Tampereen ammattikorkeakoulu

Mobile App Development 1

Study diary

by Dmitrii Bacherikov

Contents

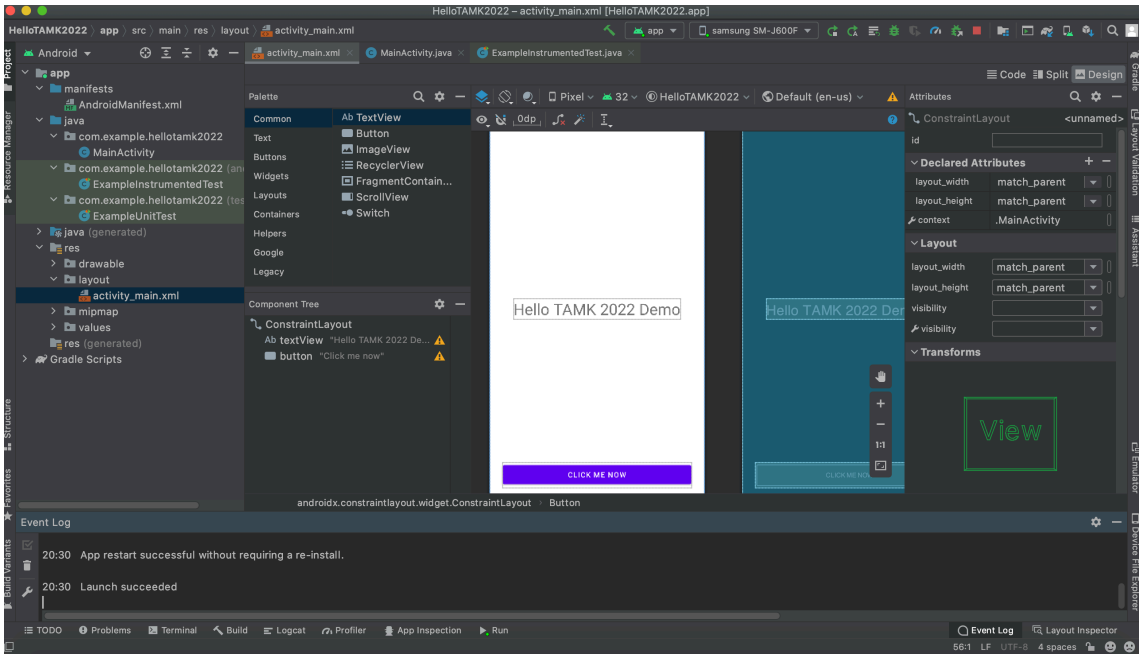
Important general note for all weekly exercises: List all external sources you’ve used to implement the task (github, tutorialspoint, other projects, network sites, blogs etc..)

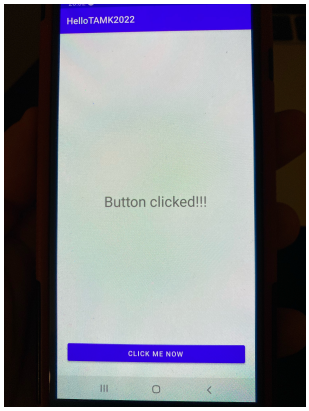
1 Week exercises

1.1 Android Studio setup and Hello World

I have installed Android Studio in MacOS (Java runtime was installed previously already).
I came across with 2 issues:
- Android Studio didn't see my Android device connected
- Failed to install the following Android SDK packages as some licences have not been accepted.
In both cases I found solutions on StackOverflow (links are given in Sources).

Here is a screenshot of successful run:





2 Week exercises

2.1

What is activity and what is the role of XML and Java files when implementing GUI?

