CCE60220

Perangkat Bergerak (TKOM)

Fakultas Ilmu Komputer Universitas Brawijaya

MATAKULIAH	: **Perangkat Bergerak (TKOM)**
KODE/ STATUS	: CCE60220
SKS	: 2
Dosen	: Dahnial Syauqy, S.T, M.T
Email	: dahnial87@ub.ac.id
Ruang	:

Agenda Perkuliahan

1. Intro dan overview perkuliahan
2. Sejarah dan perkembangan teknologi perangkat bergerak
3. Komponen perangkat keras dan perangkat lunak
4. Pengenalan dan instalasi android studio serta aplikasi sederhana
5. Intent dan passing data pada Android Studio
6. **Android Studio: Sensor reading**
7. Android Studio: Storage & shared preference
8. **========================UTS**
9. Pengenalan dan aplikasi sederhana dengan MIT AppInventor
10. Appinventor: variable, looping, conditional, tinyDB, file
11. appInventor: sensor reading & persiapan project
12. Appinventor: Akuisisi gambar dan suara
13. Appinventor: komunikasi bluetooth
14. Appinventor: basic animation
15. Presentasi kelompok
16. **========================UAS**

Using SENSORS

Few Sensor types supported by the Android platform.

Sensor	Description	Common Uses
TYPE_ACCELEROMETER	Measures the acceleration force in m/s^2 that is applied to a device on all three physical axes (x, y, and z), including the force of gravity.	Motion detection (shake, tilt, etc.).
TYPE_GRAVITY	Measures the force of gravity in m/s^2 that is applied to a device on all three physical axes (x, y, z).	Motion detection (shake, tilt, etc.).
TYPE_GYROSCOPE	Measures a device's rate of rotation in rad/s around each of the three physical axes (x, y, and z).	Rotation detection (spin, turn, etc.).
TYPE_LIGHT	Measures the ambient light level (illumination) in lx.	Controlling screen brightness.
TYPE_MAGNETIC_FIELD	Measures the ambient geomagnetic field for all three physical axes (x, y, z) in μT.	Creating a compass.
TYPE_ORIENTATION	Measures degrees of rotation that a device makes around all three physical axes (x, y, z). As of API level 3 you can obtain the inclination matrix and rotation matrix for a device by using the gravity sensor and the geomagnetic field sensor in conjunction with the getRotationMatrix() method.	Determining device position.
TYPE_PRESSURE	Measures the ambient air pressure in hPa or mbar.	Monitoring air pressure changes.
TYPE_PROXIMITY	Measures the proximity of an object in cm relative to the view screen of a device. This sensor is typically used to determine whether a handset is being held up to a person's ear.	Phone position during a call.
TYPE_RELATIVE_HUMIDITY	Measures the relative ambient humidity in percent (%).	Monitoring dewpoint, absolute, and relative humidity.

Sensor Framework

- Most Android-powered devices have built-in sensors that measure **motion**, **orientation**, and various **environmental** conditions.
- The Android sensor framework lets you access many types of sensors.

SensorManager

This class provides various methods for accessing and listing sensors, registering and unregistering sensor event listeners, and acquiring orientation information.

Sensor

This class provides various methods that let you determine a sensor's capabilities.

SensorEvent

The system uses this class to create a sensor event object, which provides information about a sensor event. A sensor event object includes the following information: the raw sensor data, the type of sensor that generated the event, the accuracy of the data, and the timestamp for the event.

SensorEventListener

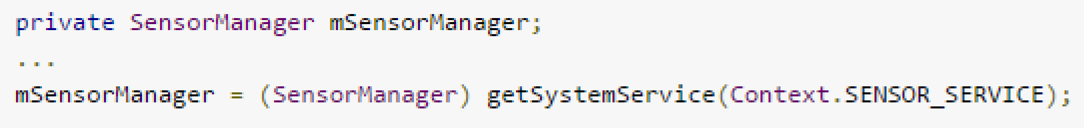
You can use this interface to create callback method that receive notifications (sensor events) when sensor values change or when sensor accuracy changes.

Main task:

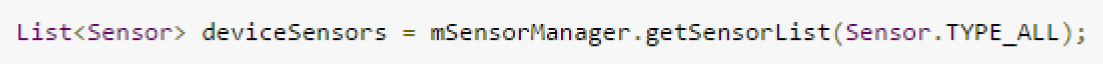
1. **Determine** which sensors are **available** on a device.
2. Register and unregister **sensor event listeners that monitor** sensor changes.

1. Identifying available sensor

Identifying Sensors and Sensor Capabilities



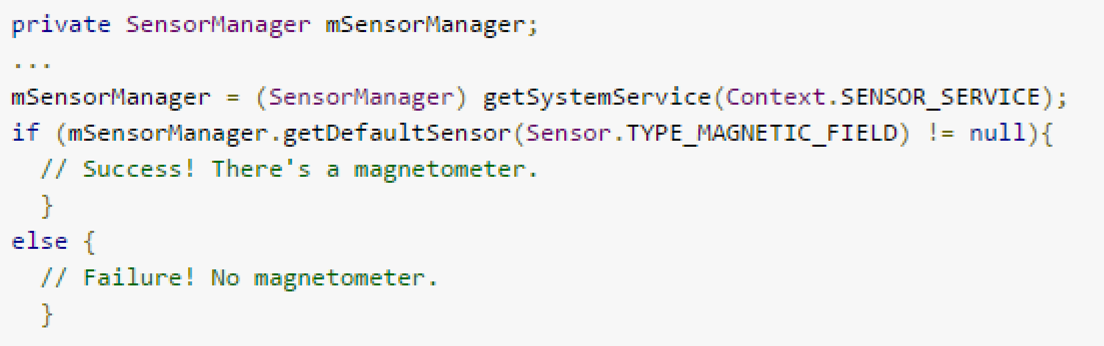
```
private SensorManager mSensorManager;
...
mSensorManager = (SensorManager) getSystemService(Context.SENSOR_SERVICE);
```



```
List<Sensor> deviceSensors = mSensorManager.getSensorList(Sensor.TYPE_ALL);
```

If you want to list all of the sensors of a given type, you could use another constant instead of TYPE_ALL such as TYPE_GYROSCOPE, TYPE_LINEAR_ACCELERATION, or TYPE_GRAVITY.

You can also determine whether a specific type of sensor exists on a device by using the getDefaultSensor() method and passing in the type constant for a specific sensor.



```
private SensorManager mSensorManager;
...
mSensorManager = (SensorManager) getSystemService(Context.SENSOR_SERVICE);
if (mSensorManager.getDefaultSensor(Sensor.TYPE_MAGNETIC_FIELD) != null){
  // Success! There's a magnetometer.
  }
else {
  // Failure! No magnetometer.
  }
```

2. Monitor Sensor Events

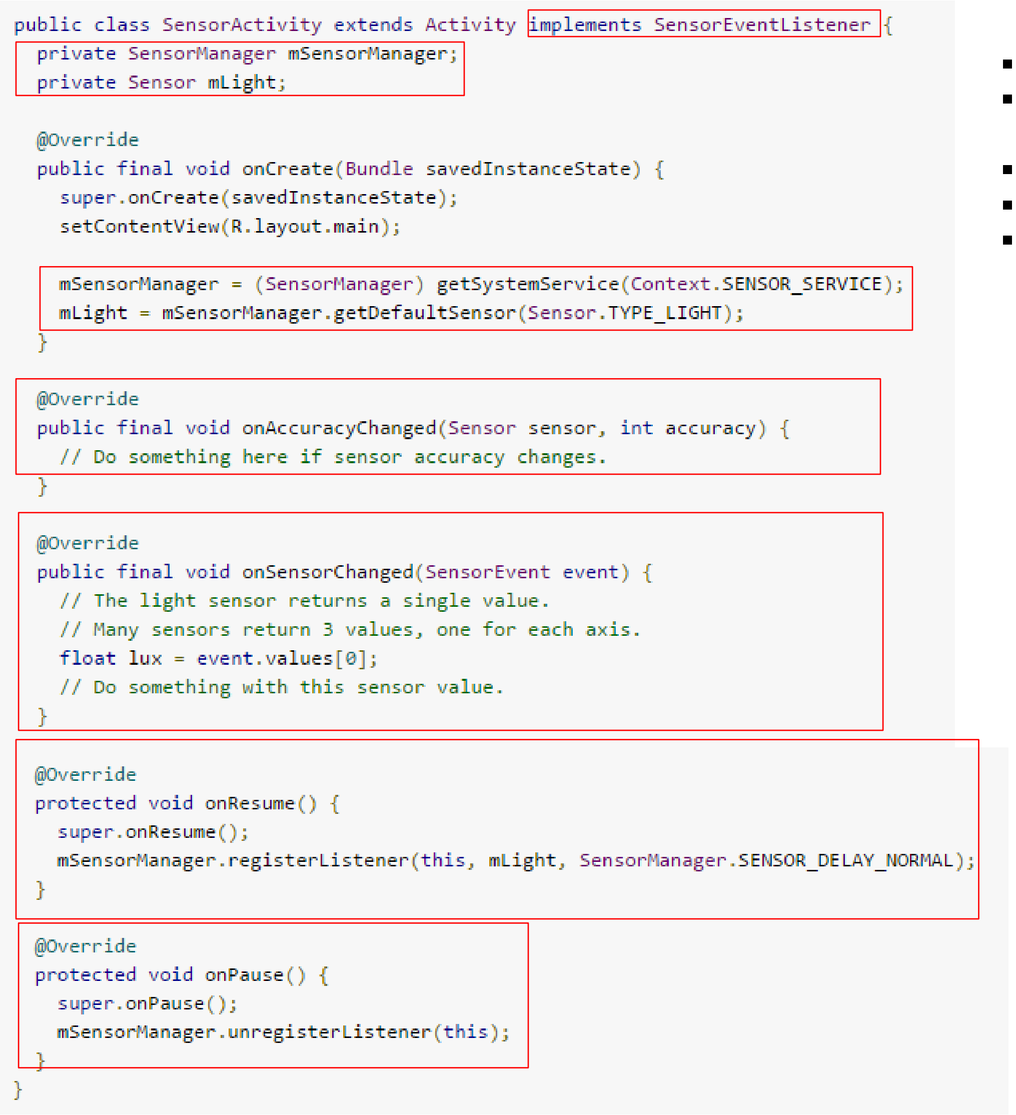
A sensor reports a new value.

In this case the system invokes the **onSensorChanged()** method, providing you with a SensorEvent object. A SensorEvent object contains information about the new sensor data, including: the accuracy of the data, the sensor that generated the data, the timestamp at which the data was generated, and the new data that the sensor recorded.

A sensor's accuracy changes.

In this case the system invokes the **onAccuracyChanged()** method, providing you with a reference to the Sensor object that changed and the new accuracy of the sensor. Accuracy is represented by one of four status constants: SENSOR_STATUS_ACCURACY_LOW, SENSOR_STATUS_ACCURACY_MEDIUM, SENSOR_STATUS_ACCURACY_HIGH, or SENSOR_STATUS_UNRELIABLE.

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);
  }

  @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];
    // Do something with this sensor value.
  }

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

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

- SENSOR_DELAY_NORMAL 200,000 microseconds
- SENSOR_DELAY_GAME (20,000 microsecond delay)
- SENSOR_DELAY_UI (60,000 microsecond delay), or
- SENSOR_DELAY_FASTEST (0 microsecond delay)
- as an absolute value (in microseconds)

It's also important to note that this example uses the onResume() and onPause() callback methods to register and unregister the sensor event listener. As a best practice **you should always disable sensors you don't need, especially when your activity is paused**. Failing to do so can drain the battery

Sensor types

Position sensors

These sensors measure the physical position of a device. This category includes orientation sensors and magnetometers.

Environmental sensors

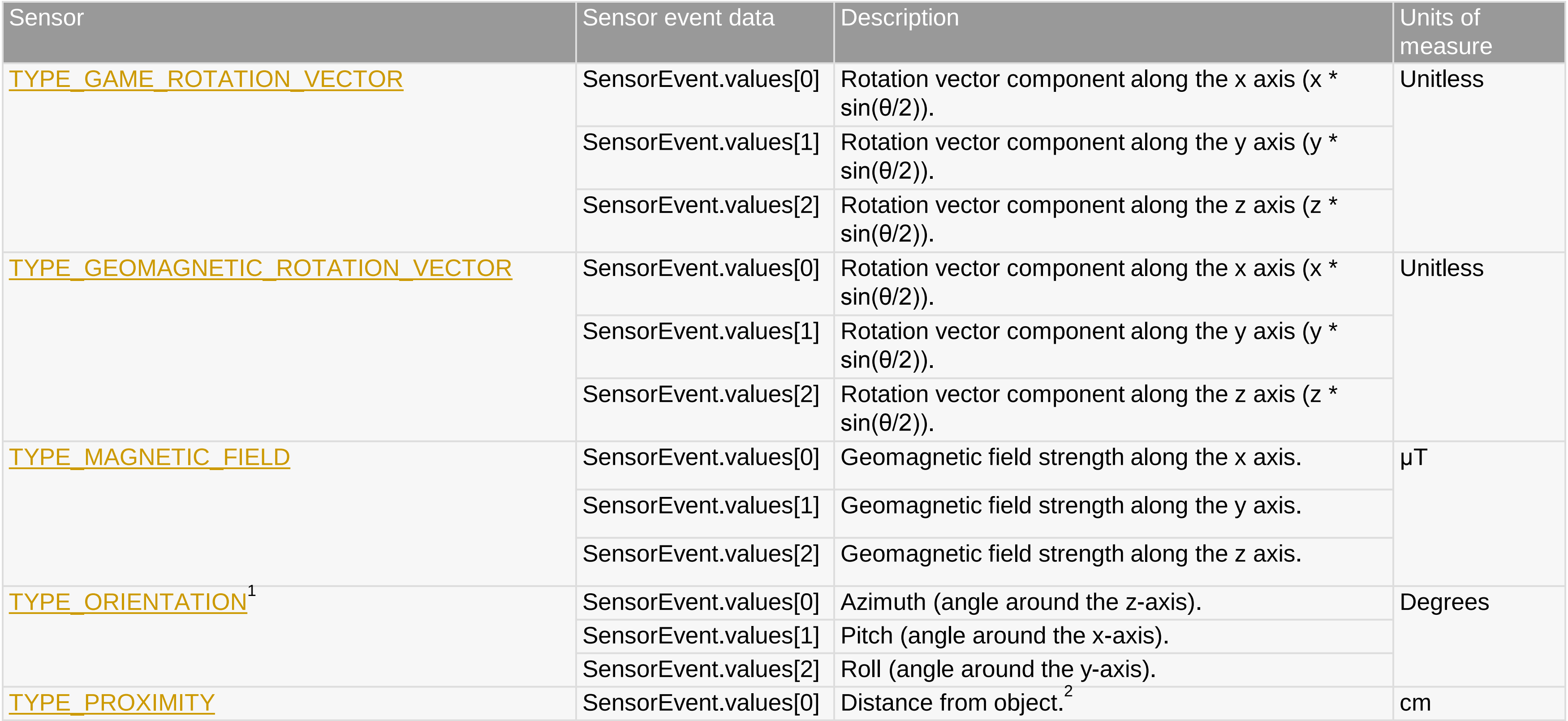
These sensors measure various environmental parameters, such as ambient air temperature and pressure, illumination, and humidity. This category includes barometers, photometers, and thermometers.

Motion sensors

These sensors measure acceleration forces and rotational forces along three axes. This category includes accelerometers, gravity sensors, gyroscopes, and rotational vector sensors.

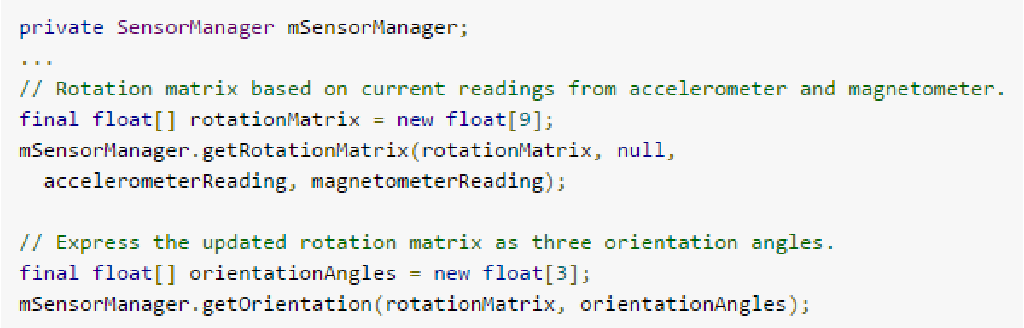
Position Sensors

Position sensors are useful for determining a device's physical position in the world's frame of reference.

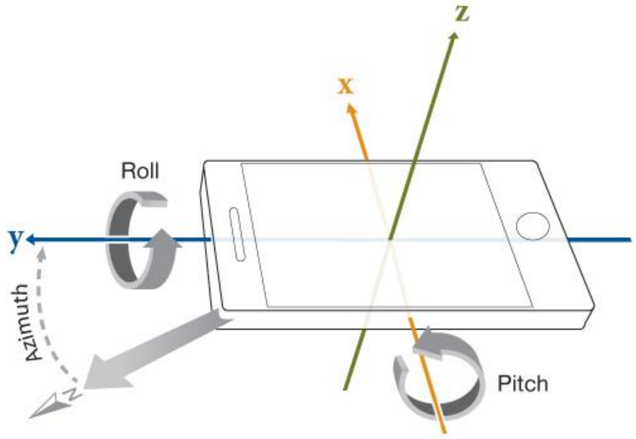


Sensor	Sensor event data	Description	Units of measure
TYPE_GAME_ROTATION_VECTOR	SensorEvent.values[0]	Rotation vector component along the x axis (x * sin(θ/2)).	Unitless
	SensorEvent.values[1]	Rotation vector component along the y axis (y * sin(θ/2)).	
	SensorEvent.values[2]	Rotation vector component along the z axis (z * sin(θ/2)).	
TYPE_GEOMAGNETIC_ROTATION_VECTOR	SensorEvent.values[0]	Rotation vector component along the x axis (x * sin(θ/2)).	Unitless
	SensorEvent.values[1]	Rotation vector component along the y axis (y * sin(θ/2)).	
	SensorEvent.values[2]	Rotation vector component along the z axis (z * sin(θ/2)).	
TYPE_MAGNETIC_FIELD	SensorEvent.values[0]	Geomagnetic field strength along the x axis.	μT
	SensorEvent.values[1]	Geomagnetic field strength along the y axis.	
	SensorEvent.values[2]	Geomagnetic field strength along the z axis.	
TYPE_ORIENTATION[1]	SensorEvent.values[0]	Azimuth (angle around the z-axis).	Degrees
	SensorEvent.values[1]	Pitch (angle around the x-axis).	
	SensorEvent.values[2]	Roll (angle around the y-axis).	
TYPE_PROXIMITY	SensorEvent.values[0]	Distance from object.[2]	cm

Computing the Device's Orientation



```
private SensorManager mSensorManager;
...
// Rotation matrix based on current readings from accelerometer and magnetometer.
final float[] rotationMatrix = new float[9];
mSensorManager.getRotationMatrix(rotationMatrix, null,
  accelerometerReading, magnetometerReading);

// Express the updated rotation matrix as three orientation angles.
final float[] orientationAngles = new float[3];
mSensorManager.getOrientation(rotationMatrix, orientationAngles);
```



Azimuth (degrees of rotation about the -z axis). This is the angle between the device's current compass direction and magnetic north. If the top edge of the device faces magnetic north, the azimuth is 0 degrees; if the top edge faces south, the azimuth is 180 degrees. Similarly, if the top edge faces east, the azimuth is 90 degrees, and if the top edge faces west, the azimuth is 270 degrees.

