?

Sobriety App Design

- Assume can make measurement when person is sober
- Make measurement when not
- Assumption:
  - Drinking to Sober difference gives measure of sobriety
  - Requires experimentation

Data from Last Night



Name	Time	Value	#2
Alex	8:53PM.	.24	.24
Dan	8:54PM	.40	.39
JR	8:45PM	.32	.5
Braida	8:56	.37	.2
Ilona	8:58	.17	.14
Jason	8:59	.3	.21
Nirtal	9:00	.3	.2
Lyndon	9:03	.48	.29
Henry	9:06	.25	.16
Vivian	9:08	.2	.26

Let’s Make the Block Diagram

- What are the main components?

Let’s Figure out How to Enhance it

- Think about the context, what could we do to make this more useful/functional/helpful?

Group Interaction

Group Interaction

- Now that groups are formed, it is very important for you to meet regularly and coordinate your activities
- Each of you will need to be assigned tasks, and make commitments to do those tasks at a particular deadline
- The expectations that each group member has of the others must be made clear
  - I suggest that these be written down, and recorded on website
  - The major goals, not small stuff

Difficulties Can and Do Happen

- Most of the difficulties that have occurred in this course have been because expectations were not made clear
- Sometimes, group members consistently failed to live up to commitments.
- If this happens, please report it to me as soon as it is something that you cannot handle with internal discussion & resolution
  - We will help you resolve it
- Disagreements are part of all human relationships
  - One distinguishes oneself by how well you deal with them

User Pepper for Internal Communication

- We have set up separate Pepper interaction spaces for each group
  - Private to the group
  - Listed under the ‘home’ community on pepper
- For posting information, videos, software, etc.
- Can use as a record of work, and a holding space for work
- A place to record the key commitments and deadlines
- Can use a medium of interaction
- Not graded

Class Participation

Class Presentations & Participation

- A key part of what happens in this course is the contribution you make to other's projects
- You will do many presentations in this class
  - Indeed, one side-effect of this project course is some real practice in giving high-quality, concise & clear communication
  - Most presentations will be 5 minutes in length
  - Must be geared so that most people in the class will understand
- Want everyone to come, listen & provide useful input
  - The grading scheme includes participation
  - Expectation that you'll listen and provide thoughtful feedback and suggestions to other's presentation, starting today

Grading Scheme

- Assignments: 16%
  - 4 assignments
- Project: 80%
  - Proposal 5%
  - Plan (incl presentation) 10%
  - Spiral 2 Presentation 10%
  - Spiral 4 Presentation 10%
  - Presentation/Demo 10%
  - Final Report 30%
- **Class Participation 9%**

External Gadgets

Last Year, Donation of Android Goggles

- We have one pair of Recon's instrumented ski goggles
  - Available for use in your project

- Ski Goggles with:
  - Head 'down' display
  - GPS
  - Full android processor with SDK
  - 9-axis accelerometer/gyroscope/compass
  - Bluetooth connection to mobile phone
  - Wrist-based selection tool for input

Features

SPEED

Calculated by GPS combined with barometric pressure data, accurately see how fast you're going as you carve the slopes.

JUMP ANALYTICS

Know your distance, height, and airtime either when pulling tricks in the park, or when hitting backcountry kickers.

VERTICAL

From the peak to the lodge, track your vertical feet by run, by day and over the course of the season.

ALTITUDE

The onboard altimeter tracks your altitude to any mountain summit. Follow your elevation up to the peak, and down to the bottom.

NAVIGATION

Find your way around resorts easily, get to know your favourite runs by name, or track down points of interest easily.

BUDDY TRACKING

Ideal for when you get separated from your group on the slopes. Now you can see where they are, and safely reunite.

SMARTPHONE CONNECTIVITY

Pair with your Smartphone to view incoming calls and read text messages immediately, as you receive them.

MUSIC

Playlist mode puts your music at your fingertips. Be in full control of what you hear as you ski or board.

- See http://www.youtube.com/watch?v=2-9UAJ4v_8M
- and http://www.youtube.com/watch?v=XpY2Oh4stM0
- and http://www.youtube.com/watch?v=VqjGsYf4upI

SDK

- Go to http://developers.reconinstruments.com
  - For tutorials, design guide and downloads

TI Sensor Tag

- Require Android 4.3; Huawei Ascend at 4.2

- Sensor Tag Sensors:
  - Accelerometer, gyroscope, magnetometer
  - Ambient Temperature
  - IR Temperature
  - Air pressure
  - Humidity

Node Sensor

- A single ‘platform’ with many different attachable sensors
- Connected by Bluetooth LE (need Android 4.3/iphone)
- See: http://variableinc.com/products/
- I have one with these attachments
  - IR temperature
  - Paint Colour
  - 9-axis: Accelerometer, Gyro, Magnetometer
  - Climate sensor (Clima)

Other Node Sensors End-Units for Purchase

THERMOCOUPLE

Thermocouple can measure surface temperature temperatures in liquids, semi-solids- foods, and meats, for quality control, and temperature monitoring.

$75 click here to buy or learn more

OXA

With a NODE OXA gas module installed your smart device becomes a super sensor. Each OXA gas module detects one of the following gases: CO, NO, NO2, Cl2, SO2, and H2S.

$149 click here to buy or learn more

CO2

NODE + CO2 is a sensor module for the NODE+ bluetooth sensor platform. The CO2 module measures the Carbon Dioxide level of the air around the sensor.

$149 click here to buy or learn more

BARCODE

We're proud to announce new NODE+Barcode sensor module! NODE+Barcode can scan any item, keeping track of your inventory, pricing and availability.

$99 click here to buy or learn more

Or, build your own:

I/O

The i/o Module allows users to connect different sensors, lights, and buttons, to access the POWER of NODE in their own projects.

