Sensores

Créditos

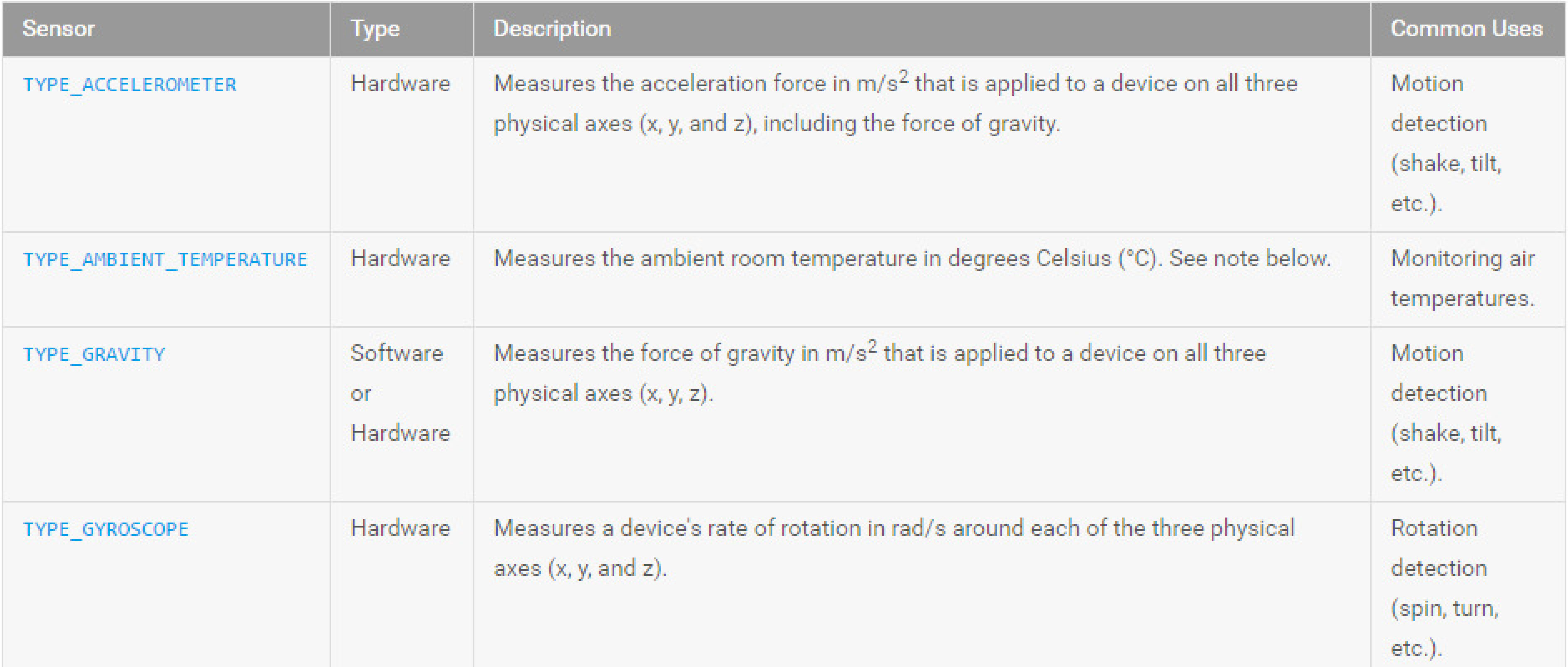
- Tutorial https://developer.android.com/guide/topics/sensors/sensors_overview.html[1]

[1] Content is licensed under Creative Commons Attribution 2.5

Clases de sensores

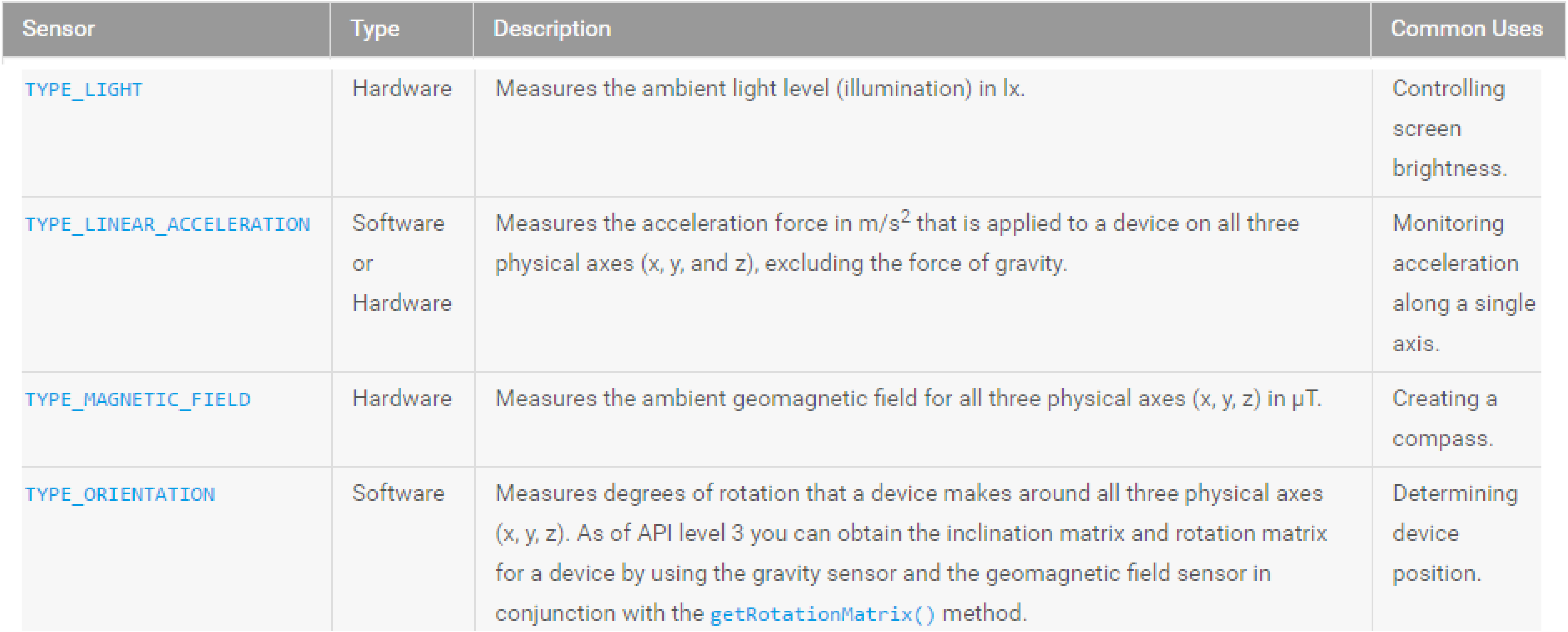
- Movimiento. Miden fuerzas de aceleración y rotacionales sobre 3 ejes. Incluyen sensor de gravedad, acelerómetro, giroscopio, etc.
- Medio ambiente. Miden parámetros del medio ambiente. Incluyen termómetro, barómetro, fotómetro, etc.
- Posición. Miden la posición física del dispositivo. Incluyen sensor de orientación, magnetómetros, etc.

