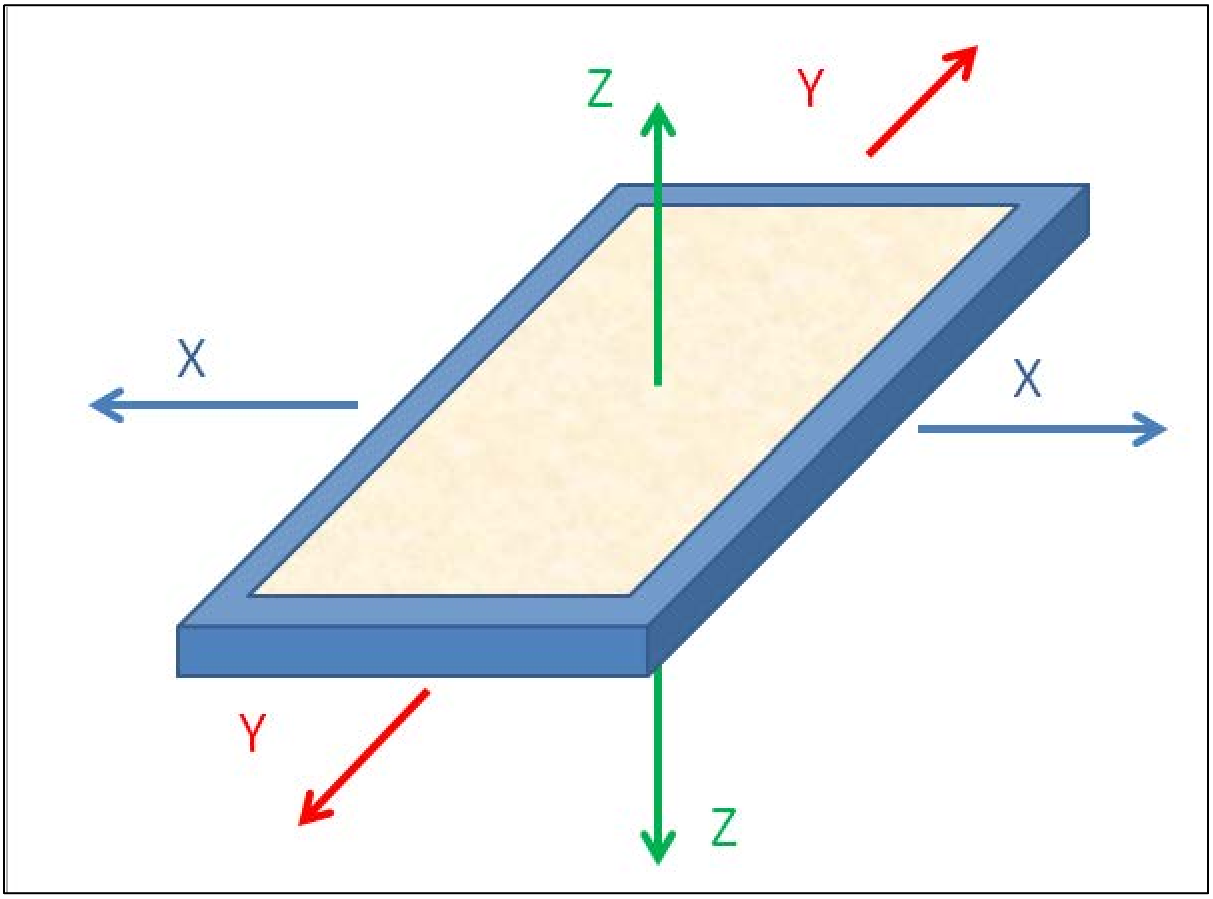
Mobile App Design Project
Pedometer Using Accelerometer Sensor

Description:

The Android Phone has a three direction accelerometer sensor that reads the change in speed along three axis (x, y, and z). Programs using the accelerometer read this information to give the phones orientation in space or the phones change in speed and direction. In addition, the gravity type for the phone can be changed based on gravitational pull on different heavenly bodies (Earth, moon, Mars, Jupiter . . .). Satellite companies have experimented with using Android device based CPU's and programming to control Satellite guidance systems. (http://www.tgdaily.com/mobility-brief/69359-android-in-space-satellite-controlled-using-nexus-one)

Image of Acceleration Values:



This Application will use the Accelerometer to display the X, Y, and Z values and use the Y value to calculate and count the steps a user takes while carrying the phone in their pocket. The App will also have a SeekBar object to set the sensitivity of the change in values on the Y Axis during a step or shake.

Phase 1: Create the App Project

Process:

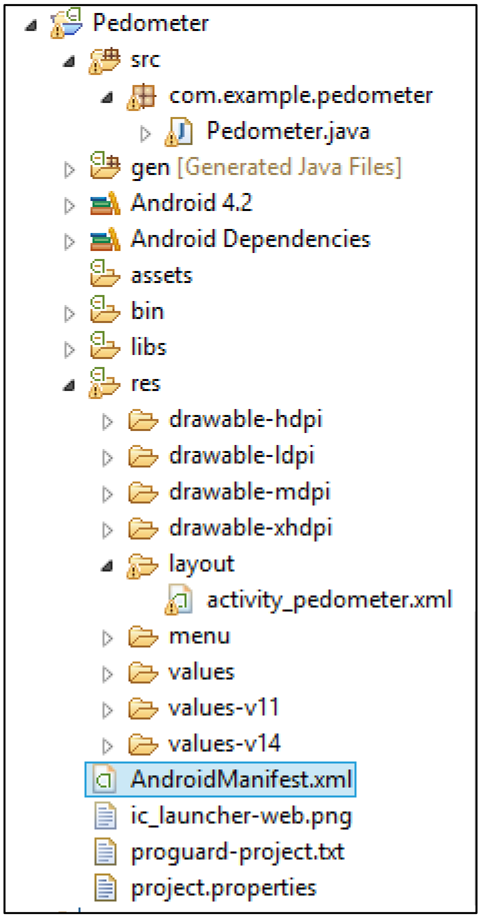
1. Start Eclipse and select New Project -> “Android Application Project”
2. Fill out the fields with the following:
    a. Application Name: Pedometer
    b. Project name: Pedometer
    c. Package name: com.example.pedometer
3. Click Next
4. Click Next at the Configure Project Screen
5. Click Next at the Configure Launcher Screen
6. Click Next at the Create Activity Screen
7. Fill out the following fields in the New Blank Activity Screen
    a. Activity Name: Pedometer
    b. LayoutName: activity_pedometer
    c. Navigation Type: None
8. Click “Finish”

Phase 2: Android Manifest XML and the XML user interface design

Because we will be shaking and moving the phone – we need to ‘freeze’ the phone’s screen orientation in the Portrait mode. This is accomplished by adding a line of code to the Android Manifest. Then we will build the User Interface on the activity_pedometer.xml file.