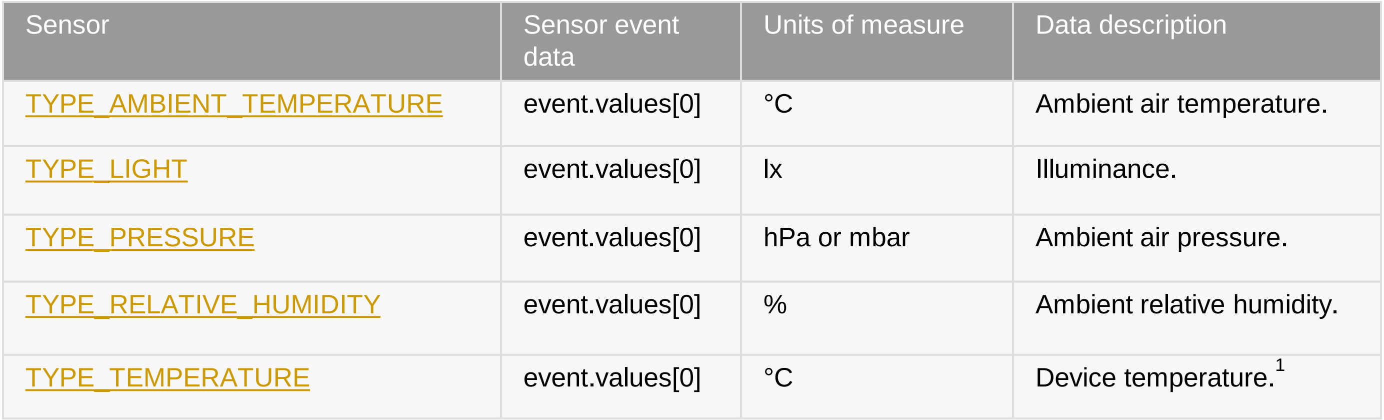
Pitch (degrees of rotation about the x axis). This is the angle between a plane parallel to the device's screen and a plane parallel to the ground. If you hold the device parallel to the ground with the bottom edge closest to you and tilt the top edge of the device toward the ground, the pitch angle becomes positive. Tilting in the opposite direction— moving the top edge of the device away from the ground— causes the pitch angle to become negative. The range of values is -180 degrees to 180 degrees.

Roll (degrees of rotation about the y axis). This is the angle between a plane perpendicular to the device's screen and a plane perpendicular to the ground. If you hold the device parallel to the ground with the bottom edge closest to you and tilt the left edge of the device toward the ground, the roll angle becomes positive. Tilting in the opposite direction—moving the right edge of the device toward the ground— causes the roll angle to become negative. The range of values is -90 degrees to 90 degrees.

Environment Sensors

The Android platform provides four sensors that let you monitor various environmental properties.

You can use these sensors to monitor relative ambient humidity, luminance, ambient pressure, and ambient temperature near an Android-powered device.

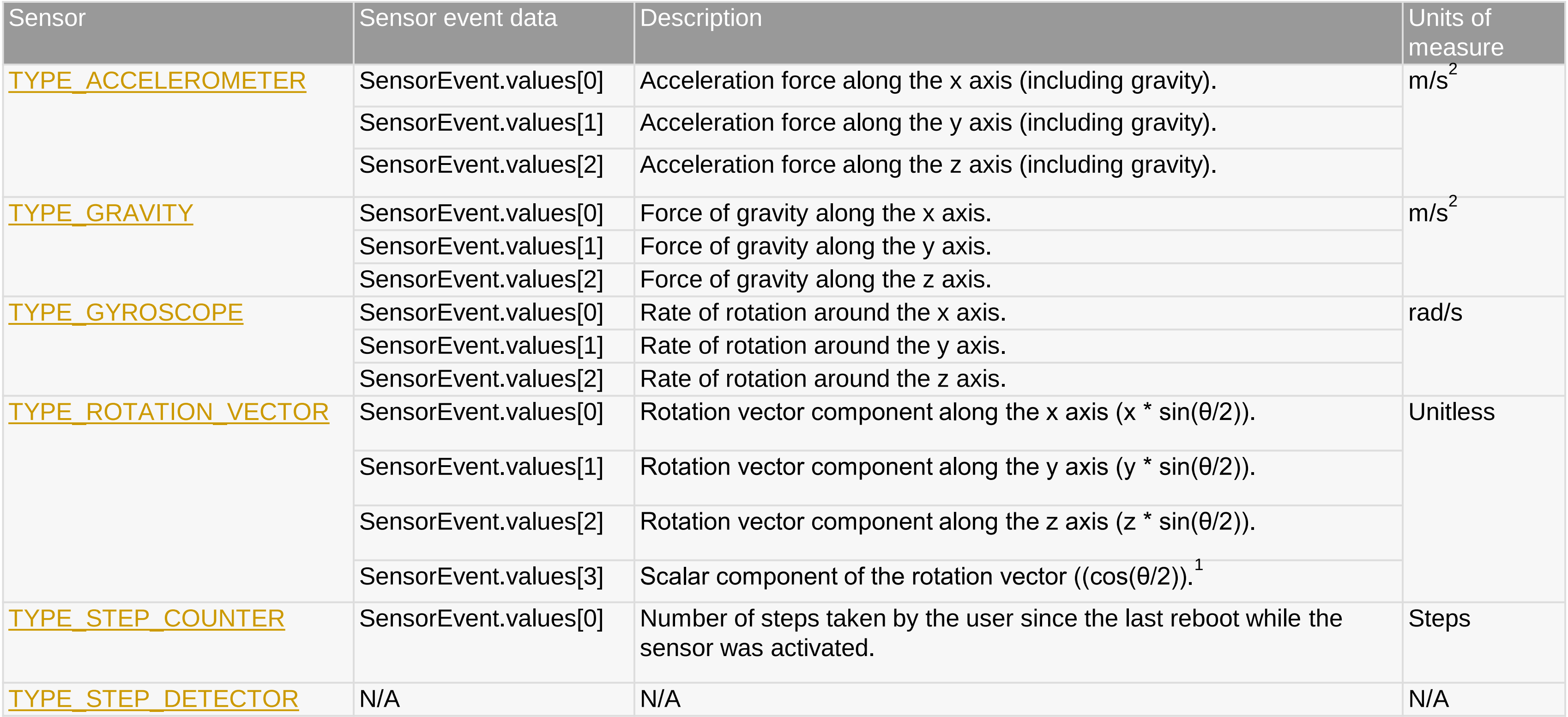


Sensor	Sensor event data	Units of measure	Data description
TYPE_AMBIENT_TEMPERATURE	event.values[0]	°C	Ambient air temperature.
TYPE_LIGHT	event.values[0]	lx	Illuminance.
TYPE_PRESSURE	event.values[0]	hPa or mbar	Ambient air pressure.
TYPE_RELATIVE_HUMIDITY	event.values[0]	%	Ambient relative humidity.
TYPE_TEMPERATURE	event.values[0]	°C	Device temperature.[1]

Motion Sensors

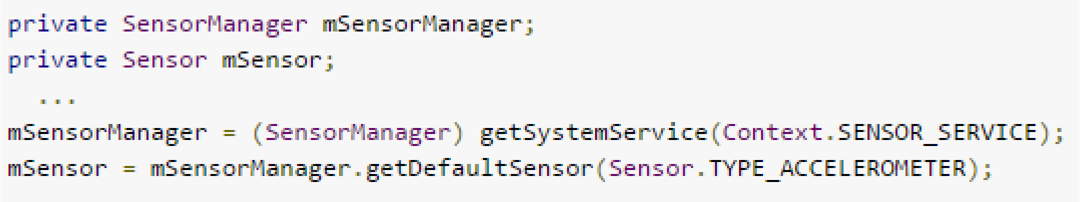
Motion sensors are useful for monitoring device movement, such as tilt, shake, rotation, or swing.

All of the motion sensors return multi-dimensional arrays of sensor values for each SensorEvent.



Sensor	Sensor event data	Description	Units of measure
TYPE_ACCELEROMETER	SensorEvent.values[0]	Acceleration force along the x axis (including gravity).	m/s^2
	SensorEvent.values[1]	Acceleration force along the y axis (including gravity).	
	SensorEvent.values[2]	Acceleration force along the z axis (including gravity).	
TYPE_GRAVITY	SensorEvent.values[0]	Force of gravity along the x axis.	m/s^2
	SensorEvent.values[1]	Force of gravity along the y axis.	
	SensorEvent.values[2]	Force of gravity along the z axis.	
TYPE_GYROSCOPE	SensorEvent.values[0]	Rate of rotation around the x axis.	rad/s
	SensorEvent.values[1]	Rate of rotation around the y axis.	
	SensorEvent.values[2]	Rate of rotation around the z axis.	
TYPE_ROTATION_VECTOR	SensorEvent.values[0]	Rotation vector component along the x axis (x * sin(θ/2)).	Unitless
	SensorEvent.values[1]	Rotation vector component along the y axis (y * sin(θ/2)).	
	SensorEvent.values[2]	Rotation vector component along the z axis (z * sin(θ/2)).	
	SensorEvent.values[3]	Scalar component of the rotation vector ((cos(θ/2)).[1]	
TYPE_STEP_COUNTER	SensorEvent.values[0]	Number of steps taken by the user since the last reboot while the sensor was activated.	Steps
TYPE_STEP_DETECTOR	N/A	N/A	N/A

Using the Accelerometer



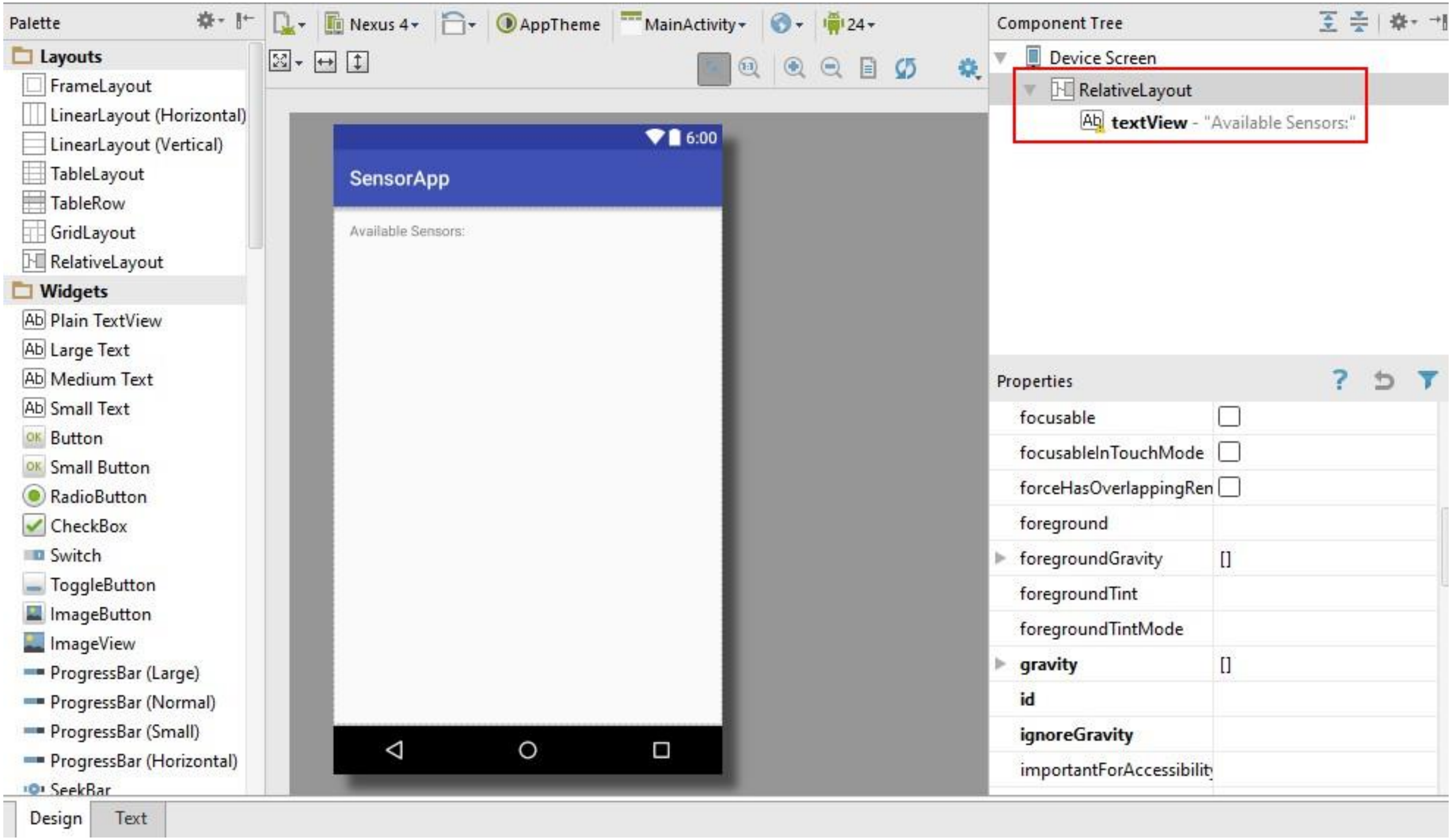
```
private SensorManager mSensorManager;
private Sensor mSensor;
  ...
mSensorManager = (SensorManager) getSystemService(Context.SENSOR_SERVICE);
mSensor = mSensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER);
```

Accelerometers use the standard sensor coordinate system. In practice, this means that the following conditions apply when a device is laying flat on a table in its natural orientation:

If you push the device on the left side (so it moves to the right), the x acceleration value is positive.
If you push the device on the bottom (so it moves away from you), the y acceleration value is positive.
If you push the device toward the sky with an acceleration of A m/s2, the z acceleration value is equal to A + 9.81, which corresponds to the acceleration of the device (+A m/s2) minus the force of gravity (-9.81 m/s2).
The stationary device will have an acceleration value of +9.81, which corresponds to the acceleration of the device (0 m/s2 minus the force of gravity, which is -9.81 m/s2).

IMPLEMENTATION

Listing available sensors



Palette

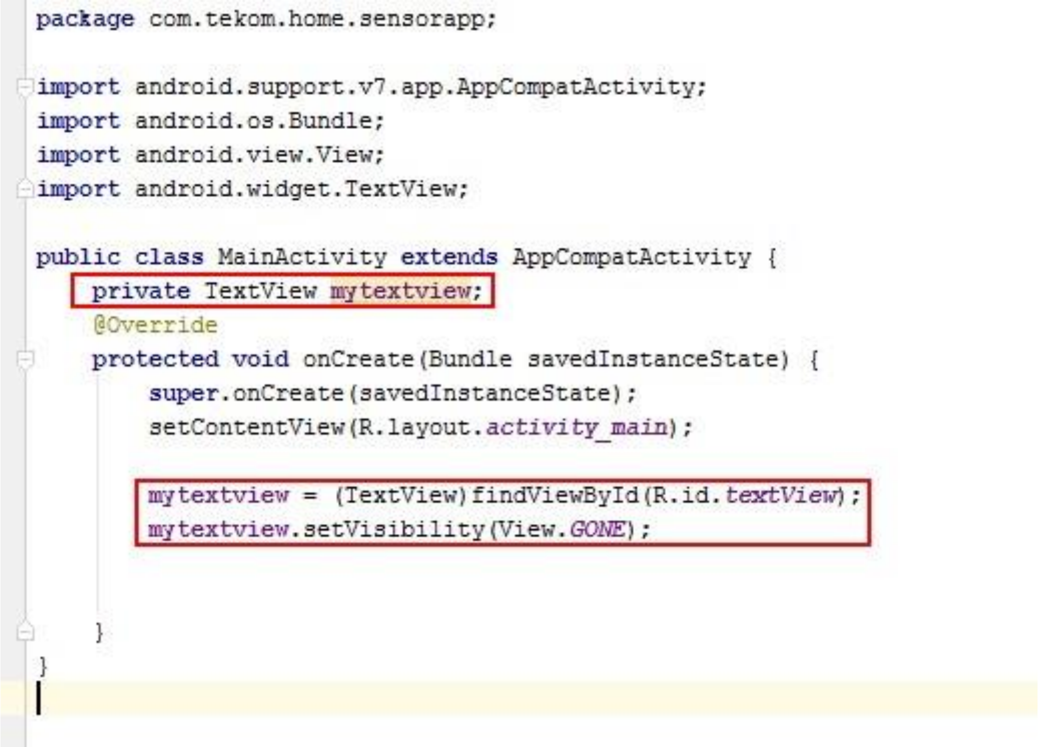
Layouts
- FrameLayout
- LinearLayout (Horizontal)
- LinearLayout (Vertical)
- TableLayout
- TableRow
- GridLayout
- RelativeLayout

Widgets
- Plain TextView
- Large Text
- Medium Text
- Small Text
- Button
- Small Button
- RadioButton
- CheckBox
- Switch
- ToggleButton
- ImageButton
- ImageView
- ProgressBar (Large)
- ProgressBar (Normal)
- ProgressBar (Small)
- ProgressBar (Horizontal)
- SeekBar

Component Tree
- Device Screen
  - RelativeLayout
    - textView - "Available Sensors:"

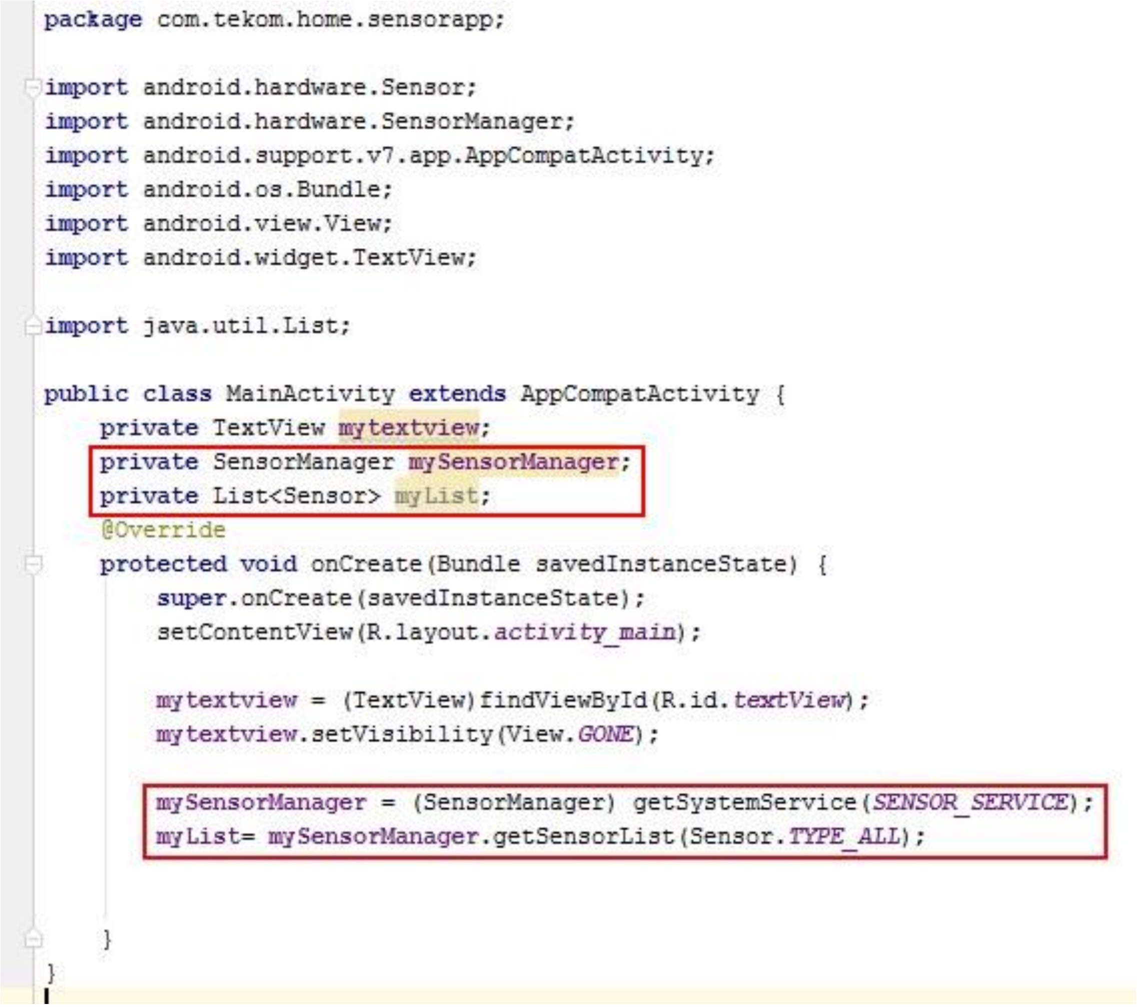
Properties

focusable	□
focusableInTouchMode	□
forceHasOverlappingRen	□
foreground	
foregroundGravity	[]
foregroundTint	
foregroundTintMode	
gravity	[]
id	
ignoreGravity	
importantForAccessibilit	



