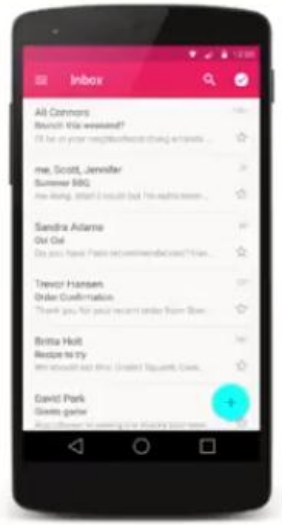
Lists and Sensors

Listviews

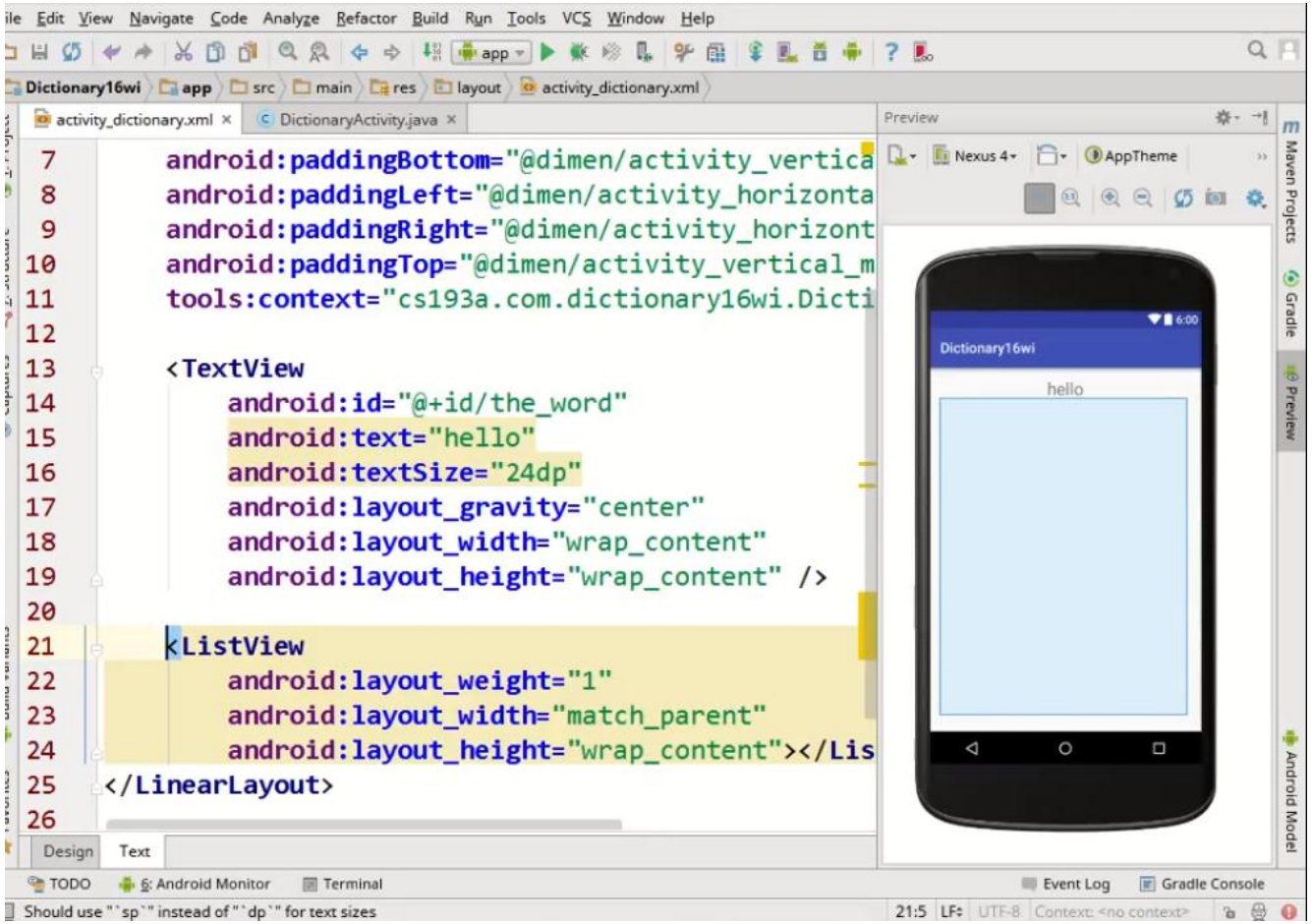
An ordered collection of selectable choices