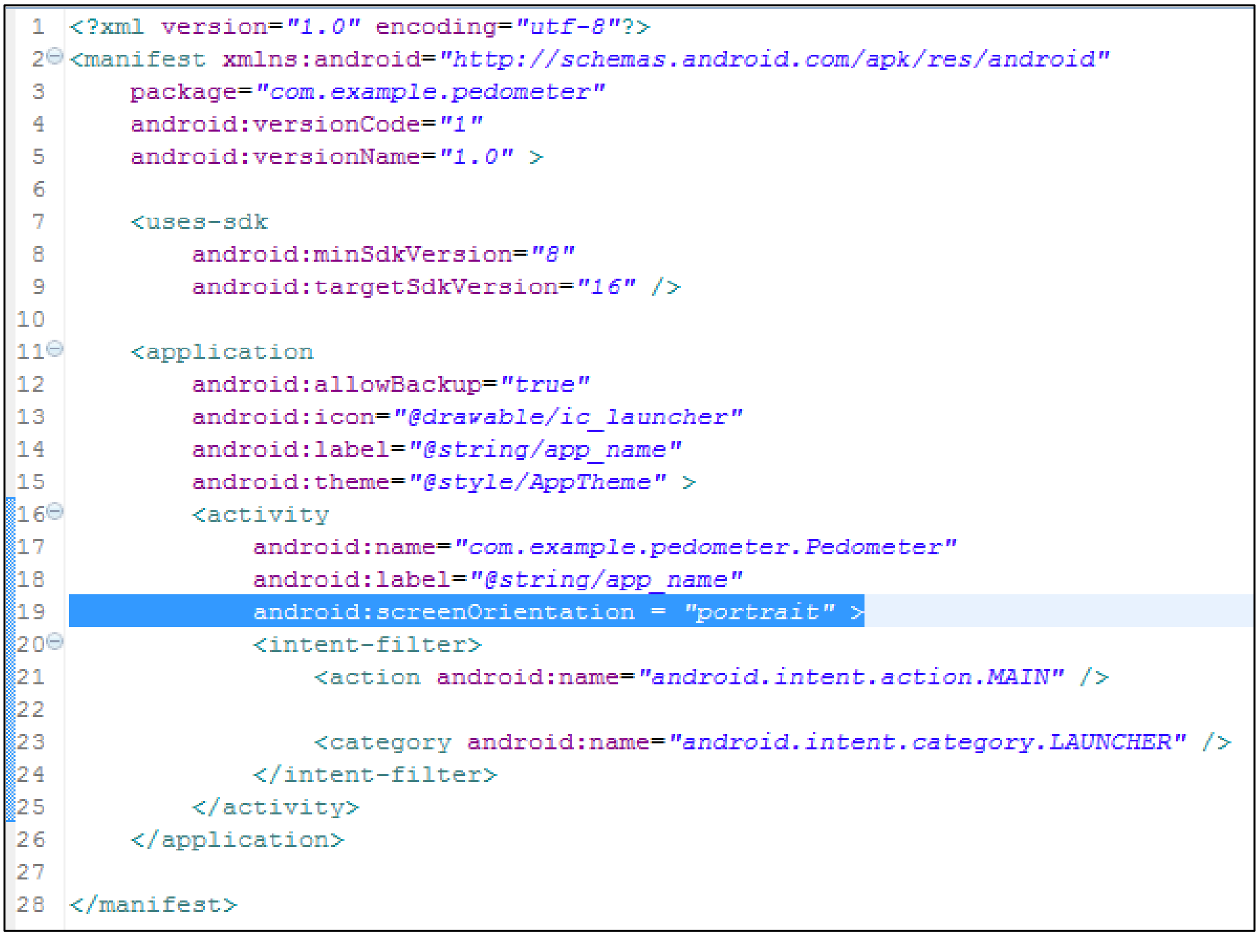
Process:

1. Open the Android Manifest.xml file



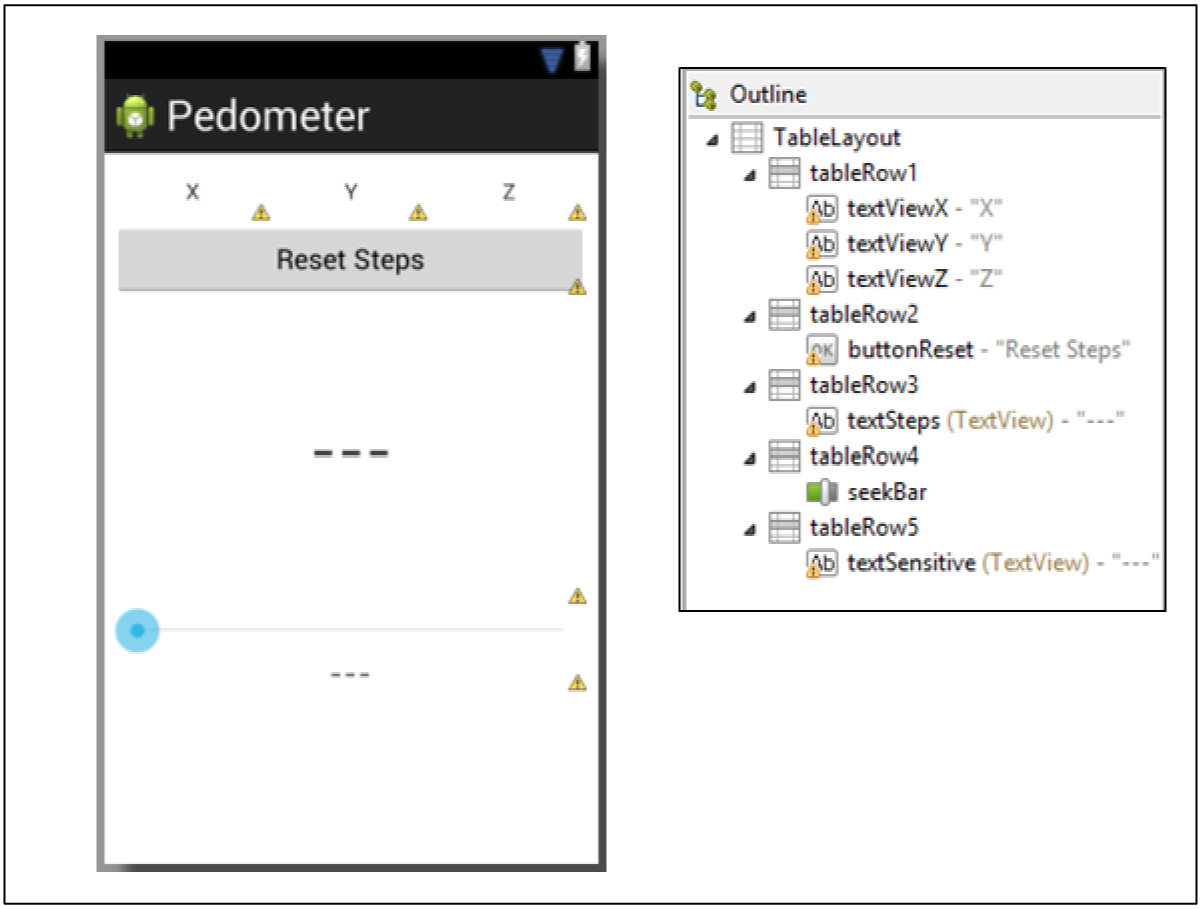
```
Pedometer
    src
        com.example.pedometer
            Pedometer.java
    gen [Generated Java Files]
    Android 4.2
    Android Dependencies
    assets
    bin
    libs
    res
        drawable-hdpi
        drawable-ldpi
        drawable-mdpi
        drawable-xhdpi
        layout
            activity_pedometer.xml
        menu
        values
        values-v11
        values-v14
    AndroidManifest.xml
    ic_launcher-web.png
    proguard-project.txt
    project.properties
```

2. Modify the existing code and insert the screen orientation setting at Line 19 inside the <activity tag with the code: `android:screenOrientation = "portrait"`



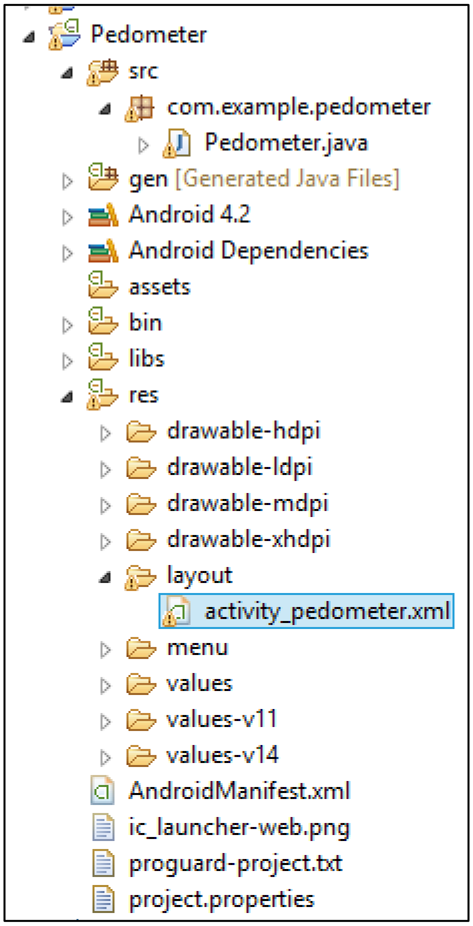
```
1  <?xml version="1.0" encoding="utf-8"?>
2  <manifest xmlns:android="http://schemas.android.com/apk/res/android"
3      package="com.example.pedometer"
4      android:versionCode="1"
5      android:versionName="1.0" >
6  
7      <uses-sdk
8          android:minSdkVersion="8"
9          android:targetSdkVersion="16" />
10 
11     <application
12         android:allowBackup="true"
13         android:icon="@drawable/ic_launcher"
14         android:label="@string/app_name"
15         android:theme="@style/AppTheme" >
16         <activity
17             android:name="com.example.pedometer.Pedometer"
18             android:label="@string/app_name"
19             android:screenOrientation = "portrait" >
20             <intent-filter>
21                 <action android:name="android.intent.action.MAIN" />
22 
23                 <category android:name="android.intent.category.LAUNCHER" />
24             </intent-filter>
25         </activity>
26     </application>
27 
28 </manifest>
```