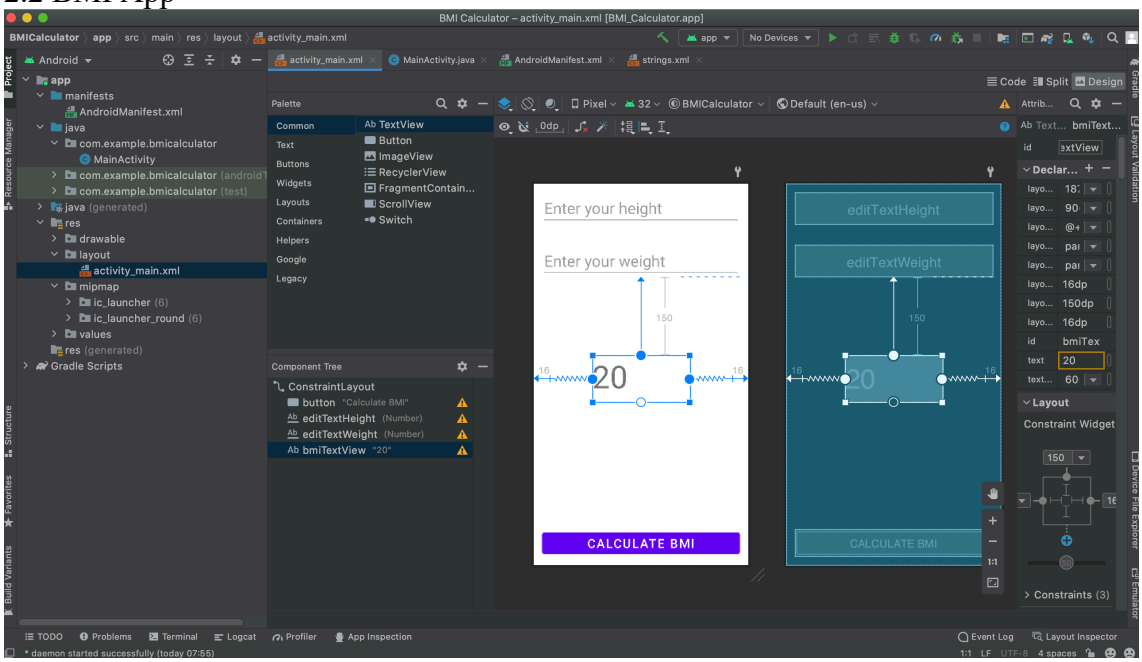
Activity is a specific screen of an applications (can be multiple activities in one applications) which consists of XML and Java files. XML is for GUI layout (design), and Java is for logic.

What is meant by an activity lifecycle. When is activity created/closed in Android?

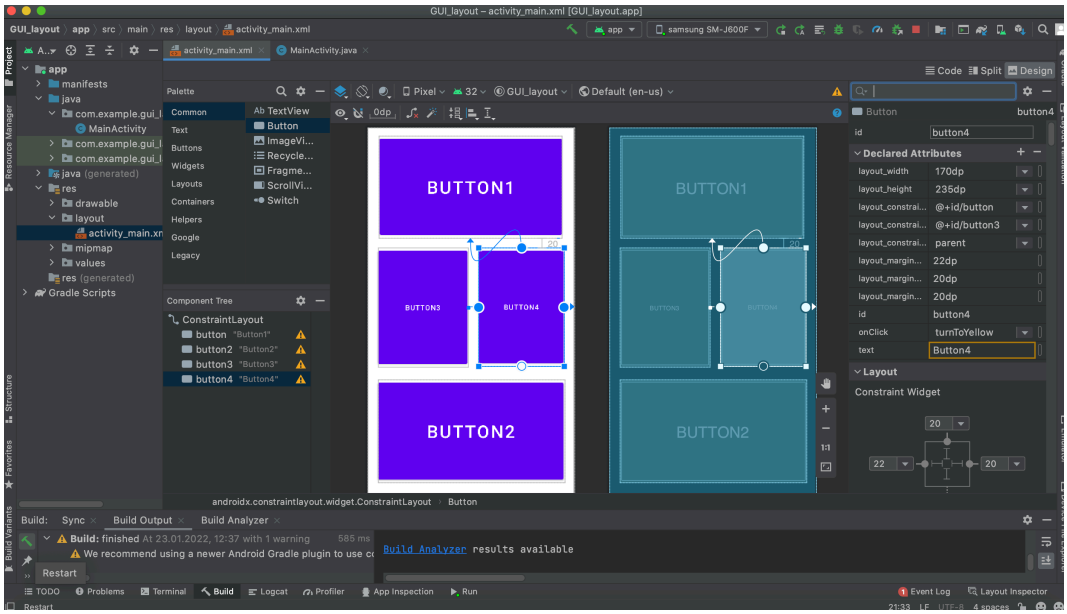
An activity lifecycle is based on 7 methods: onCreate(), onStart(), onResume(), onPause(), onStop(), onRestart(), onDestroy(). Thus it can be created by onCreate() method and closed/finished by onDestroy() method.

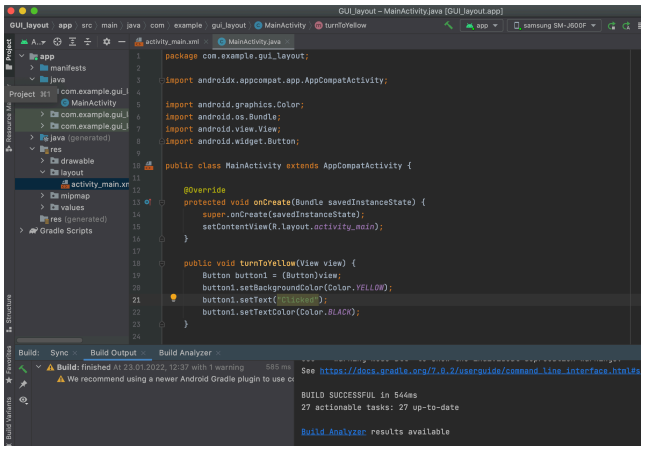
Main activity is created when the application is launched (can be other activities which are created when the user switches to another view), and closed when the app is stopped or another activity is started.