Sensores soportados



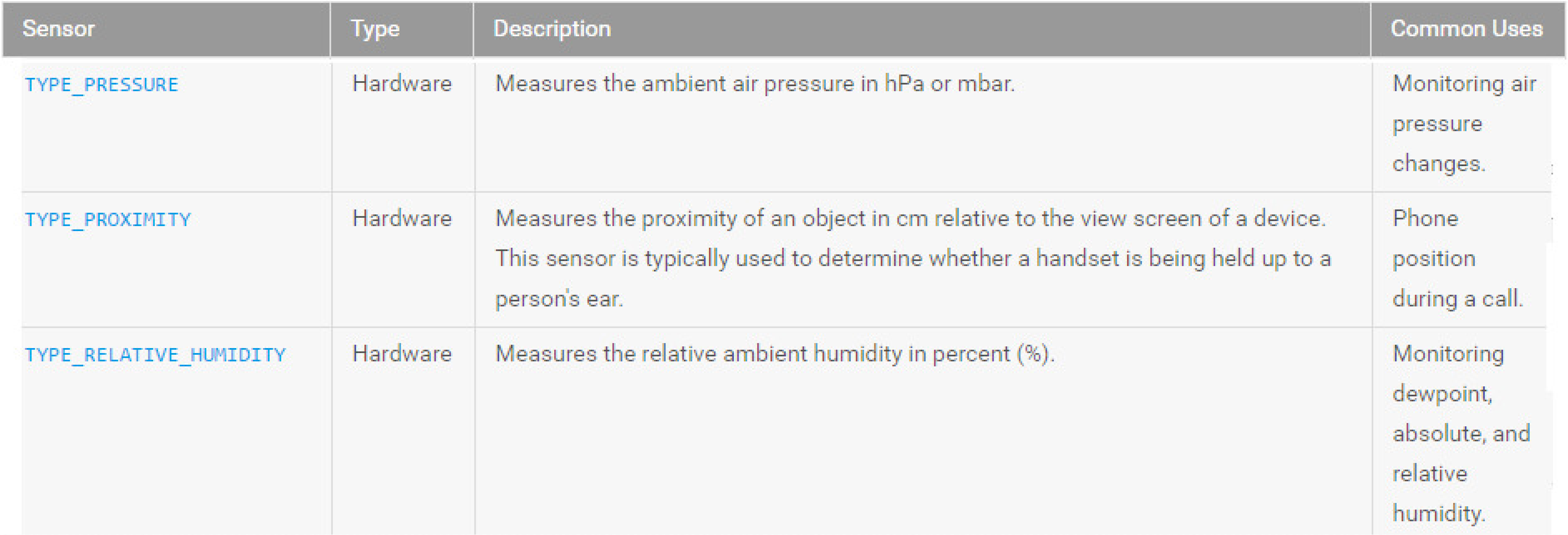
Sensor	Type	Description	Common Uses
`TYPE_ACCELEROMETER`	Hardware	Measures the acceleration force in m/s^2 that is applied to a device on all three physical axes (x, y, and z), including the force of gravity.	Motion detection (shake, tilt, etc.).
`TYPE_AMBIENT_TEMPERATURE`	Hardware	Measures the ambient room temperature in degrees Celsius (°C). See note below.	Monitoring air temperatures.
`TYPE_GRAVITY`	Software or Hardware	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`	Hardware	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.).

Sensores soportados



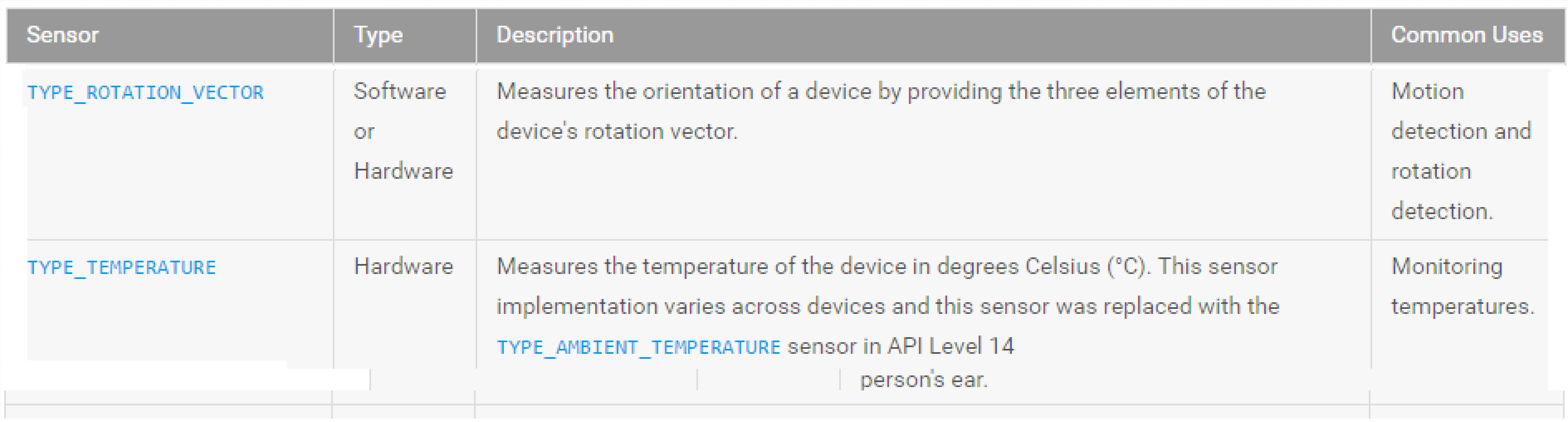
Sensor	Type	Description	Common Uses
`TYPE_LIGHT`	Hardware	Measures the ambient light level (illumination) in lx.	Controlling screen brightness.
`TYPE_LINEAR_ACCELERATION`	Software or Hardware	Measures the acceleration force in m/s^2 that is applied to a device on all three physical axes (x, y, and z), excluding the force of gravity.	Monitoring acceleration along a single axis.
`TYPE_MAGNETIC_FIELD`	Hardware	Measures the ambient geomagnetic field for all three physical axes (x, y, z) in μT.	Creating a compass.
`TYPE_ORIENTATION`	Software	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.

Sensores soportados



Sensor	Type	Description	Common Uses
`TYPE_PRESSURE`	Hardware	Measures the ambient air pressure in hPa or mbar.	Monitoring air pressure changes.
`TYPE_PROXIMITY`	Hardware	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`	Hardware	Measures the relative ambient humidity in percent (%).	Monitoring dewpoint, absolute, and relative humidity.

Sensores soportados



Sensor	Type	Description	Common Uses
`TYPE_ROTATION_VECTOR`	Software or Hardware	Measures the orientation of a device by providing the three elements of the device's rotation vector.	Motion detection and rotation detection.
`TYPE_TEMPERATURE`	Hardware	Measures the temperature of the device in degrees Celsius (°C). This sensor implementation varies across devices and this sensor was replaced with the `TYPE_AMBIENT_TEMPERATURE` sensor in API Level 14	Monitoring temperatures.

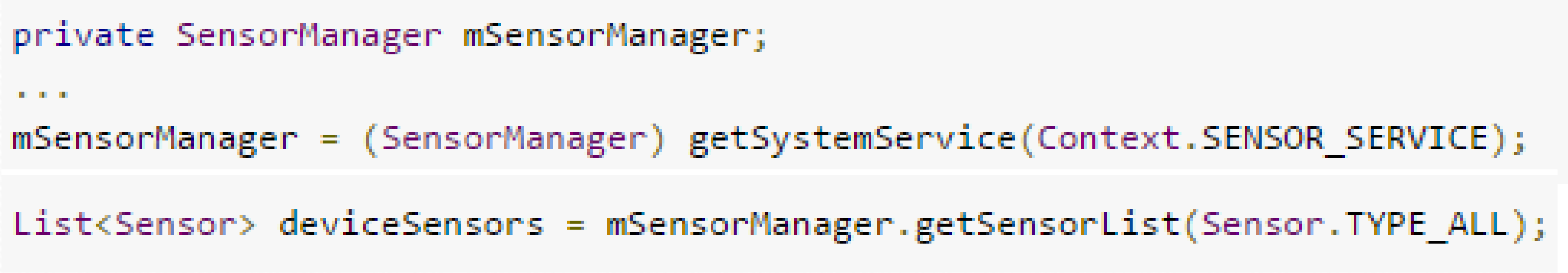
person's ear.

Framework de sensores

- *SensorManager*. Servicio de sensores.
- *Sensor*. Sensores específicos.
- *SensorEvent*. Información sobre eventos de sensor.
- *SensorEventListener*. Callbacks para recibir eventos de sensor.