- key attributes in XML:

`android:clickable="bool"`	set to false to disable the list
`android:id="@+id/theID"`	unique ID for use in Java code
`android:entries="@array/array"`	set of options to appear in the list (must match an array in `strings.xml`)

Listview Example Xml



Static List

- **static list**: Content is fixed and known before the app runs.
  - Declare the list elements in the **strings.xml** resource file.

```
<!-- res/values/strings.xml -->
<resources>
    <string-array name="oses">
        <item>Android</item>
        <item>iPhone</item>
        ...
        <item>Max OS X</item>
    </string-array>
</resources>

<!-- res/layout/activity_main.xml -->
<ListView ... android:id="@+id/mylist"
    android:entries="@array/oses" />
```

Dynamic List

- **dynamic list**: Content is read or generated as the program runs.
  - Comes from a data file, or from the internet, etc.
  - Must be set in the Java code.

  - Suppose we have the following file and want to make a list from it:

```
// res/raw/oses.txt
Android
iPhone
...
Max OS X
```

List Adapters

- **adapter**: Helps turn list data into list view items.
  - common adapters: `ArrayAdapter`, `CursorAdapter`

- Syntax for creating an adapter:

```
ArrayAdapter<String> name =
  new ArrayAdapter<>(activity, layout, array);
```

  - the ***activity*** is usually `this`
  - the default ***layout*** for lists is `android.R.layout.simple_list_item_1`
  - get the ***array*** by reading your file or data source of choice
    (it can be an array like `String[]`, or a list like `ArrayList<String>`)

  - Once you have an adapter, you can attach it to your list by calling the `setAdapter` method of the `ListView` object in the Java code.

Dynamic List Code

```
ArrayList<String> myArray = ...;  // load data from file

ArrayAdapter<String> adapter =
  new ArrayAdapter<>(
      this,
      android.R.layout.simple_list_item_1,
      myArray);

ListView list = (ListView) findViewById(R.id.mylist);
list.setAdapter(myAdapter);
```

Handling events

- Unfortunately lists don't use a simple `onClick` event.
  - Several fancier GUI widgets use other kinds of events.
  - The event listeners must be attached in the Java code, not in the XML.
  - Understanding how to attach these event listeners requires the use of Java **anonymous inner classes**.

- **anonymous inner class**: A shorthand syntax for declaring a small class without giving it an explicit name.
  - The class can be made to extend a given super class or implement a given interface.
  - Typically the class is declared and a single object of it is constructed and used all at once.

Attaching Event Listener in JAVA

```
<!-- activity_main.xml -->
<Button ... android:onClick="mybuttonOnClick" />
<Button ... android:id="@+id/mybutton" />

// MainActivity.java
public void mybuttonOnClick() { ... }
Button button = (Button) findViewById(R.id.mybutton);
button.setOnClickListener(new View.OnClickListener() {
    public void onClick(View v) {
        // code to run when the button gets clicked
    }
});

// this was the required style for event listeners
// in older versions of Android :-/
```

List Events

- List views respond to the following events:
  - **setOnItemClickListener**(AdapterView.OnItemClickListener)
    Listener for when an item in the list has been clicked.
  - **setOnItemLongClickListener**(AdapterView.OnItemLongClickListener)
    Listener for when an item in the list has been clicked and held.
  - **setOnItemSelectedListener**(AdapterView.OnItemSelectedListener)
    Listener for when an item in the list has been selected.
- Others:
  - onDrag, onFocusChanged, onHover, onKey, onScroll, onTouch, ...