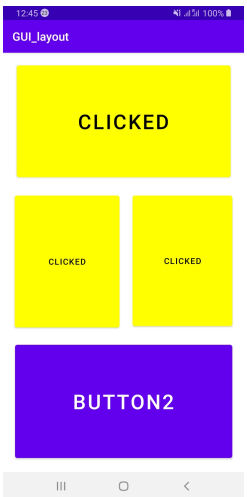
2.2 BMI App



2.3 Create GUI layout with Constraint layout







3 Week exercises

3.1

What advantages you see in defining user interfaces in UI resource files (.xml) instead of coding the UI (e.g. with Java)?

It is easier to design

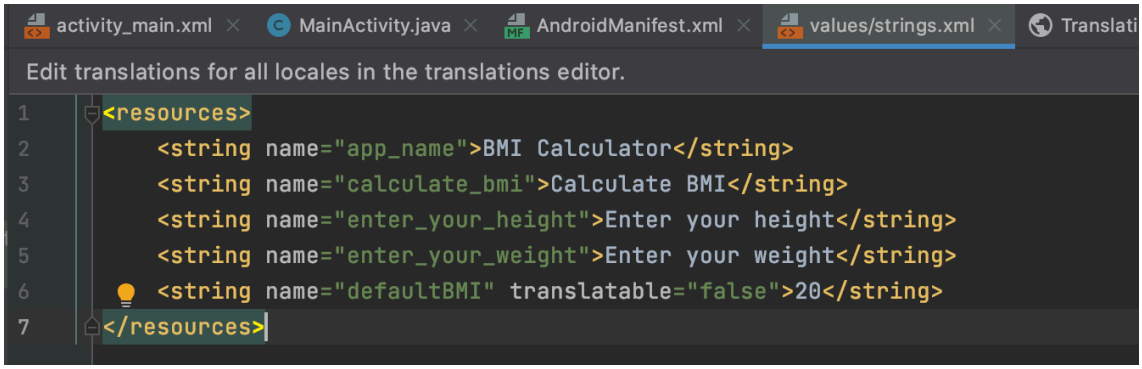
- Less code is needed and no hardcode
- Easy to design layout using XML in Android Studio
- Easy to localize for different devices
- Easier to maintain
- It is faster

In Android studio, how localization and language variants are created for an Android project?

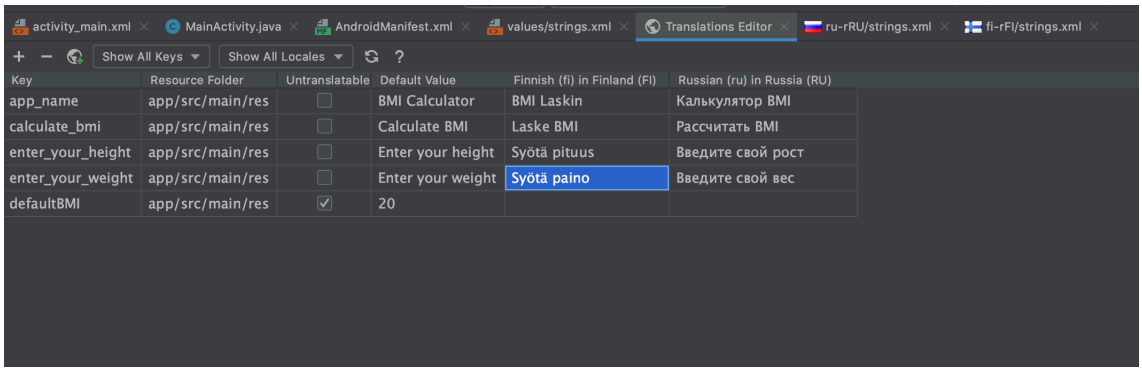
In the editor we add Locale/languages needed (another XML file is created), then we add translations.