$25 click here to buy or learn more

Android Essentials

- Life Cycle
- Sensors
- Debugging
- Pop-Up Messages – Toast & Alerts

Android Activity ‘Life Cycle’

Android Application Life Cycle

- Recall: Activities are screens that the user sees, and associated process
- Android manages these Activities as a **stack**.
- When a new activity is started, it is placed on the top of the stack and becomes the running activity

- The previous activity always remains below it in the stack,
  - and will not come to the foreground again until the new activity exits.

Important to Pay Attention to ‘LifeCycle’

- To ensure app behaves well in several ways, including:

1. Does not crash if the user receives a phone call or switches to another app
2. Does not consume valuable system resources when the user is not actively using your app
3. Does not lose the user's progress if they leave your app and return to it at a later time
4. Does not crash or lose the user's progress when the screen rotates between landscape and portrait orientation.

An Activity Can Be in 1 of 4 ‘States’

State 1: Active/Running

- Activity in the foreground of the screen (at the top of the stack)
- Has ‘focus’, meaning user interactions go to it.

State 2: Paused

- activity has lost focus but is still visible
- a new smaller or transparent activity has focus on top of the activity)
- A paused activity is completely alive (it maintains all state and member information and remains attached to the window manager), but can be killed by the system in extreme low memory situations.

Activity States 3 and 4

State 3: Stopped

- activity is completely obscured by another activity
- retains all state and member information
- no longer visible to the user so its window is hidden
- it will often be killed by the system when memory is needed elsewhere.

State 4: Destroyed

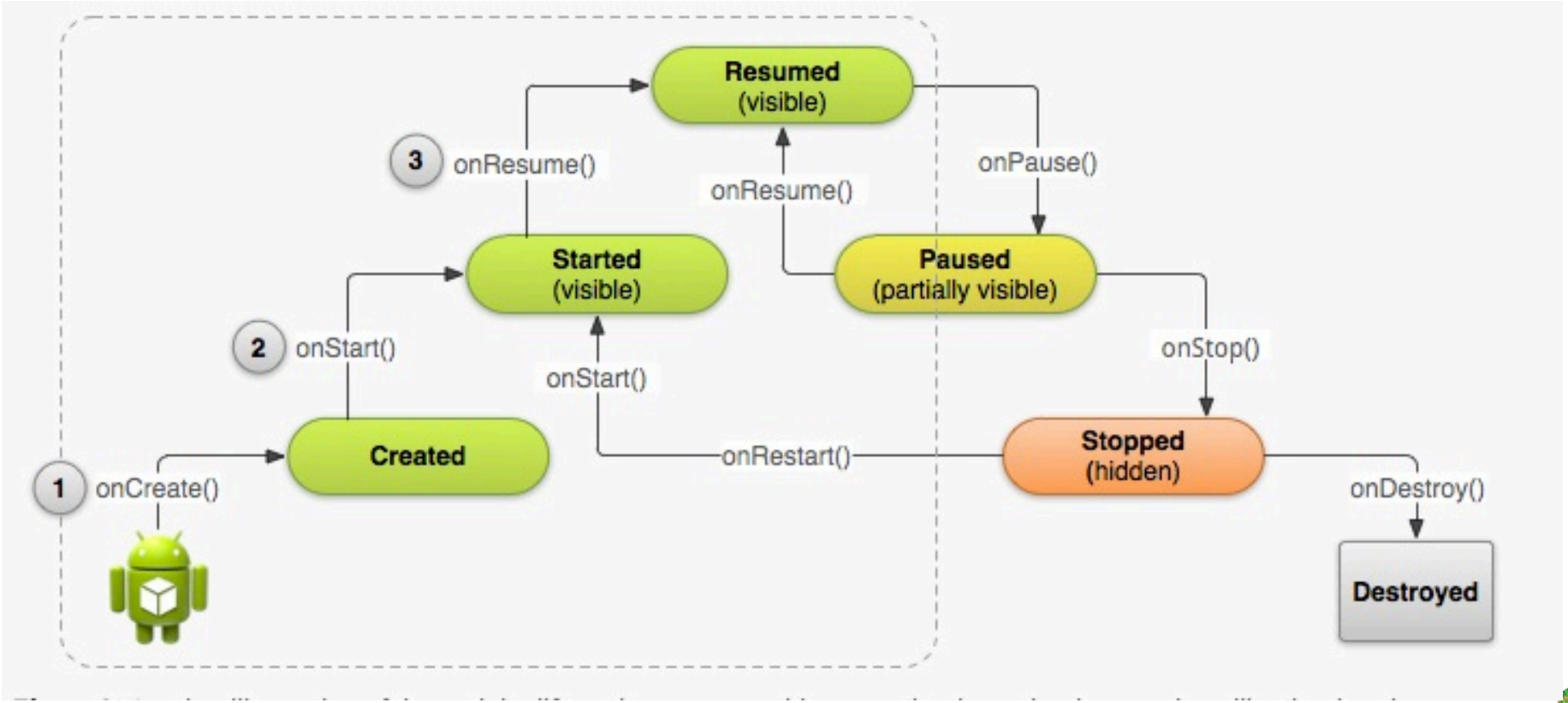
- If an activity is paused or stopped, the system can drop the activity from memory by either asking it to finish, **or simply killing its process.**
- When displayed again to the user, it must be completely restarted and restored to its previous state.

Android Talking to Your App

- The Android operating system asks (or tells) your app to go into those different states by invoking methods associated with your Activity

Methods Called By Android to Change States

- Diagram shows states and methods called to change state
  - Colours: the states
  - MethodName(): methods called by Android OS



Three Key States

- Activity can be in 1 of 3 states for long period of time:

1. Resumed
   - In this state, the activity is in the foreground and the user can interact with it. (Also sometimes referred to as the "running" state.)
2. Paused
   - In this state, the activity is partially obscured by another activity—the other activity that's in the foreground is semi-transparent or doesn't cover the entire screen. The paused activity does not receive user input and cannot execute any code.
3. Stopped
   - In this state, the activity is completely hidden and not visible to the user; it is considered to be in the background. While stopped, the activity instance and all its state information such as member variables is retained, but it cannot execute any code.

