



## MOBILNE APLIKACIJE

### Vežbe 11

#### Senzori

2022/2023

# Sadržaj

1. Senzori	3
1.1 Sensor Framework	3
2. Domaći	10

## 1. Senzori

Senzor je uređaj koji pretvara jednu fizičku veličinu u drugu fizičku veličinu, koju čovek može neposredno da opazi ili koju računar može da očita.

Android podržava tri tipa senzora:

1. Senzori pozicije
2. Senzori pokreta
3. Senzori okruženja

**Senzori pozicije** mere fizičku poziciju uređaja. Ova kategorija uključuje senzore za orijentaciju i magnetometre (MAGNETIC\_FIELD, PROXIMITY).

**Senzori pokreta** mere sile ubrzanja i rotacione sile duž tri ose. Senzori pokreta su: akcelerometri, gravitacioni senzori, žiroskopi i senzor rotacionog vektora (ACCELEROMETER, GRAVITY, GYROSCOPE, LINEAR\_ACCELERATION, ROTATION\_VECTOR).

**Senzori okruženja** mere različite parametre okoline, kao što su temperatura, pritisak, osvetljenje, vlaga. Ova kategorija uključuje barometre, fotometre i termometre (AMBIENT\_TEMPERATURE, LIGHT, PRESSURE, RELATIVE\_HUMIDITY).

Hardverski senzori: ACCELEROMETER, AMBIENT\_TEMPERATURE, GYROSCOPE, LIGHT, MAGNETIC\_FIELD, PRESSURE, PROXIMITY, RELATIVE\_HUMIDITY

Softverski ili hardverski senzori: GRAVITY, LINEAR\_ACCELERATION, ROTATION\_VECTOR

### 1.1 Sensor Framework

*Sensor Framework* omogućava pristup različitim tipovima senzora i rad sa njima. *Sensor Framework* je deo paketa android.hardware i uključuje sledeće klase i interfejs:

- *SensorManager*,
- *Sensor*,
- *SensorEvent* i
- *SensorEventListener*.

#### **SensorManager**

Ova klasa omogućava različite metode za pristupanje, izlistavanje senzora itd.

#### **Sensor**

Klasa *Sensor* može da se koristi da se kreira instanca nekog specifičnog senzora. Ova klasa sadrži informacije o svojstvima određenog senzora i nudi niz metoda koje omogućavaju da se ustanove mogućnosti senzora.

#### **SensorEvent**

Sistem koristi klasu *SensorEvent* da kreira sensor event objekat, koji sadrži informacije o određenom merenju.

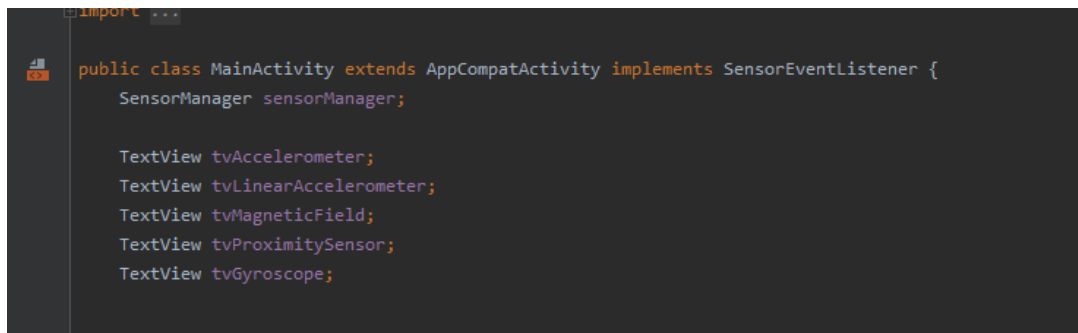
### SensorEventListener

Interfejs *SensorEventListener* služi za primanje notifikacija od *SensorManager*-a. On sadrži obrađivače *SensorEvent* događaja.

Sa *Sensor Framework*-om možemo da:

- Odredimo koji senzori su dostupni na uređaju.
- Odredimo mogućnosti dostupnih senzora.
- Napišemo obrađivače događaja, koji reaguju na promenu fizičke veličine ili tačnosti merenja.
- Registrujemo i odregistrujemo obrađivače događaja, koji prate promene u merenju.

U primeru za vežbe 9 kreirali smo klasu *MainActivity* koja implementira *SensorEventListener* (slika 1).



```
import ...

public class MainActivity extends AppCompatActivity implements SensorEventListener {
    SensorManager sensorManager;

    TextView tvAccelerometer;
    TextView tvLinearAccelerometer;
    TextView tvMagneticField;
    TextView tvProximitySensor;
    TextView tvGyroscope;
}
```

Slika 1. Kreiranje klase koja implementira interfejs *SensorEventListener*

Da bi se ustanovilo koji sve senzori postoje na uređaju, potrebno je dobiti referencu do sensor servisa (slika 2). Kreira se instanca klase *SensorManager* pozivanjem metode *getSystemService* i prosleđivanjem konstante *SENSOR\_SERVICE* (linija 26). Nad objektom *sensorManager*-a pozivaćemo metode da registrujemo *listener*-e, da bismo dobijali merenja.