Identificar sensores

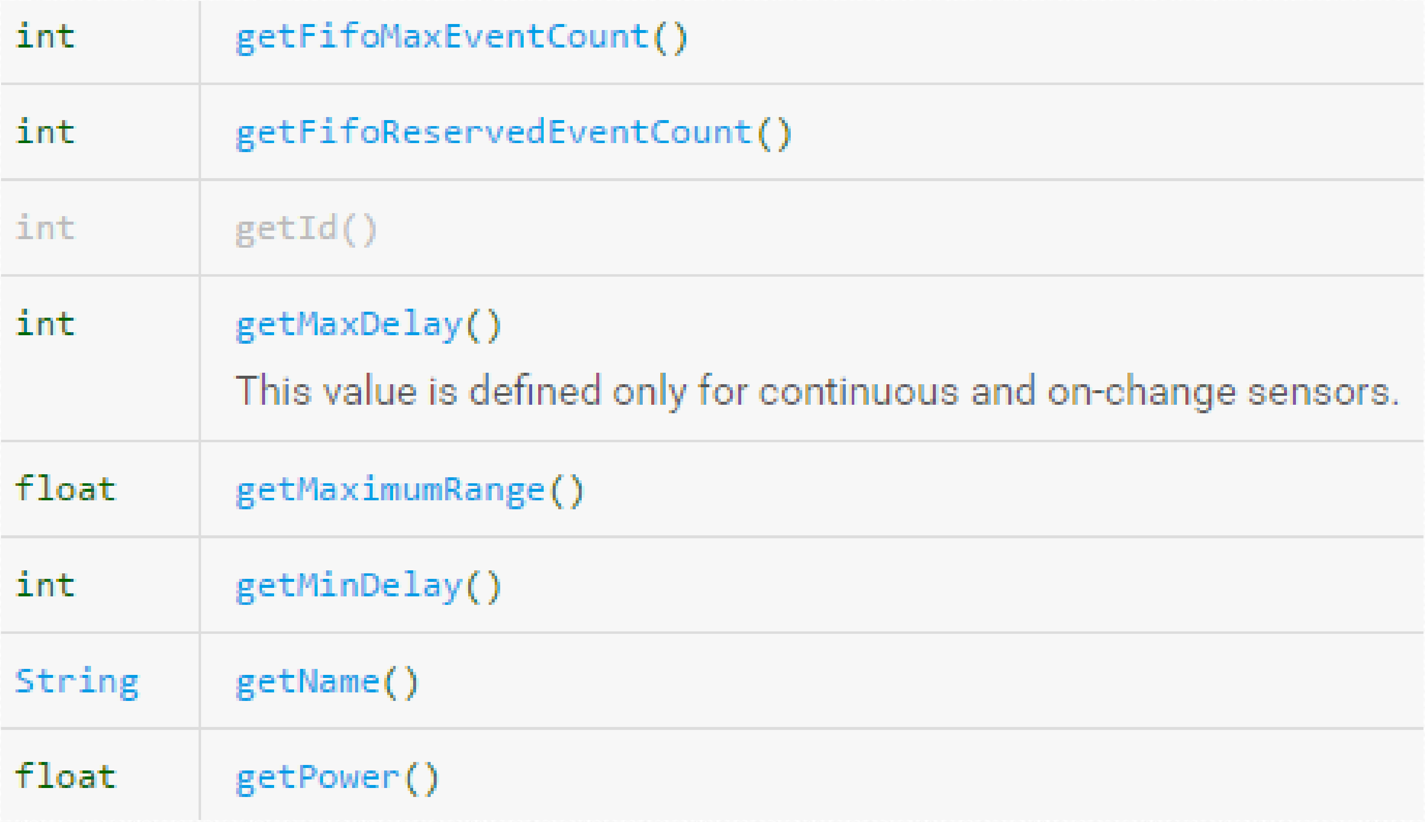
- Una lista con todos los sensores.



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

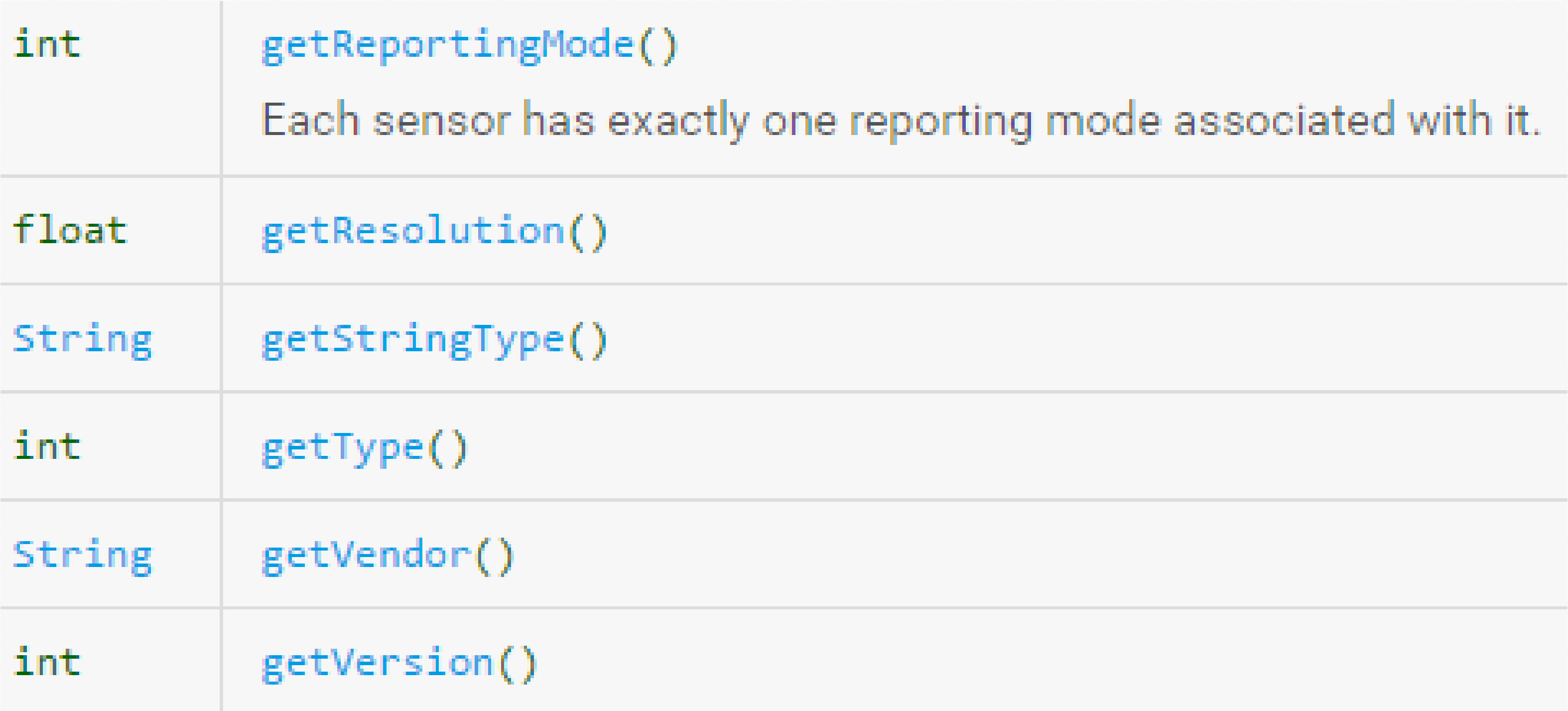
```
List<Sensor> deviceSensors = mSensorManager.getSensorList(Sensor.TYPE_ALL);
```

Métodos de *Sensor*



`int`	`getFifoMaxEventCount()`
`int`	`getFifoReservedEventCount()`
`int`	`getId()`
`int`	`getMaxDelay()` This value is defined only for continuous and on-change sensors.
`float`	`getMaximumRange()`
`int`	`getMinDelay()`
`String`	`getName()`
`float`	`getPower()`

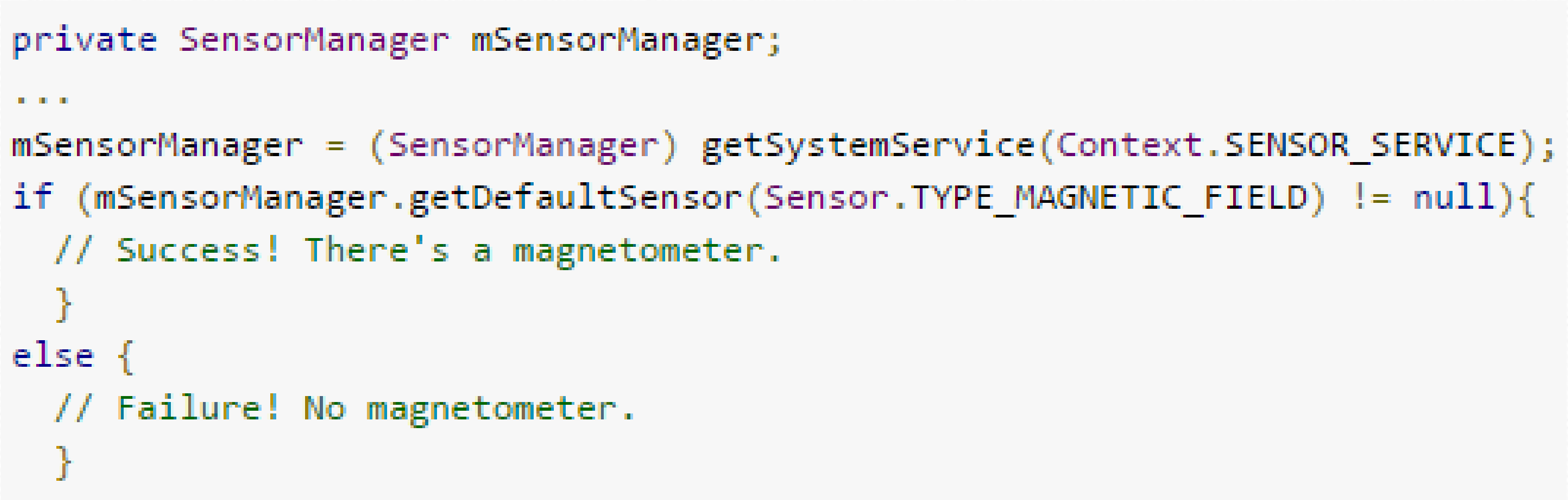
Métodos de sensor



`int`	`getReportingMode()` Each sensor has exactly one reporting mode associated with it.
`float`	`getResolution()`
`String`	`getStringType()`
`int`	`getType()`
`String`	`getVendor()`
`int`	`getVersion()`