3. We will now design the User interface. The objects and locations within the user interface are shown below:

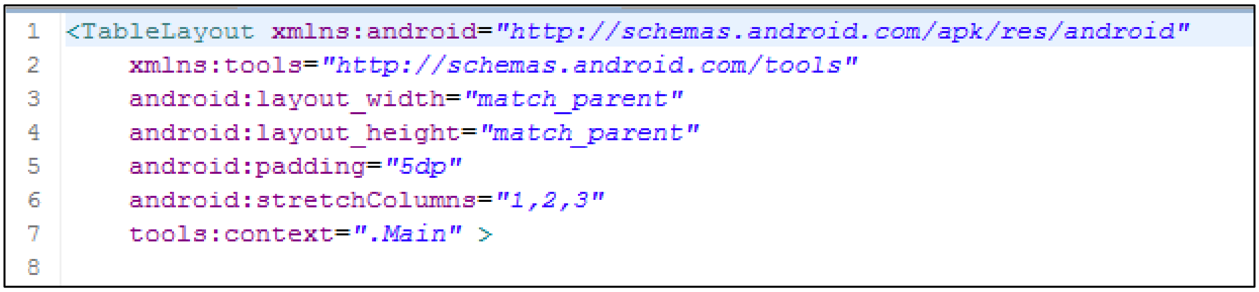


```
Outline
  TableLayout
    tableRow1
      textViewX - "X"
      textViewY - "Y"
      textViewZ - "Z"
    tableRow2
      buttonReset - "Reset Steps"
    tableRow3
      textSteps (TextView) - "---"
    tableRow4
      seekBar
    tableRow5
      textSensitive (TextView) - "---"
```

4. Open the activity_pedometer.xml file

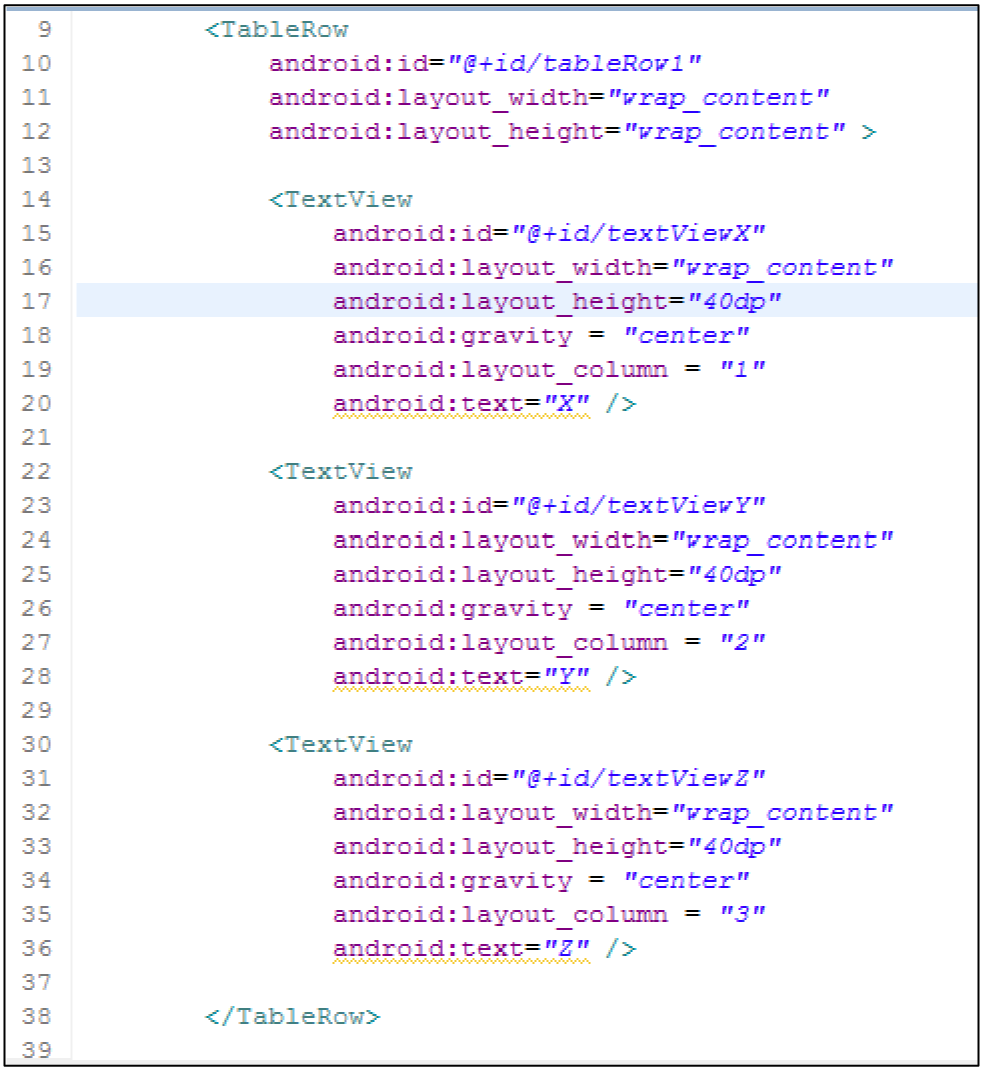


5. Start the XML file with a Table Layout Tag:



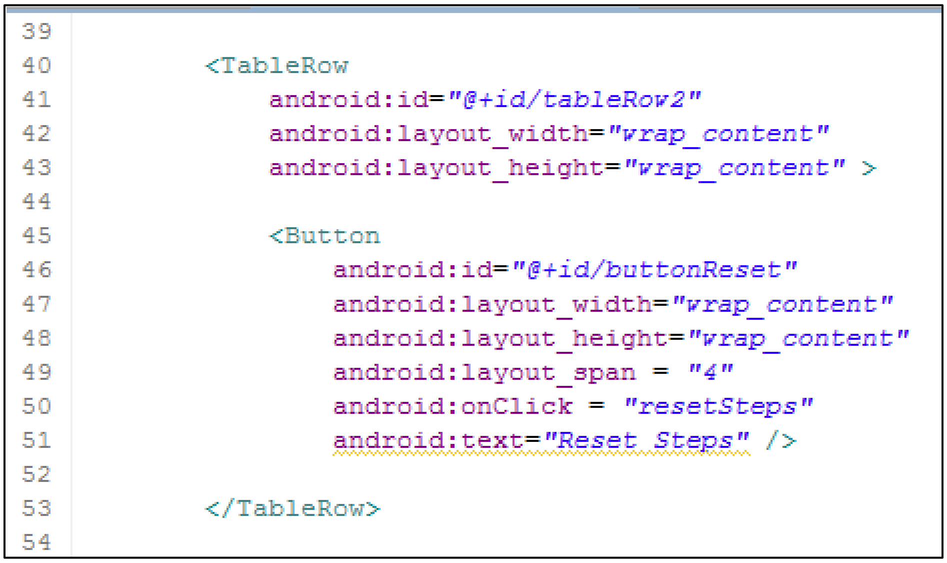
```
<TableLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:padding="5dp"
    android:stretchColumns="1,2,3"
    tools:context=".Main" >