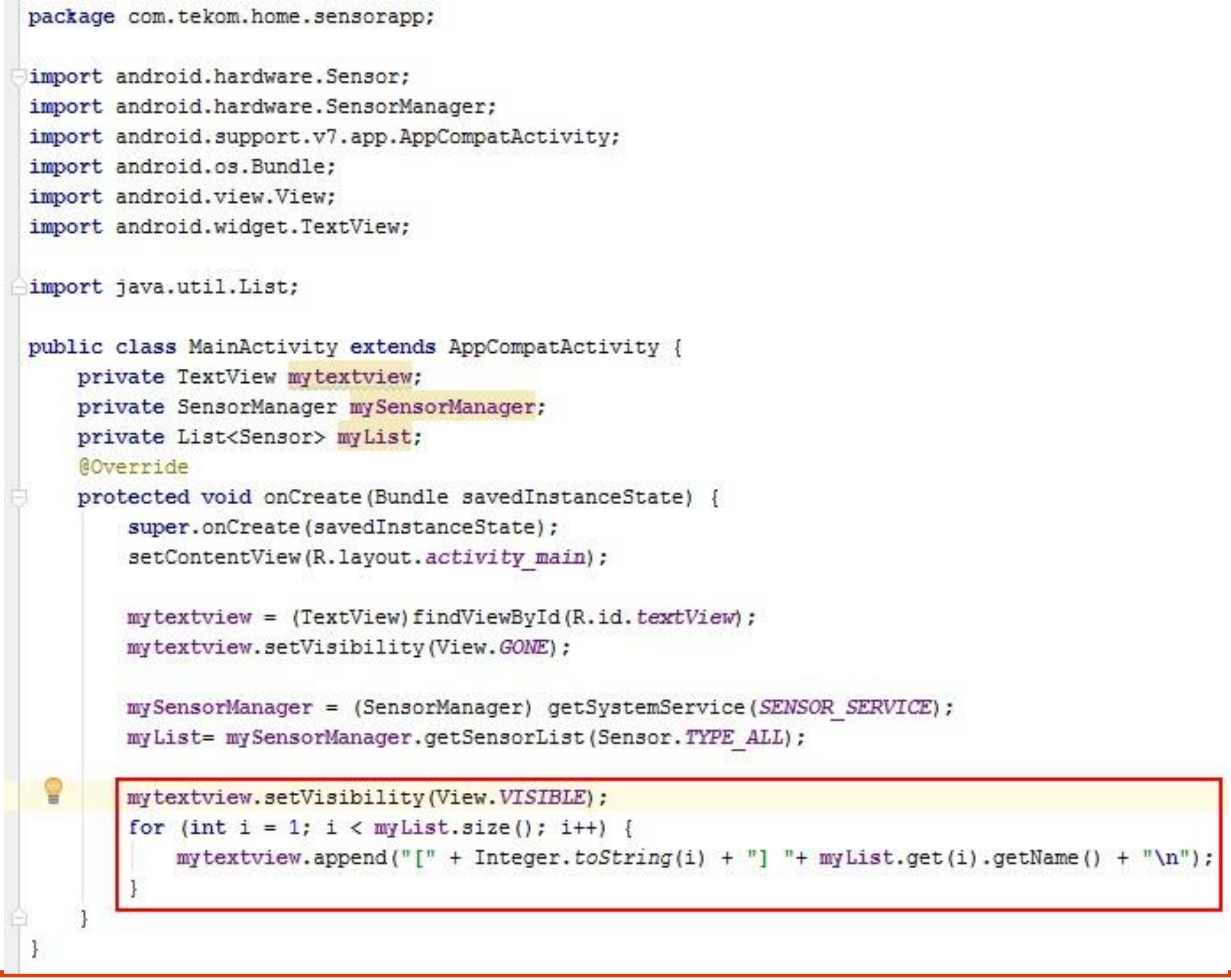
```
package com.tekom.home.sensorapp;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.TextView;

public class MainActivity extends AppCompatActivity {
    private TextView mytextview;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        mytextview = (TextView)findViewById(R.id.textView);
        mytextview.setVisibility(View.GONE);


    }
}
```

List available sensors



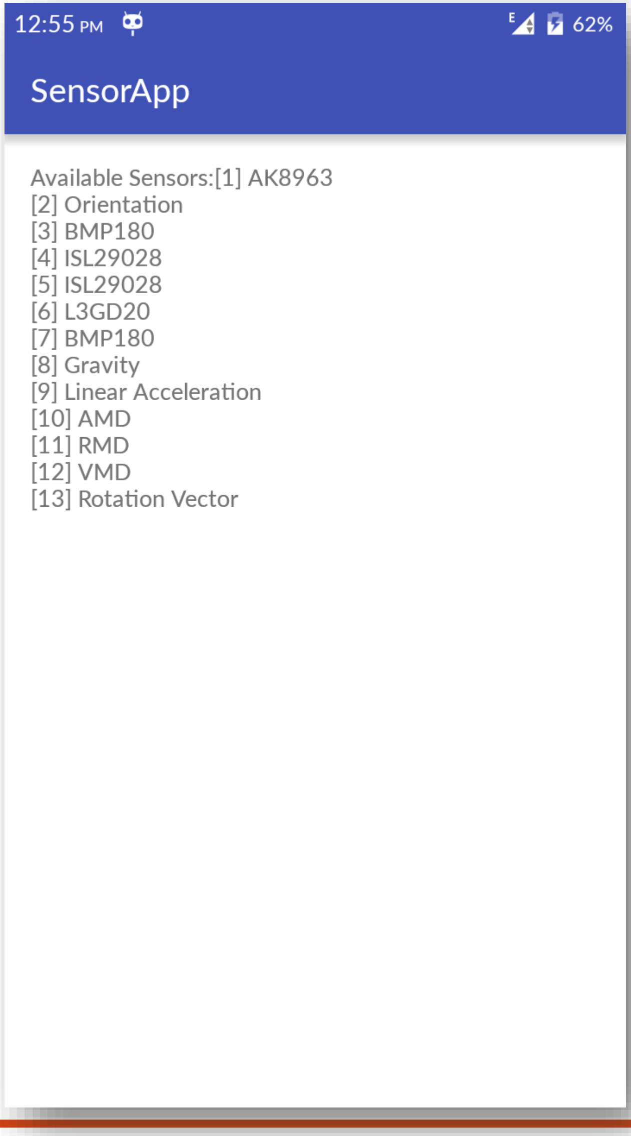
```java
package com.tekom.home.sensorapp;

import android.hardware.Sensor;
import android.hardware.SensorManager;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.TextView;

import java.util.List;

public class MainActivity extends AppCompatActivity {
    private TextView mytextview;
    private SensorManager mySensorManager;
    private List<Sensor> myList;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        mytextview = (TextView)findViewById(R.id.textView);
        mytextview.setVisibility(View.GONE);

        mySensorManager = (SensorManager) getSystemService(SENSOR_SERVICE);
        myList= mySensorManager.getSensorList(Sensor.TYPE_ALL);


    }
}
```

List available sensors



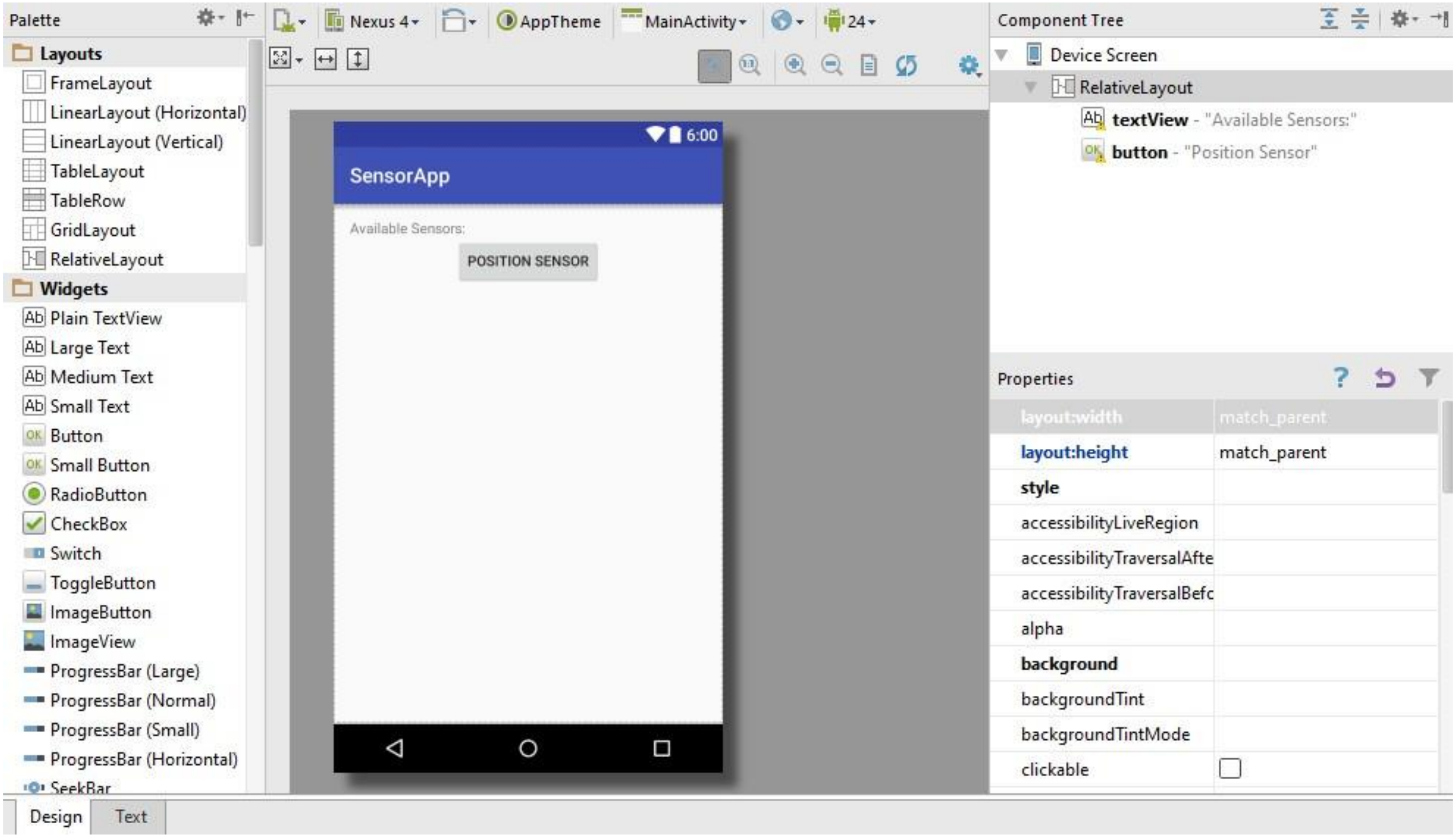
```
package com.tekom.home.sensorapp;

import android.hardware.Sensor;
import android.hardware.SensorManager;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.TextView;

import java.util.List;

public class MainActivity extends AppCompatActivity {
    private TextView mytextview;
    private SensorManager mySensorManager;
    private List<Sensor> myList;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        mytextview = (TextView)findViewById(R.id.textView);
        mytextview.setVisibility(View.GONE);

        mySensorManager = (SensorManager) getSystemService(SENSOR_SERVICE);
        myList= mySensorManager.getSensorList(Sensor.TYPE_ALL);

        mytextview.setVisibility(View.VISIBLE);
        for (int i = 1; i < myList.size(); i++) {
            mytextview.append("[" + Integer.toString(i) + "] "+ myList.get(i).getName() + "\n");
        }
    }
}
```

Result



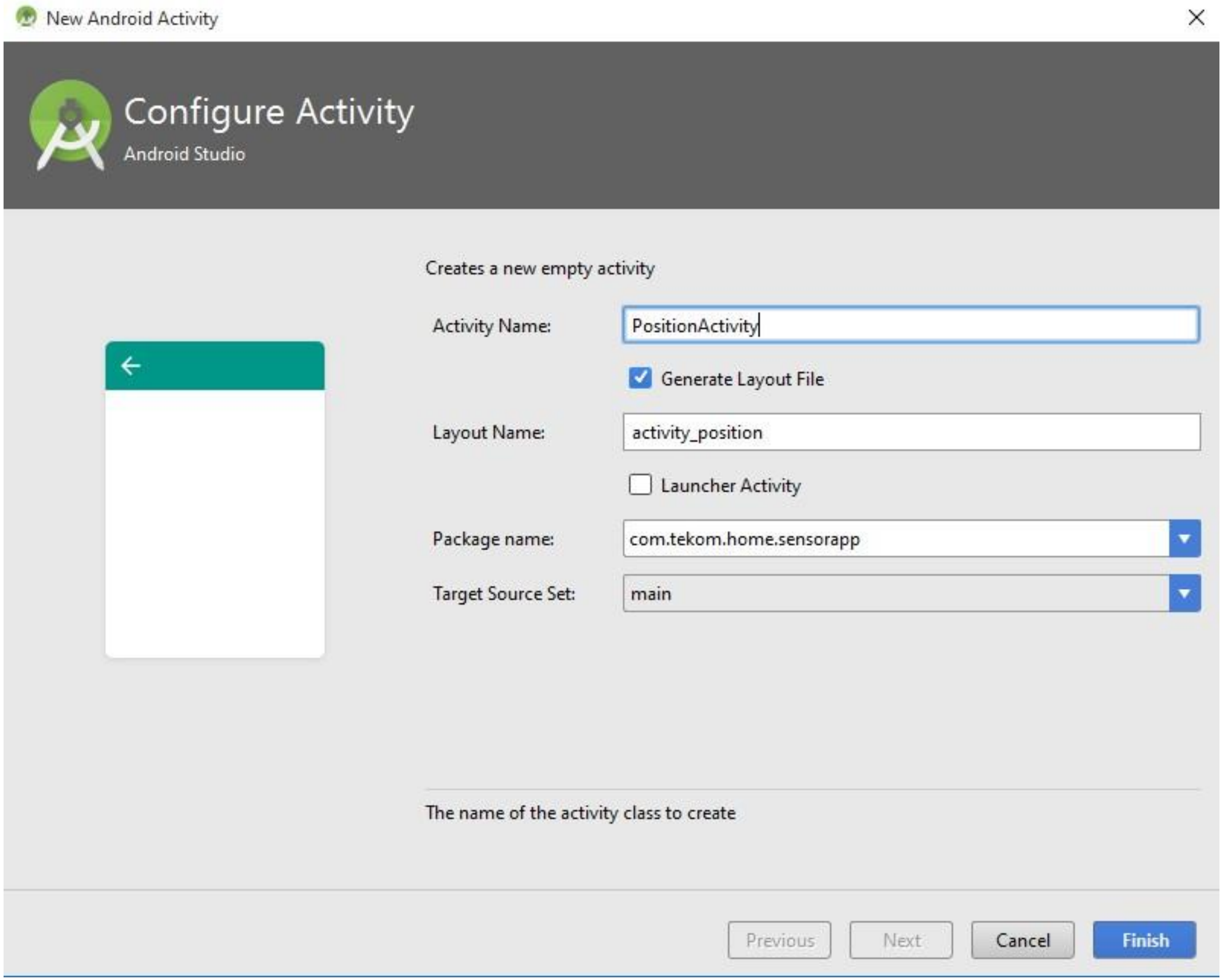
```
Available Sensors:[1] AK8963
[2] Orientation
[3] BMP180
[4] ISL29028
[5] ISL29028
[6] L3GD20
[7] BMP180
[8] Gravity
[9] Linear Acceleration
[10] AMD
[11] RMD
[12] VMD
[13] Rotation Vector
```

Read position Sensor - proximity



- Device Screen
  - RelativeLayout
    - textView - "Available Sensors:"
    - button - "Position Sensor"

layout:width	match_parent
layout:height	match_parent
style	
accessibilityLiveRegion	
accessibilityTraversalAfte	
accessibilityTraversalBefc	
alpha	
background	
backgroundTint	
backgroundTintMode	
clickable	□