Detectar un sensor en particular

- Puede haber más de un sensor del mismo tipo.
- Uno de ellos el sensor por default.
- Usar *getDefaultSensor*()



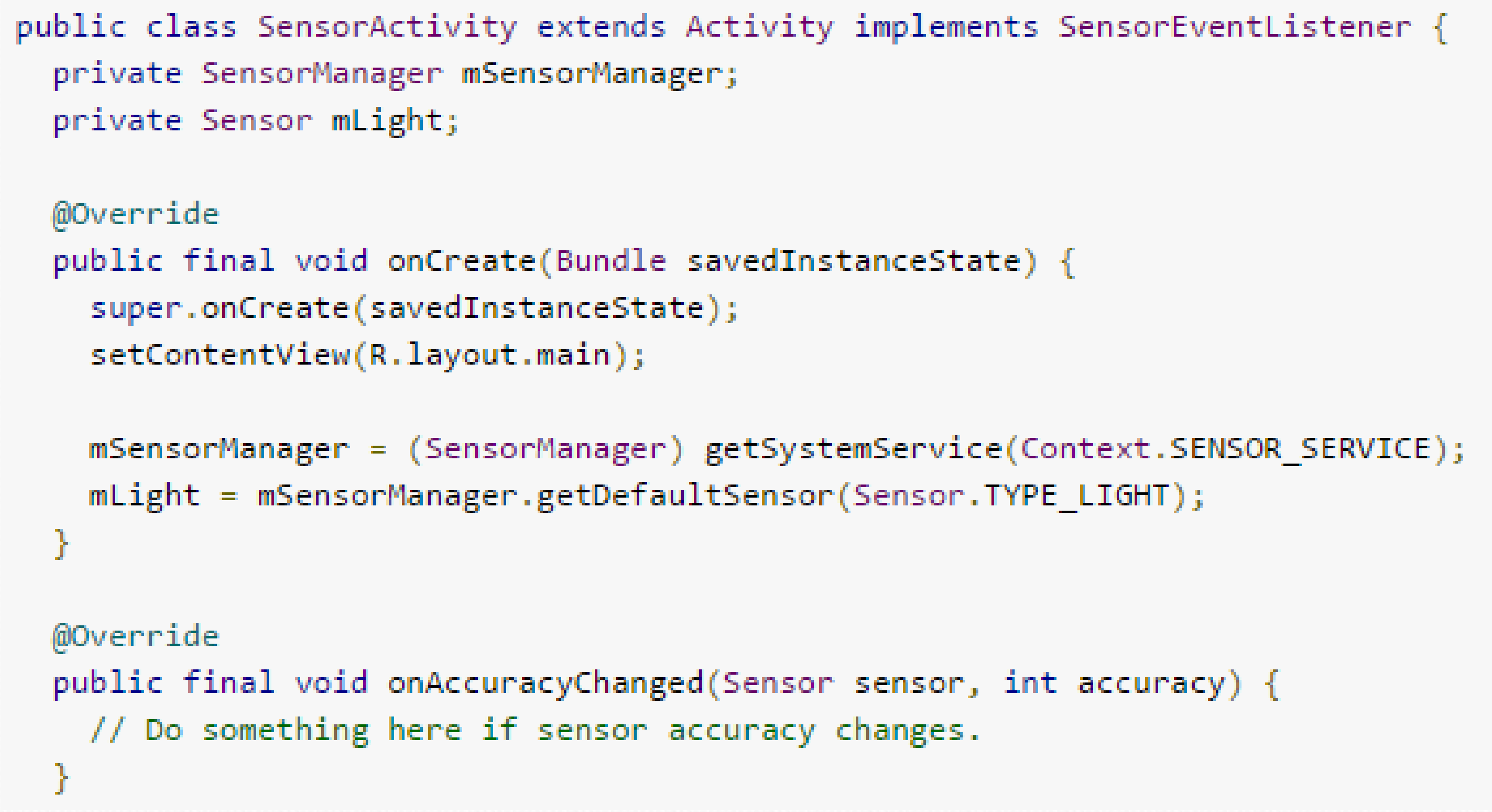
```
private SensorManager mSensorManager;
...
mSensorManager = (SensorManager) getSystemService(Context.SENSOR_SERVICE);
if (mSensorManager.getDefaultSensor(Sensor.TYPE_MAGNETIC_FIELD) != null){
  // Success! There's a magnetometer.
  }
else {
  // Failure! No magnetometer.
  }
```

Eventos de sensor

- Implementar *SensorEventListener*.
- Implementar dos métodos:
  - *onSensorChanged*() – el sensor detectó un valor nuevo y la información viene en una variable de tipo *SensorEvent*.
  - *onAccuracyChanged*() – el sensor cambió su precisión.

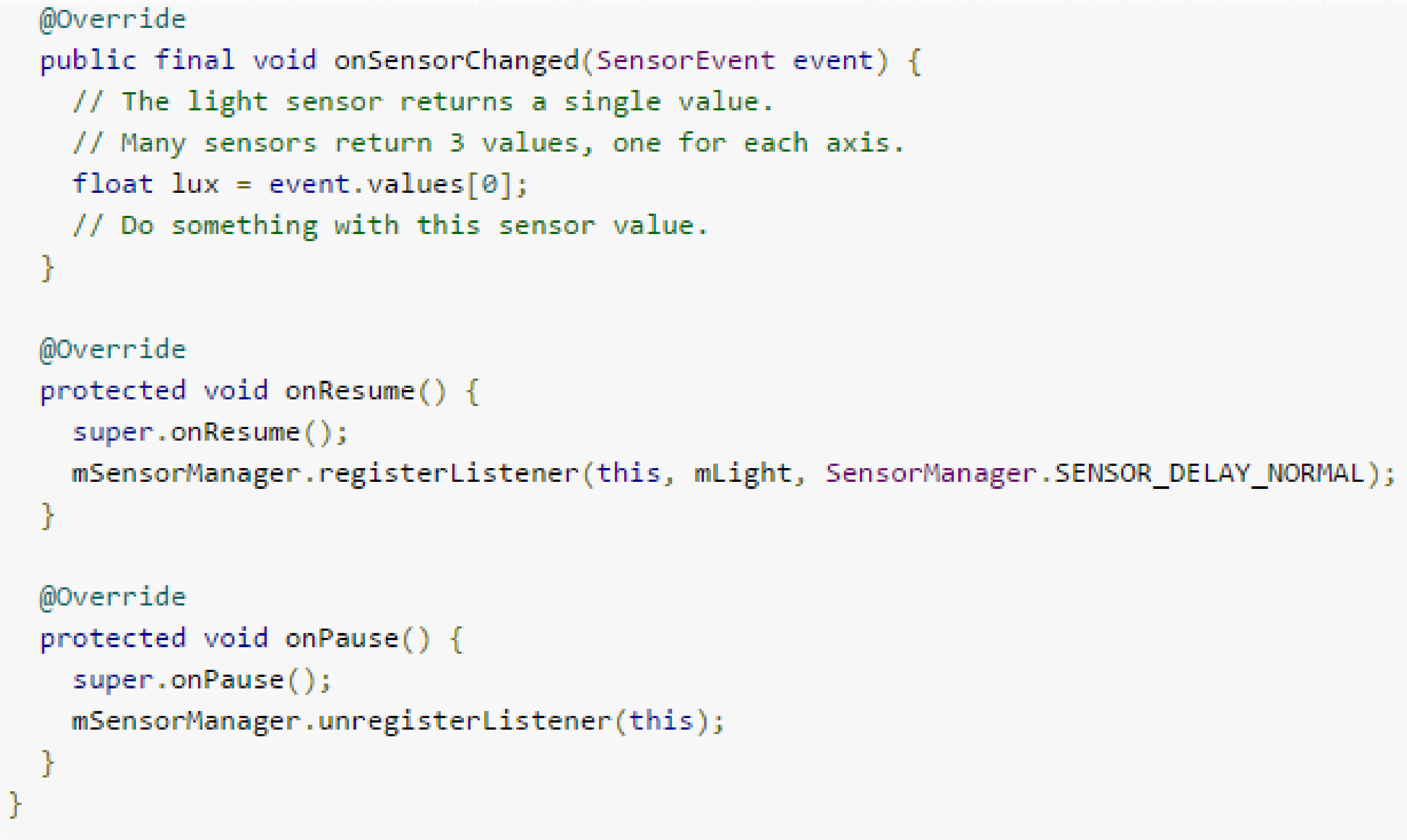
    SENSOR_STATUS_ACCURACY_LOW, *SENSOR_STATUS_ACCURACY_MEDIUM*, *SENSOR_STATUS_ACCURACY_HIGH*, *SENSOR_STATUS_UNRELIABLE*.