```

6. In Table Row 1 we will have three TextView objects to display the X, Y, and Z values from the Accelerometer Sensor. Write the code for TableRow 1



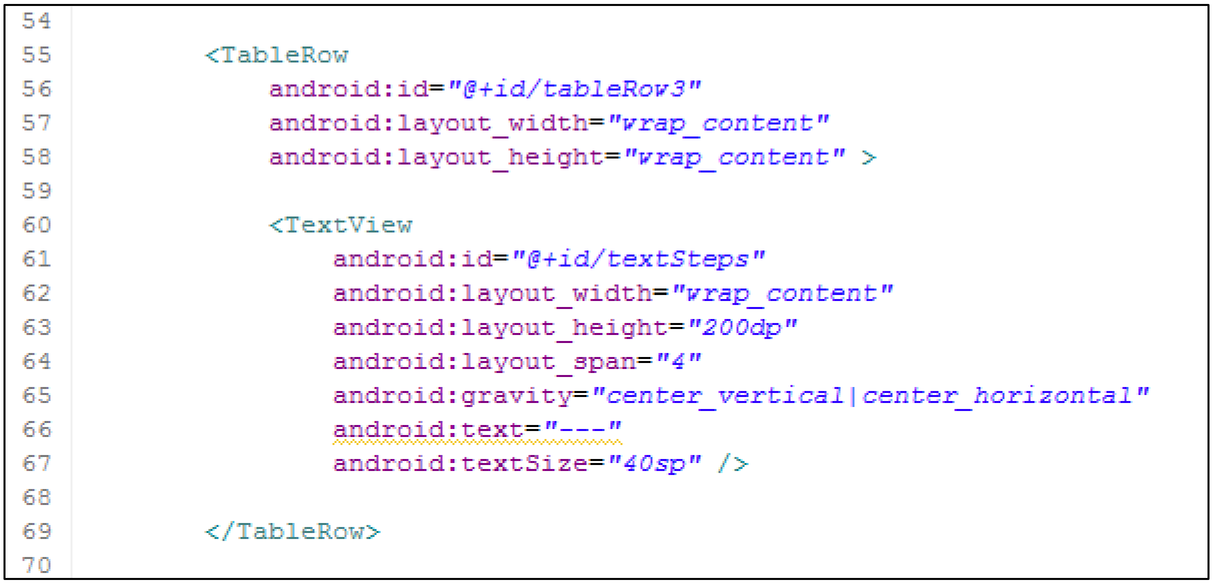
```
        <TableRow
            android:id="@+id/tableRow1"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content" >

            <TextView
                android:id="@+id/textViewX"
                android:layout_width="wrap_content"
                android:layout_height="40dp"
                android:gravity = "center"
                android:layout_column = "1"
                android:text="X" />

            <TextView
                android:id="@+id/textViewY"
                android:layout_width="wrap_content"
                android:layout_height="40dp"
                android:gravity = "center"
                android:layout_column = "2"
                android:text="Y" />

            <TextView
                android:id="@+id/textViewZ"
                android:layout_width="wrap_content"
                android:layout_height="40dp"
                android:gravity = "center"
                android:layout_column = "3"
                android:text="Z" />

        </TableRow>

```

7. In Table Row 2 we will place the Reset Button that will call the method ‘resetSteps’ to reset the step counter.



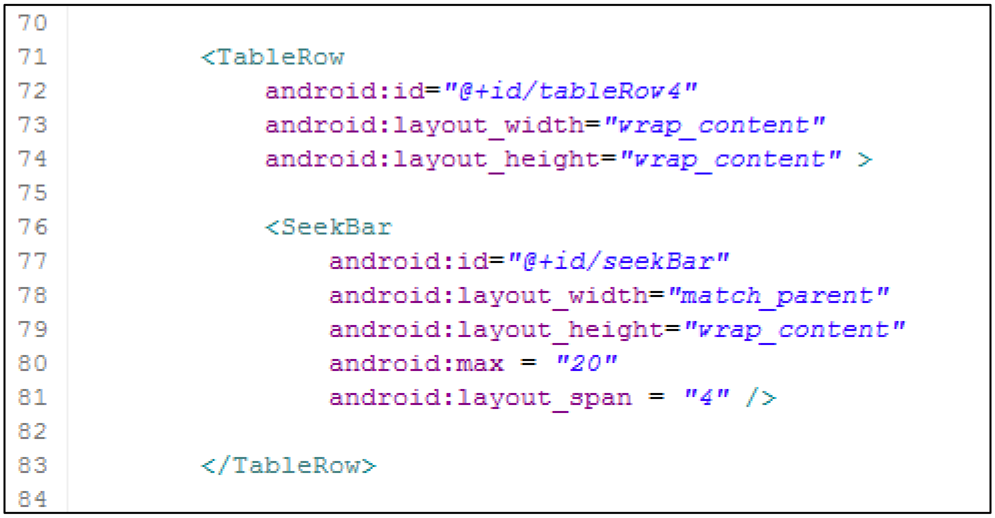
```

        <TableRow
            android:id="@+id/tableRow2"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content" >

            <Button
                android:id="@+id/buttonReset"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_span = "4"
                android:onClick = "resetSteps"
                android:text="Reset Steps" />

        </TableRow>

```

8. In Table Row Three we will have a Large TextView object that will display the Steps Centered horizontally, vertically, and in a large font size.



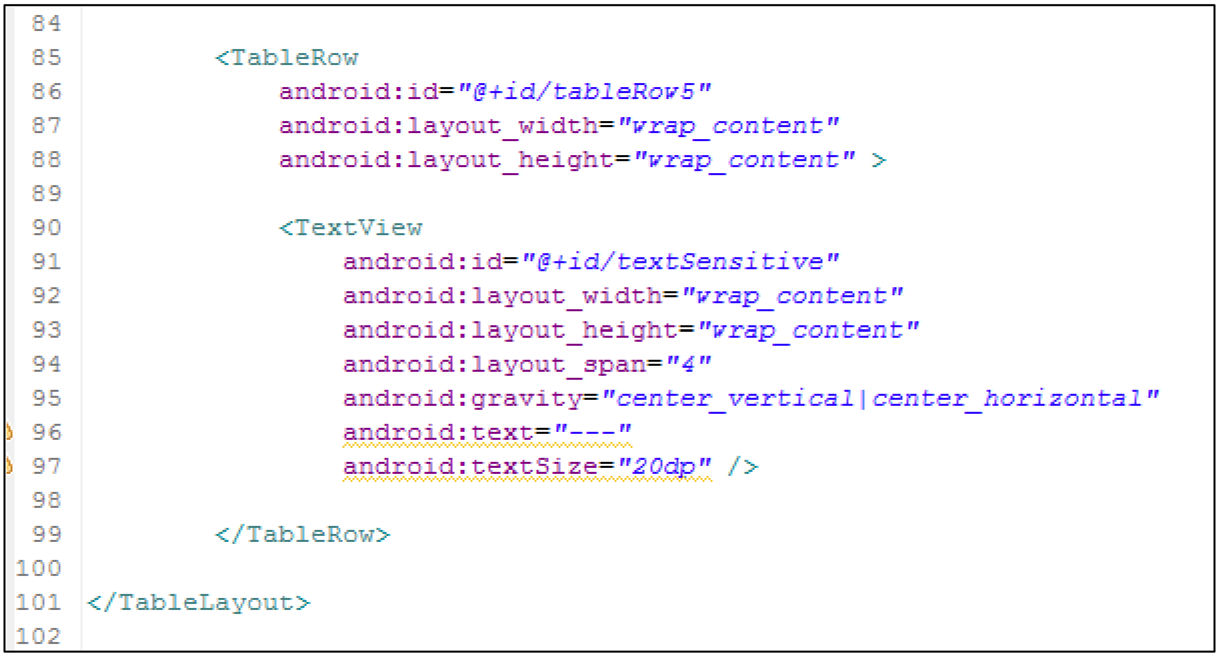
```

        <TableRow
            android:id="@+id/tableRow3"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content" >

            <TextView
                android:id="@+id/textSteps"
                android:layout_width="wrap_content"
                android:layout_height="200dp"
                android:layout_span="4"
                android:gravity="center_vertical|center_horizontal"
                android:text="---"
                android:textSize="40sp" />

        </TableRow>