3.2

Removed all hard coded strings and put them into strings.xml



Added 2 more languages (Russian and Finnish)



3.3 Java app consisting of Main.java and Student.java files

```java
public class Student {
    private String sName;
    private int sAge;
    private String sID;
    private int sCredits;

    public Student() {
        System.out.println("Default constructor was activated");
    }

    public Student(String aName, int aAge, String aID) {
        sName = aName;
        sAge = aAge;
        sID = aID;
        sCredits = 0;
    }

    public void printStudentData() {
        System.out.println("Name: " + this.sName);
        System.out.println("Age: " + this.sAge);
        System.out.println("ID: " + this.sID);
        System.out.println("Credit points: " + this.sCredits);
    }

    public void addCreditPoints(int addedCredits) {
        this.sCredits += addedCredits;
        System.out.println("Student " + this.sName + " received " + addedCredits + " credit points");
    }

}
```

```java
public class Main {
    Run | Debug
    public static void main( String args[] ) {

        System.out.println("Test of default constructor:");
        Student st1 = new Student();
        System.out.println();

        Student st2 = new Student ("Jack", 25, "ID1");
        Student st3 = new Student ("Mike", 21, "ID2");
        Student st4 = new Student ("Angela", 18, "ID3");

        st2.printStudentData();
        System.out.println();
        st3.printStudentData();
        System.out.println();
        st4.printStudentData();
        System.out.println();
        st2.addCreditPoints(5);
        System.out.println();
        st2.printStudentData();
    }
}
```

```
MacBook-Pro-Dmitrij:Week3 dialex2006$ java Main
Test of default constructor:
Default constructor was activated

Name: Jack
Age: 25
ID: ID1
Credit points: 0

Name: Mike
Age: 21
ID: ID2
Credit points: 0

Name: Angela
Age: 18
ID: ID3
Credit points: 0

Student Jack received 5 credit points

Name: Jack
Age: 25
ID: ID1
Credit points: 5
```

4 Week exercises

4.1 Address class

StudentApp.java Trainee.java Student.java Address.java

```
public class Address {
    private String mStreet;
    private String mCity;

    public Address ( String aStreet, String aCity) {
        mStreet = aStreet;
        mCity = aCity;
    }

    public void setCity( String aCity ) {
        mCity = aCity;
    }

    public void setStreet( String aStreet ) {
        mStreet = aStreet;
    }

    public String getCity() {
        return mCity;
    }

    public String getStreet() {
        return mStreet;
    }

    void printAddress() {
        System.out.println("Address: " + mStreet + ", " + mCity);
    }
}
```

Extra constructor containing address parameter:

```
public Student( String aName, int aAge, String aID, Address aAddress ) {
    mName = aName;
    mAge = aAge;
    mID = aID;
    mCreditPoints = 0;
    mAddress = aAddress;
}
```

Creating students in the main program:

```
import java.util.ArrayList;

public class StudentApp {
    public static void main( String args[]) {
        Student pekka = new Student( aName: "Pekka", aAge: 20, aID: "32222", aCreditPoints: 0 );
        Student tiina = new Student( aName: "Tiina", aAge: 25, aID: "11227");
        Student ville = new Student( aName: "Ville", aAge: 30, aID: "12345", aCreditPoints: 5);
        Address marjasAddress = new Address( aStreet: "Ilmarinkatu", aCity: "Tampere" );
        Student marja = new Student( aName: "Marja", aAge: 23, aID: "37777", aCreditPoints: 0, marjasAddress );
        pekka.printStudentData();
        tiina.printStudentData();
        ville.printStudentData();
        marja.printStudentData();
```

4.2 Trainee class

StudentApp.java | Trainee.java | Student.java | Address.java

```java
public class Trainee extends Student {

    private String mEmployer;
    private float mSalary;

    public Trainee() {
        super();
    }

    public Trainee(String aName, int aAge, String aID, String aEmployer, float aSalary) {
        super(aName, aAge, aID);
        mEmployer = aEmployer;
        mSalary = aSalary;
    }

    public Trainee(String aName, int aAge, String aID, int aCreditPoints, String aEmployer, float aSalary) {
        super(aName, aAge, aID, aCreditPoints);
        mEmployer = aEmployer;
        mSalary = aSalary;
    }

```

StudentApp.java | Trainee.java | Student.java | Address.java

```java
    }

    public String getEmployer() {
        return mEmployer;
    }

    public void setEmployer( String sEmployer ) {
        mEmployer = sEmployer;
    }

    public float getSalary() {
        return mSalary;
    }

    public void setSalary( float sSalary ) {
        mSalary = sSalary;
    }

    @Override
    public void printStudentData() {
        super.printStudentData();
        System.out.println("Employer: " + this.mEmployer);
        System.out.println("Salary: " + this.mSalary);
    }
}
```

4.3 Array of students

```
        //dynamic array of students
        ArrayList<Student> students = new ArrayList<Student>();

        students.add (new Student( aName: "Miko",  aAge: 24,  aID: "88888",  aCreditPoints: 14));
        students.add (new Trainee( aName: "Katja",  aAge: 22,  aID: "99999",  aEmployer: "Bitwise",  aSalary: 2100));

        for (Student s:students) {
            s.printStudentData();
        }

    }
```