Ejemplo: sensor de luz



```
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.
  }
```

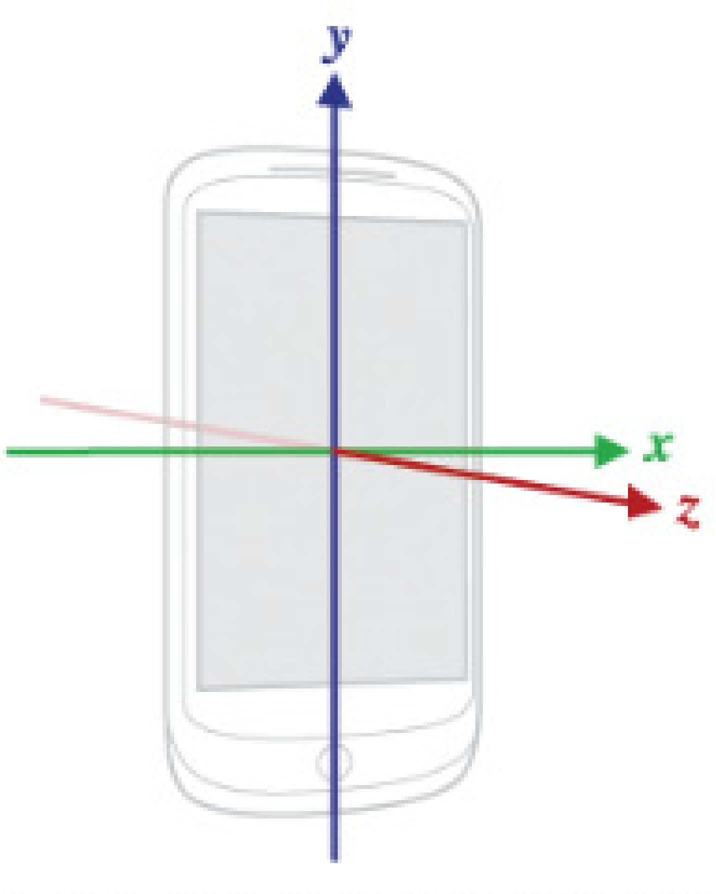
Ejemplo: sensor de luz



```
  @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);
  }
}
```

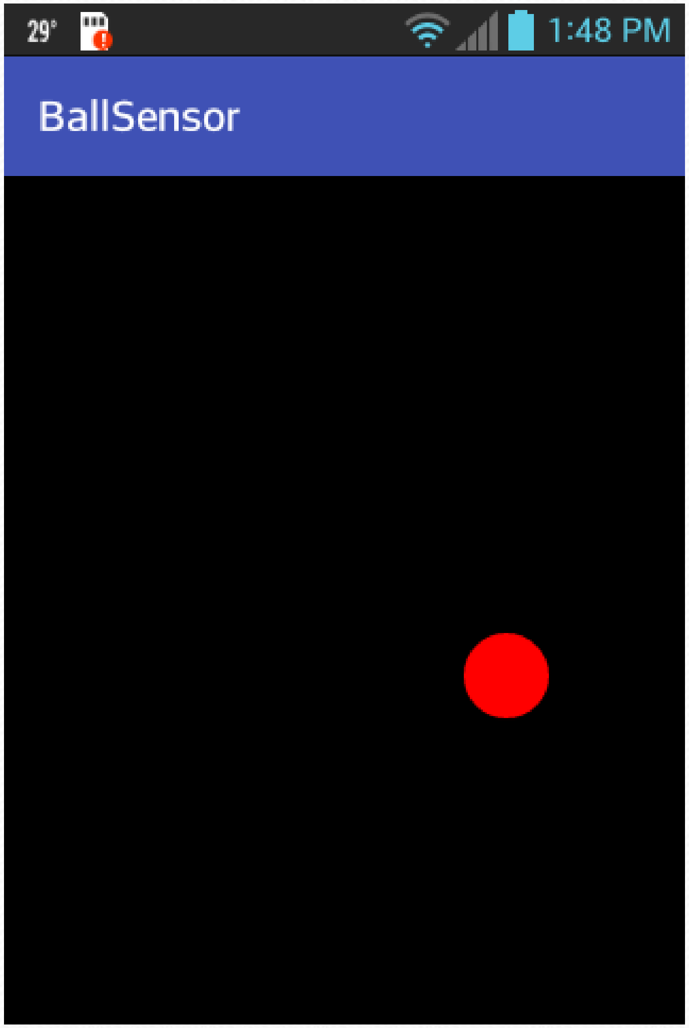
Sistema de coordenadas

- Acceleration sensor
- Gravity sensor
- Gyroscope
- Linear acceleration sensor
- Geomagnetic field sensor

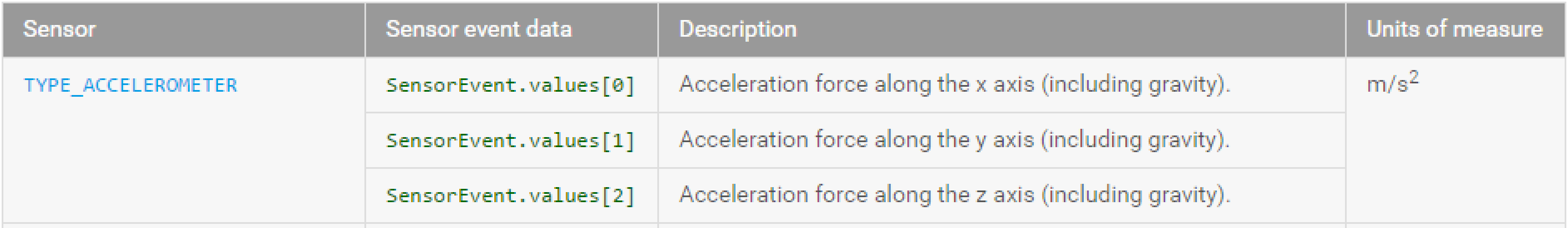


Ejemplo

- Usar el sensor de aceleración para mover una pelota.