List Event Example

```
ListView list = (ListView) findViewById(R.id.id);
list.setOnItemClickListener(
    new AdapterView.OnItemClickListener() {
        @Override
        public void onItemClick(AdapterView<?> list,
                                View row,
                                int index,
                                long rowID) {
            // code to run when user clicks that item
            ...
        }
    }
);
```

List View Alternate Methods

- Or
  - Pass parameter this,
  - Use IDE to help (impements….)
  - Write the code inside generated function..

```
public class MyActivity extends Activity
        implements AdapterView.OnItemClickListener {

        ...
        ListView list = (ListView) findViewById(R.id.id);
        list.setOnItemClickListener(this);
        ...

    @Override
    public void onItemClick(AdapterView<?> list,
                            View row,
                            int index,
                            long rowID) {
        // code to run when user clicks that item
        ...
    }
}
```

ListView Methods

Method	Description
`getAdapter()`	returns array adapter
`getCount()`	# of items in list
`getSelectedItem()`	item currently selected, if any
`getSelectedItemIndex()`	currently selected item's index
`performItemClick(`***`view, index, id`***`)`	simulate a click
`setAdapter(`***`adapter`***`)`	sets array adapter
`setFocusable(`***`boolean`***`)`	whether items can be focused on
`setOnItemClickListener(`***`L`***`)`	attaching event listeners
`setOnItemLongClickListener(`***`L`***`)`	
`setOnItemSelectedListener(`***`L`***`)`	
`setSelection(`***`index`***`)`	sets which element is selected

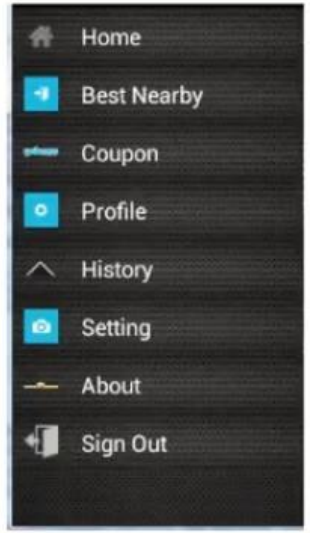
When List Data Changed

- If the data in the list changes, you must notify the list.

```
ArrayList<String> myArray = ...;
ArrayAdapter<String> adapter = ...;
ListView list = (ListView) findViewById(R.id.mylist);
list.setAdapter(myAdapter);

// data changes in some way
myArray.remove(0);
...

// need to notify the ListView of the change!
myAdapter.notifyDataSetChanged();
```