State Management

- The other states (Created and Started) are transient and the system quickly moves from them to the next state by calling the next lifecycle callback method. That is, after the system calls onCreate(), it quickly calls onStart(), which is quickly followed by onResume().

- Depending on the complexity of your activity, you probably don't need to implement all the lifecycle methods.
- However, it's important that you understand each one and implement those that ensure your app behaves the way users expect.

References

1. The Android Documentation:

http://developer.android.com/training/basics/activity-lifecycle/index.html

2. Murphy, Busy Coder's Android, Pages 280-301
   - "Activities and their Lifecycles"

- Once your project gets going, it is really important to read through this and understand it
  - Previous years' students pointed out that this was the key thing they had not understood in Android, that caused the most problems

The Key 'LifeCycle' Methods

OnCreate()

- Familiar with already – brings the activity to life

OnPause()

- Another Activity has gained the 'focus'
- Should stop any background threads, release large resources (such as a camera)
- **No guarantee that OnDestroy() will be called**, so best to save **all** state here

OnResume()

- Called as activity starts, **or** is restarted from a pause
- Can recall state from file, refresh the User Interface – see example

Relating to Sensors

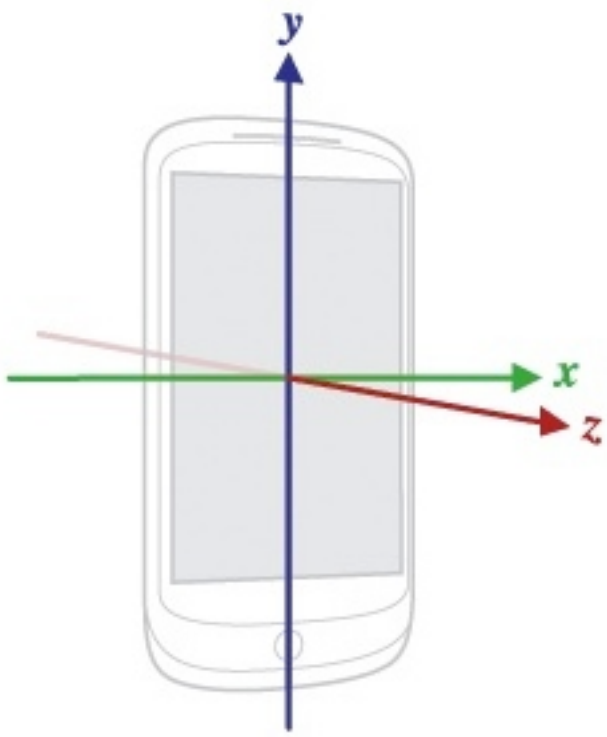
For Assignment P3 (due in 2 weeks)

And general sensor stuff

The Accelerometer

The Accelerometer

- The Accelerometer in phones is quite an exciting sensor!
- It can be used to feel motion in three dimensions
  - It samples the acceleration at least 100 times/second
  - But not reliably!
  - It is very sensitive
- It measures acceleration in **m/s^2**



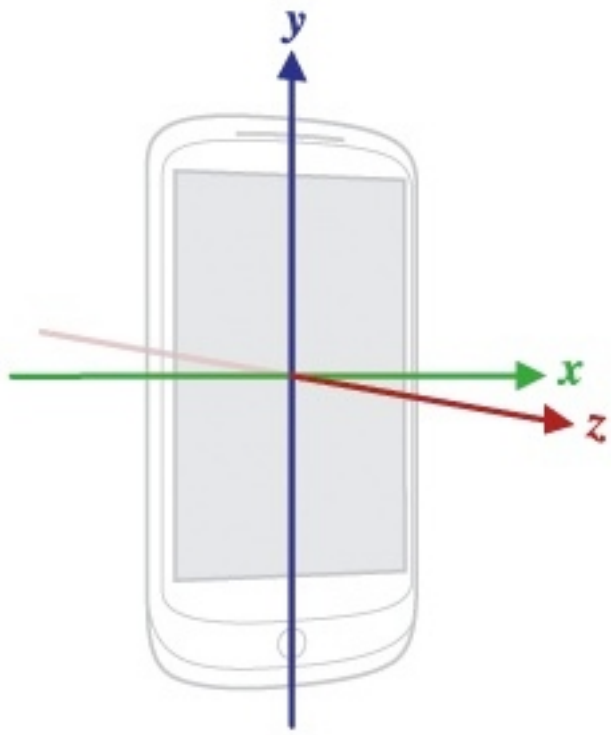
Gravity

- **Recall**: acceleration due to gravity is 9.8 m/s^2
- When the phone isn't moving, one or more of the axes will 'feel' this acceleration due to gravity or part of it.
- A phone that is falling, will feel 0 acceleration on all three axes

- Online Android documentation describes how to isolate out both gravity (to figure out which axes it is on) and motion aside from gravity:
  - http://developer.android.com/reference/android/hardware/Sensor.html

The Accelerometer Coordinate Space

- Coordinate-system is defined relative to the screen of the phone in its default orientation.
  - axes are **not** swapped when the device's screen orientation changes.
- X axis is horizontal & points right
- Y axis is vertical & points up
- Z axis points towards the outside of the front face of the screen.
  - coordinates behind the screen have negative Z values.