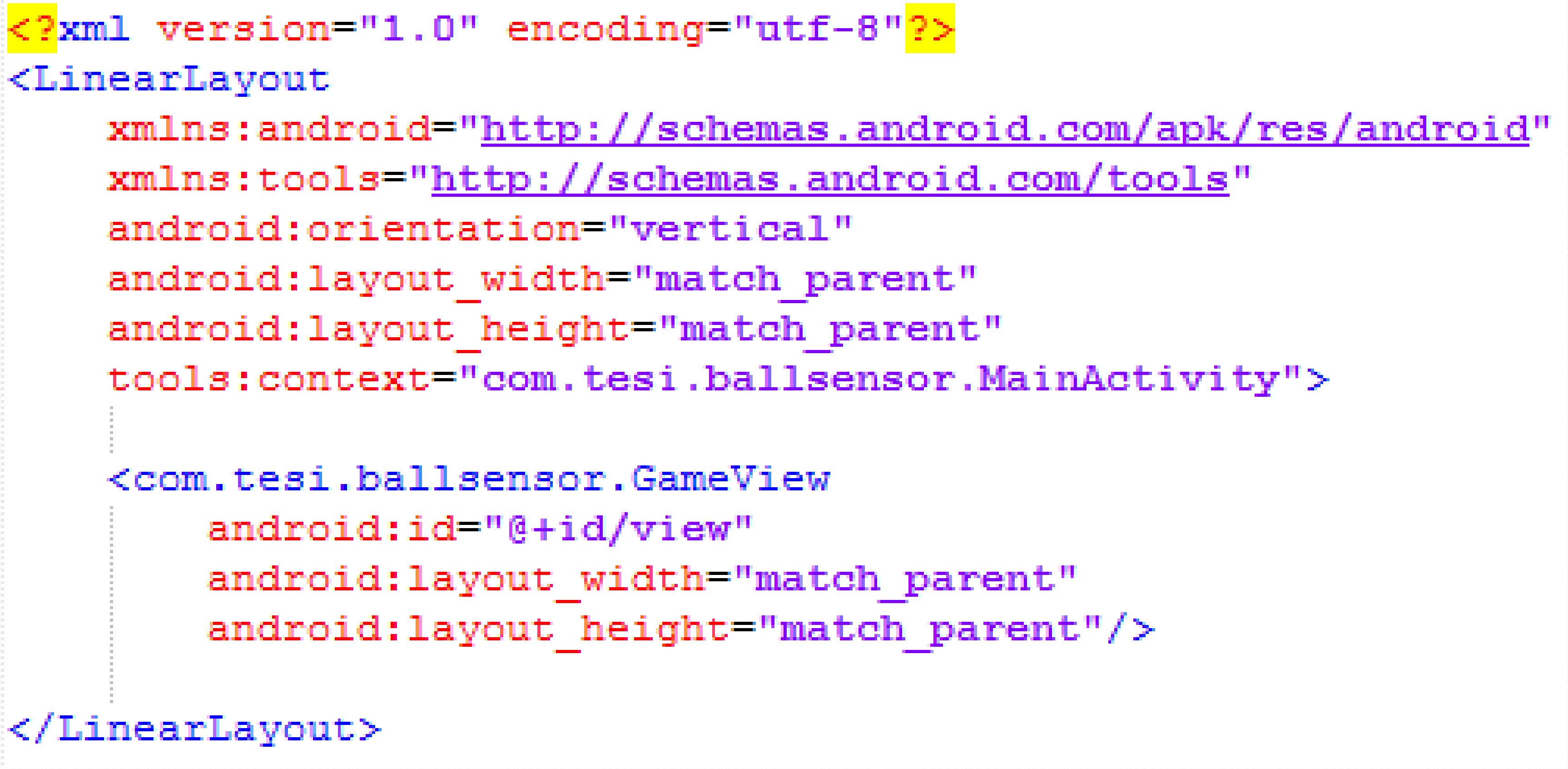

29°
1:48 PM
BallSensor

Sensor de aceleración



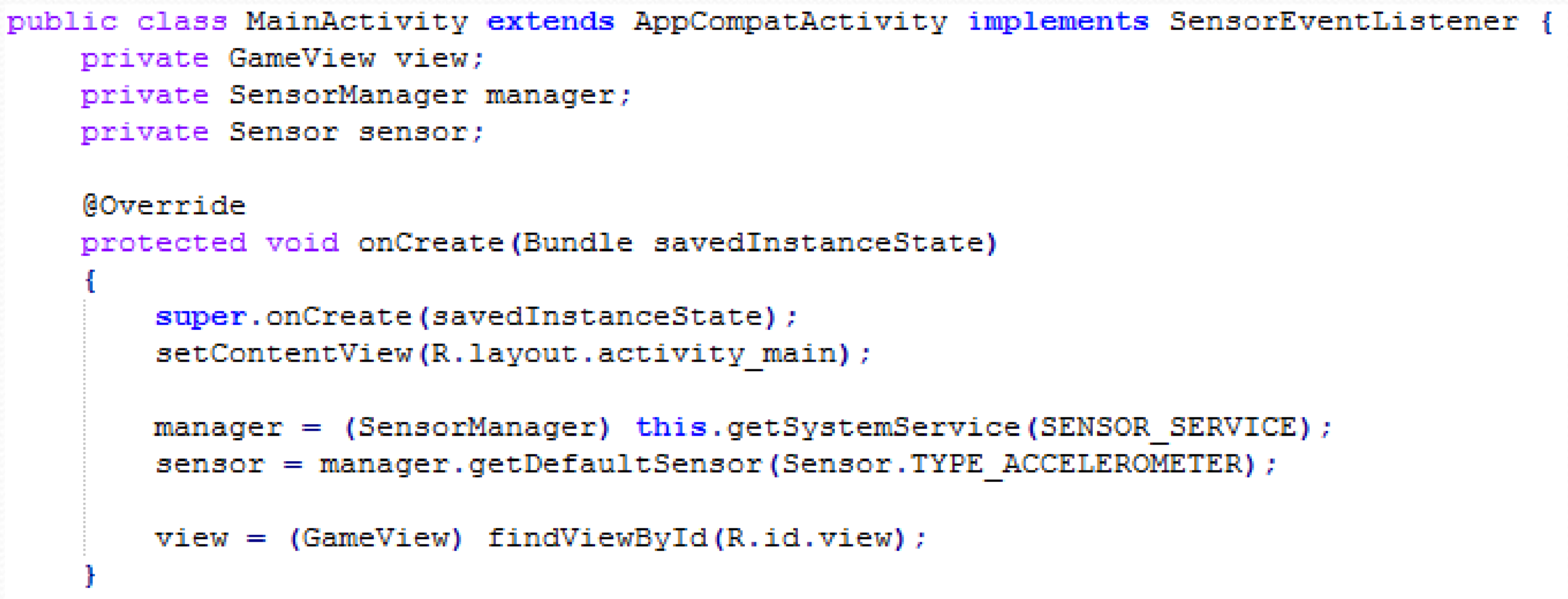
Sensor	Sensor event data	Description	Units of measure
`TYPE_ACCELEROMETER`	`SensorEvent.values[0]`	Acceleration force along the x axis (including gravity).	m/s^2
	`SensorEvent.values[1]`	Acceleration force along the y axis (including gravity).	
	`SensorEvent.values[2]`	Acceleration force along the z axis (including gravity).	

Layout



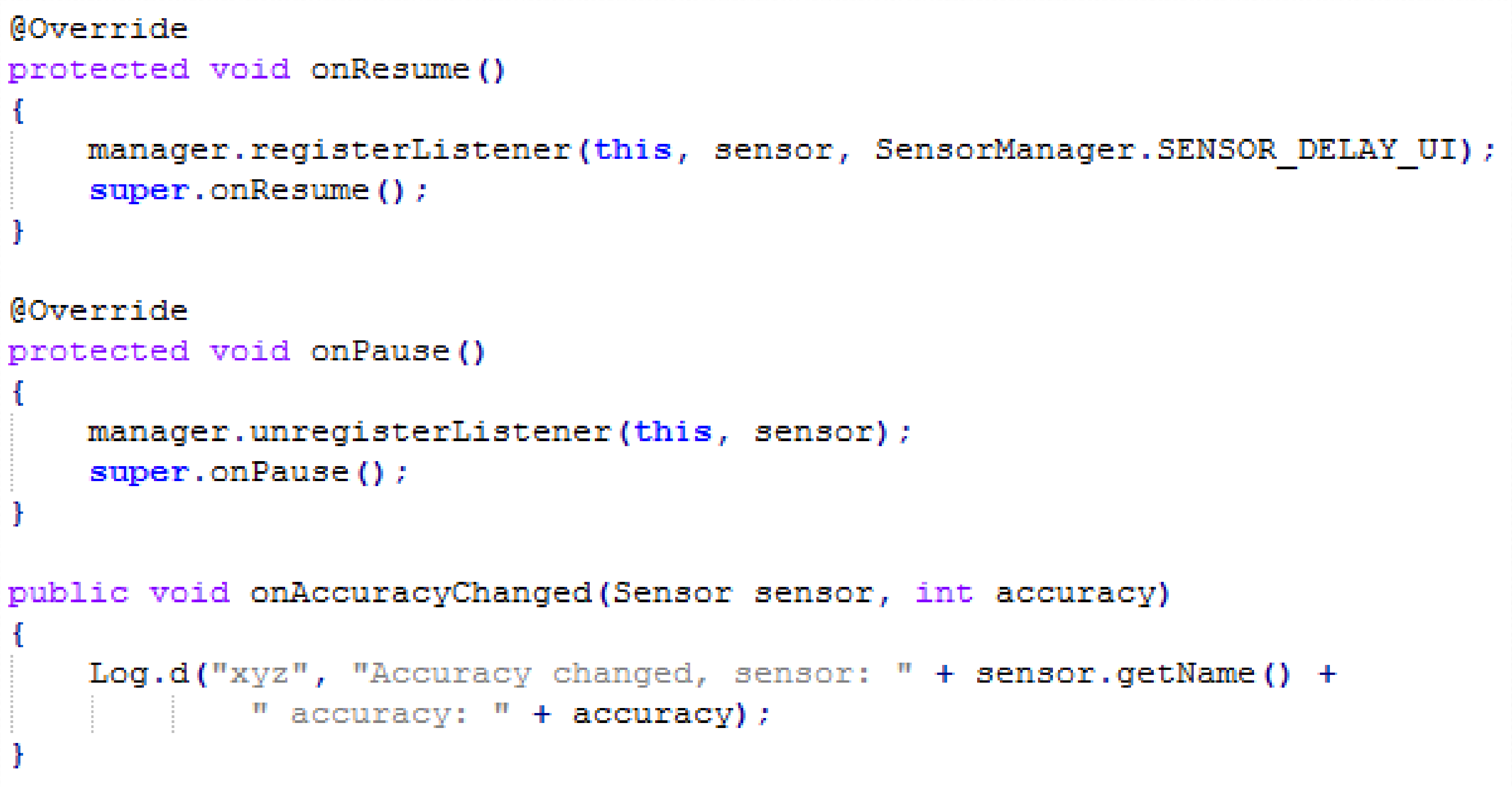
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="com.tesi.ballsensor.MainActivity">