New Android Activity

Configure Activity

Android Studio

Creates a new empty activity

Activity Name: PositionActivity

☑ Generate Layout File

Layout Name: activity_position

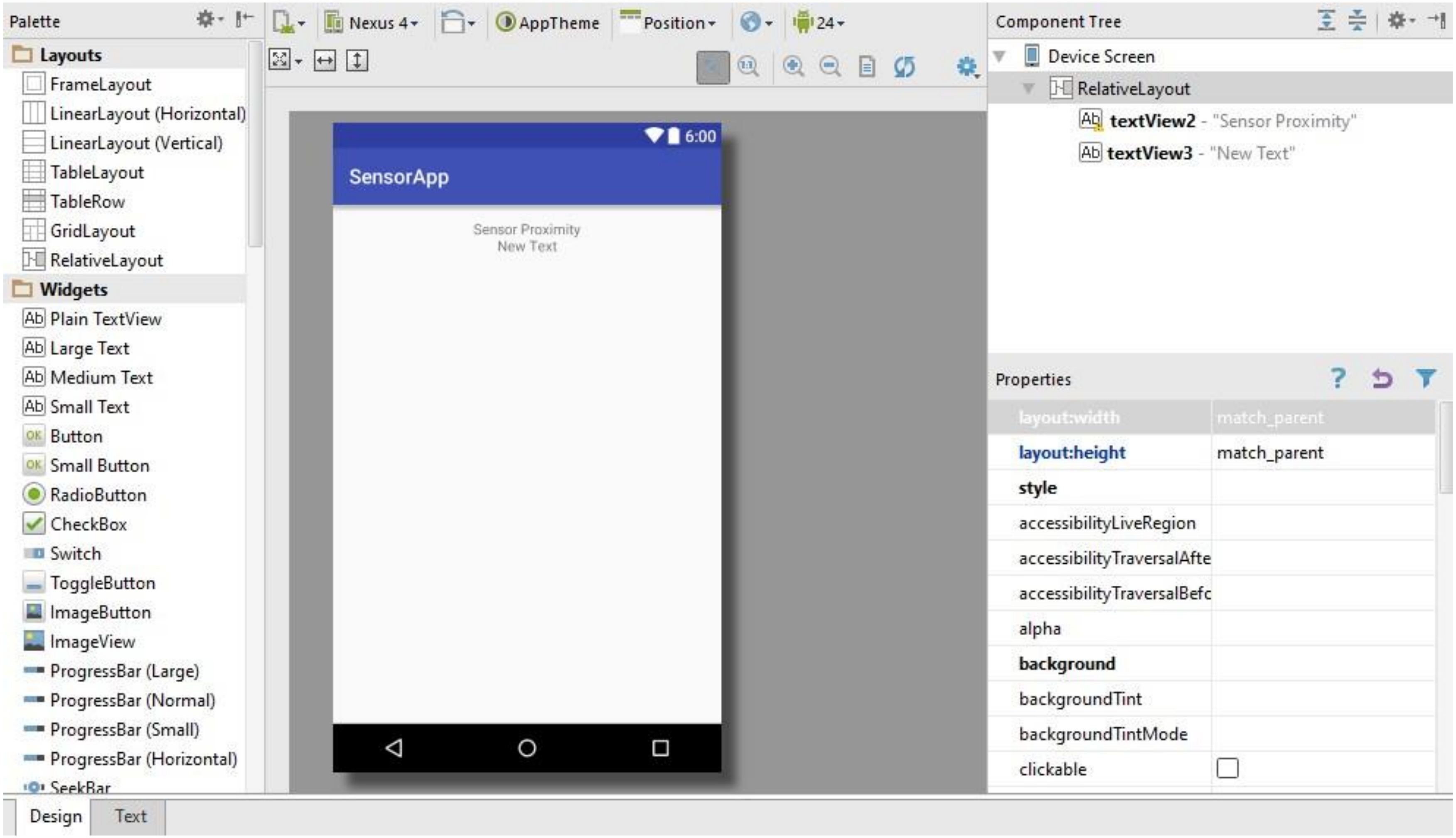
□ Launcher Activity

Package name: com.tekom.home.sensorapp

Target Source Set: main

The name of the activity class to create

Position activity



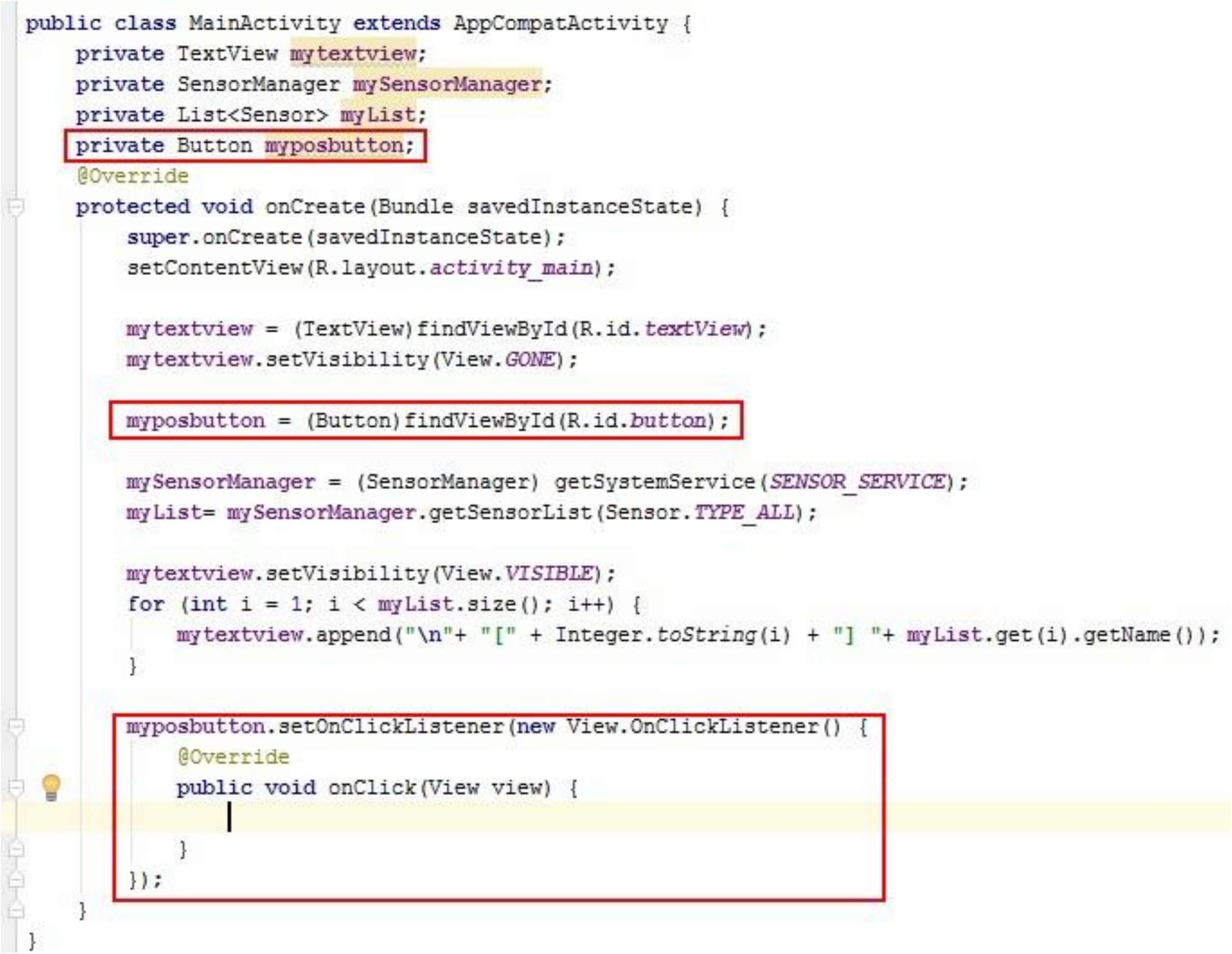
Component Tree

- Device Screen
  - RelativeLayout
    - textView2 - "Sensor Proximity"
    - textView3 - "New Text"

Properties

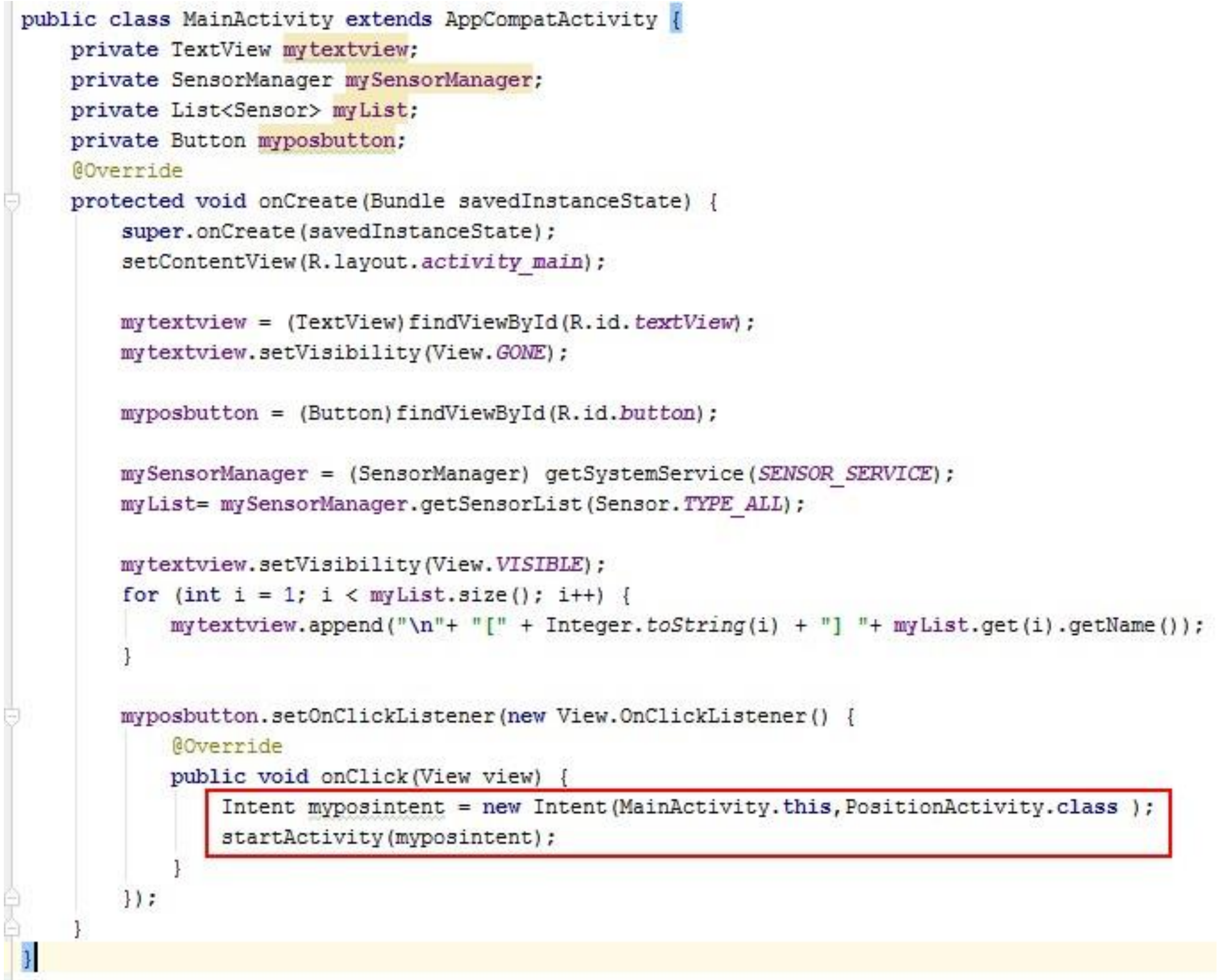
layout:width	match_parent
layout:height	match_parent
style	
accessibilityLiveRegion	
accessibilityTraversalAfte	
accessibilityTraversalBefc	
alpha	
background	
backgroundTint	
backgroundTintMode	
clickable	□

Back and Modify MainActivity Java file



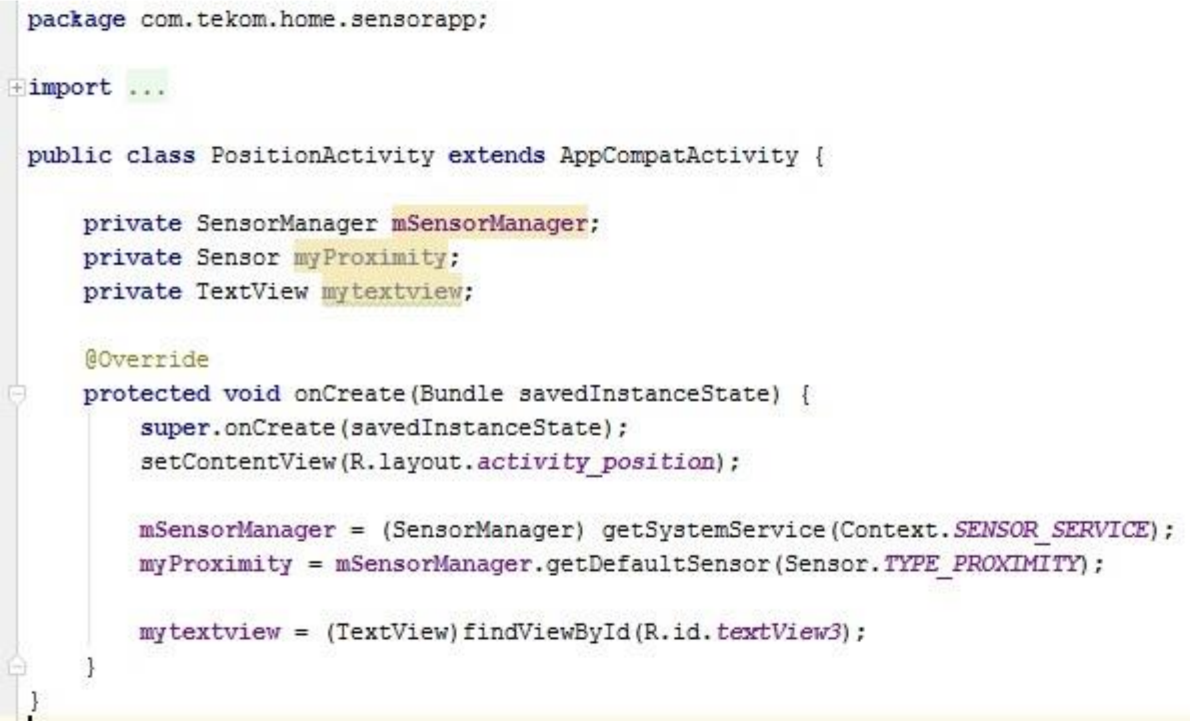
```
public class MainActivity extends AppCompatActivity {
    private TextView mytextview;
    private SensorManager mySensorManager;
    private List<Sensor> myList;
    private Button myposbutton;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        mytextview = (TextView)findViewById(R.id.textView);
        mytextview.setVisibility(View.GONE);

        myposbutton = (Button)findViewById(R.id.button);

        mySensorManager = (SensorManager) getSystemService(SENSOR_SERVICE);
        myList= mySensorManager.getSensorList(Sensor.TYPE_ALL);

        mytextview.setVisibility(View.VISIBLE);
        for (int i = 1; i < myList.size(); i++) {
            mytextview.append("\n"+ "[" + Integer.toString(i) + "] "+ myList.get(i).getName());
        }

        myposbutton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {

            }
        });
    }
}
```

OnClick Handler: start the PositionActivity.class

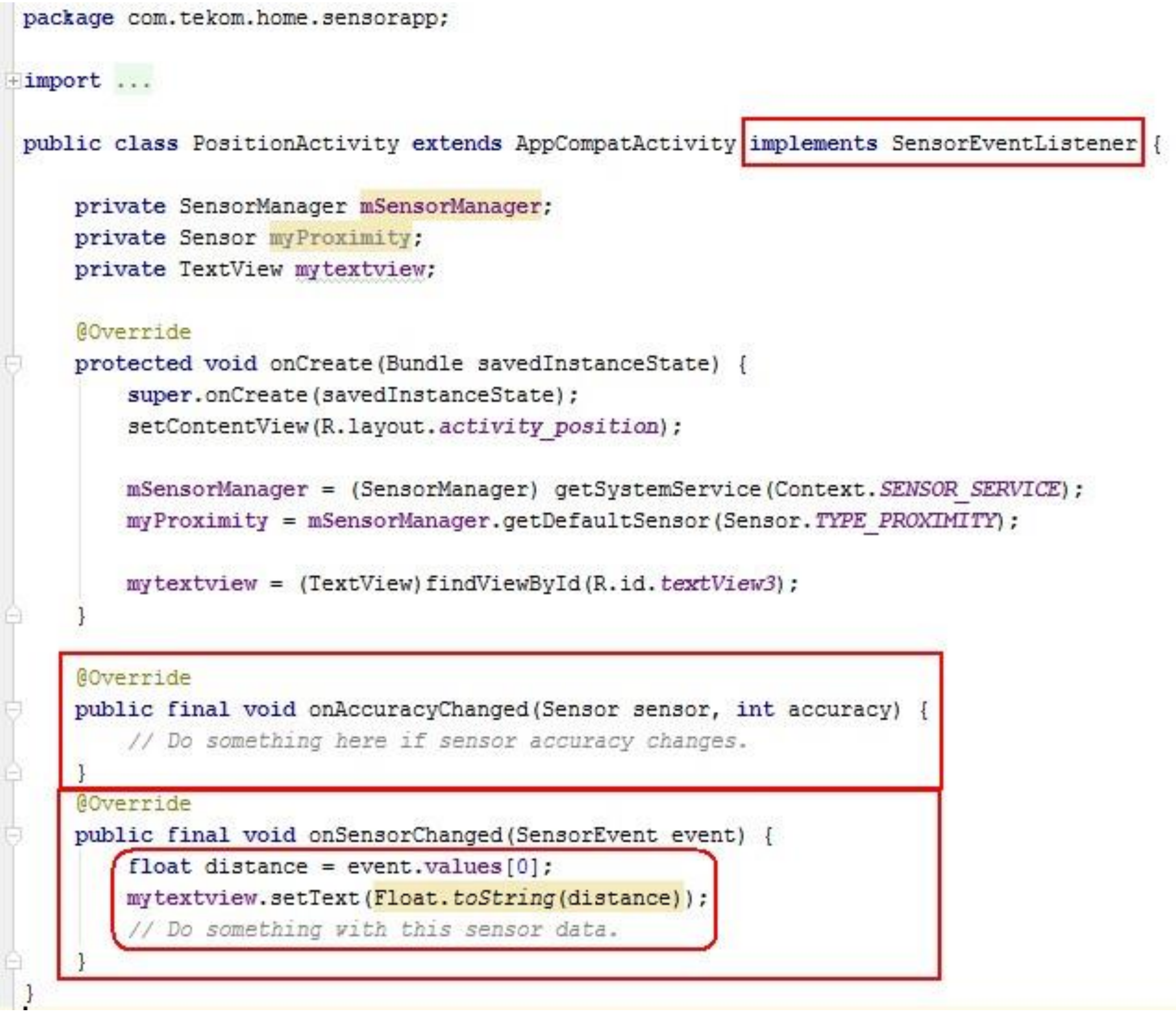


```
public class MainActivity extends AppCompatActivity {
    private TextView mytextview;
    private SensorManager mySensorManager;
    private List<Sensor> myList;
    private Button myposbutton;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        mytextview = (TextView)findViewById(R.id.textView);
        mytextview.setVisibility(View.GONE);

        myposbutton = (Button)findViewById(R.id.button);

        mySensorManager = (SensorManager) getSystemService(SENSOR_SERVICE);
        myList= mySensorManager.getSensorList(Sensor.TYPE_ALL);

        mytextview.setVisibility(View.VISIBLE);
        for (int i = 1; i < myList.size(); i++) {
            mytextview.append("\n"+ "[" + Integer.toString(i) + "] "+ myList.get(i).getName());
        }

        myposbutton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Intent myposintent = new Intent(MainActivity.this,PositionActivity.class );
                startActivity(myposintent);
            }
        });
    }
}
```

Now edit the PositionActivity java file to get the sensor proximity data

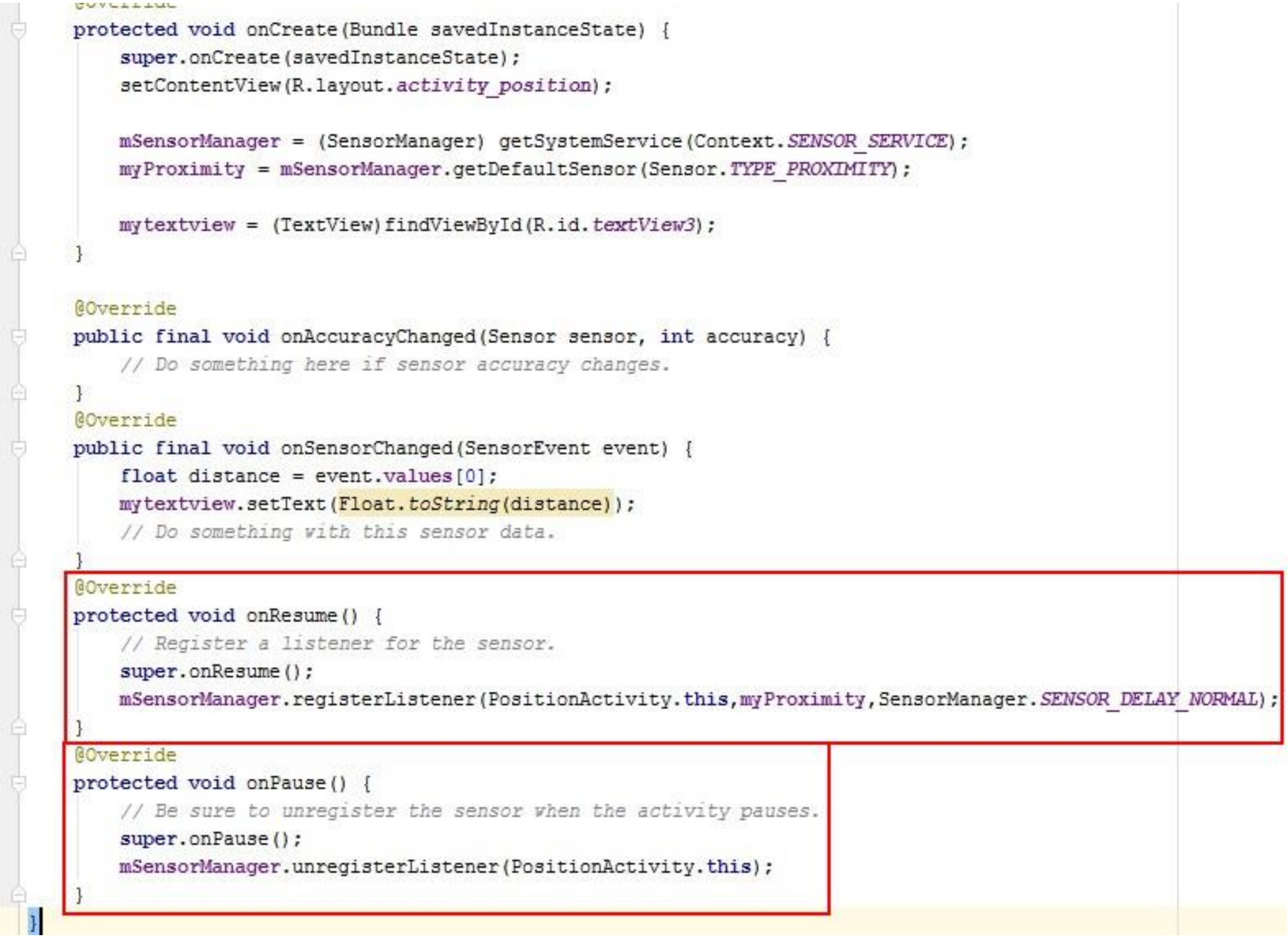


```
package com.tekom.home.sensorapp;

import ...

public class PositionActivity extends AppCompatActivity {

    private SensorManager mSensorManager;
    private Sensor myProximity;
    private TextView mytextview;

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

        mSensorManager = (SensorManager) getSystemService(Context.SENSOR_SERVICE);
        myProximity = mSensorManager.getDefaultSensor(Sensor.TYPE_PROXIMITY);

        mytextview = (TextView)findViewById(R.id.textView3);
    }
}
```