Using the Accelerometer

- Bones of an application to
  - Read the accelerometers every time they change
  - Output the raw values in each dimension
  - Will include gravity

To Access a Sensor

Three key classes:

1. SensorManager
2. Sensor
3. SensorEvent

- Need to create a SensorManager class

SensorManager

- Is a class that lets you figure out what sensors are on the device:
  - produces a list of all the sensors available,
  - your program must need to check and see if the one you want is actually on the device:
- First, create the general sensor manager

```
myManager = (SensorManager) getSystemService
        (Context.SENSOR_SERVICE);
```

- Then, ask if the sensor you want is on the phone, e.g. the Accelerometer:

```
sensors = myManager.getSensorList
        (Sensor.TYPE_ACCELEROMETER);
```

SensorEvent Class

- ‘event’ is signaled when there is a new reading
- SensorEvent class contains the reading of the sensor, including:
  1. Accuracy of the measurement
  2. Sensor that generated the event
  3. Timestamp
     - The time in nanoseconds at which the event happened
     - Can check the spacing, in time, of your readings!
  4. Values of the reading itself
     - e.g the three values of the acceleration along each axis

Declarations & usual

```
public class readaccel2 extends Activity {
    private TextView accText;
    private SensorManager myManager;
    private List<Sensor> sensors;
    private Sensor accSensor;
@Override
   public void onCreate(Bundle savedInstanceState) {
       super.onCreate(savedInstanceState);
       setContentView(R.layout.main);
       accText = (TextView)findViewById(R.id.accText);
```

Set-up Sensor

```
// Set Sensor + Manager
myManager = (SensorManager)
getSystemService(Context.SENSOR_SERVICE);


sensors =
myManager.getSensorList(Sensor.TYPE_ACCELEROMETER);
    if(sensors.size() > 0)
     {
        accSensor = sensors.get(0);
      }
  }
```

Listening, Reading, Output

```
private final SensorEventListener mySensorListener = new
SensorEventListener()  {
    public void onSensorChanged(SensorEvent event)  {
        float x = event.values[0];
        float y = event.values[1];
        float z = event.values[2];
        String output = String
         .format(”x: %f / y: %f / z: %f", x, y, z);
         accText.setText(output);  }
public void onAccuracyChanged(Sensor sensor, int
accuracy) {}};
```

Registering the Listener

```
@Override
   protected void onResume()
   {
    super.onResume();
    myManager.registerListener(mySensorListener,
    accSensor, SensorManager.SENSOR_DELAY_GAME);
   }
```

- Last parameter sets the frequency of updates

OnPause – Turn off accelerometer

```
@Override
    protected void onPause()
    {
     myManager.unregisterListener(mySensorListener);
     super.onStop();
    }
}
```

- i.e. unregister the listener

To Use the Compass (Magnetometer)

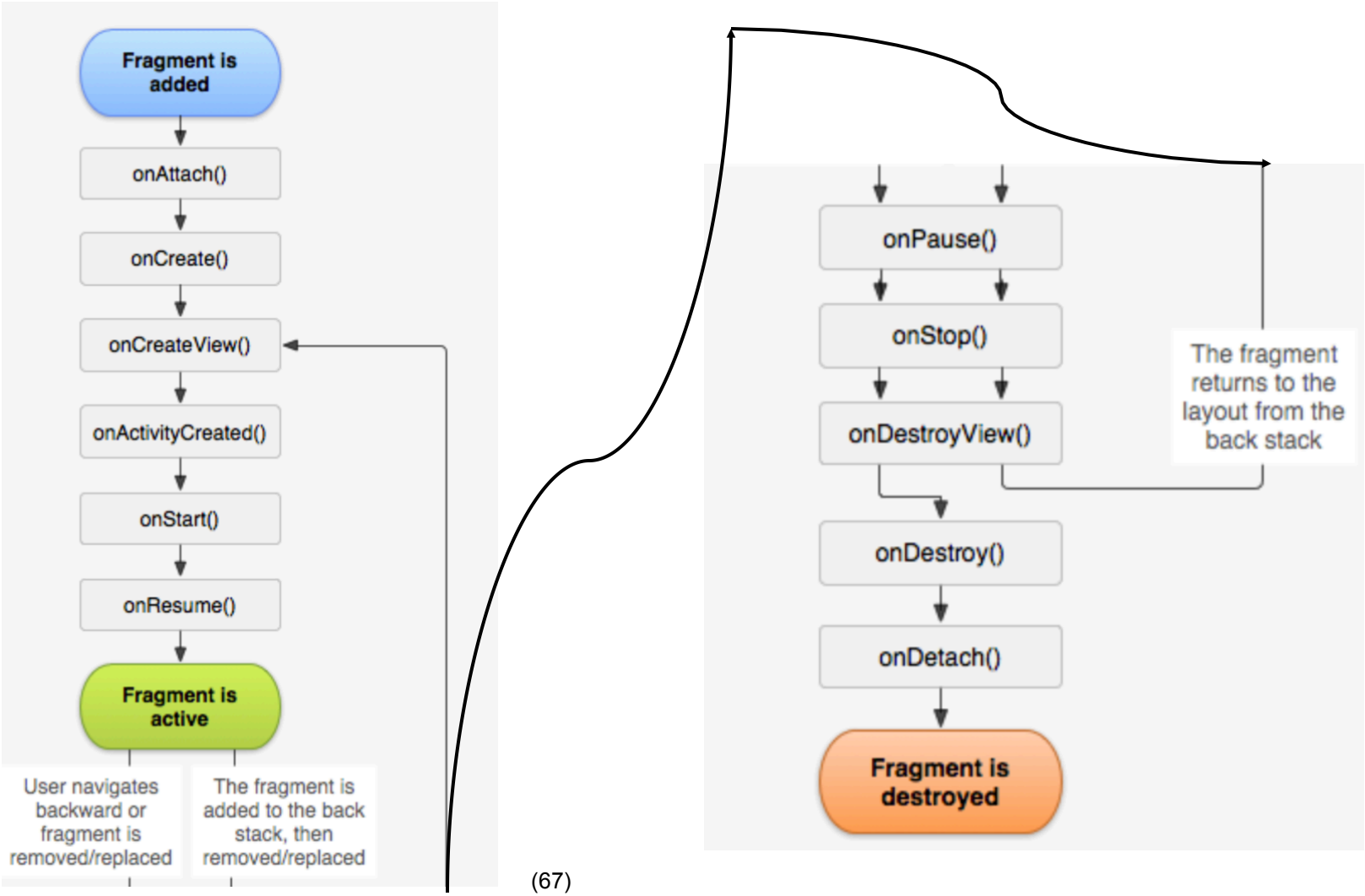
- Change:

```
sensors = myManager.getSensorList
                    (Sensor.TYPE_ACCELEROMETER);
```

- To:

```
sensors = myManager.getSensorList
                    (Sensor. TYPE_MAGNETIC_FIELD);
```

Fragments Behave Similarly



The Other Sensors

There are quite a few Sensors!