```

9. In Table Row Four we will have a SeekBar object that we will use to set the threshold value



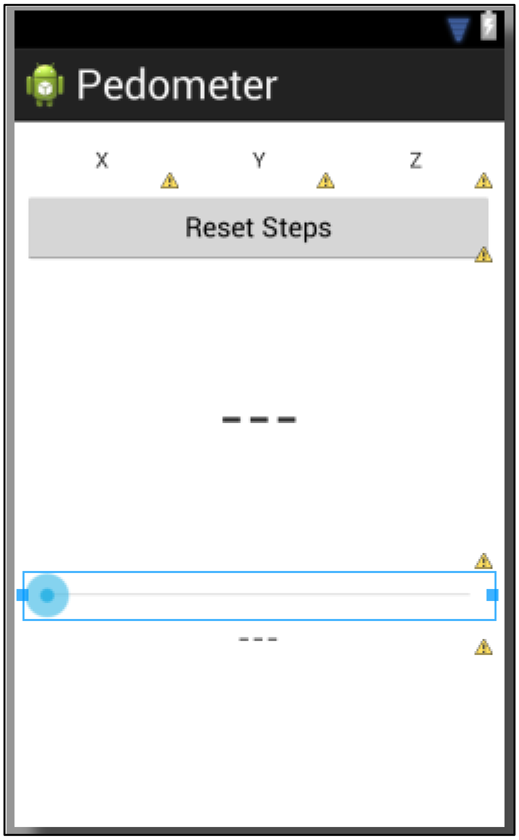
```

        <TableRow
            android:id="@+id/tableRow4"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content" >

            <SeekBar
                android:id="@+id/seekBar"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:max = "20"
                android:layout_span = "4" />

        </TableRow>

```

10. In Table Row 5 we will have a TextView object to display the threshold. We will also close out the TableLayout object and finish the XML



```

        <TableRow
            android:id="@+id/tableRow5"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content" >

            <TextView
                android:id="@+id/textSensitive"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_span="4"
                android:gravity="center_vertical|center_horizontal"
                android:text="---"
                android:textSize="20dp" />

        </TableRow>

</TableLayout>

```

11. The User Interface should look like this:

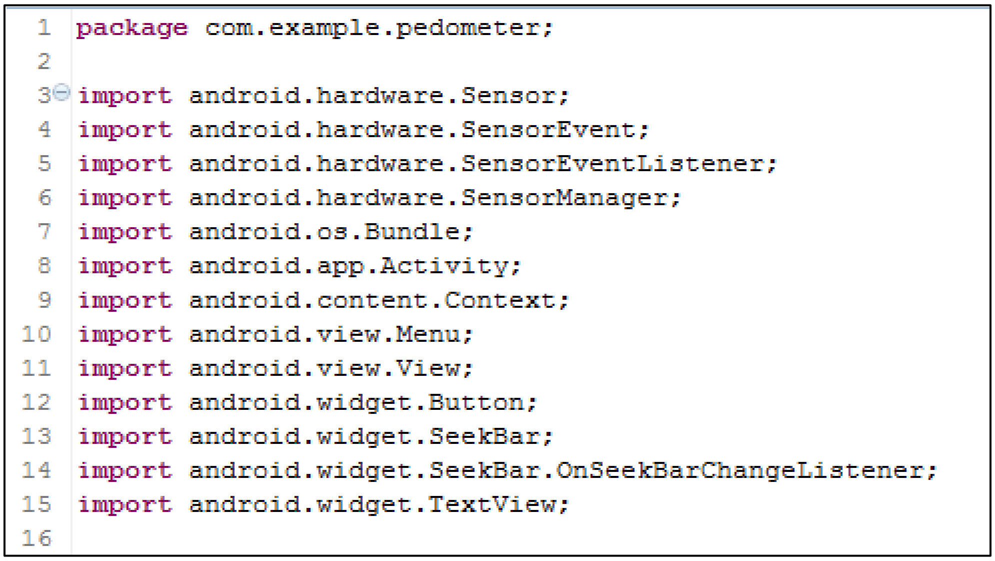


Phase 3: Writing the Code for the Pedometer Class

The Pedometer Class will create the model to display the accelerometer X, Y, and Z data, have a Sensor Listener to read the Accelerometer, and have the algorithm to detect steps using the threshold value.

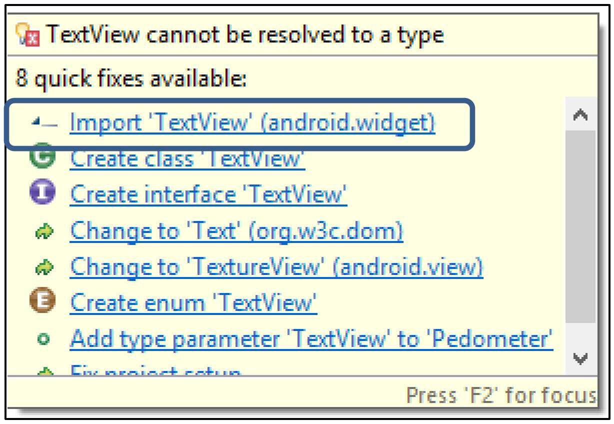
Process:

1. Write the import statements for the Android Classes used within the Pedometer Class



```
package com.example.pedometer;

import android.hardware.Sensor;
import android.hardware.SensorEvent;
import android.hardware.SensorEventListener;
import android.hardware.SensorManager;
import android.os.Bundle;
import android.app.Activity;
import android.content.Context;
import android.view.Menu;
import android.view.View;
import android.widget.Button;
import android.widget.SeekBar;
import android.widget.SeekBar.OnSeekBarChangeListener;
import android.widget.TextView;

```

Note! You can skip this step and 'include' the imports as you write the code. When an error occurs, hover over the error and select 'Import ' (Usually the first suggestion from Eclipse)