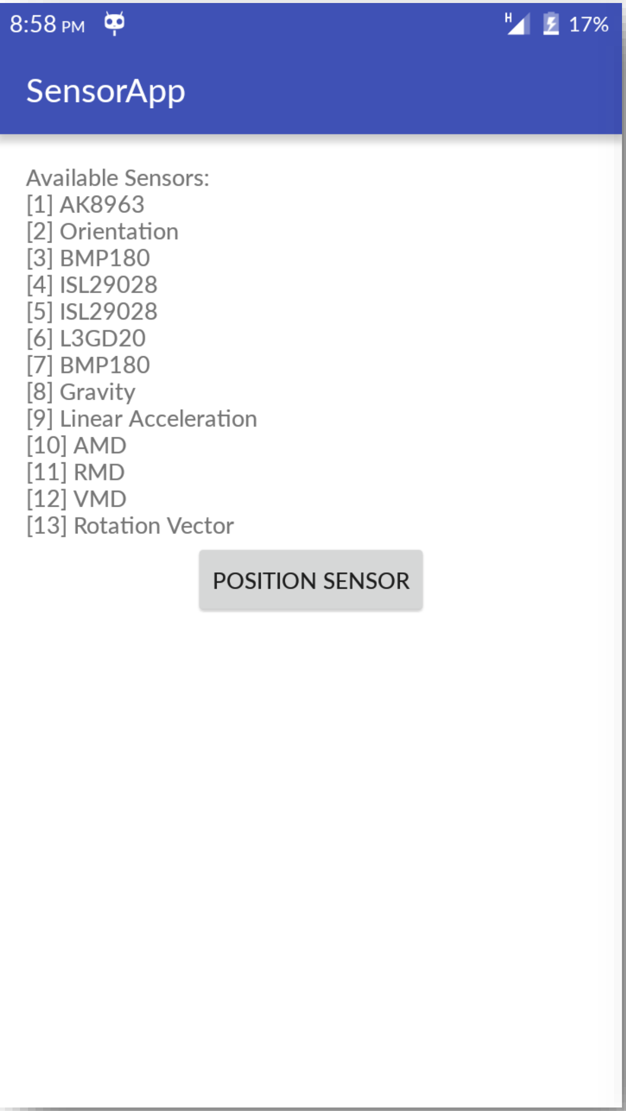
```
package com.tekom.home.sensorapp;

import ...

public class PositionActivity extends AppCompatActivity implements SensorEventListener {

    private SensorManager mSensorManager;
    private Sensor myProximity;
    private TextView mytextview;

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

        mSensorManager = (SensorManager) getSystemService(Context.SENSOR_SERVICE);
        myProximity = mSensorManager.getDefaultSensor(Sensor.TYPE_PROXIMITY);

        mytextview = (TextView)findViewById(R.id.textView3);
    }

    @Override
    public final void onAccuracyChanged(Sensor sensor, int accuracy) {
        // Do something here if sensor accuracy changes.
    }
    @Override
    public final void onSensorChanged(SensorEvent event) {
        float distance = event.values[0];
        mytextview.setText(Float.toString(distance));
        // Do something with this sensor data.
    }
}
```

Don’t forget implement onPause() and onResume() methods

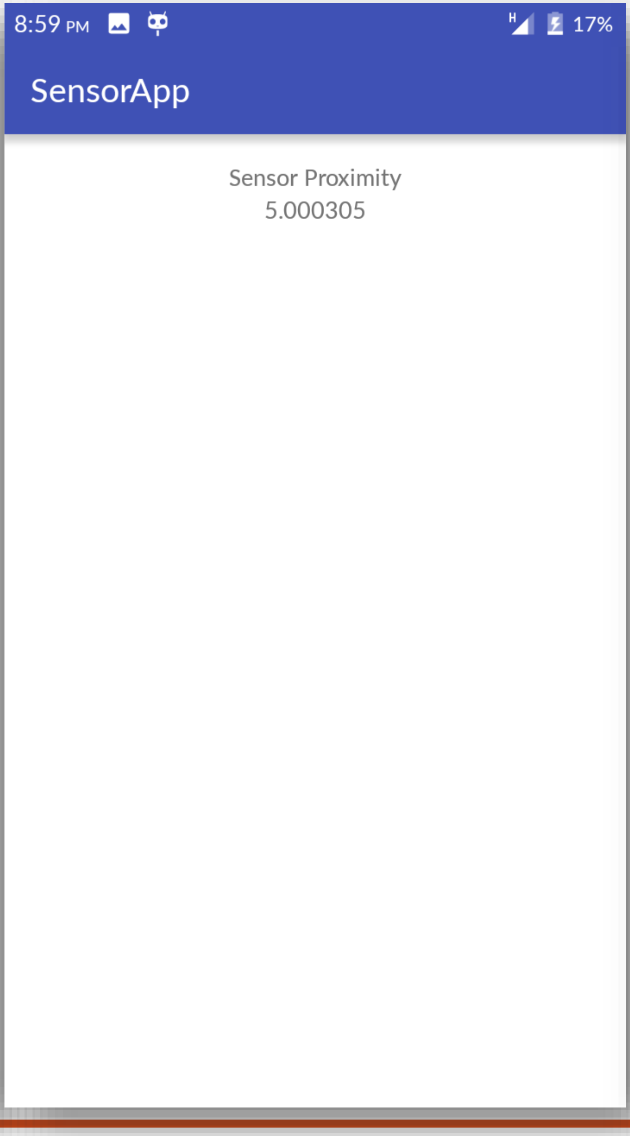


```java
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_position);

    mSensorManager = (SensorManager) getSystemService(Context.SENSOR_SERVICE);
    myProximity = mSensorManager.getDefaultSensor(Sensor.TYPE_PROXIMITY);

    mytextview = (TextView)findViewById(R.id.textView3);
}

@Override
public final void onAccuracyChanged(Sensor sensor, int accuracy) {
    // Do something here if sensor accuracy changes.
}
@Override
public final void onSensorChanged(SensorEvent event) {
    float distance = event.values[0];
    mytextview.setText(Float.toString(distance));
    // Do something with this sensor data.
}
@Override
protected void onResume() {
    // Register a listener for the sensor.
    super.onResume();
    mSensorManager.registerListener(PositionActivity.this,myProximity,SensorManager.SENSOR_DELAY_NORMAL);
}
@Override
protected void onPause() {
    // Be sure to unregister the sensor when the activity pauses.
    super.onPause();
    mSensorManager.unregisterListener(PositionActivity.this);
}
}
```

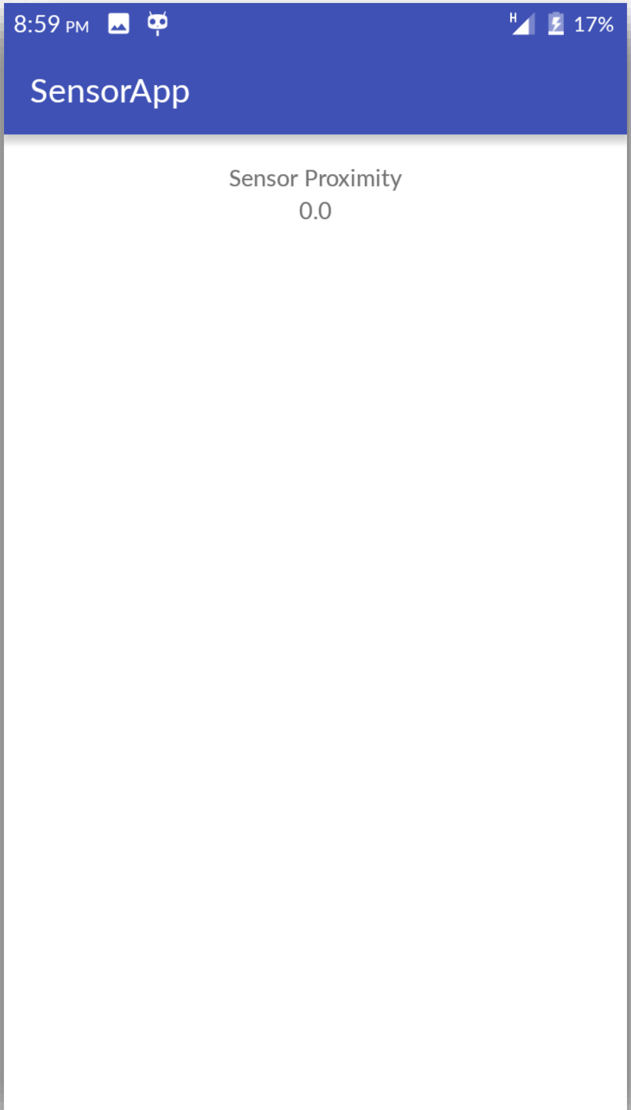
Result



Available Sensors:
[1] AK8963
[2] Orientation
[3] BMP180
[4] ISL29028
[5] ISL29028
[6] L3GD20
[7] BMP180
[8] Gravity
[9] Linear Acceleration
[10] AMD
[11] RMD
[12] VMD
[13] Rotation Vector

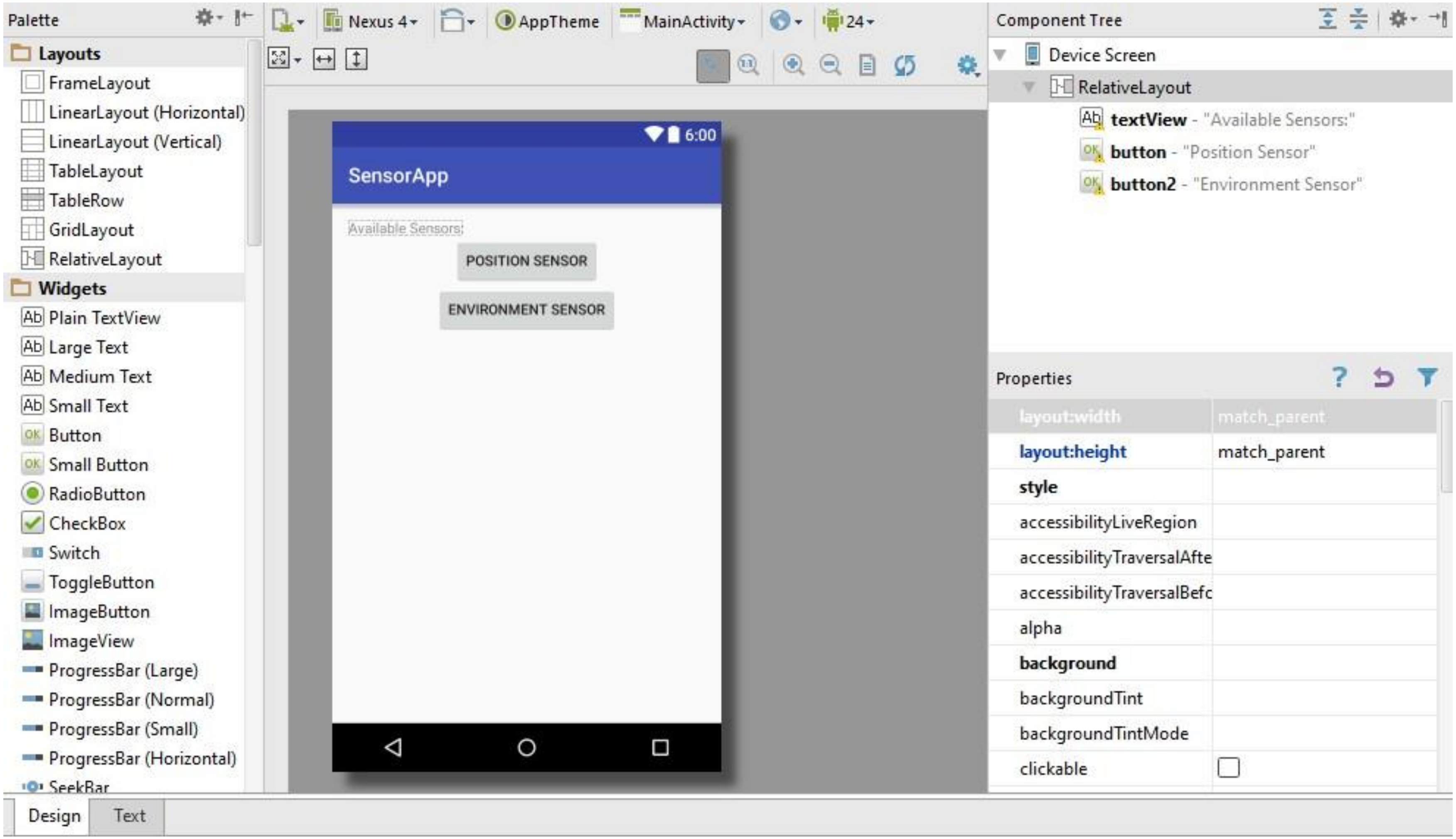


Sensor Proximity
5.000305



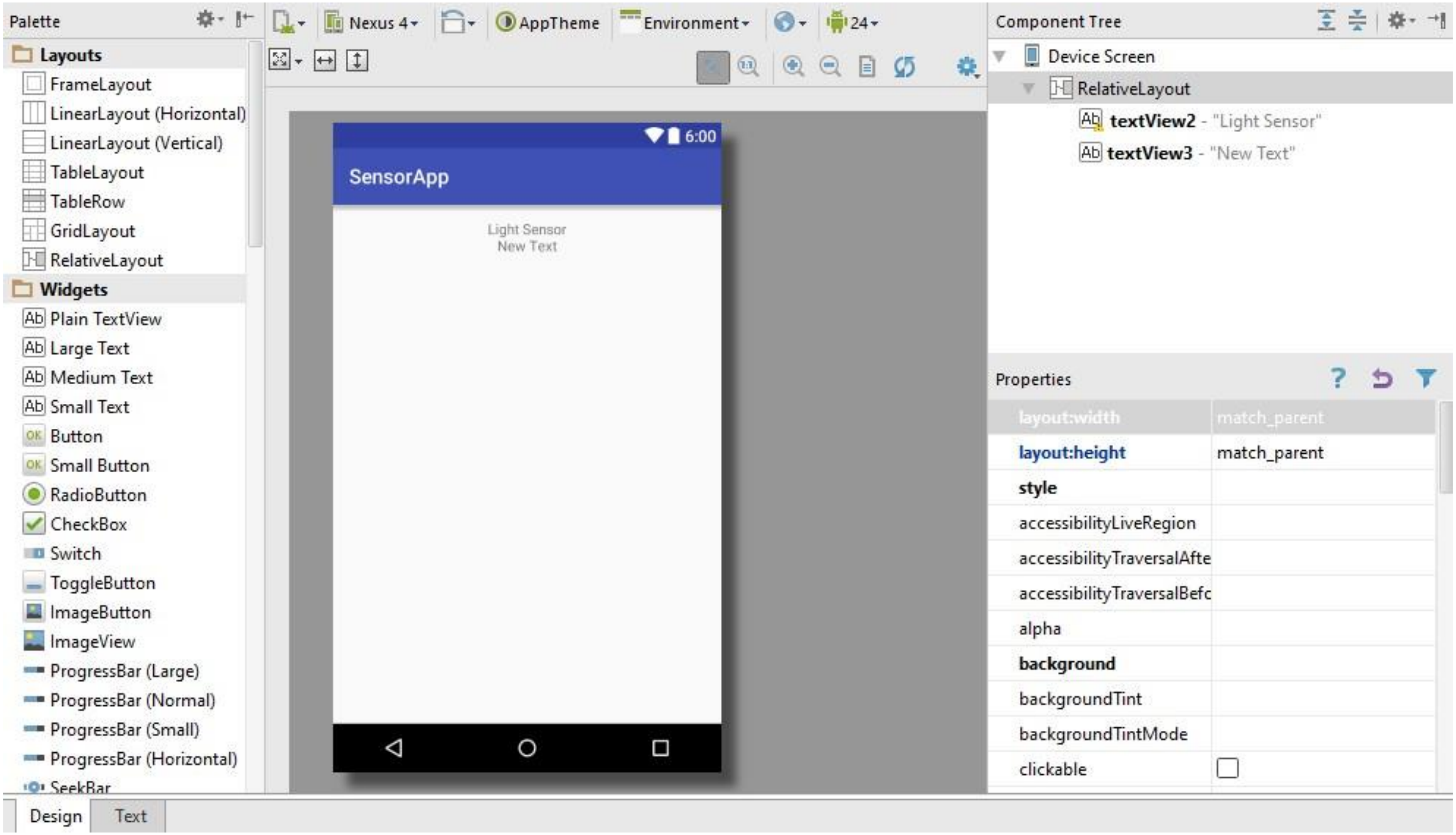
Sensor Proximity
0.0

Environment Sensor - Light



layout:width	match_parent
layout:height	match_parent
style	
accessibilityLiveRegion	
accessibilityTraversalAfte	
accessibilityTraversalBefc	
alpha	
background	
backgroundTint	
backgroundTintMode	
clickable	□

