Федеральное агентство связи
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Сибирский государственный университет
телекоммуникаций и информатики»

Лабораторная работа по теме:
«Компас»

Выполнили:
студентки 3 курса
ИВТ, гр. ИП-712
Гервас А.В.
Онищенко А.В.

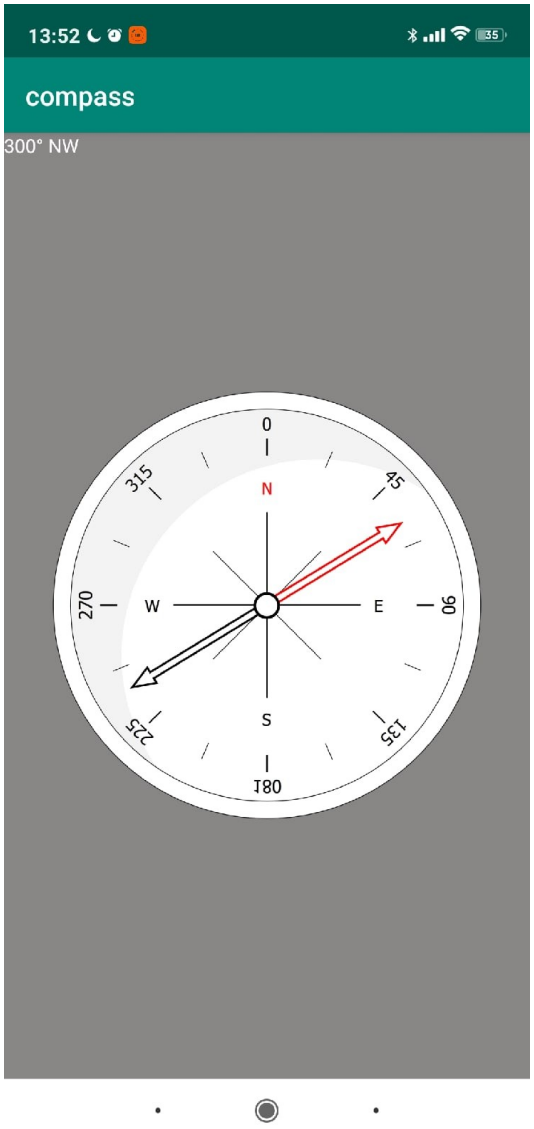
Новосибирск 2020

Оглавление

Задание

Создайте приложение "Компас". На экране отображается циферблат компаса, вращение циферблата осуществляется в зависимости от работы датчика местоположения.

Скриншоты



Листинг кода
MainActivity.java

```
package ru.lab4.compass;

import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;
import android.util.Log;
import android.view.animation.Animation;
import android.view.animation.RotateAnimation;
import android.widget.ImageView;
import android.widget.TextView;

public class MainActivity extends AppCompatActivity {

    private static final String TAG = "MainActivity";

    private Compass compass;
    private ImageView arrowView;
    private TextView sotwLabel;

    private float currentAzimuth;
    private SOTWFormatter sotwFormatter;

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

        sotwFormatter = new SOTWFormatter(this);

        arrowView = findViewById(R.id.main_image_hands);
        sotwLabel = findViewById(R.id.sotw_label);
        setupCompass();
    }

    @Override
    protected void onStart() {
        super.onStart();
        Log.d(TAG, "start compass");
        compass.start();
    }

    @Override
    protected void onPause() {
        super.onPause();
        compass.stop();
    }

    @Override
    protected void onResume() {
        super.onResume();
        compass.start();
    }

    @Override
    protected void onStop() {
        super.onStop();
        Log.d(TAG, "stop compass");
        compass.stop();
    }

    private void setupCompass() {
        compass = new Compass(this);
```

```
        Compass.CompassListener cl = getCompassListener();
        compass.setListener(cl);
    }

    private void adjustArrow(float azimuth) {
        Log.d(TAG, "will set rotation from " + currentAzimuth + " to "
                + azimuth);

        Animation an = new RotateAnimation(-currentAzimuth, -azimuth,
                Animation.RELATIVE_TO_SELF, 0.5f, Animation.RELATIVE_TO_SELF,
                0.5f);
        currentAzimuth = azimuth;

        an.setDuration(500);
        an.setRepeatCount(0);
        an.setFillAfter(true);

        arrowView.startAnimation(an);
    }

    private void adjustSotwLabel(float azimuth) {
        sotwLabel.setText(sotwFormatter.format(azimuth));
    }

    private Compass.CompassListener getCompassListener() {
        return new Compass.CompassListener() {
            @Override
            public void onNewAzimuth(final float azimuth) {
                // UI updates only in UI thread
                // https://stackoverflow.com/q/11140285/444966
                runOnUiThread(new Runnable() {
                    @Override
                    public void run() {
                        adjustArrow(azimuth);
                        adjustSotwLabel(azimuth);
                    }
                });
            }
        };
    }

}
```