Listu svih senzora možemo dobiti uz pomoć metode *getSensorList()*. Toj metodi prosleđujemo konstantu *TYPE\_ALL*. Primer:

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

Za dobavljanje senzora određenog tipa umesto *TYPE\_ALL* prosleđuje se *TYPE\_GYROSCOPE* ili *TYPE\_GRAVITY* ili *TYPE\_ACCELERATION* itd.

Osim što možemo da dobavimo listu senzora, moguće je koristiti *public* metode klase *Sensor* da dobavimo informacije o svojstvima određenog senzora. Nemaju svi uređaji iste vrste senzora,

niti ista vrsta senzora na različitim uređajima ima jednaka svojstva. Metode koje klasa *Sensor* nudi mogu biti veoma korisne, ako želimo da se naša aplikacija ponaša različito u zavisnosti od tipova senzora i njihovih svojstava na različitim uređajima. Neke od metoda koje mogu da se koriste su *getResolution()*, *getMaximumRange()*, *getPower()* itd.

Na narednim linijama (od 28 do 32) dobavljamo tekstualna polja sa našeg *layout*-a (slika 3), u koja ćemo upisivati vrednosti merenja. *activity\_main.xml layout* sadrži 5 tekstualnih polja, po jedno tekstualno polje za svaki senzor koji ćemo pratiti.

```
21      @Override
22      protected void onCreate(Bundle savedInstanceState) {
23          super.onCreate(savedInstanceState);
24          setContentView(R.layout.activity_main);
25
26          sensorManager = (SensorManager) getSystemService(SENSOR_SERVICE);
27
28          tvAccelerometer = (TextView) findViewById(R.id.tvAccelerometer);
29          tvLinearAccelerometer = (TextView) findViewById(R.id.tvLinearAccelerometer);
30          tvMagneticField = (TextView) findViewById(R.id.tvMagneticField);
31          tvProximitySensor = (TextView) findViewById(R.id.tvProximitySensor);
32          tvGyroscope = (TextView) findViewById(R.id.tvGyroscope);
33      }
34
```

Slika 2. Metoda *onCreate*

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
    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"
    tools:context="ftn.rs.sensors.MainActivity">

    <TextView
        android:id="@+id/tvAccelerometer"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_alignParentTop="true"
        android:layout_marginTop="70dp"
        android:text="Accelerometer" />

    <TextView
        android:id="@+id/tvMagneticField"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/tvAccelerometer"
        android:layout_marginTop="65dp"
        android:text="Magnetic field" />
```

Slika 3. *activity\_main.xml*

U primeru za ove vežbe pratimo merenja 5 senzora:

1. ACCELEROMETER
2. LINEAR\_ACCELERATION
3. MAGNETIC\_FIELD
4. PROXIMITY
5. GYROSCOPE

U metodi *onResume* registrujemo *listener*-e na senzore da bismo dobijali merenja. *Listener*-e registrujemo uz pomoć metode *registerListener*. Ovu metodu pozivamo nad objektom klase *SensorManager* i prosleđujemo joj tri parametra:

1. *Listener* – obrađivač događaja
2. *Sensor* – je senzor
3. *SamplingPeriodUs* – predstavlja period uzorkovanja

Na slici 4 se nalazi primer registrovanja *listener*-a.

```
@Override
protected void onResume() {
    super.onResume();
    // register this class as a listener for the orientation and
    // accelerometer sensors
    sensorManager.registerListener( listener: this,
                                   sensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER),
                                   SensorManager.SENSOR_DELAY_NORMAL);

    sensorManager.registerListener( listener: this,
                                   sensorManager.getDefaultSensor(Sensor.TYPE_LINEAR_ACCELERATION),
                                   SensorManager.SENSOR_DELAY_NORMAL);

    sensorManager.registerListener( listener: this,
                                   sensorManager.getDefaultSensor(Sensor.TYPE_MAGNETIC_FIELD),
                                   SensorManager.SENSOR_DELAY_NORMAL);

    sensorManager.registerListener( listener: this,
                                   sensorManager.getDefaultSensor(Sensor.TYPE_PROXIMITY),
                                   SensorManager.SENSOR_DELAY_NORMAL);