    <com.tesi.ballsensor.GameView
        android:id="@+id/view"
        android:layout_width="match_parent"
        android:layout_height="match_parent"/>

</LinearLayout>
```

Actividad principal



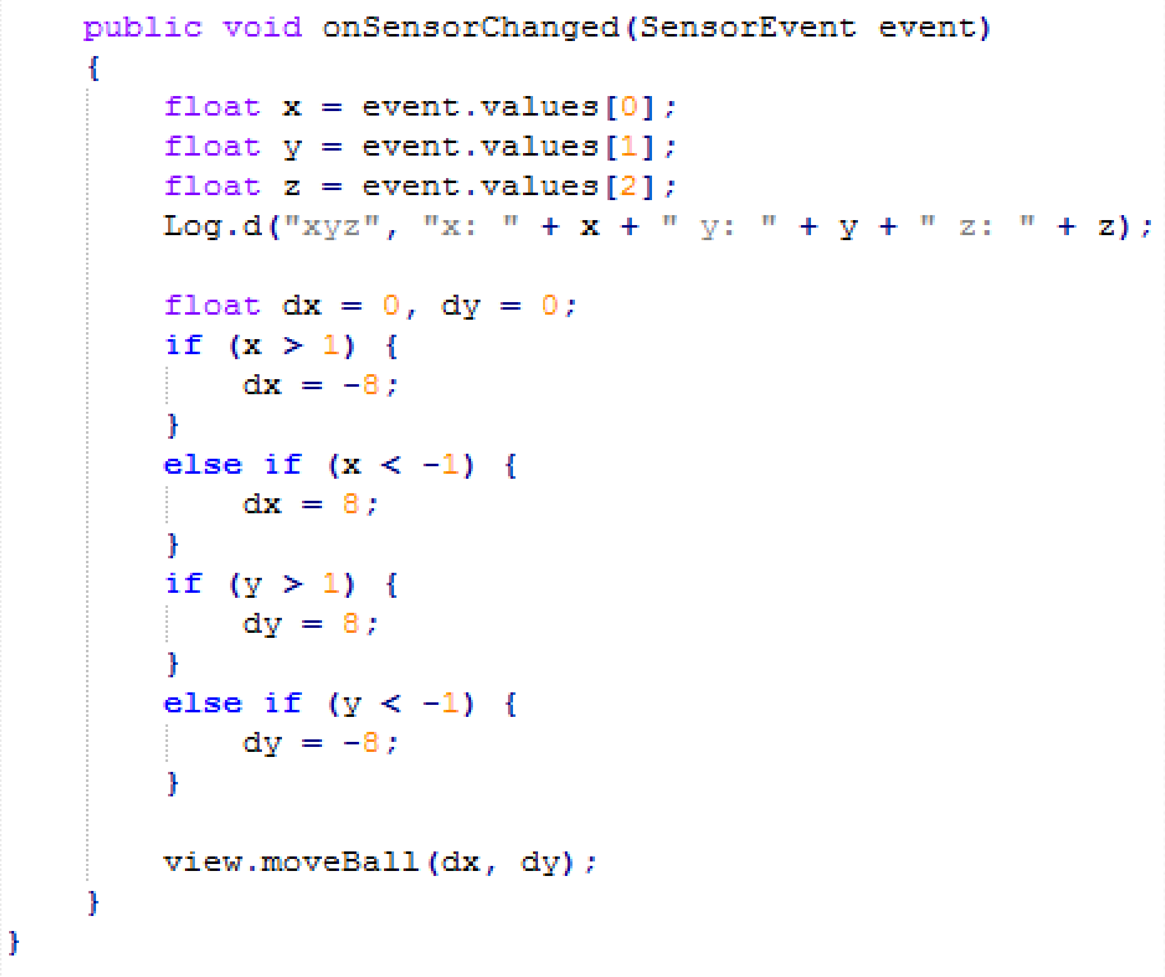
```
public class MainActivity extends AppCompatActivity implements SensorEventListener {
    private GameView view;
    private SensorManager manager;
    private Sensor sensor;

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

        manager = (SensorManager) this.getSystemService(SENSOR_SERVICE);
        sensor = manager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER);

        view = (GameView) findViewById(R.id.view);
    }
```

Actividad principal



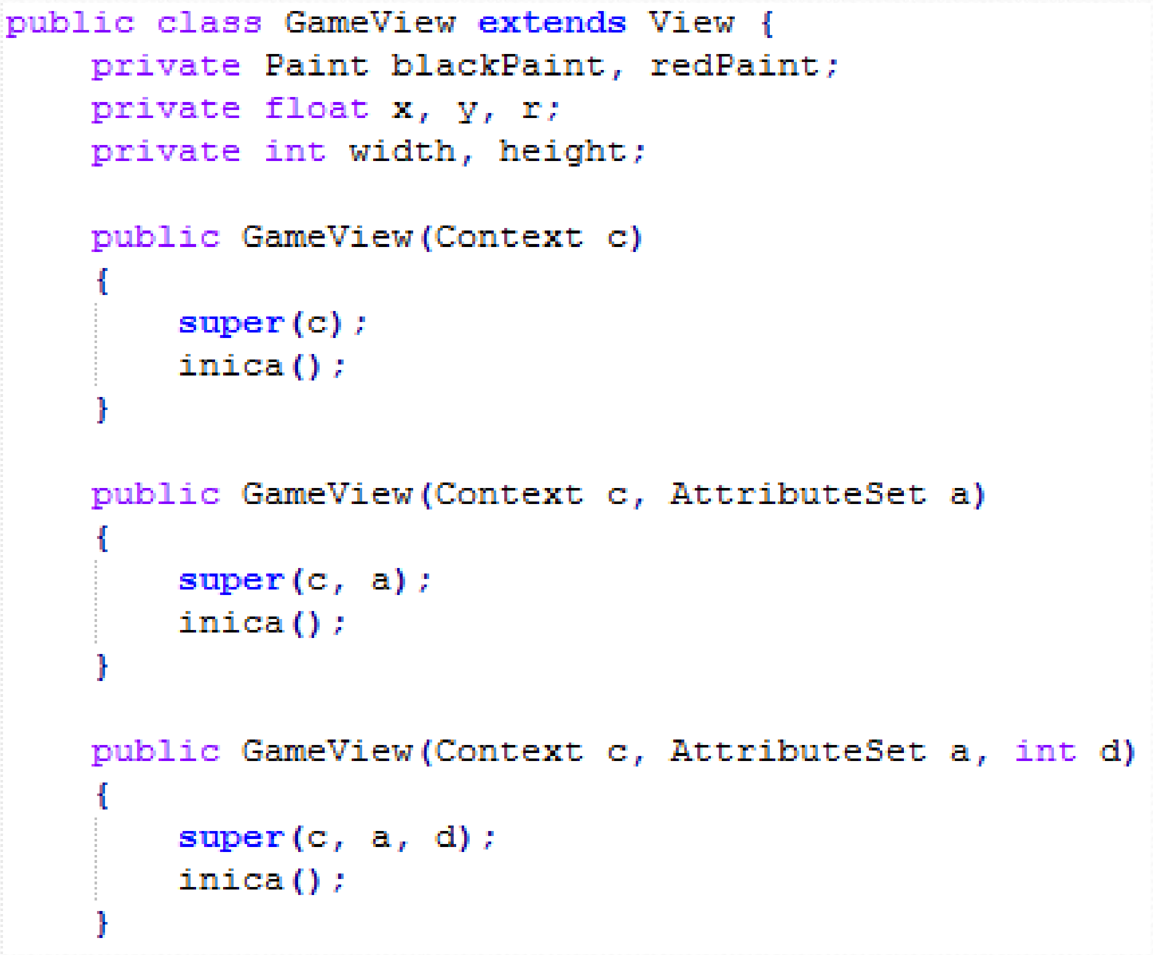
```
@Override
protected void onResume()
{
    manager.registerListener(this, sensor, SensorManager.SENSOR_DELAY_UI);
    super.onResume();
}