Compass.java

```
package ru.lab4.compass;

import android.content.Context;
import android.hardware.Sensor;
import android.hardware.SensorEvent;
import android.hardware.SensorEventListener;
import android.hardware.SensorManager;

public class Compass implements SensorEventListener {
    private static final String TAG = "Compass";

    public interface CompassListener {
        void onNewAzimuth(float azimuth);
    }

    private CompassListener listener;

    private SensorManager sensorManager;
    private Sensor gsensor;
    private Sensor msensor;

    private float[] mGravity = new float[3];
```

```
private float[] mGeomagnetic = new float[3];
private float[] R = new float[9];
private float[] I = new float[9];

private float azimuth;
private float azimuthFix;

public Compass(Context context) {
    sensorManager = (SensorManager) context
            .getSystemService(Context.SENSOR_SERVICE);
    gsensor = sensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER);
    msensor = sensorManager.getDefaultSensor(Sensor.TYPE_MAGNETIC_FIELD);
}

public void start() {
    sensorManager.registerListener(this, gsensor,
            SensorManager.SENSOR_DELAY_GAME);
    sensorManager.registerListener(this, msensor,
            SensorManager.SENSOR_DELAY_GAME);
}

public void stop() {
    sensorManager.unregisterListener(this);
}

public void setAzimuthFix(float fix) {
    azimuthFix = fix;
}

public void resetAzimuthFix() {
    setAzimuthFix(0);
}

public void setListener(CompassListener l) {
    listener = l;
}

@Override
public void onSensorChanged(SensorEvent event) {
    final float alpha = 0.97f;

    synchronized (this) {
        if (event.sensor.getType() == Sensor.TYPE_ACCELEROMETER) {

            mGravity[0] = alpha * mGravity[0] + (1 - alpha)
                    * event.values[0];
            mGravity[1] = alpha * mGravity[1] + (1 - alpha)
                    * event.values[1];
            mGravity[2] = alpha * mGravity[2] + (1 - alpha)
                    * event.values[2];

            // mGravity = event.values;

            // Log.e(TAG, Float.toString(mGravity[0]));
        }

        if (event.sensor.getType() == Sensor.TYPE_MAGNETIC_FIELD) {
            // mGeomagnetic = event.values;

            mGeomagnetic[0] = alpha * mGeomagnetic[0] + (1 - alpha)
                    * event.values[0];
            mGeomagnetic[1] = alpha * mGeomagnetic[1] + (1 - alpha)
                    * event.values[1];
            mGeomagnetic[2] = alpha * mGeomagnetic[2] + (1 - alpha)
                    * event.values[2];
            // Log.e(TAG, Float.toString(event.values[0]));

        }
```

```
            boolean success = SensorManager.getRotationMatrix(R, I, mGravity,
                    mGeomagnetic);
            if (success) {
                float orientation[] = new float[3];
                SensorManager.getOrientation(R, orientation);
                // Log.d(TAG, "azimuth (rad): " + azimuth);
                azimuth = (float) Math.toDegrees(orientation[0]); // orientation
                azimuth = (azimuth + azimuthFix + 360) % 360;
                // Log.d(TAG, "azimuth (deg): " + azimuth);
                if (listener != null) {
                    listener.onNewAzimuth(azimuth);
                }
            }
        }
    }

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

SOTWFormatter.java

```
package ru.lab4.compass;

import android.content.Context;

class SOTWFormatter {
    private static final int[] sides = {0, 45, 90, 135, 180, 225, 270, 315, 360};
    private static String[] names = null;

    public SOTWFormatter(Context context) {
        initLocalizedNames(context);
    }

    public String format(float azimuth) {
        int iAzimuth = (int) azimuth;
        int index = findClosestIndex(iAzimuth);
        return iAzimuth + "° " + names[index];
    }

    private void initLocalizedNames(Context context) {
        // {"N", "NE", "E", "SE", "S", "SW", "W", "NW", "N"}
        // yes, N is twice, for 0 and for 360

        if (names == null) {
            names = new String[]{
                    context.getString(R.string.sotw_north),
                    context.getString(R.string.sotw_northeast),
                    context.getString(R.string.sotw_east),
                    context.getString(R.string.sotw_southeast),
                    context.getString(R.string.sotw_south),
                    context.getString(R.string.sotw_southwest),
                    context.getString(R.string.sotw_west),
                    context.getString(R.string.sotw_northwest),
                    context.getString(R.string.sotw_north)
            };
        }
    }

    private static int findClosestIndex(int target) {
        int i = 0, j = sides.length, mid = 0;
        while (i < j) {
            mid = (i + j) / 2;
            if (target < sides[mid]) {
                if (mid > 0 && target > sides[mid - 1]) {
                    return getClosest(mid - 1, mid, target);
```

```
            }

            j = mid;
        } else {
            if (mid < sides.length - 1 && target < sides[mid + 1]) {
                return getClosest(mid, mid + 1, target);
            }
            i = mid + 1;
        }
    }
    return mid;
}

private static int getClosest(int index1, int index2, int target) {
    if (target - sides[index1] >= sides[index2] - target) {
        return index2;
    }
    return index1;
}
}
```