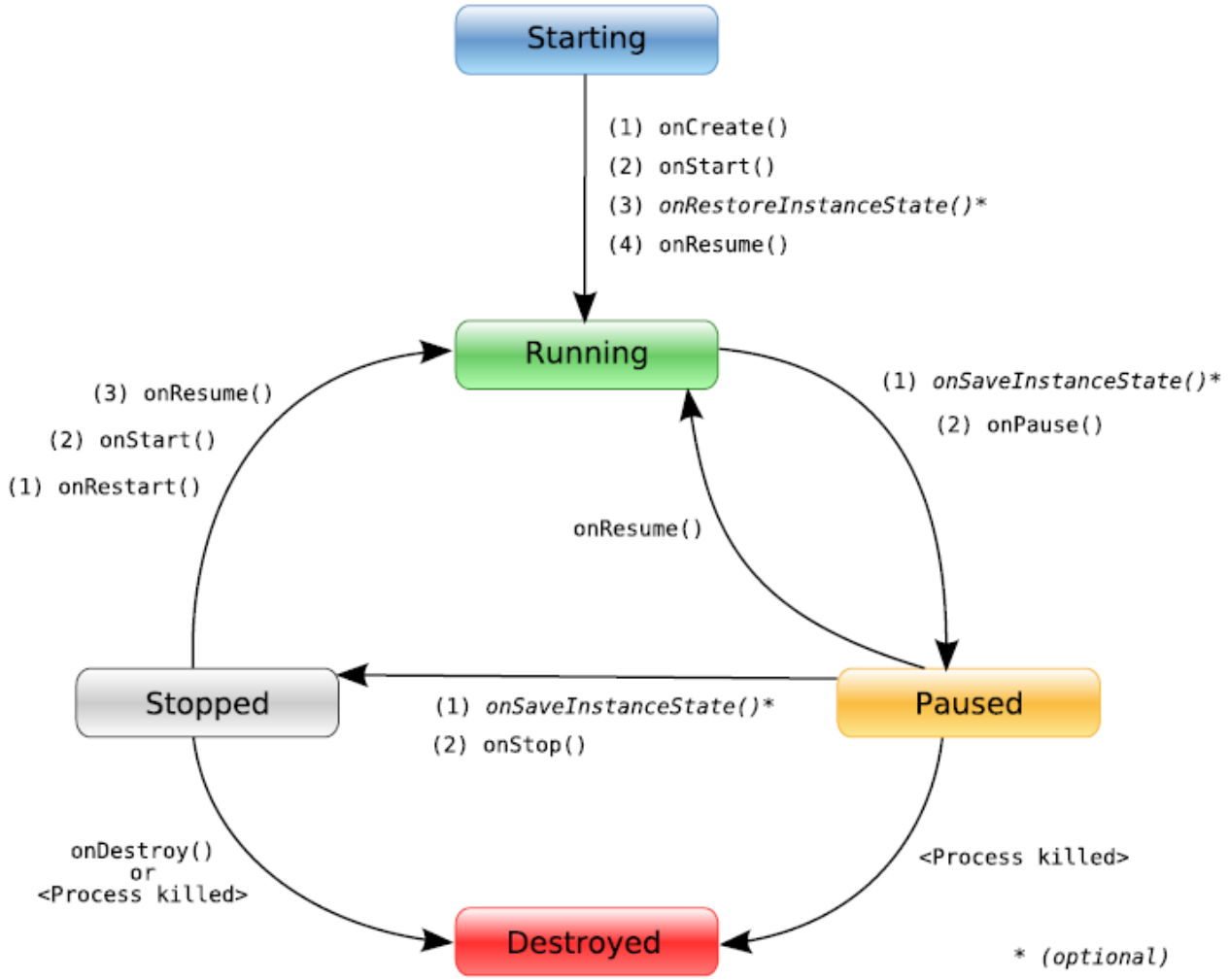
Custom List Layout

- If you want your list to look different than the default appearance (of just a text string for each line), you must:
  - Write a short **layout XML file** describing the layout for each row.
  - Tell your **`ArrayAdapter`** how to supply the item text to each item.



Sensors

Activity Life-cycle



Types of Sensors

- Accelerometer
- Gravity sensor
- Linear Acceleration sensor
- Magnetic Field sensor
- Orientation sensor
- Gyroscope
- Light sensor
- Proximity sensor
- Temprature sensor
- Pressure sensor
- Other sensor