Environment activity



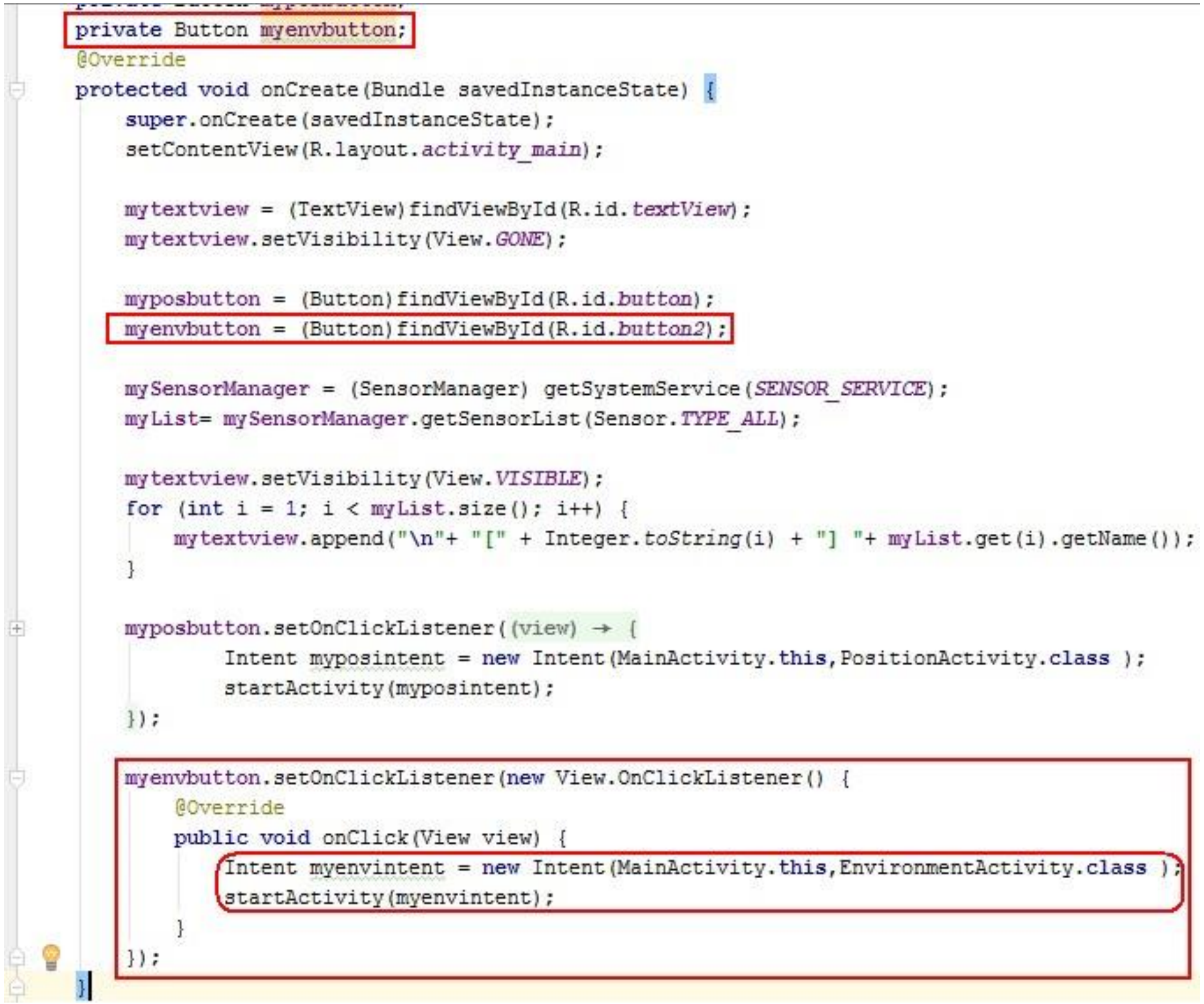
Component Tree

- Device Screen
  - RelativeLayout
    - textView2 - "Light Sensor"
    - textView3 - "New Text"

Properties

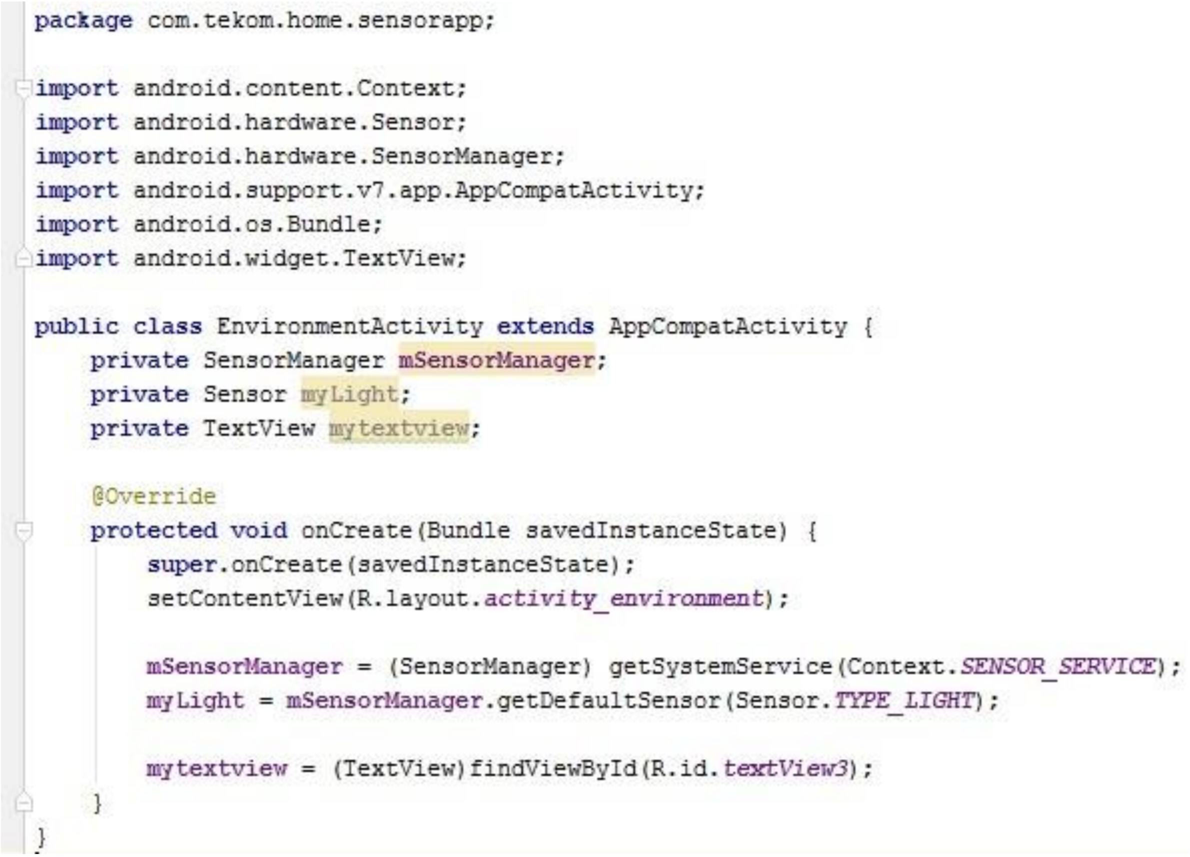
layout:width	match_parent
layout:height	match_parent
style	
accessibilityLiveRegion	
accessibilityTraversalAfte	
accessibilityTraversalBefc	
alpha	
background	
backgroundTint	
backgroundTintMode	
clickable	□

Back and modify the MainActivity.java

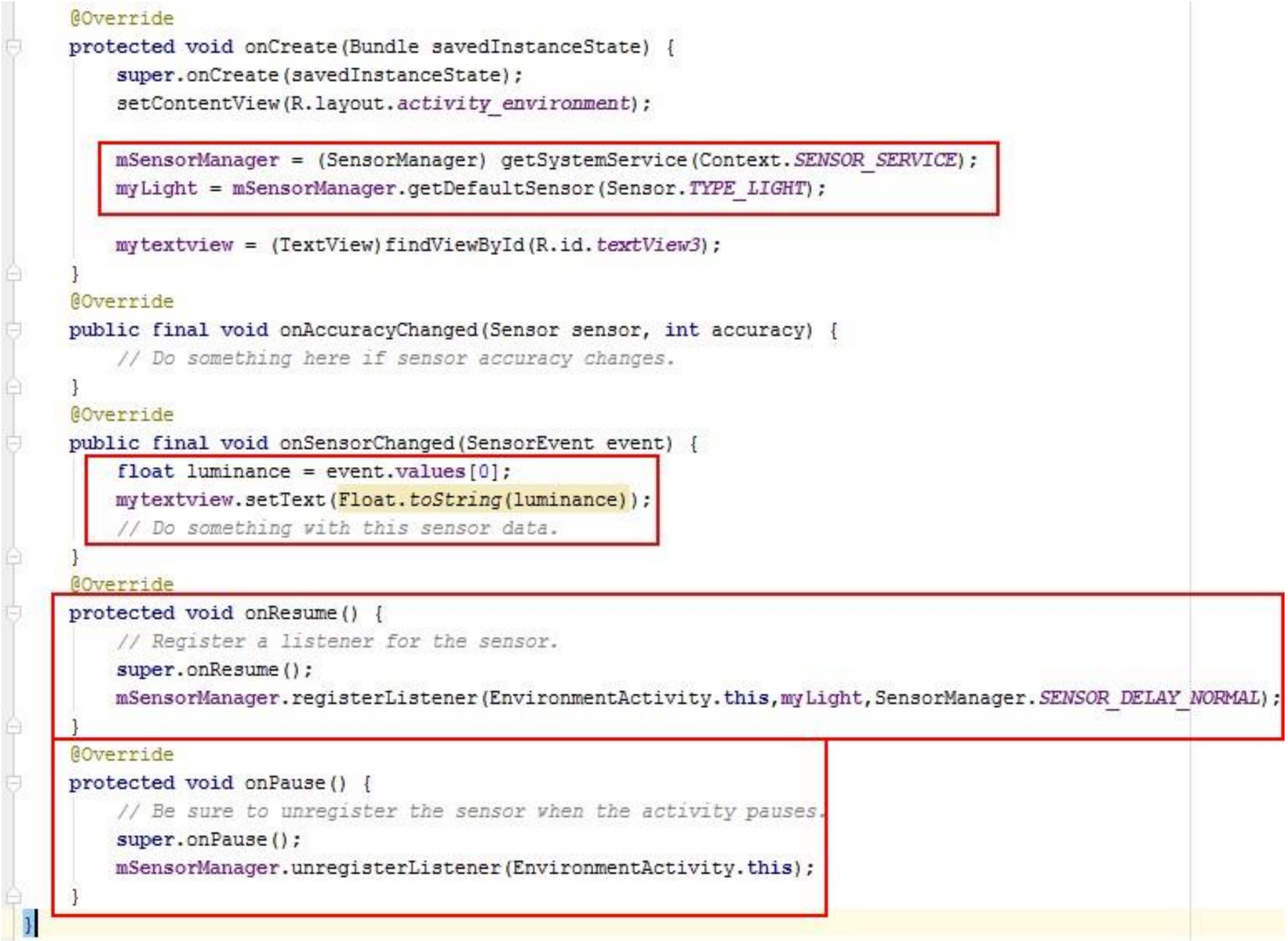


```
private Button myposbutton;
private Button myenvbutton;
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    mytextview = (TextView)findViewById(R.id.textView);
    mytextview.setVisibility(View.GONE);

    myposbutton = (Button)findViewById(R.id.button);
    myenvbutton = (Button)findViewById(R.id.button2);

    mySensorManager = (SensorManager) getSystemService(SENSOR_SERVICE);
    myList= mySensorManager.getSensorList(Sensor.TYPE_ALL);

    mytextview.setVisibility(View.VISIBLE);
    for (int i = 1; i < myList.size(); i++) {
        mytextview.append("\n"+ "[" + Integer.toString(i) + "] "+ myList.get(i).getName());
    }

    myposbutton.setOnClickListener((view) -> {
            Intent myposintent = new Intent(MainActivity.this,PositionActivity.class );
            startActivity(myposintent);
    });

    myenvbutton.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            Intent myenvintent = new Intent(MainActivity.this,EnvironmentActivity.class );
            startActivity(myenvintent);
        }
    });
}
```

Now modify the EnvironmentActivity.java

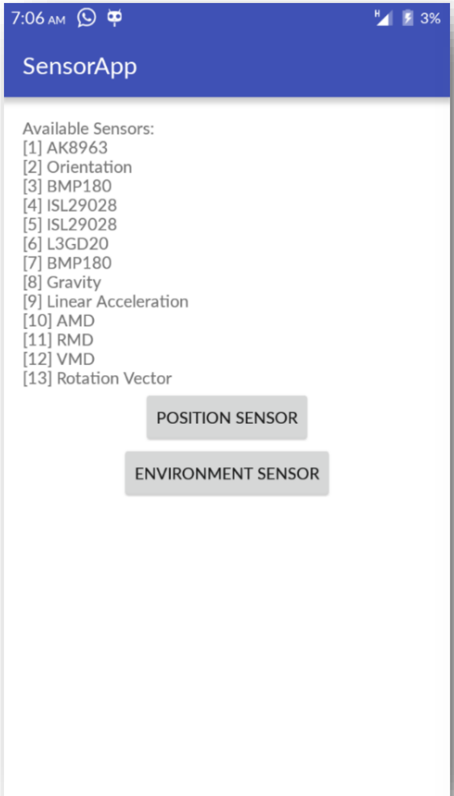


```
package com.tekom.home.sensorapp;

import android.content.Context;
import android.hardware.Sensor;
import android.hardware.SensorManager;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.widget.TextView;

public class EnvironmentActivity extends AppCompatActivity {
    private SensorManager mSensorManager;
    private Sensor myLight;
    private TextView mytextview;

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

        mSensorManager = (SensorManager) getSystemService(Context.SENSOR_SERVICE);
        myLight = mSensorManager.getDefaultSensor(Sensor.TYPE_LIGHT);

        mytextview = (TextView)findViewById(R.id.textView3);
    }
}
```

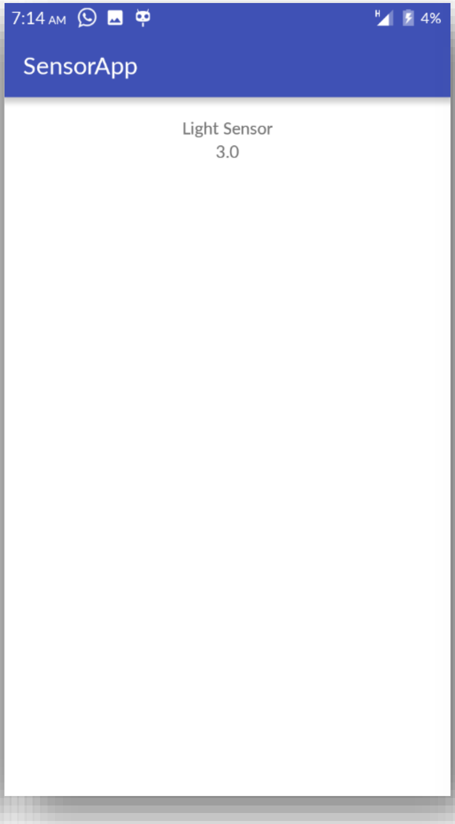


```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_environment);

    mSensorManager = (SensorManager) getSystemService(Context.SENSOR_SERVICE);
    myLight = mSensorManager.getDefaultSensor(Sensor.TYPE_LIGHT);

    mytextview = (TextView)findViewById(R.id.textView3);
}
@Override
public final void onAccuracyChanged(Sensor sensor, int accuracy) {
    // Do something here if sensor accuracy changes.
}
@Override
public final void onSensorChanged(SensorEvent event) {
    float luminance = event.values[0];
    mytextview.setText(Float.toString(luminance));
    // Do something with this sensor data.
}
@Override
protected void onResume() {
    // Register a listener for the sensor.
    super.onResume();
    mSensorManager.registerListener(EnvironmentActivity.this,myLight,SensorManager.SENSOR_DELAY_NORMAL);
}
@Override
protected void onPause() {
    // Be sure to unregister the sensor when the activity pauses.
    super.onPause();
    mSensorManager.unregisterListener(EnvironmentActivity.this);
}
}
```

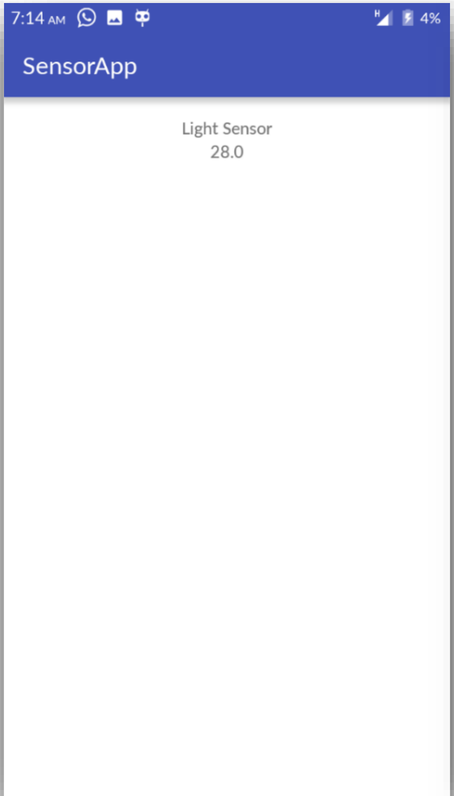
Result



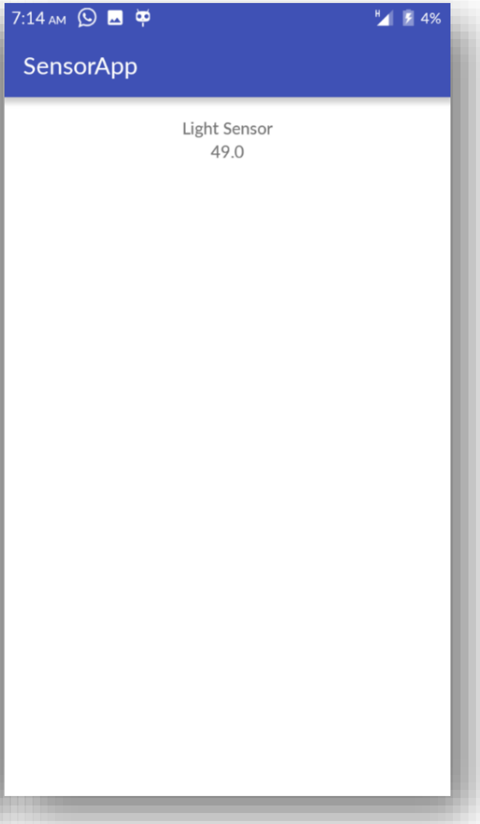
Available Sensors:
[1] AK8963
[2] Orientation
[3] BMP180
[4] ISL29028
[5] ISL29028
[6] L3GD20
[7] BMP180
[8] Gravity
[9] Linear Acceleration
[10] AMD
[11] RMD
[12] VMD
[13] Rotation Vector