5 Week exercises

5.1 App GUI with 2 screens

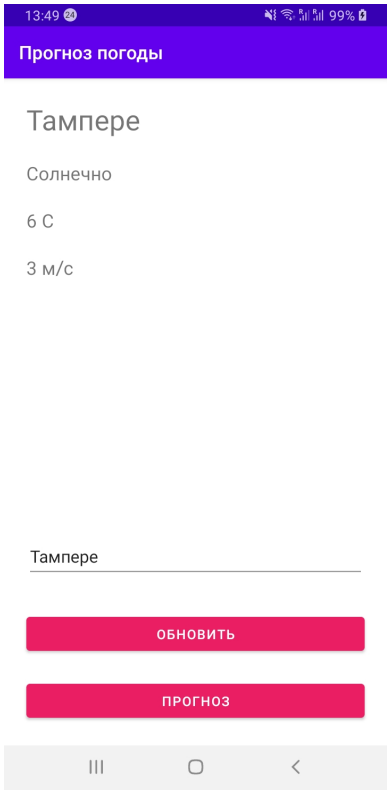
Creating the second activity

```java
package com.example.weatherapplication;

import ...

public class ForecastActivity extends AppCompatActivity {

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

        String cityName = getIntent().getStringExtra( name: "CITY_NAME");

        TextView forecastCityName = findViewById(R.id.textViewCityNameForecast);
        if (cityName != null) {
            forecastCityName.setText(cityName);
        }
        else {
            forecastCityName.setText("No city");
        }
    }
}
```

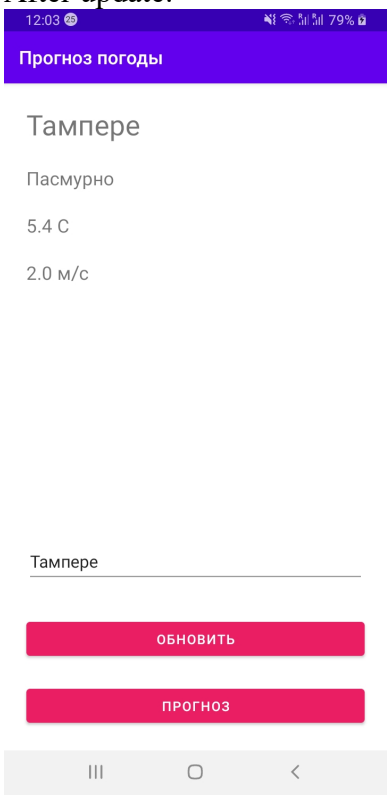
Creating the intent

```java
public class MainActivity extends AppCompatActivity {

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

    public void openForecast(View view) {
        //1. Create an intent for opening ForecastActivity
        Intent intent = new Intent( packageContext: this, ForecastActivity.class);
        //2. Send some data
        //parameters: key for the data and value for the data
        intent.putExtra( name: "CITY_NAME",  value: "Tampere");
        //3. Start the activity through the intent
        startActivity(intent);
    }

    public void updateWeather(View view) {
        String weatherDescription = "Cloudy";
        float temperature = 5.4f;
        float windSpeed = 2.0f;
        TextView weatherDescriptionTextView = findViewById(R.id.textViewWeatherDescription);
        weatherDescriptionTextView.setText( weatherDescription );

        TextView temperatureTextView = findViewById(R.id.textViewTemperature);
        temperatureTextView.setText("" + temperature + " C");

        TextView windSpeedTextView = findViewById(R.id.textViewWindSpeed);
        windSpeedTextView.setText("" + windSpeed + " m/s");

    }
```

Application screens (localized Russian version):



After update:



Second activity:

5.2 Written exercises

Activity is a screen of an application (can be multiple activities in one application) which consists of XML and Java files. XML is for GUI layout (design), and Java is for logic.

An activity lifecycle is based on 7 methods: onCreate(), onStart(), onResume(), onPause(), onStop(), onRestart(), onDestroy(). Thus it can be created by onCreate() method and closed/finished by onDestroy() method.
Main activity is created when the application is launched (can be other activities which are created when the user switches to another view), and closed when the app is stopped or another activity is started.

Intent is a tool used to switch from one activity to another. It's a message that triggers Activity. Intent is a call to activate something + data to send with the call. Data may contain key-value pairs for additional information to be delivered to the component handling the intent.

6 Week exercises

6.1 Fetching HTTP data and parsing JSON

To get REST data from a HTTP server, we'll need to use an API in the Internet with a specific address. Usually Fetch or Axios methods are used to get data. For Android the most used library is Volley. Thus in Android we create RequestQueue, pass it Request objects. Once we add the request it moves through the pipeline, gets serviced, and has its raw response parsed and delivered. As a callback function we get result – either successful or failed.