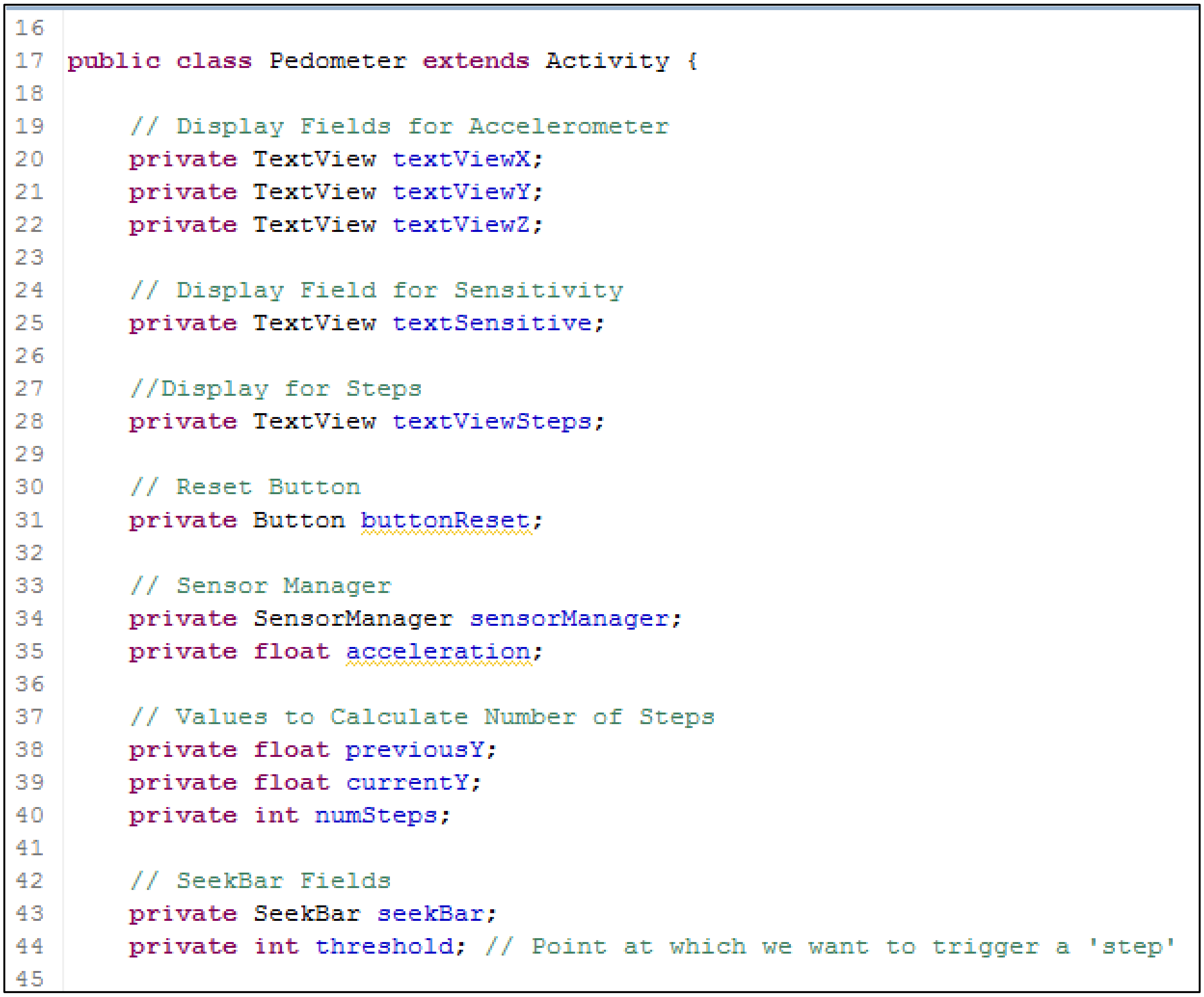
TextView cannot be resolved to a type

8 quick fixes available:

Import 'TextView' (android.widget)
Create class 'TextView'
Create interface 'TextView'
Change to 'Text' (org.w3c.dom)
Change to 'TextureView' (android.view)
Create enum 'TextView'
Add type parameter 'TextView' to 'Pedometer'
Fix project setup

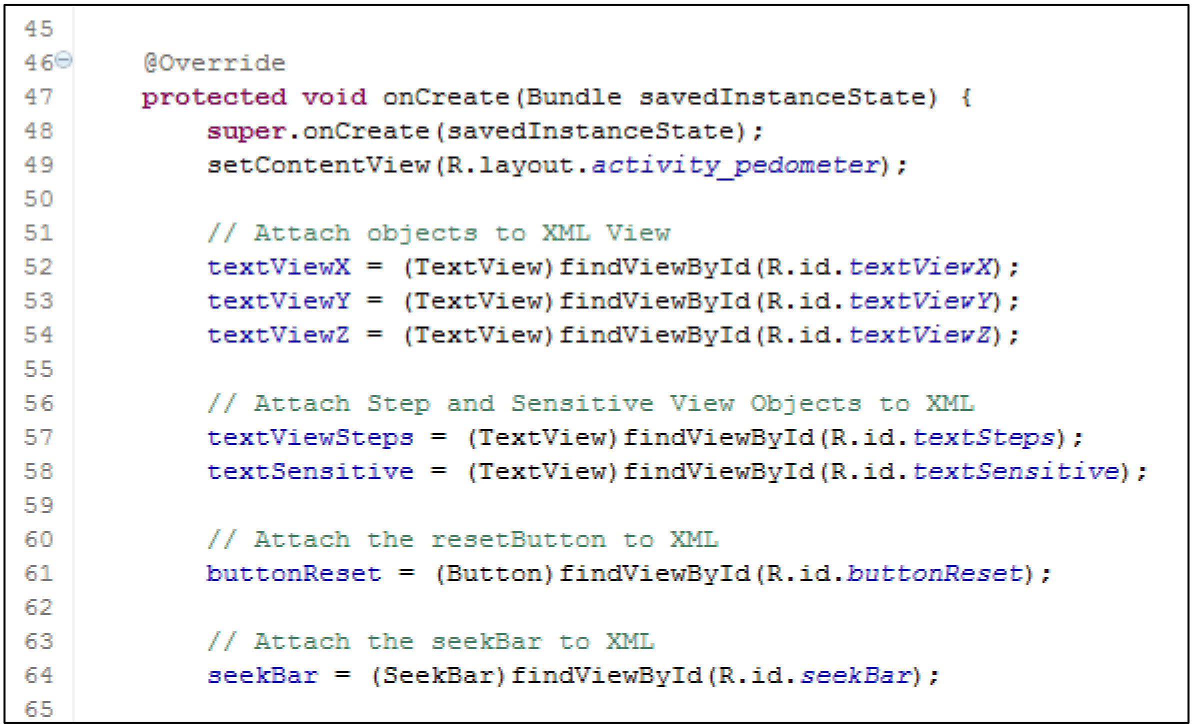
Press 'F2' for focus

2. Define the Fields for the Pedometer Class



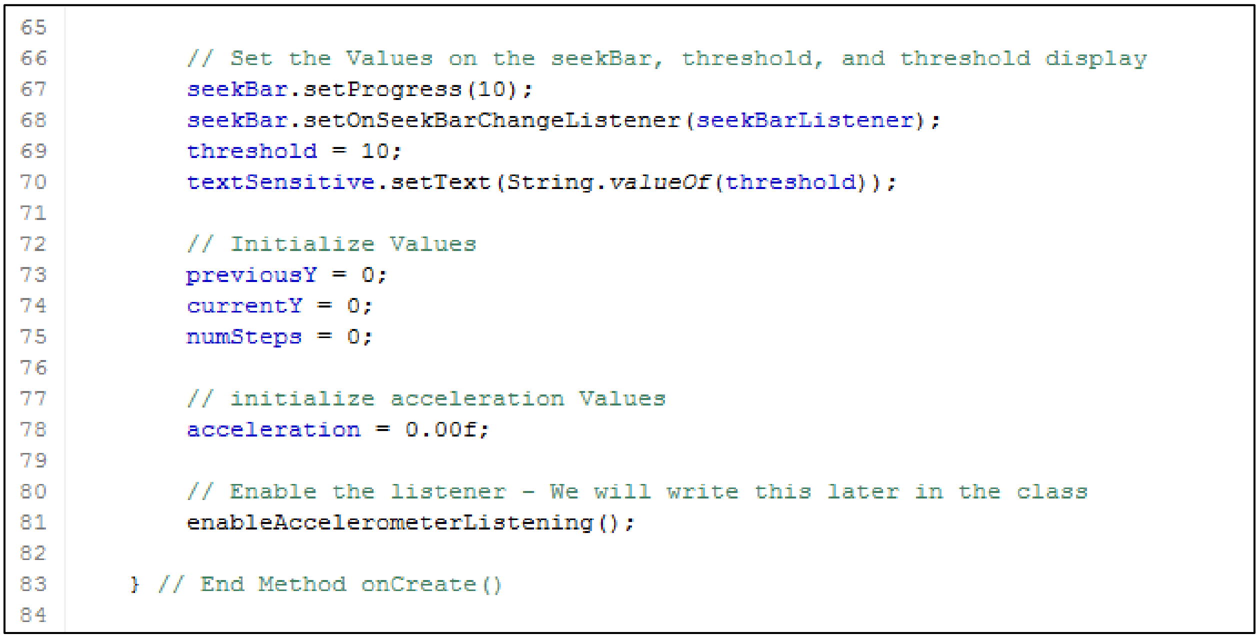
```

public class Pedometer extends Activity {

    // Display Fields for Accelerometer
    private TextView textViewX;
    private TextView textViewY;
    private TextView textViewZ;

    // Display Field for Sensitivity
    private TextView textSensitive;

    //Display for Steps
    private TextView textViewSteps;

    // Reset Button
    private Button buttonReset;

    // Sensor Manager
    private SensorManager sensorManager;
    private float acceleration;

    // Values to Calculate Number of Steps
    private float previousY;
    private float currentY;
    private int numSteps;

    // SeekBar Fields
    private SeekBar seekBar;
    private int threshold; // Point at which we want to trigger a 'step'