- Some are just different calculations with same sensor

1. Accelerometer:
   - Sensor.TYPE_ACCELEROMETER
2. Gravity
   - The accelerometer with non-gravity acceleration filtered out?
   - Sensor.TYPE_GRAVITY
   - Only available on Android 2.3 and later
3. Linear Acceleration
   - The accelerometer with gravity removed (filtered)
   - Sensor.TYPE_LINEAR_ACCELERATION
   - Only available on Android 2.3 and later

Compass/Magnetic field

- Sensor.TYPE_MAGNETIC_FIELD:
- Measures the magnetic field in three dimensions,
  - Measured in micro Tesla
- So, just Change:

```
sensors = myManager.getSensorList
                    (Sensor.TYPE_ACCELEROMETER);
```

- To:

```
sensors = myManager.getSensorList
                    (Sensor. TYPE_MAGNETIC_FIELD)
```

Gyroscope

- Sensor.TYPE_GYROSCOPE:
- Gives pitch, roll and yaw in radians/second
  - Measures motion more directly

Light Sensor

- Used for measuring ambient light to set screen brightness
- Measures the light, in Lux, coarsely
  - Seem like roughly 8 different values, maybe, when I tried it on the Nexus One
- Sensor. TYPE_LIGHT
- Available on Huawei Ascend P6
- DEMO

Proximity Sensor

- Used for measuring how close the phone is to person's ear
- To turn off the touch screen
- Usually binary: **close** or **far**
- Sensor.TYPE_PROXIMITY

- DEMO

Useful Method – sensor maximum

- MaxRange = accSensor.getMaximumRange();
- Tells the largest value a sensor can deliver on a device

Using the Debugger in Eclipse/Android

Debugging

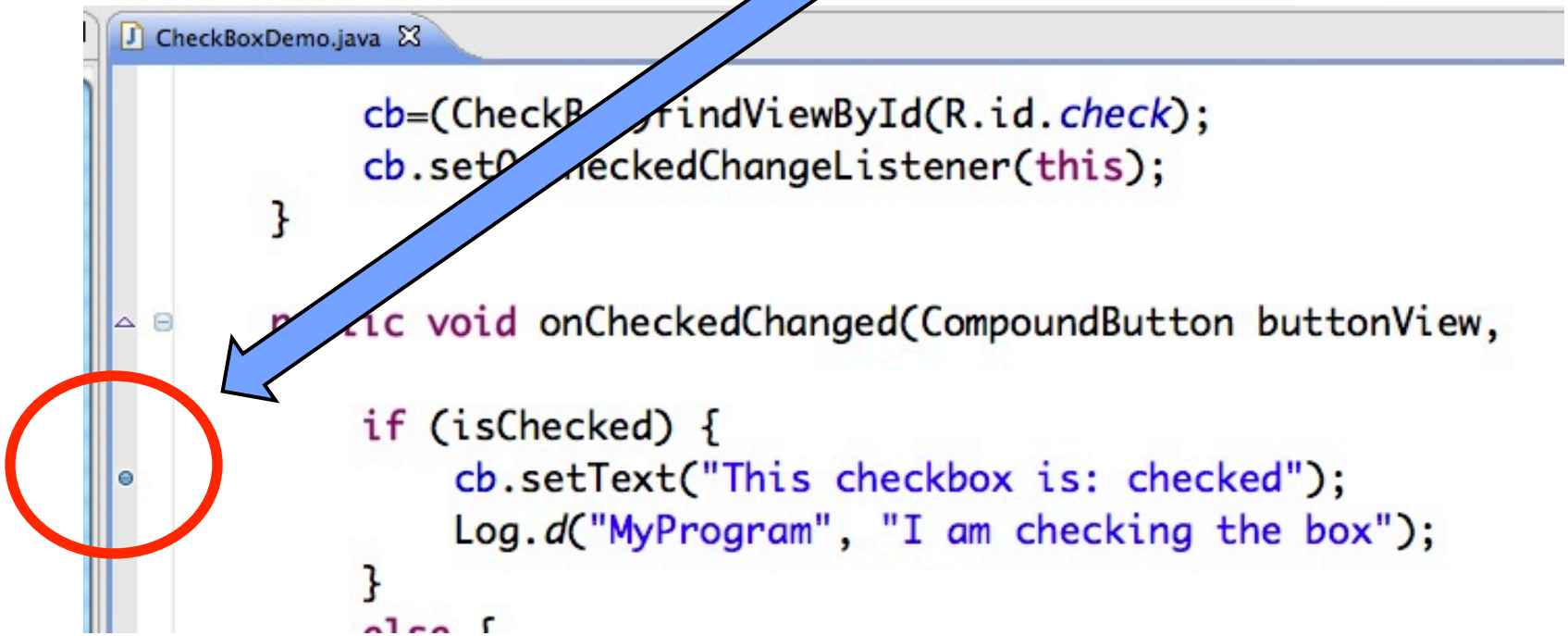
- Simple debug: use Log.d(TAG, “String”);
- Better: use the debugger in Eclipse/Android

- First, make sure your phone or emulator is set to enable ‘USB debugging’
- From the home screen:
  - Settings->Applications->Development
  - Check USB Debugging (have to do that 7-times magic incantation)