Light Sensor
3.0

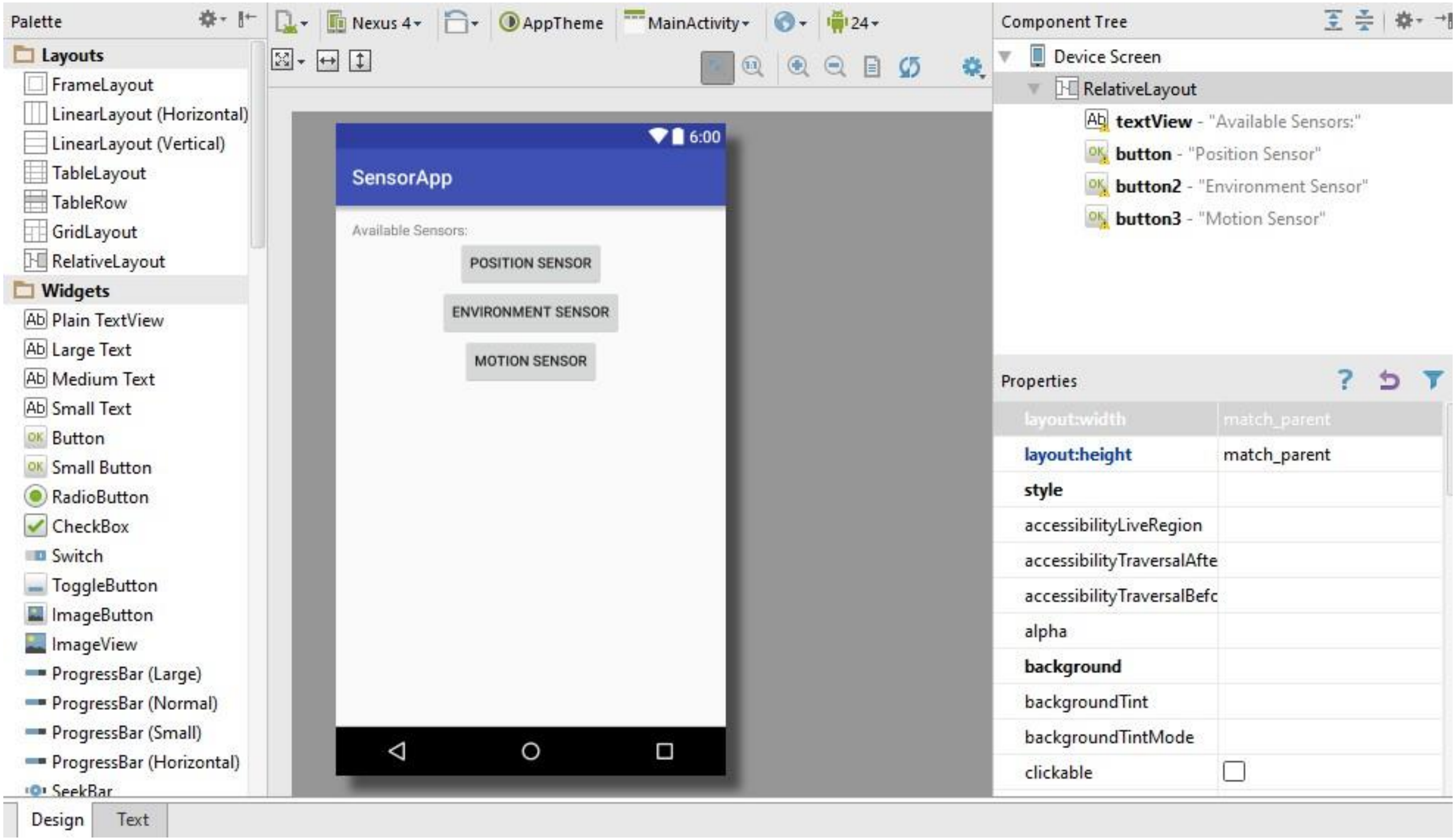


Light Sensor
28.0



Light Sensor
49.0

Motion Sensor - Accelerometer



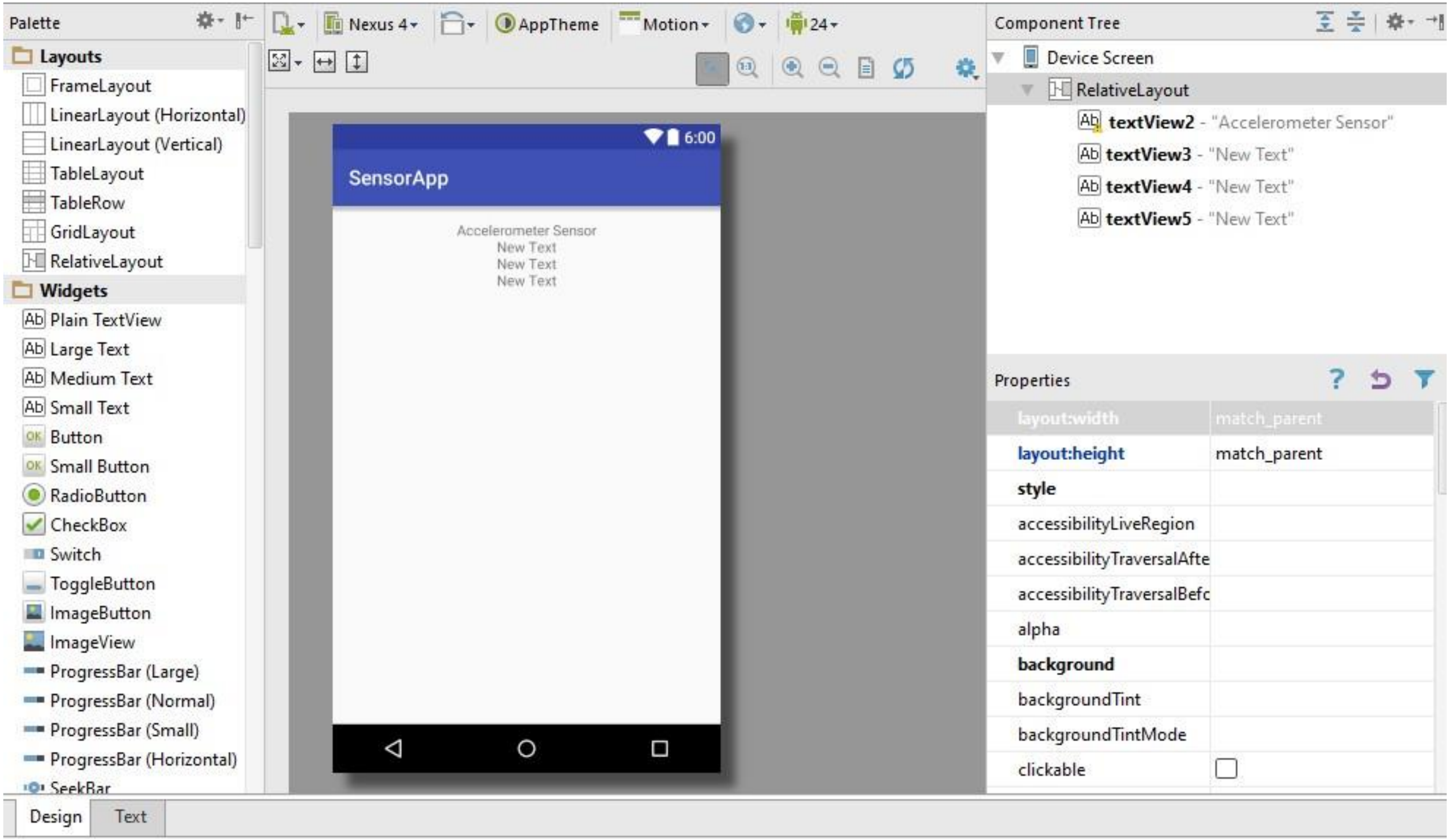
Component Tree

- Device Screen
  - RelativeLayout
    - textView - "Available Sensors:"
    - button - "Position Sensor"
    - button2 - "Environment Sensor"
    - button3 - "Motion Sensor"

Properties

layout:width	match_parent
layout:height	match_parent
style	
accessibilityLiveRegion	
accessibilityTraversalAfte	
accessibilityTraversalBefc	
alpha	
background	
backgroundTint	
backgroundTintMode	
clickable	□

Motion activity



Palette

Layouts
- FrameLayout
- LinearLayout (Horizontal)
- LinearLayout (Vertical)
- TableLayout
- TableRow
- GridLayout
- RelativeLayout

Widgets
- Plain TextView
- Large Text
- Medium Text
- Small Text
- Button
- Small Button
- RadioButton
- CheckBox
- Switch
- ToggleButton
- ImageButton
- ImageView
- ProgressBar (Large)
- ProgressBar (Normal)
- ProgressBar (Small)
- ProgressBar (Horizontal)
- SeekBar

Nexus 4 · AppTheme · Motion · 24

SensorApp

Accelerometer Sensor
New Text
New Text
New Text

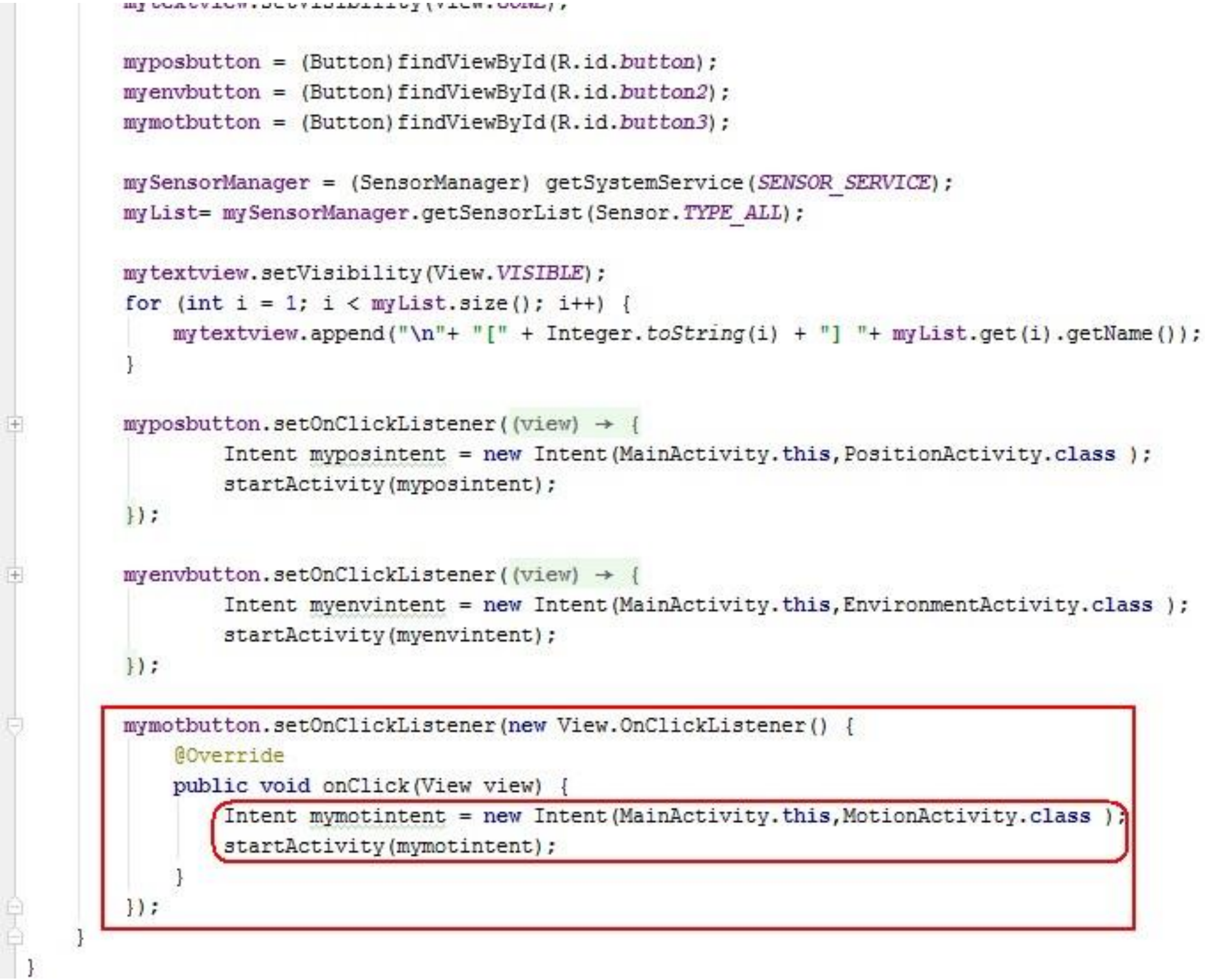
Component Tree

- Device Screen
  - RelativeLayout
    - textView2 - "Accelerometer Sensor"
    - textView3 - "New Text"
    - textView4 - "New Text"
    - textView5 - "New Text"

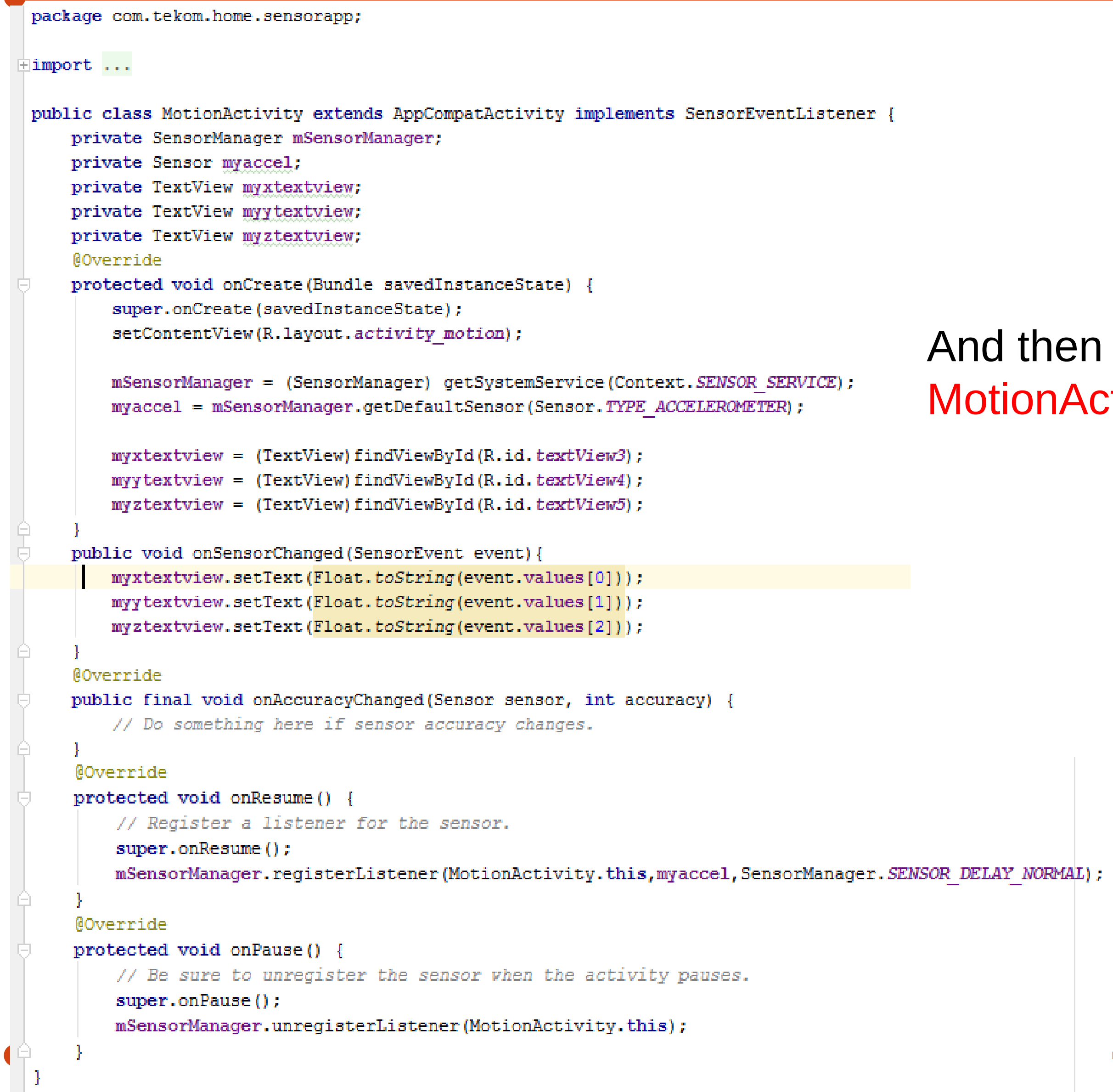
Properties

layout:width	match_parent
layout:height	match_parent
style	
accessibilityLiveRegion	
accessibilityTraversalAfte	
accessibilityTraversalBefc	
alpha	
background	
backgroundTint	
backgroundTintMode	
clickable	□

Back and modify the MainActivity.java



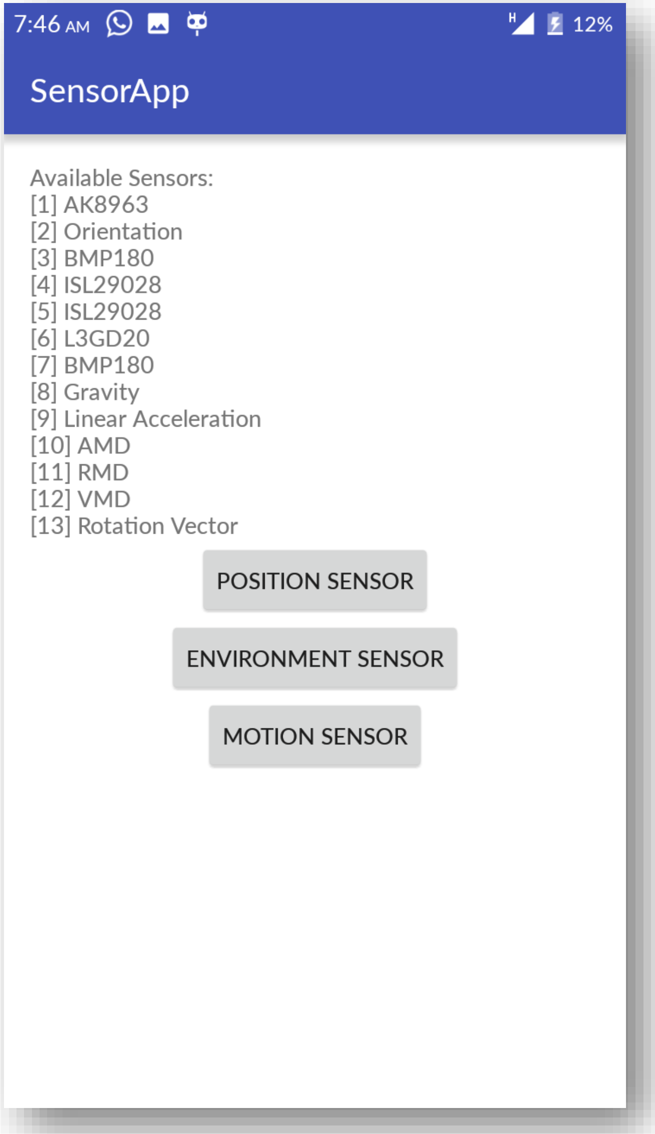
```
        mytextview.setVisibility(View.GONE);

        myposbutton = (Button)findViewById(R.id.button);
        myenvbutton = (Button)findViewById(R.id.button2);
        mymotbutton = (Button)findViewById(R.id.button3);

        mySensorManager = (SensorManager) getSystemService(SENSOR_SERVICE);
        myList= mySensorManager.getSensorList(Sensor.TYPE_ALL);

        mytextview.setVisibility(View.VISIBLE);
        for (int i = 1; i < myList.size(); i++) {
            mytextview.append("\n"+ "[" + Integer.toString(i) + "] "+ myList.get(i).getName());
        }

        myposbutton.setOnClickListener((view) → {
                Intent myposintent = new Intent(MainActivity.this,PositionActivity.class );
                startActivity(myposintent);
        });

        myenvbutton.setOnClickListener((view) → {
                Intent myenvintent = new Intent(MainActivity.this,EnvironmentActivity.class );
                startActivity(myenvintent);
        });

        mymotbutton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Intent mymotintent = new Intent(MainActivity.this,MotionActivity.class );
                startActivity(mymotintent);
            }
        });
    }
}
```



```java
package com.tekom.home.sensorapp;

import ...

public class MotionActivity extends AppCompatActivity implements SensorEventListener {
    private SensorManager mSensorManager;
    private Sensor myaccel;
    private TextView myxtextview;
    private TextView myytextview;
    private TextView myztextview;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_motion);

        mSensorManager = (SensorManager) getSystemService(Context.SENSOR_SERVICE);
        myaccel = mSensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER);

        myxtextview = (TextView)findViewById(R.id.textView3);
        myytextview = (TextView)findViewById(R.id.textView4);
        myztextview = (TextView)findViewById(R.id.textView5);
    }
    public void onSensorChanged(SensorEvent event){
        myxtextview.setText(Float.toString(event.values[0]));
        myytextview.setText(Float.toString(event.values[1]));
        myztextview.setText(Float.toString(event.values[2]));
    }
    @Override
    public final void onAccuracyChanged(Sensor sensor, int accuracy) {
        // Do something here if sensor accuracy changes.
    }
    @Override
    protected void onResume() {
        // Register a listener for the sensor.
        super.onResume();
        mSensorManager.registerListener(MotionActivity.this,myaccel,SensorManager.SENSOR_DELAY_NORMAL);
    }
    @Override
    protected void onPause() {
        // Be sure to unregister the sensor when the activity pauses.
        super.onPause();
        mSensorManager.unregisterListener(MotionActivity.this);
    }
}
```