```

3. Start the onCreate() method by attaching the fields to the XML User Interface objects



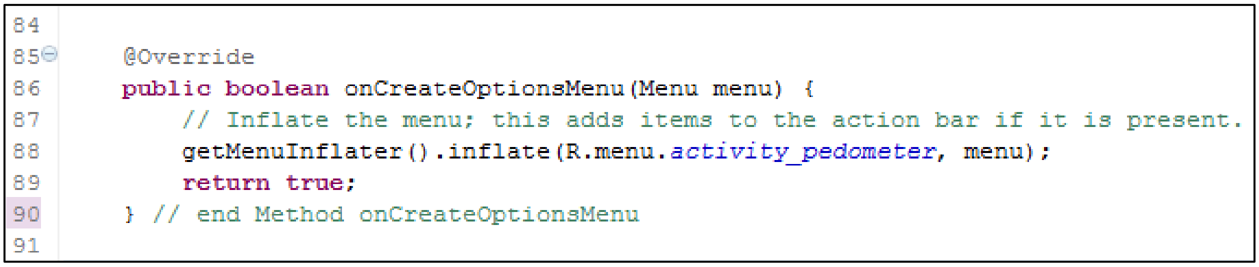
```java

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_pedometer);

        // Attach objects to XML View
        textViewX = (TextView)findViewById(R.id.textViewX);
        textViewY = (TextView)findViewById(R.id.textViewY);
        textViewZ = (TextView)findViewById(R.id.textViewZ);

        // Attach Step and Sensitive View Objects to XML
        textViewSteps = (TextView)findViewById(R.id.textSteps);
        textSensitive = (TextView)findViewById(R.id.textSensitive);

        // Attach the resetButton to XML
        buttonReset = (Button)findViewById(R.id.buttonReset);

        // Attach the seekBar to XML
        seekBar = (SeekBar)findViewById(R.id.seekBar);

```

4. Finish writing the onCreate() method by initializing the values for the seekBar, threshold, and the values used to calculate the steps



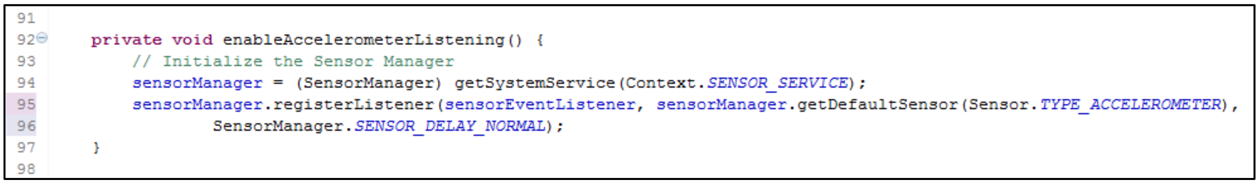
```java

        // Set the Values on the seekBar, threshold, and threshold display
        seekBar.setProgress(10);
        seekBar.setOnSeekBarChangeListener(seekBarListener);
        threshold = 10;
        textSensitive.setText(String.valueOf(threshold));

        // Initialize Values
        previousY = 0;
        currentY = 0;
        numSteps = 0;

        // initialize acceleration Values
        acceleration = 0.00f;

        // Enable the listener - We will write this later in the class
        enableAccelerometerListening();

    } // End Method onCreate()

```

5. The onCreateOptionsMenu is included by default. Leave this method as written by Eclipse.



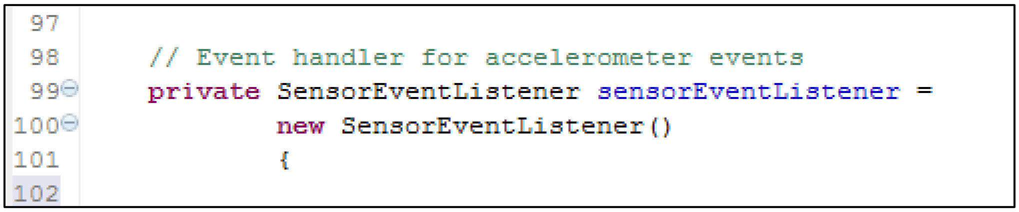
```

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.activity_pedometer, menu);
        return true;
    } // end Method onCreateOptionsMenu

```

6. Write the enableAccelerometerListening() function to enable Accelerometer and register the listener.



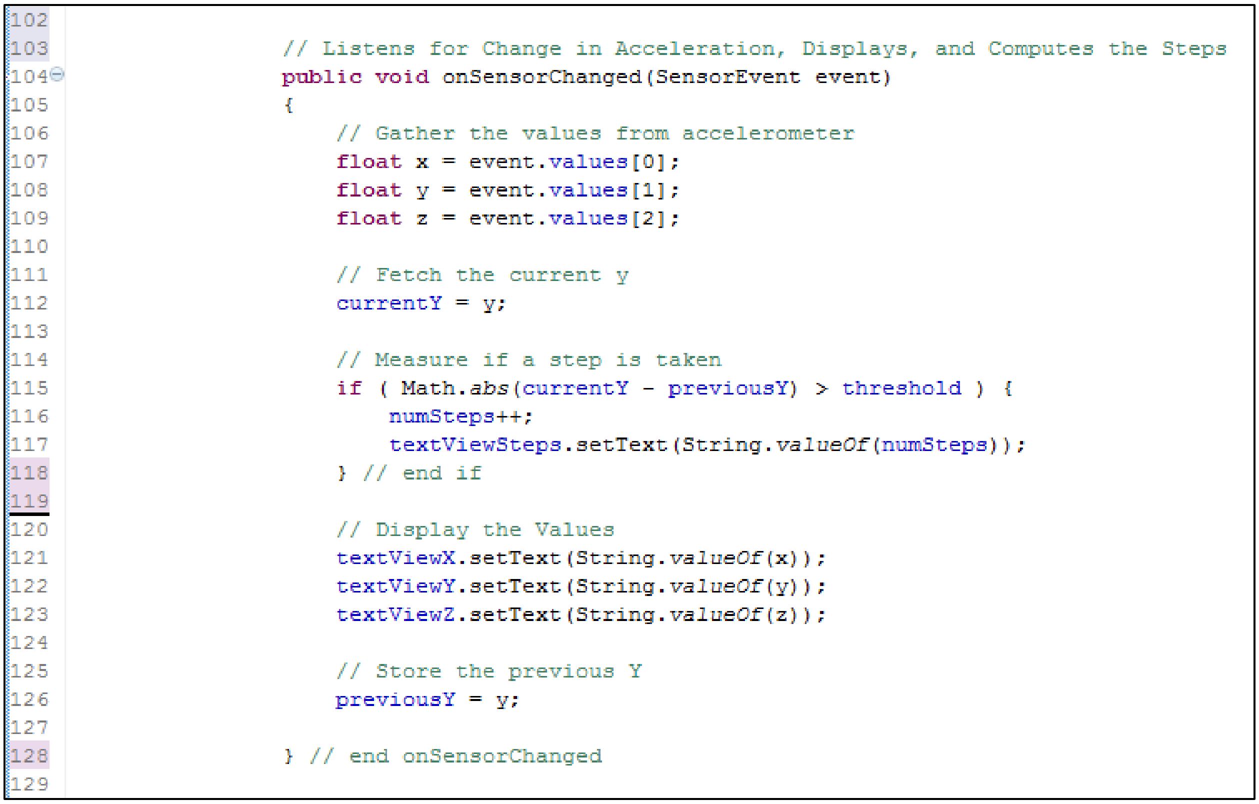
```

    private void enableAccelerometerListening() {
        // Initialize the Sensor Manager
        sensorManager = (SensorManager) getSystemService(Context.SENSOR_SERVICE);
        sensorManager.registerListener(sensorEventListener, sensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER),
                SensorManager.SENSOR_DELAY_NORMAL);
    }