Set a Breakpoint

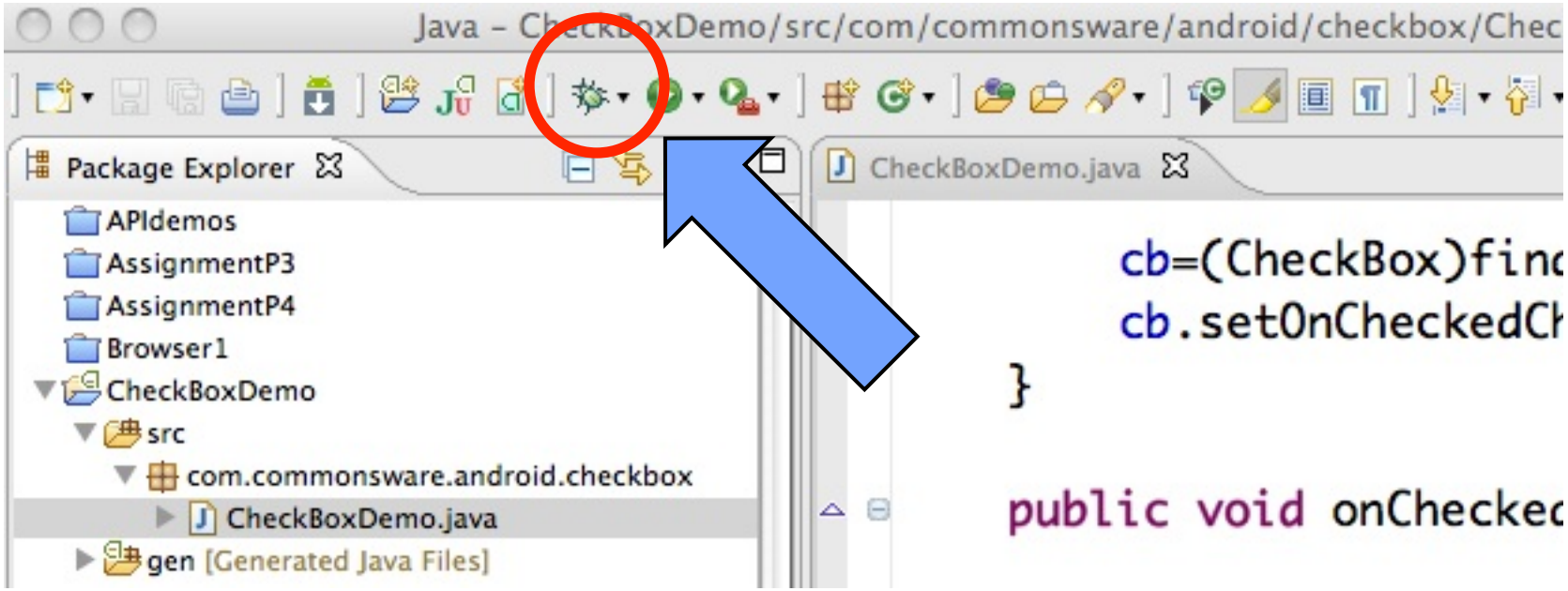
- Go to your source file,
- right click in the left-most column
- Select ‘Toggle Break Point’
  - Will set a breakpoint at that source code line (little blue dot)