    sensorManager.registerListener( listener: this,
                                   sensorManager.getDefaultSensor(Sensor.TYPE_GYROSCOPE),
                                   SensorManager.SENSOR_DELAY_NORMAL);
}
```

Slika 4. Metoda *onResume*

Interfejs *SensorEventListener* nam nudi 2 metode koje implementiramo:

1. *onSensorChanged*
2. *onAccuracyChanged*

### **onSensorChanged**

Metoda *onSensorChanged* se poziva kada se vrednosti merenja promene. Ova metoda prima *event*, objekat klase *SensorEvent*, koji sadrži informacije o merenju. Kod na slici 5, u zavisnosti od tipa senzora, popunjava tekstualna polja *layout*-a. U svako polje unosi se *string* sa nazivom senzora, koji se prati, i sa pročitanim vrednostima. Vrednosti čitamo iz niza *values*, koji sadrži objekat *event*.

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

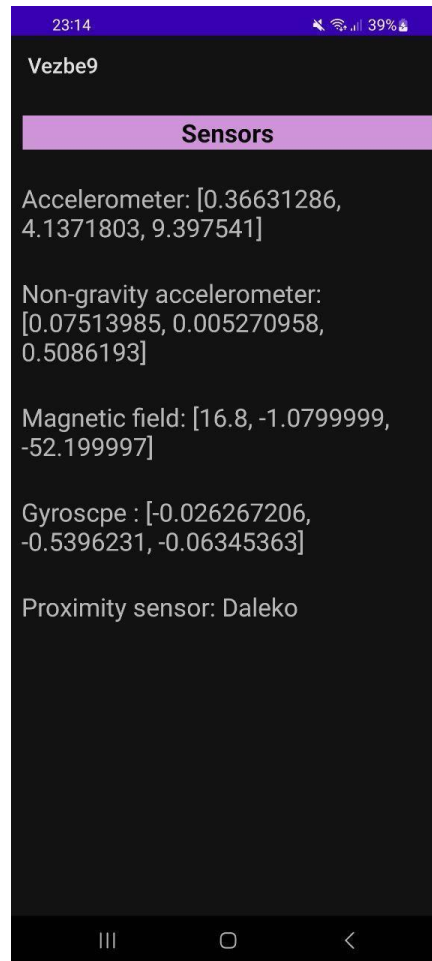
        tvAccelerometer.setText("Accelerometer: [" + x + ", " + y + ", " + z + "]");
    } else if (event.sensor.getType() == Sensor.TYPE_LINEAR_ACCELERATION) {
        float[] values = event.values;
        float x = values[0];
        float y = values[1];
        float z = values[2];

        tvLinearAccelerometer.setText("Non-gravity accelerometer: [" + x + ", " + y + ", " + z + "]");
    } else if (event.sensor.getType() == Sensor.TYPE_MAGNETIC_FIELD) {
        float[] values = event.values;
        float x = values[0];
        float y = values[1];
        float z = values[2];

        tvMagneticField.setText("Magnetic field: [" + x + ", " + y + ", " + z + "]");
    } else if (event.sensor.getType() == Sensor.TYPE_GYROSCOPE) {
        float[] values = event.values;
        float x = values[0];
        float y = values[1];
        float z = values[2];
    }
}
```

Slika 5. Metoda *onSensorChanged*

Na slici 6 se nalazi primer kako se merenja prikazuju u našoj aplikaciji.



Slika 6. Prikaz merjenja

### **onAccuracyChanged**

Metoda *onAccuracyChanged* (slika 7) se poziva kada se promeni tačnost merjenja registrovanog senzora. Za razliku od metode *onSensorChanged*, ova metoda se poziva samo kad se tačnost promeni.



```

124
125      @Override
126      public void onAccuracyChanged(Sensor sensor, int accuracy) {
127          if(sensor.getType() == Sensor.TYPE_ACCELEROMETER){
128              Log.i( tag: "REZ_ACCELEROMETER", String.valueOf(accuracy));
129          }else if(sensor.getType() == Sensor.TYPE_LINEAR_ACCELERATION){
130              Log.i( tag: "REZ_LINEAR_ACCELERATION", String.valueOf(accuracy));
131          }else if(sensor.getType() == Sensor.TYPE_MAGNETIC_FIELD){
132              Log.i( tag: "REZ_MAGNETIC_FIELD", String.valueOf(accuracy));
133          }else if(sensor.getType() == Sensor.TYPE_GYROSCOPE){
134              Log.i( tag: "REZ_GYROSCOPE", String.valueOf(accuracy));
135          }else if(sensor.getType() == Sensor.TYPE_PROXIMITY){
136              Log.i( tag: "REZ_TYPE_PROXIMITY", String.valueOf(accuracy));
137          }else{
138              Log.i( tag: "REZ_OTHER_SENSOR", String.valueOf(accuracy));
139          }
140      }
141  }
142

```

Slika 7. Metoda *onAccuracyChanged*

Kada više ne koristimo senzore, sve *listener*-e treba otključiti i to radimo u metodi *onPause* (slika 8).

```

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

```

Slika 8. Metoda *onPause*

## 2. Domaći

Domaći se nalazi na *Canvas-u* (*canvas.ftn.uns.ac.rs*) na putanji *Vežbe/11 Zadatak.pdf*

Primer *Vežbe11* možete preuzeti na sledećem linku:

<https://gitlab.com/antesevicceca/mobilne-aplikacije-sit>

Za dodatna pitanja možete se obratiti asistentima:

- Svetlana Antešević ([svetlanaantesevic@uns.ac.rs](mailto:svetlanaantesevic@uns.ac.rs))
- Jelena Matković ([matkovic.jelena@uns.ac.rs](mailto:matkovic.jelena@uns.ac.rs))