And then Modify the MotionActivity java file

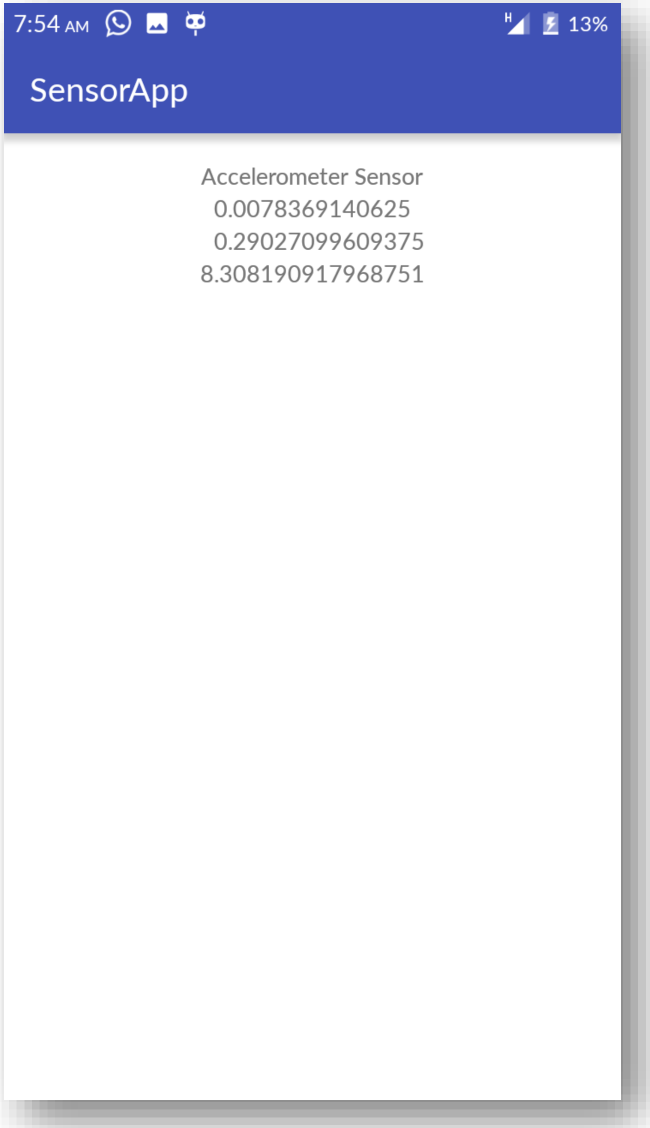
Result



7:46 AM 12%

SensorApp

```
Available Sensors:
[1] AK8963
[2] Orientation
[3] BMP180
[4] ISL29028
[5] ISL29028
[6] L3GD20
[7] BMP180
[8] Gravity
[9] Linear Acceleration
[10] AMD
[11] RMD
[12] VMD
[13] Rotation Vector
```



7:54 AM 13%

SensorApp

```
Accelerometer Sensor
0.0078369140625
0.29027099609375
8.308190917968751
```

Best Practices for Accessing and Using Sensors

1. Verify sensors before you use them
2. Unregister sensor listeners when the sensor activity pauses
3. Choose sensor delays carefully
4. Don’t block the onSensorChanged() method

    *Sensor data can change at a high rate, which means the system may call the onSensorChanged(SensorEvent) method quite often. As a best practice, **you should do as little as possible within the onSensorChanged(SensorEvent) method so you don't block it**.*

5. Don't test your code on the emulator

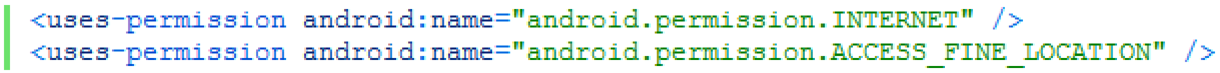
GPS sensor

Permissions for accessing Location:

To run our GPS Location Manager application, we need to provide the permissions given below.

***ACCESS_FINE_LOCATION:* This permission will give the application access to the GPS location coordinates.**

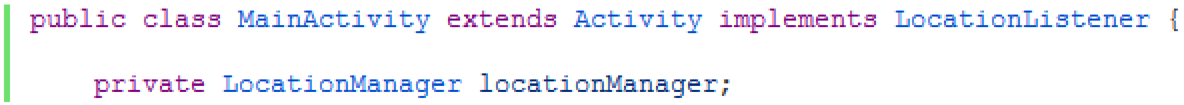
***INTERNET:* This permission will allow the application to use the Internet. Add the lines of code below to your Android manifest file**



```xml
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
```

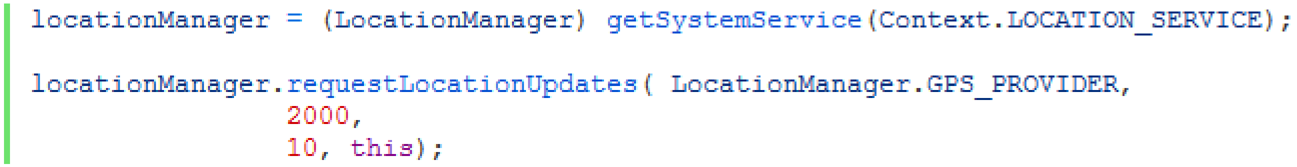
Working with the java file

Implement a LocationListener and make a global object for the LocationManager and implement all the unimplemented methods



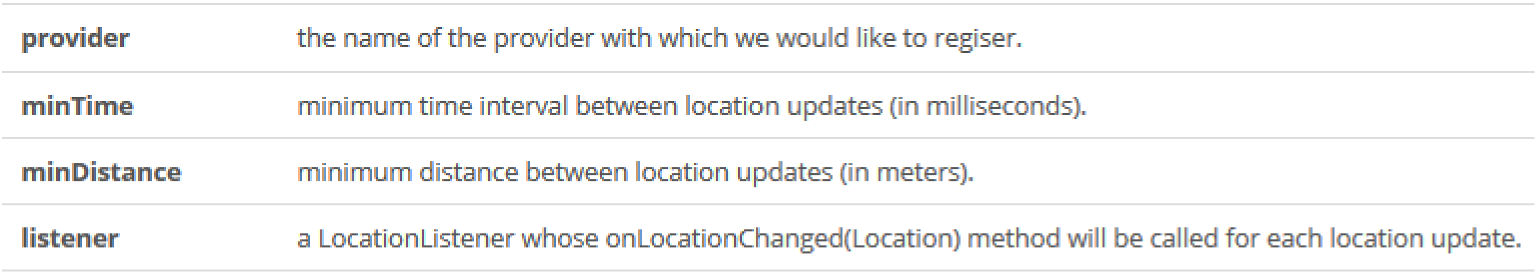
```
public class MainActivity extends Activity implements LocationListener {

    private LocationManager locationManager;
```

We will need to call the method requestLocationUpdates to get the current location as it’s updated by the user.



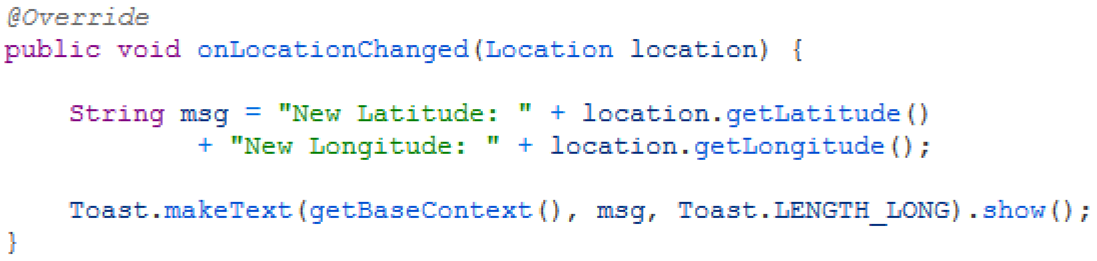
```
locationManager = (LocationManager) getSystemService(Context.LOCATION_SERVICE);

locationManager.requestLocationUpdates( LocationManager.GPS_PROVIDER,
                2000,
                10, this);
```

The parameters of this function are as follows:



provider	the name of the provider with which we would like to regiser.
minTime	minimum time interval between location updates (in milliseconds).
minDistance	minimum distance between location updates (in meters).
listener	a LocationListener whose onLocationChanged(Location) method will be called for each location update.

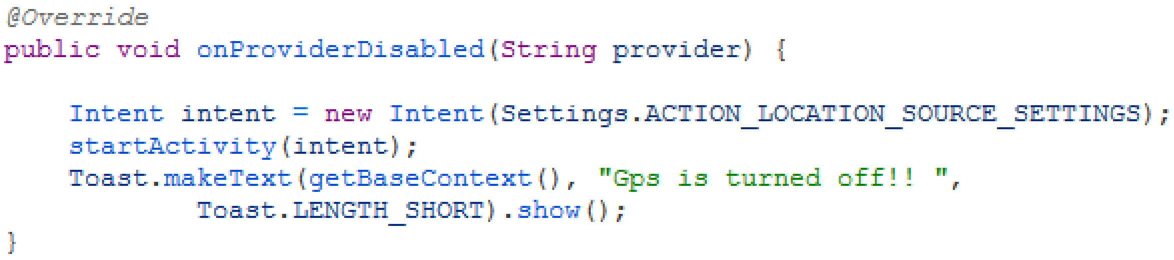
Use the below code to print the location.



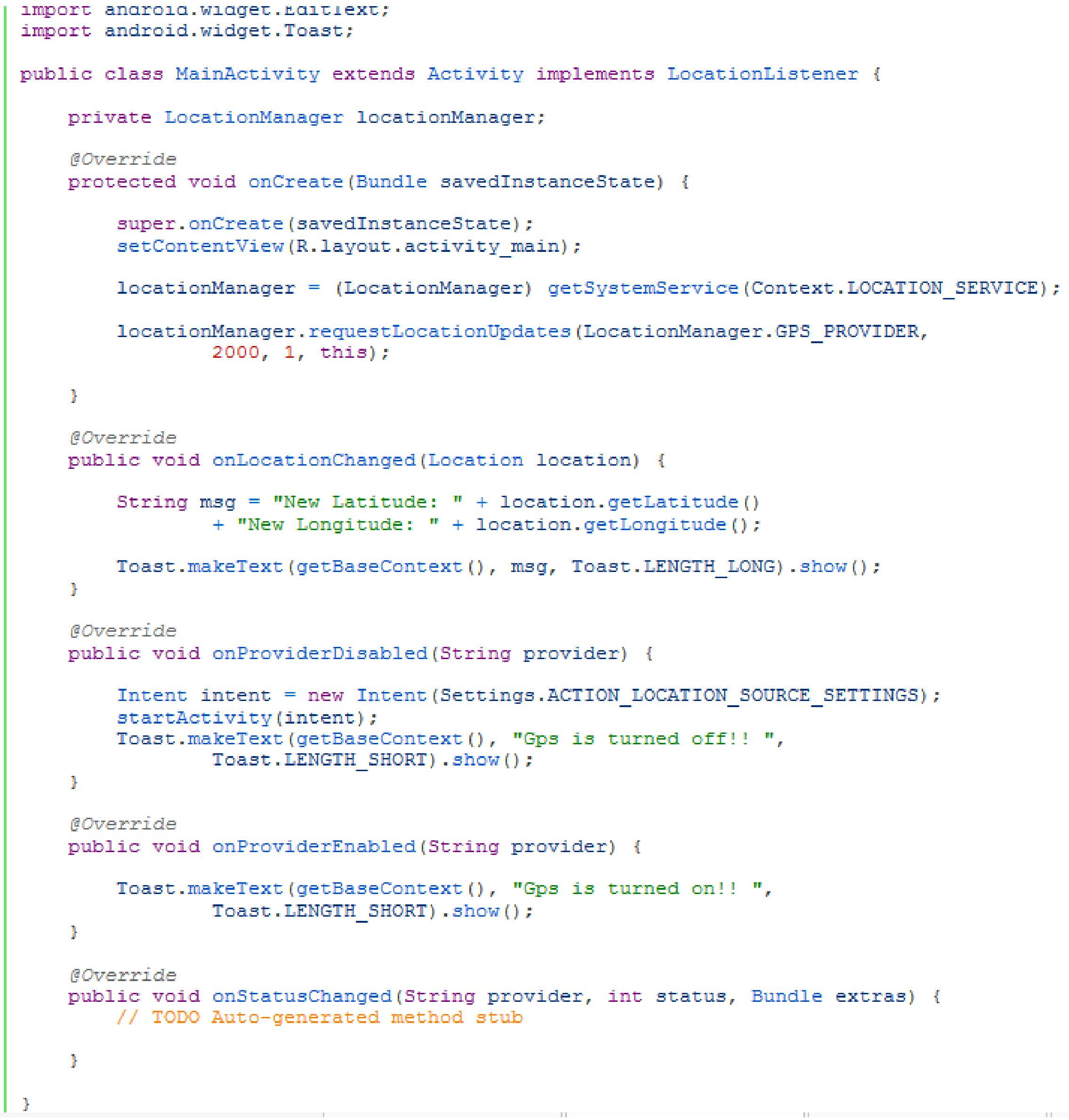
```
@Override
public void onLocationChanged(Location location) {

    String msg = "New Latitude: " + location.getLatitude()
            + "New Longitude: " + location.getLongitude();

    Toast.makeText(getBaseContext(), msg, Toast.LENGTH_LONG).show();
}
```

But what if the GPS is not enabled?

We can handle this event in the **onProviderDisabled** function. We need to redirect our application to the Location settings of the device if the GPS has been disabled.

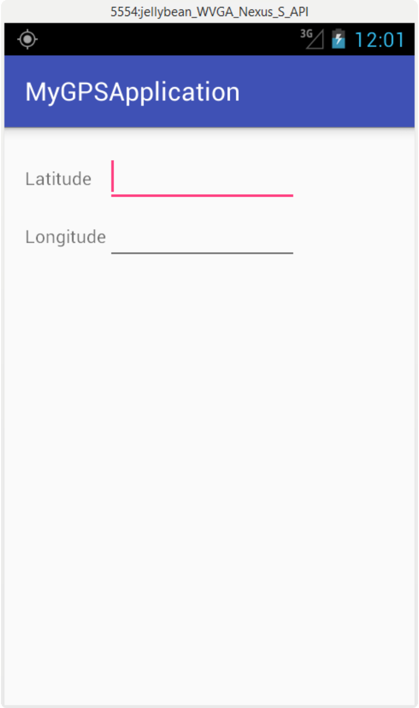


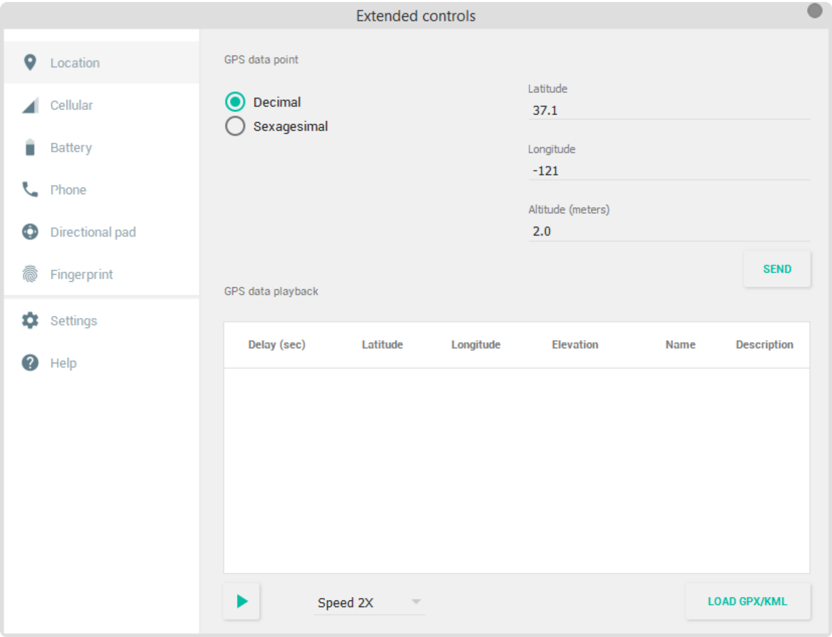
```
@Override
public void onProviderDisabled(String provider) {

    Intent intent = new Intent(Settings.ACTION_LOCATION_SOURCE_SETTINGS);
    startActivity(intent);
    Toast.makeText(getBaseContext(), "Gps is turned off!! ",
            Toast.LENGTH_SHORT).show();
}
```



```
import android.widget.EditText;
import android.widget.Toast;

public class MainActivity extends Activity implements LocationListener {

    private LocationManager locationManager;

    @Override
    protected void onCreate(Bundle savedInstanceState) {

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

        locationManager = (LocationManager) getSystemService(Context.LOCATION_SERVICE);

        locationManager.requestLocationUpdates(LocationManager.GPS_PROVIDER,
                2000, 1, this);

    }

    @Override
    public void onLocationChanged(Location location) {

        String msg = "New Latitude: " + location.getLatitude()
                + "New Longitude: " + location.getLongitude();

        Toast.makeText(getBaseContext(), msg, Toast.LENGTH_LONG).show();
    }

    @Override
    public void onProviderDisabled(String provider) {

        Intent intent = new Intent(Settings.ACTION_LOCATION_SOURCE_SETTINGS);
        startActivity(intent);
        Toast.makeText(getBaseContext(), "Gps is turned off!! ",
                Toast.LENGTH_SHORT).show();
    }

    @Override
    public void onProviderEnabled(String provider) {

        Toast.makeText(getBaseContext(), "Gps is turned on!! ",
                Toast.LENGTH_SHORT).show();
    }

    @Override
    public void onStatusChanged(String provider, int status, Bundle extras) {
        // TODO Auto-generated method stub

    }

}
```

Testing (using AVD)


5554:jellybean_WVGA_Nexus_S_API


Extended controls

GPS data point

● Decimal
○ Sexagesimal

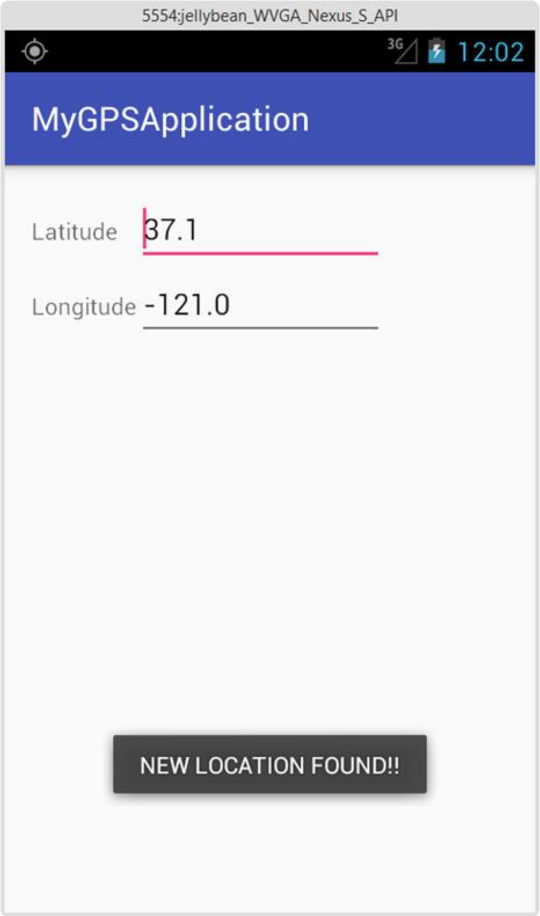
Latitude: 37.1

Longitude: -121

Altitude (meters): 2.0

GPS data playback

Delay (sec)	Latitude	Longitude	Elevation	Name	Description


5554:jellybean_WVGA_Nexus_S_API

Latitude 37.1

Longitude -121.0

NEW LOCATION FOUND!!

TERIMA KASIH