Types of Sensors

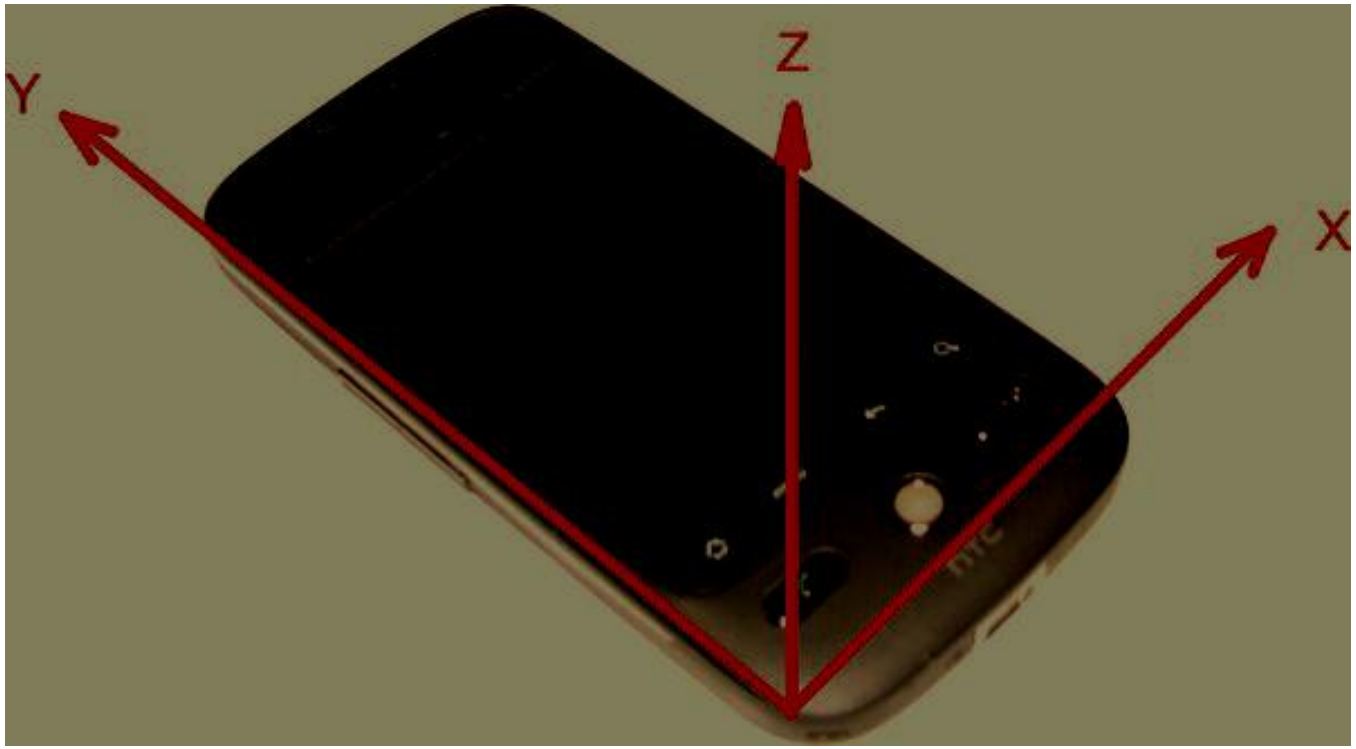
int	TYPE_ACCELEROMETER
int	TYPE_ALL
int	TYPE_AMBIENT_TEMPERATURE
int	TYPE_GAME_ROTATION_VECTOR
int	TYPE_GEOMAGNETIC_ROTATION_VECTOR
int	TYPE_GRAVITY
int	TYPE_GYROSCOPE
int	TYPE_GYROSCOPE_UNCALIBRATED
int	TYPE_HEART_RATE
int	TYPE_LIGHT
int	TYPE_LINEAR_ACCELERATION
int	TYPE_MAGNETIC_FIELD
int	TYPE_MAGNETIC_FIELD_UNCALIBRATED

int	TYPE_PRESSURE
int	TYPE_PROXIMITY
int	TYPE_RELATIVE_HUMIDITY
int	TYPE_ROTATION_VECTOR
int	TYPE_SIGNIFICANT_MOTION
int	TYPE_STEP_COUNTER
int	TYPE_STEP_DETECTOR