```

7. Begin the Event handler for the Accelerometer. This is an inner class



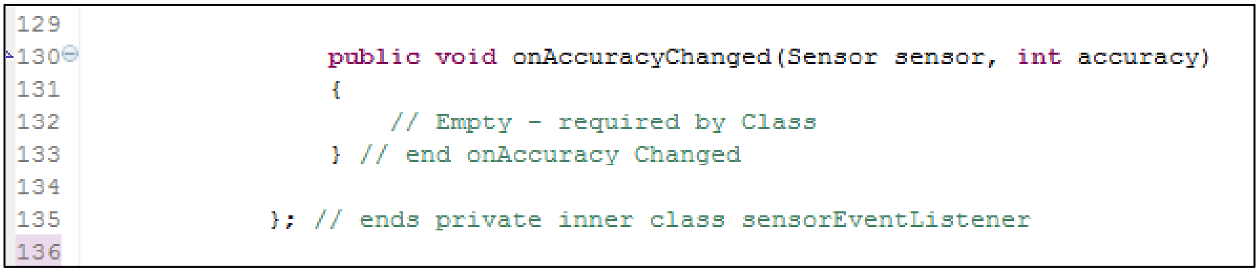
```

    // Event handler for accelerometer events
    private SensorEventListener sensorEventListener =
            new SensorEventListener()
            {

```

8. Write the onSensorChanged method. This holds the majority of the logic and flow for the App. The values are gathered from the event object and displayed. A conditional statement compared the previous and current acceleration to the threshold to count the steps.



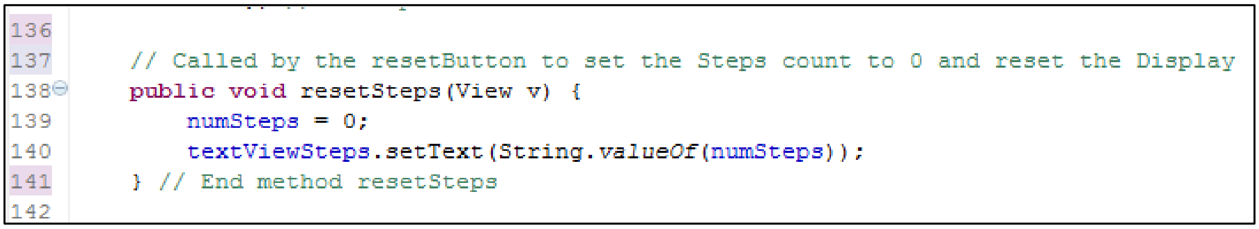
```java
        // Listens for Change in Acceleration, Displays, and Computes the Steps
        public void onSensorChanged(SensorEvent event)
        {
            // Gather the values from accelerometer
            float x = event.values[0];
            float y = event.values[1];
            float z = event.values[2];

            // Fetch the current y
            currentY = y;

            // Measure if a step is taken
            if ( Math.abs(currentY - previousY) > threshold ) {
                numSteps++;
                textViewSteps.setText(String.valueOf(numSteps));
            } // end if

            // Display the Values
            textViewX.setText(String.valueOf(x));
            textViewY.setText(String.valueOf(y));
            textViewZ.setText(String.valueOf(z));

            // Store the previous Y
            previousY = y;

        } // end onSensorChanged
```

9. Finish the SensorEventListener with a required method and close with a bracket and semicolon



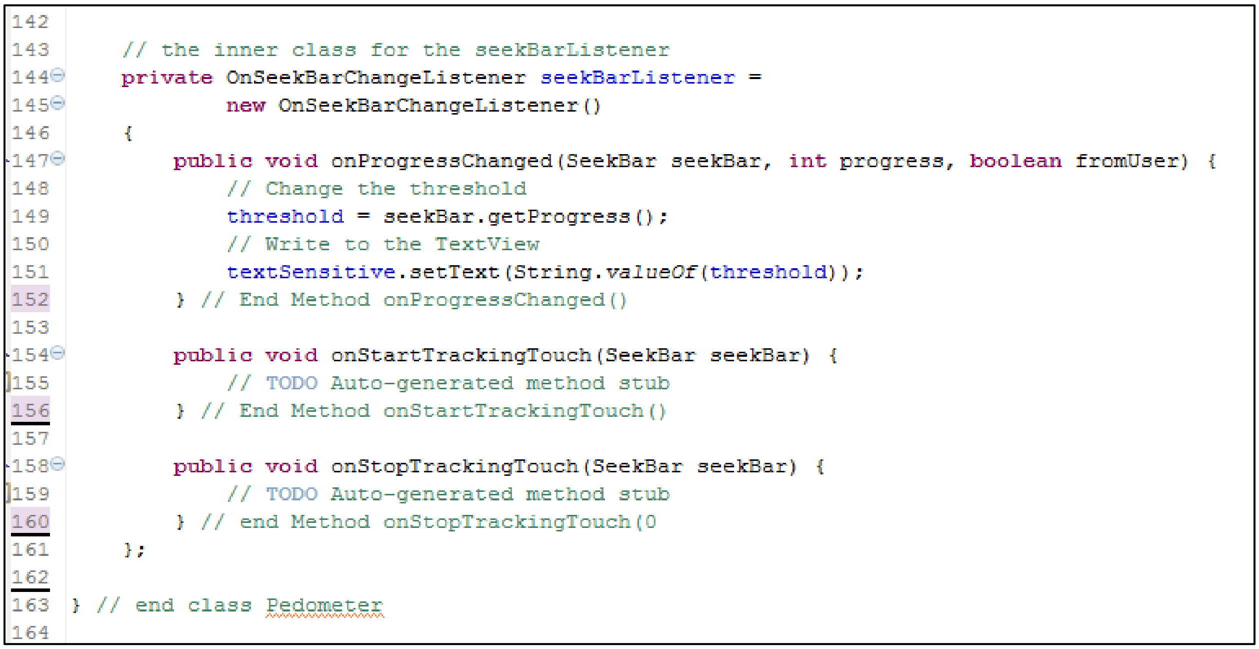
```java
        public void onAccuracyChanged(Sensor sensor, int accuracy)
        {
            // Empty - required by Class
        } // end onAccuracy Changed

    }; // ends private inner class sensorEventListener
```

10. Write the resetSteps() method. The resetButton calls this method when clicked to reset the step counter and the display.



```java
    // Called by the resetButton to set the Steps count to 0 and reset the Display
    public void resetSteps(View v) {
        numSteps = 0;
        textViewSteps.setText(String.valueOf(numSteps));
    } // End method resetSteps
```

11. Finish the Pedometer class with a private class OnSeekBarChangeListener. This Listener reacted to changes to the SeekBar, adjusting the threshold value used by the SensorEventListener. The Pedometer class is finished with a bracket at line 163.



```java

    // the inner class for the seekBarListener
    private OnSeekBarChangeListener seekBarListener =
            new OnSeekBarChangeListener()
    {
        public void onProgressChanged(SeekBar seekBar, int progress, boolean fromUser) {
            // Change the threshold
            threshold = seekBar.getProgress();
            // Write to the TextView
            textSensitive.setText(String.valueOf(threshold));
        } // End Method onProgressChanged()

        public void onStartTrackingTouch(SeekBar seekBar) {
            // TODO Auto-generated method stub
        } // End Method onStartTrackingTouch()

        public void onStopTrackingTouch(SeekBar seekBar) {
            // TODO Auto-generated method stub
        } // end Method onStopTrackingTouch(0
    };

} // end class Pedometer

```

12. Save and Test your App! Experiment to see how accurate the App counts steps.

13. Improvements you can make!
    a. Add sounds to the App to give an aural cue every time a step is recorded.
    b. Change the User interface to add more colors and interest.
    c. Create a better algorithm to count steps more accurately.
    d. Combine with the GPS Sensor to record distance traveled.
    e. Add a function to Count Calories based on Number of steps and weight of User.