When JSON object is received, we just parse data from the object by using getJSONObject and getJSONArray methods. We use "try … catch" statement to handle it just to make sure it is a valid JSON object.

6.2 Fetching HTTP data and parsing JSON

As an application I used previously developed weather app and weather API with this address:
https://api.openweathermap.org/data/2.5/weather?q=tampere&units=metric&appid=5bb5570e03fef03fc9963343649ae985

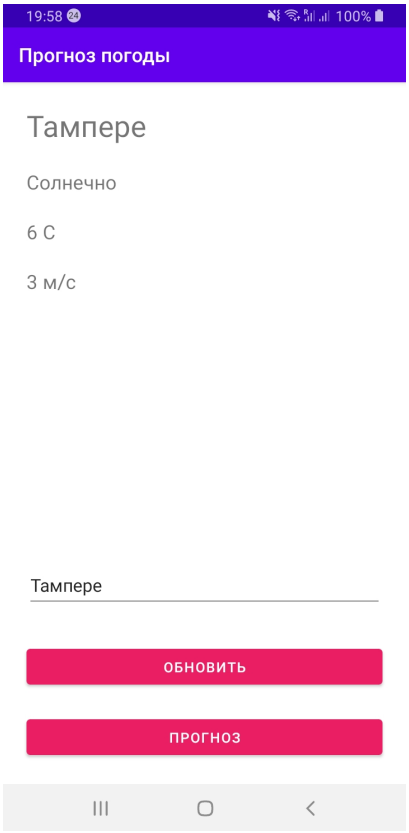
We add RequestQueue:

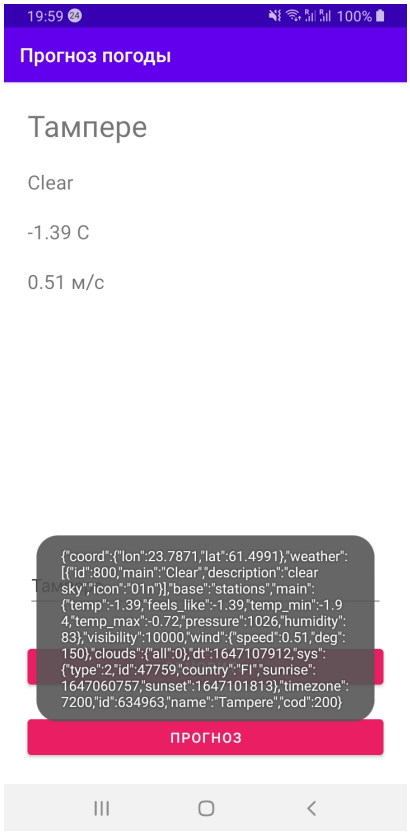
```
45    public void updateWeather(View view) {
46
47        StringRequest stringRequest = new StringRequest(Request.Method.GET, url,
48                response -> {
49                    //Show the network response within an android toast
50                    Toast.makeText( context: this, response, Toast.LENGTH_LONG).show();
51
52                    //parsing the data from the JSON Object
53                    parseJsonAndUpdateUI( response );
54
55                }, error -> {
56                    Toast.makeText( context: this,  text: "Error!", Toast.LENGTH_LONG).show();
57                }
58        );
59        //Sending the request by adding it to the queue
60        queue.add(stringRequest);
```

Then we parse data from JSON object:

```
77    private void parseJsonAndUpdateUI(String response) {
78        String weatherDescription = "Sunny";
79        double temperature = -20;
80        double windSpeed = 5;
81
82        //parsing from JSON
83        try {
84            JSONObject weather = new JSONObject(response);
85            temperature = weather.getJSONObject("main").getDouble( name: "temp");
86            weatherDescription = weather.getJSONArray( name: "weather").getJSONObject( index: 0).getString( name: "main");
87            windSpeed = weather.getJSONObject("wind").getDouble( name: "speed");
88
89        } catch (JSONException e) {
90            e.printStackTrace();
91        }
92
93        TextView weatherDescriptionTextView = findViewById(R.id.textViewWeatherDescription);
94        weatherDescriptionTextView.setText( weatherDescription );
95
96        TextView temperatureTextView = findViewById(R.id.textViewTemperature);
97        temperatureTextView.setText("" + temperature + getString(R.string.tempMeasurement));
```

Here are screenshots of the app:

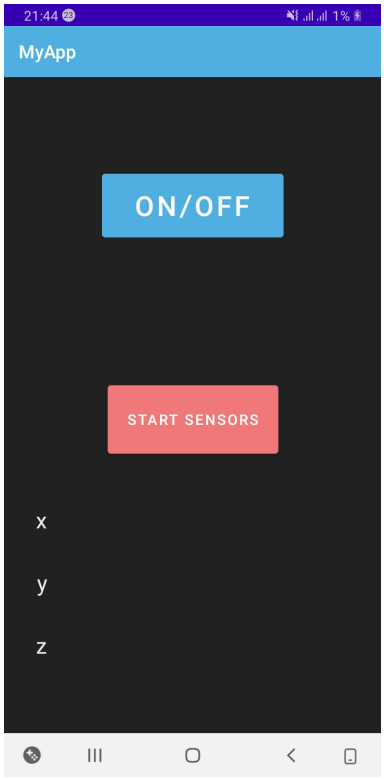

19:58
100%
Прогноз погоды
Тампере
Солнечно
6 C
3 м/с
Тампере
ОБНОВИТЬ
ПРОГНОЗ


19:59
100%
Прогноз погоды
Тампере
Clear
-1.39 C
0.51 м/с
{"coord":{"lon":23.7871,"lat":61.4991},"weather":[{"id":800,"main":"Clear","description":"clear sky","icon":"01n"}],"base":"stations","main":{"temp":-1.39,"feels_like":-1.39,"temp_min":-1.94,"temp_max":-0.72,"pressure":1026,"humidity":83},"visibility":10000,"wind":{"speed":0.51,"deg":150},"clouds":{"all":0},"dt":1647107912,"sys":{"type":2,"id":47759,"country":"FI","sunrise":1647060757,"sunset":1647101813},"timezone":7200,"id":634963,"name":"Tampere","cod":200}
ПРОГНОЗ

7 Week exercises

7.1 Synchronous device API Example – Camera HW and “torch app”

Flashlight App screenshots

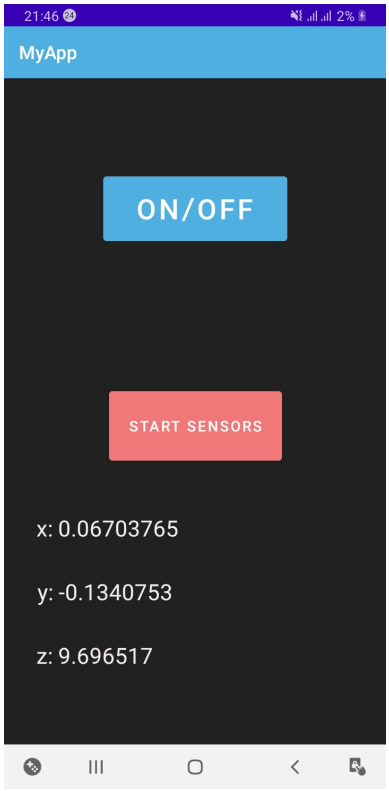



```
public void setTorchOnOff(View view) {

    //Todo: connect Camera Manager: which manages camera flashlight
    CameraManager cameraManager = (CameraManager) getSystemService(CAMERA_SERVICE);
    try {

        //Todo: ask if there is a camera, which has flashlight
        for (String id : cameraManager.getCameraIdList()) {
            CameraCharacteristics cameraChar = cameraManager.getCameraCharacteristics(id);
            if (cameraChar.get(CameraCharacteristics.FLASH_INFO_AVAILABLE)) {

                if ( torchOn )
                    cameraManager.setTorchMode(id,  enabled: false);
                //Todo: set the flashlight of that camera on (torch made to true)
                else cameraManager.setTorchMode(id,  enabled: true);
                torchOn = !torchOn;
            }
        }
    }
    catch (CameraAccessException e) {
        e.printStackTrace();
    }
```

In the emulator we can test the app using logcat in Android Studio.