Accelerometer

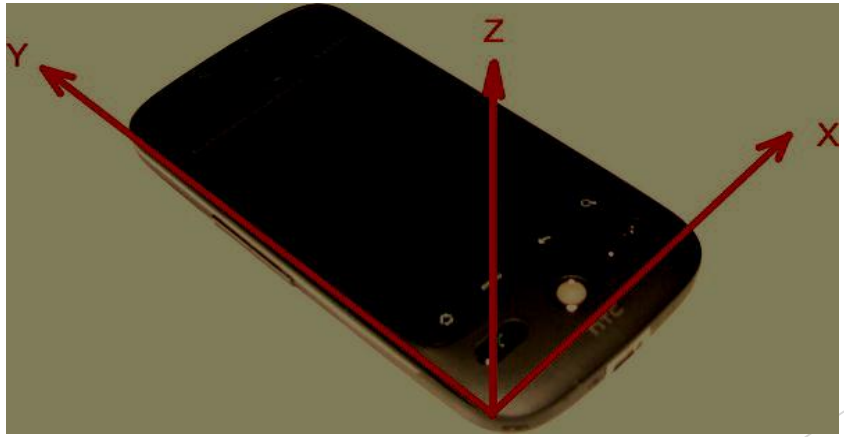
- Provides the total force applied on the device.
- When device is stationary, this gives +1g (gravitational force) reading in one axis and its components on other axes.
- Accelerometer values minus the gravity factor gives the induced acceleration.
- Generally it is used to determine device orientation.
- The measurment unit is SI $\mathbf{m/s^2}$.

Accelerometer Coordinate System



Magnetic field Sensor

- The magnetic field sensor measures the ambient magnetic field in the x, y, and z axes.
- The units of the magnetic field sensor are microteslas (uT). This
- sensor can detect the Earth's magnetic field and therefore tell us where north is. This sensor is also referred to as the compass
- Magnetic field sensor is used in combination with accelerometer.
- The coordinate system



Gyroscope sensor

- Gyroscopes measure the rate of rotation around an axis. When the device is not rotating, the sensor values will be zeroes.
- It gives us 3 value.
  - Pitch (around X)
  - Roll (around y)
  - Azimuth (around z)

Light Sensor

- Located at front of mobile near to front facing camera.
- gives a reading of the light level detected by the light sensor of the device. As the light level changes, the sensor readings change.
- The units of the data are in SI lux units
- Range is from 0 to maximum value for sensor.

Proximity Sensor

- The proximity sensor either measures the distance that some object is from the device (in centimeters) or represents a flag to say whether an object is close or far.
- Some proximity sensors will give a value ranging from 0.0 to the maximum in increments, while others return either 0.0 or he maximum value only.
- Interesting fact about proximity sensors : the proximity sensor is sometimes the same hardware as the light sensor. Android still treats them as logically separate sensors.

Temprature Sensor

- The old temperature sensor provided a temperature reading and also returned just a API level 13single value in values[0]. This sensor usually read an internal temperature, such as at the battery. Till
- From API level 14 it is replaced by TYPE_AMBIENT_TEMPERATURE.
- The new sensor tell us about the room temprature in degree Celsius.

Pressure sensor

- This sensor measures barometric pressure, which could detect altitude can be used for weather predictions.
- The unit of measurement for a pressure sensor is atmospheric pressure in hPa (millibar).

Sensor fusion

- Sensor fusion is a technique to filter the data and obtain the result by combining two or more sensor's data.

- Well known algorithm for sensor fusion
  - Kalman filter algorithm
  - Complimentary filter algorithm

Sensor Example

```
public class SensorActivity extends Activity implements
SensorEventListener {

  private SensorManager mSensorManager;
  private Sensor mLight;

  @Override
  public final void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);

    MSensorManager =(SensorManager)
getSystemService(Context.SENSOR_SERVICE);

    mLight = mSensorManager.getDefaultSensor(Sensor.TYPE_LIGHT);
  }
```

SensorEventListener call back

```
@Override
  public final void onAccuracyChanged(Sensor sensor, int
accuracy) {
    // Do something here if sensor accuracy changes.
  }

  @Override
  public final void onSensorChanged(SensorEvent event) {
    // The light sensor returns a single value.
    // Many sensors return 3 values, one for each axis.
    float lux =;
  } event.values[0]
```