Right Click over here



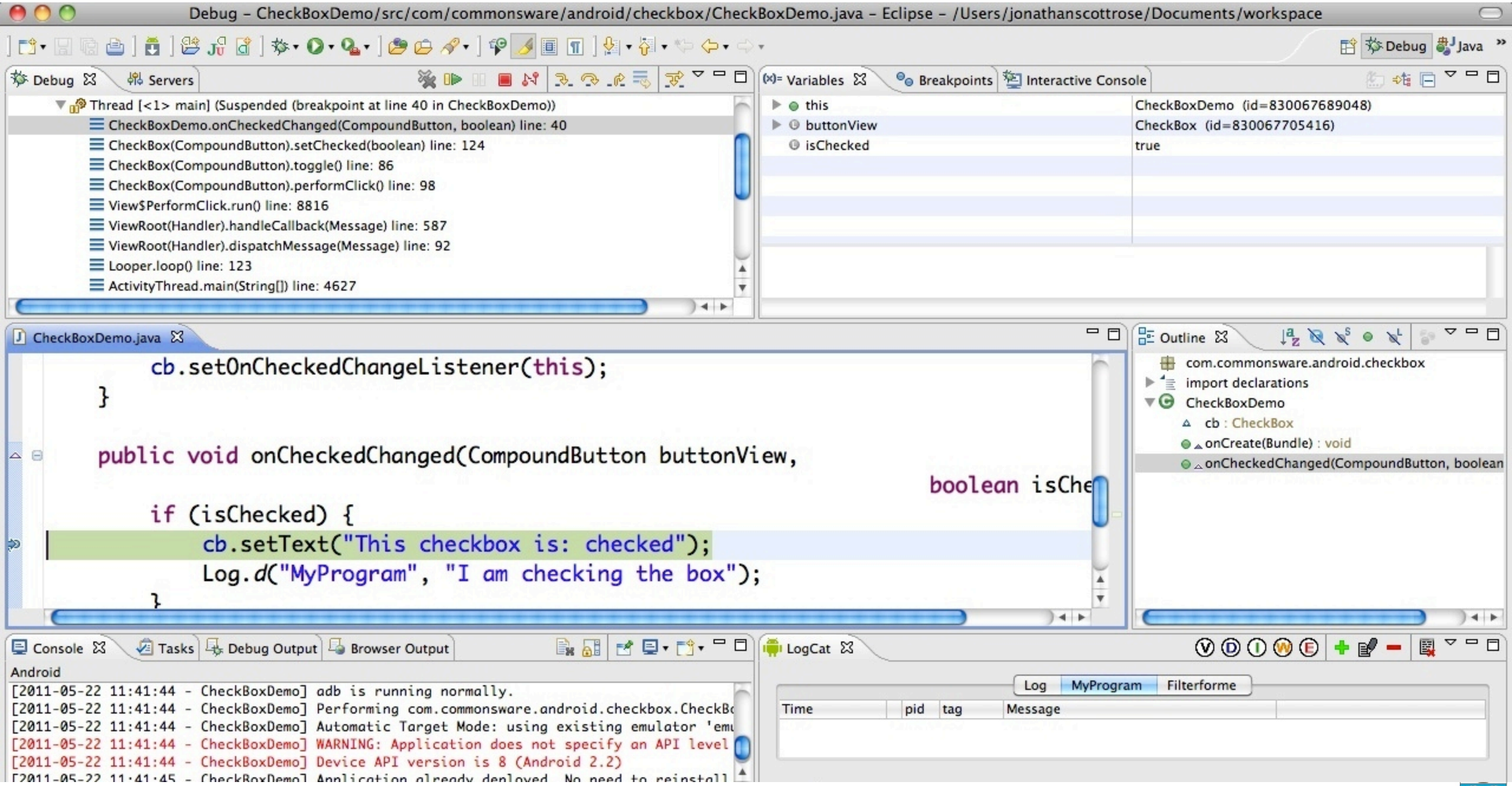
To Run the Debugger

- Click the ‘green bug’ to the left of to the ‘play’ button



Once the Breakpoint is Hit

- The whole Debugger perspective shows up:



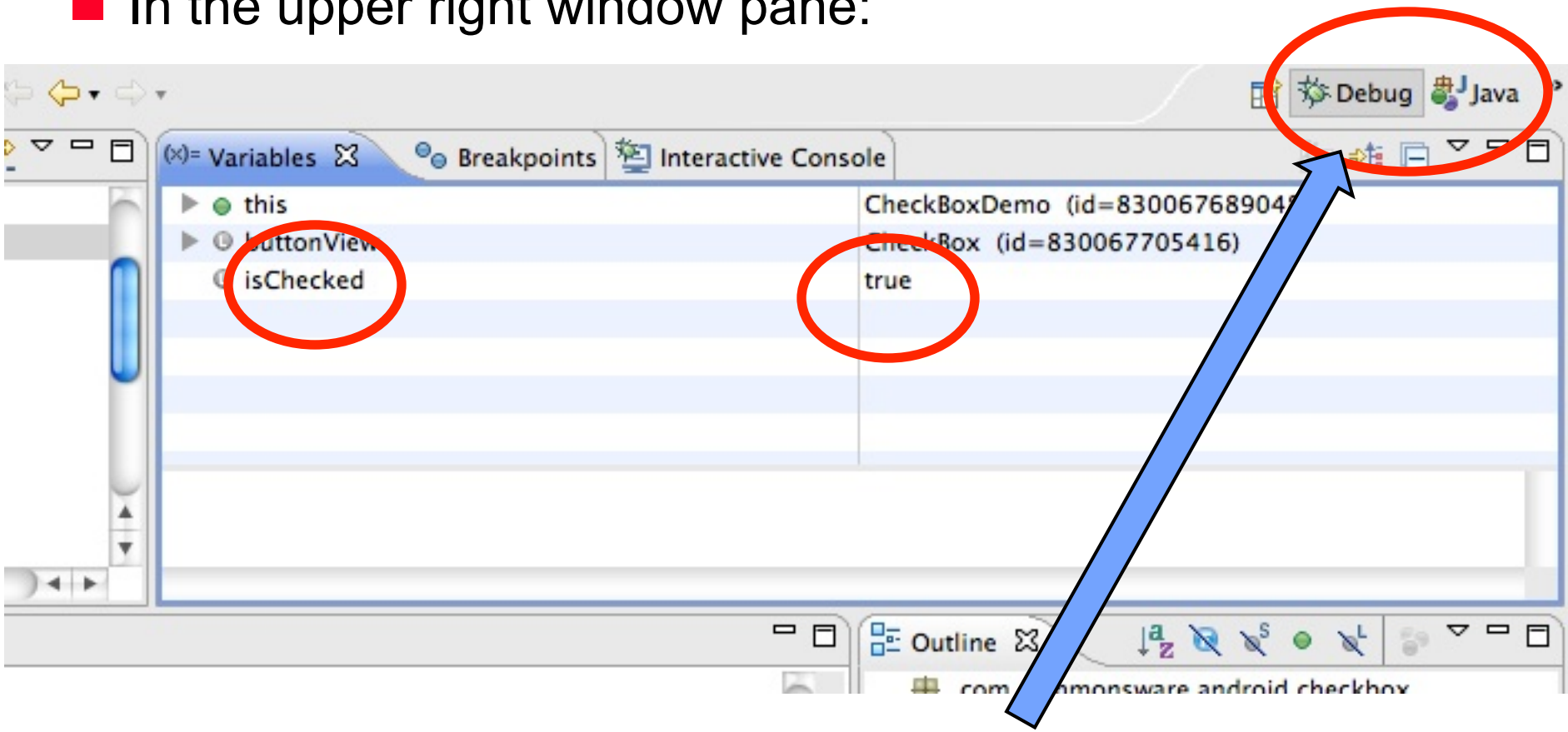
Can Single Step over, Step In, Run

- See this graphic near the top:

1. Continue running from Breakpoint
2. Step in to a function
3. Step over a line (single step)

Can View Value of Variables

- In the upper right window pane:



- To switch between regular and debug perspectives

API Demos!