@Override
protected void onPause()
{
    manager.unregisterListener(this, sensor);
    super.onPause();
}

public void onAccuracyChanged(Sensor sensor, int accuracy)
{
    Log.d("xyz", "Accuracy changed, sensor: " + sensor.getName() +
            " accuracy: " + accuracy);
}
```

Actividad principal



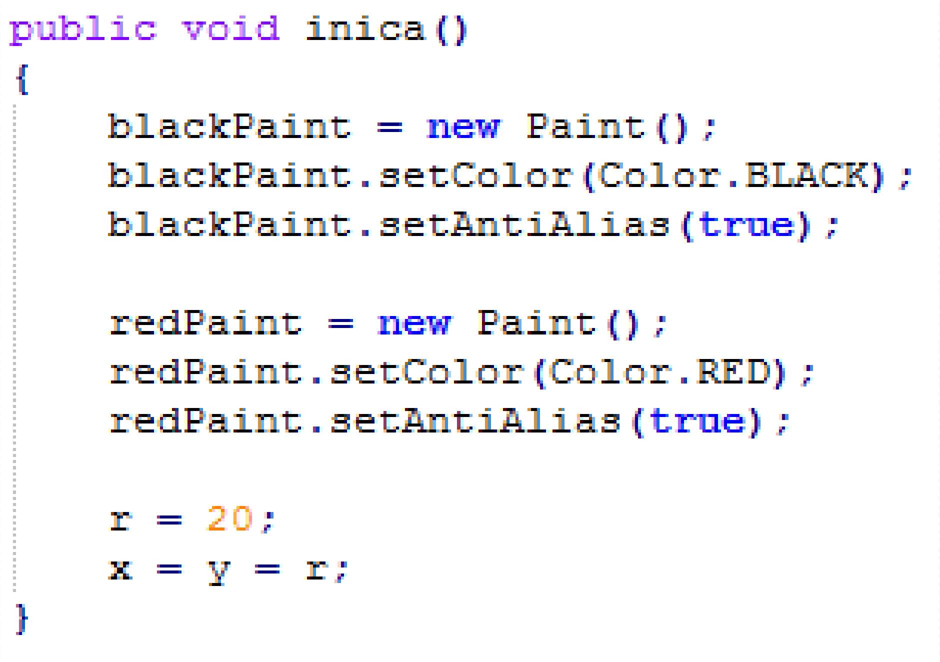
```
    public void onSensorChanged(SensorEvent event)
    {
        float x = event.values[0];
        float y = event.values[1];
        float z = event.values[2];
        Log.d("xyz", "x: " + x + " y: " + y + " z: " + z);

        float dx = 0, dy = 0;
        if (x > 1) {
            dx = -8;
        }
        else if (x < -1) {
            dx = 8;
        }
        if (y > 1) {
            dy = 8;
        }
        else if (y < -1) {
            dy = -8;
        }

        view.moveBall(dx, dy);
    }
}
```

GameView



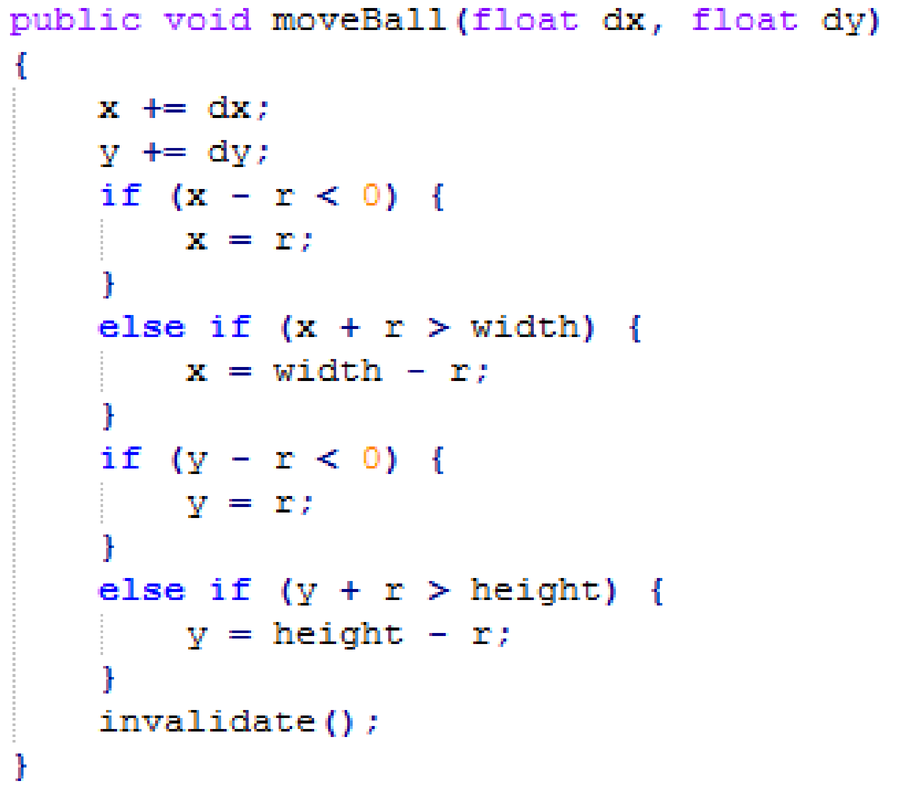
```
public class GameView extends View {
    private Paint blackPaint, redPaint;
    private float x, y, r;
    private int width, height;

    public GameView(Context c)
    {
        super(c);
        inica();
    }

    public GameView(Context c, AttributeSet a)
    {
        super(c, a);
        inica();
    }

    public GameView(Context c, AttributeSet a, int d)
    {
        super(c, a, d);
        inica();
    }
```

GameView



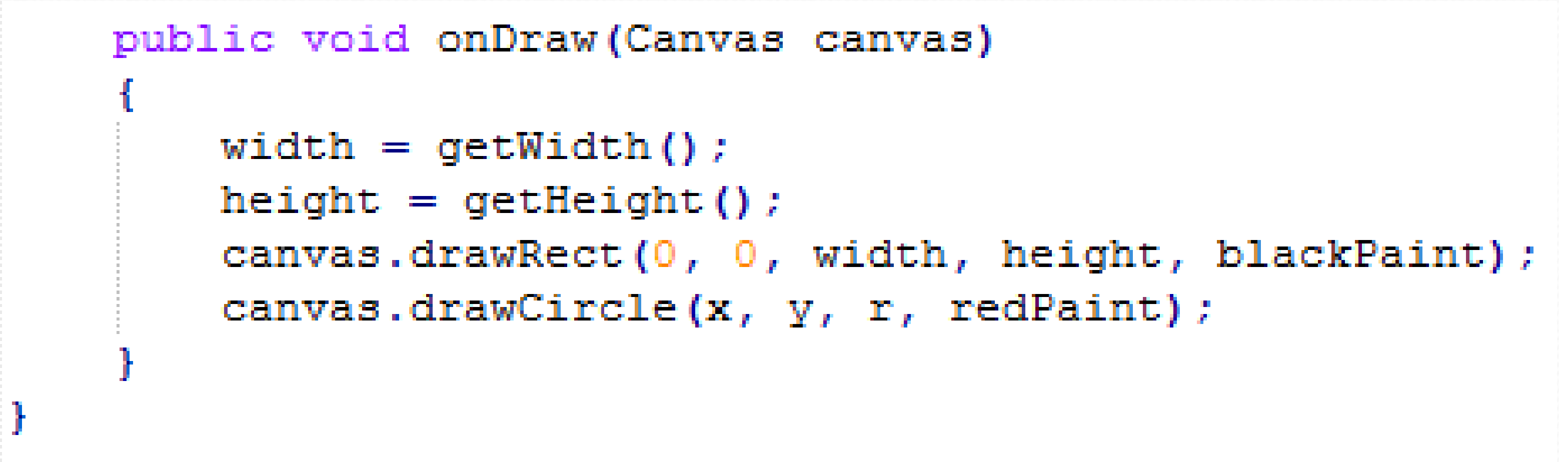
```
public void inica()
{
    blackPaint = new Paint();
    blackPaint.setColor(Color.BLACK);
    blackPaint.setAntiAlias(true);

    redPaint = new Paint();
    redPaint.setColor(Color.RED);
    redPaint.setAntiAlias(true);

    r = 20;
    x = y = r;
}
```

GameView



```
public void moveBall(float dx, float dy)
{
    x += dx;
    y += dy;
    if (x - r < 0) {
        x = r;
    }
    else if (x + r > width) {
        x = width - r;
    }
    if (y - r < 0) {
        y = r;
    }
    else if (y + r > height) {
        y = height - r;
    }
    invalidate();
}
```

GameView



```
    public void onDraw(Canvas canvas)
    {
        width = getWidth();
        height = getHeight();
        canvas.drawRect(0, 0, width, height, blackPaint);
        canvas.drawCircle(x, y, r, redPaint);
    }
}
```

Más información

- Sensores de movimiento: https://developer.android.com/guide/topics/sensors/sensors_motion.html
- Sensores de posición: https://developer.android.com/guide/topics/sensors/sensors_position.html
- Sensores de medio ambiente: https://developer.android.com/guide/topics/sensors/sensors_environment.html