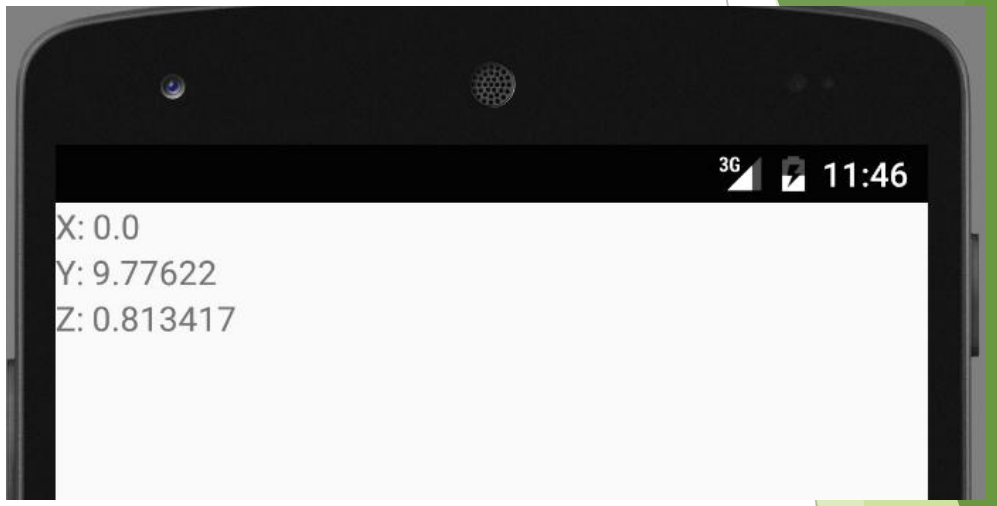
Good Habits for Sensors

```
@Override
  protected void onResume() {
    super.onResume();
    mSensorManager.registerListener(this, mLight,
SensorManager.SENSOR_DELAY_NORMAL);
  }

  @Override
  protected void onPause() {
    super.onPause();
    mSensorManager.unregisterListener(this);
  }
}
```

Code for Acc.

```
private void updateTV(float x, float y, float z) {
    xLabel.setText("X: " + x);
    yLabel.setText("Y: " + y);
    zLabel.setText("z: " + z);

}

private final SensorEventListener mySensorListener = new SensorEventListener() {

    public void onSensorChanged(SensorEvent event) {
        updateTV(event.values[0], event.values[1], event.values[2]);
    }

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

};
```

```java
public class SensorActivity extends Activity implements SensorEventListener {
    private SensorManager sensorManager;

    TextView xCoor; // declare X axis object
    TextView yCoor; // declare Y axis object
    TextView zCoor; // declare Z axis object

    @Override
    public void onCreate(Bundle savedInstanceState){

        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_sensor);

        xCoor=(TextView)findViewById(R.id.xcoor); // create X axis object
        yCoor=(TextView)findViewById(R.id.ycoor); // create Y axis object
        zCoor=(TextView)findViewById(R.id.zcoor); // create Z axis object

        sensorManager=(SensorManager)getSystemService(SENSOR_SERVICE);
        // add listener.
        sensorManager.registerListener(this,
                sensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER),
                SensorManager.SENSOR_DELAY_NORMAL);
    }

    @Override
    protected void onPause(){
        super.onPause();
        sensorManager.unregisterListener(this);
    }

    public void onAccuracyChanged(Sensor sensor,int accuracy){
        // accuracy The new accuracy of this sensor, one of SensorManager.SENSOR_STATUS_*

    }

    public void onSensorChanged(SensorEvent event){

        // check sensor type
        if(event.sensor.getType()==Sensor.TYPE_ACCELEROMETER){
            // assign directions
            float x=event.values[0];
            float y=event.values[1];
            float z=event.values[2];

            xCoor.setText("X: "+x);
            yCoor.setText("Y: "+y);
            zCoor.setText("Z: "+z);
        }
    }
}
```



Sensor Event Listener

```
public final SensorEventListener mSensorListener = new SensorEventListener() {

    public void onSensorChanged(SensorEvent se) {
        if (se.sensor.getType() == Sensor.TYPE_ACCELEROMETER) {
            float x = se.values[0];
            float y = se.values[1];
            float z = se.values[2];
        }
    }

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

Thanks for your attention…