Google's API Demos

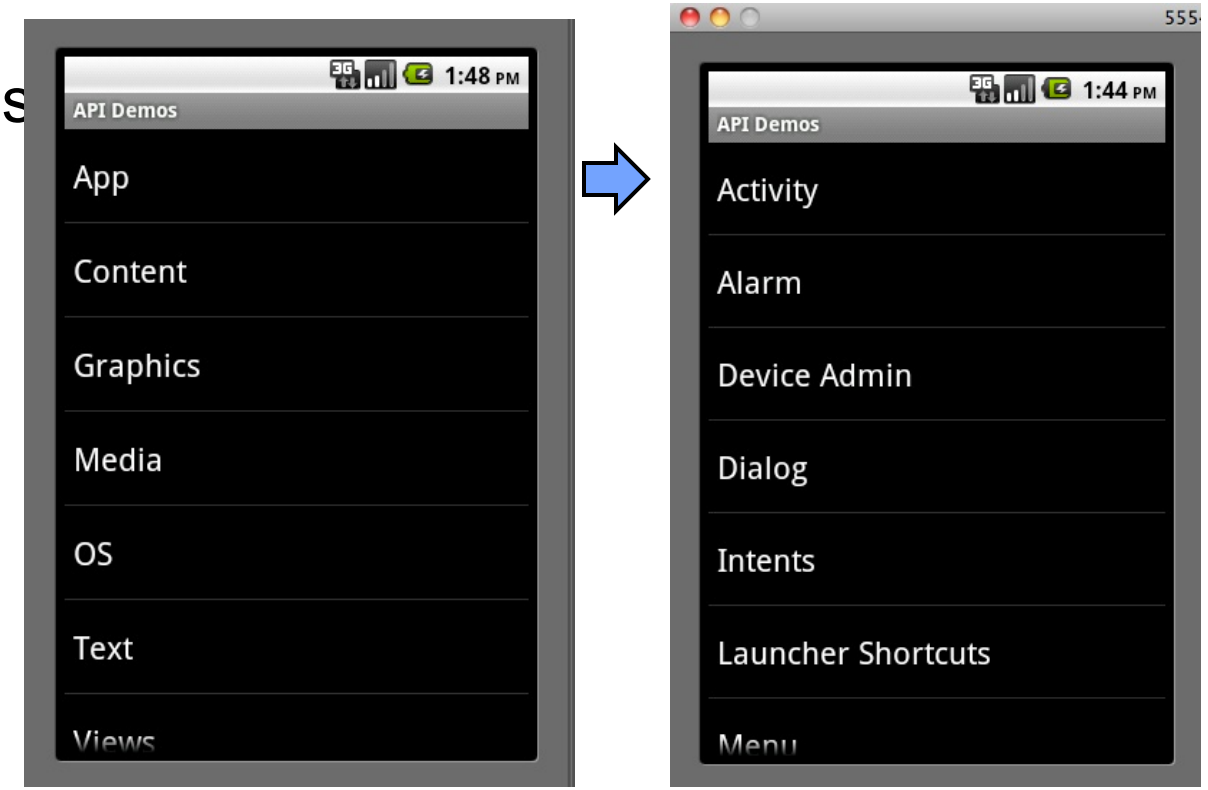
- Are great for learning everything Android
- Every feature of the phone is used, in a simple example

How:

- Be sure to have downloaded the 'Samples for SDK' from the SDK Manager available in Eclipse
- To see them, create a new project in Eclipse (not an Android Project)
  - Under 'Android Project'
  - Select 'Android Project from Existing Code'
- Load in the project from this directory:
  - adt-bundle/sdk/samples/android-N/APIdemos

Then, Experiment and Learn!

- Each feature – activity, camera, sensor, is shown as a simple example, along with code
- Lots!
- Demo



Pop-up Messages

Pop-Up Messages

- Are really handy for conveying an alert or information to the user.

- Two kinds in Android:
  1. **Toasts**
  2. **Alerts**

Toasts

- A temporary message that appears and then disappears after a fixed amount of time
  - e.g. battery low warning
  - Someone signed in to Skype
  - Won lottery
- Purpose is to be un-obtrusive
  - Doesn't take 'focus' away from your activity

Toast Code – Easy!

- Create and show, all in one:

```
Toast.makeText(this,
                    "Eureka, you win!",
                    Toast.LENGTH_LONG).show();
```

Three fields:

1. 'this' is the current class context
   - Can also use **getApplicationContext**() instead
2. The text to be shown
3. Length of time to show
   - LENGTH_LONG or LENGTH_SHORT constants

Alerts

- A specific message that changes focus
- User must respond by clicking

- Three parts:
  - Message title
  - Actual message
  - Up to three response buttons:
    - Positive
    - Neutral
    - Negative
- Use OnClick to respond to buttons

Alert Code – using Toast to respond

```
new AlertDialog.Builder(this)
        .setTitle("MessageDemo")
        .setMessage("How are you feeling?")
        .setPositiveButton("I'm Positive",

 new DialogInterface.OnClickListener() {
    public void onClick(DialogInterface dlg, int sumthin) {
    Toast.makeText(MessageDemo.this, "Eureka, you win!",
                    Toast.LENGTH_LONG).show();
        }
})
```

Construction of the part of the Alert

A Toast in response to the positive click

Neutral Button in Alert

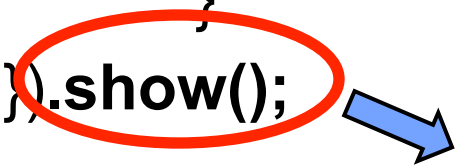
- A cascade of method calls

```
.setNeutralButton("Either Way",
    new DialogInterface.OnClickListener() {
    public void onClick(DialogInterface dlg, int sumthin) {
    Toast.makeText(MessageDemo.this, "Neutral Feeling",
                Toast.LENGTH_SHORT).show();
        }
})
```

Neutral Button in Alert

- Another cascade

```
.setNegativeButton("Either Way",
    new DialogInterface.OnClickListener() {
    public void onClick(DialogInterface dlg, int sumthin) {
   Toast.makeText(MessageDemo.this, ”I’m feeling bad",
                Toast.LENGTH_SHORT).show();
        }
}).show();
```



This shows the Alert

DEMO: MessageAlert