7.2 Asynchronous API call example (listener) – Sensor API and “Level App”

App screenshots

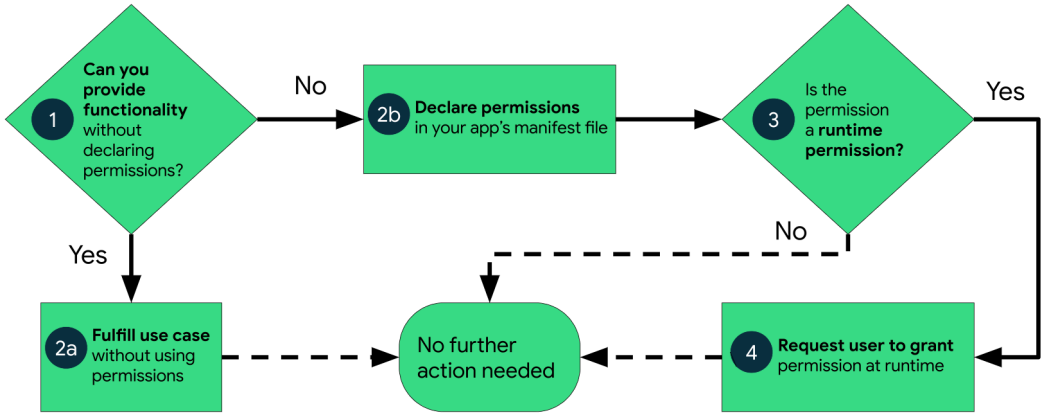



```java
public void startSensors(View view) {
    //Start listening

    //Todo: connect sensor Manager
    SensorManager sensorManager = (SensorManager) getSystemService(Context.SENSOR_SERVICE);
    List<Sensor> sensorList = sensorManager.getSensorList(Sensor.TYPE_ALL);
    for ( Sensor s : sensorList) {
        Toast.makeText( context: this, s.getName(), Toast.LENGTH_SHORT);
    }

    //Todo: Register to Listen to accelerometer sensor events from the HW
    Sensor accelerometer = sensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER);

    //Todo: React to sensor events by updating the text on the activity screen
    sensorManager.registerListener( listener: this, accelerometer, SensorManager.SENSOR_DELAY_NORMAL);
}

@Override
public void onSensorChanged(SensorEvent sensorEvent) {
    //here we get sensor
    float x = sensorEvent.values[0];
    float y = sensorEvent.values[1];
    float z = sensorEvent.values[2];

    TextView sensorTextViewX = (TextView) findViewById(R.id.sensorTextView);
    TextView sensorTextViewY = (TextView) findViewById(R.id.sensorTextView2);
    TextView sensorTextViewZ = (TextView) findViewById(R.id.sensorTextView3);
    sensorTextViewX.setText("x: " + x);
    sensorTextViewY.setText("y: " + y);
    sensorTextViewZ.setText("z: " + z);
}
```

8 Week exercises

8.1 Android app permission workflow

Flashlight App screenshots



If an app doesn't use any permissions, then nothing is needed. But an app uses permissions, then they are declared in the app's manifest regardless of the type of permission (at this step these are install-time permissions). When one declares install-time permissions in your app, the system automatically grants the app the permissions when the user installs the app. If a runtime (dangerous) permission is needed then the app needs to request this permission additionally in the app from the user before the app can access the restricted data or perform restricted actions. Many runtime permissions access private user data, a special type of restricted data that includes potentially sensitive information. Examples of private user data include location and contact information.

Example from the app using GPS location data:

Install-time permissions in the manifest file:

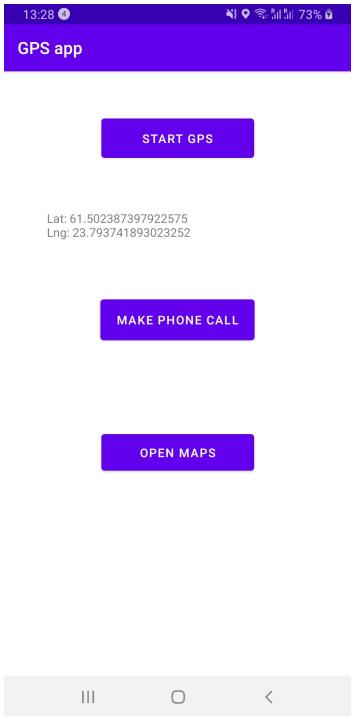
```
    <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"/>
    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
```

Requesting a runtime permission and checking if it was granted or not (in MainActivity.java file)

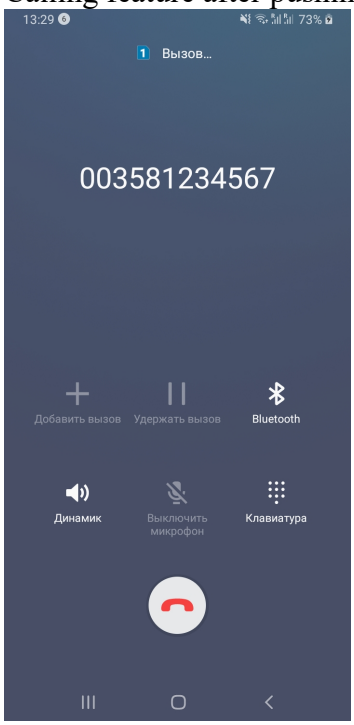
```
// Todo: Check if the user has granted permission
if (ActivityCompat.checkSelfPermission( context: this, Manifest.permission.ACCESS_FINE_LOCATION) != PackageManager.PERMISSION_GRANTED && Ac

    ActivityCompat.requestPermissions( activity: this,
            new String[] {Manifest.permission.ACCESS_FINE_LOCATION, Manifest.permission.ACCESS_COARSE_LOCATION},
             requestCode: 0
    );

    return;
}
locationMaganer.requestLocationUpdates(LocationManager.GPS_PROVIDER,  minTimeMs: 0,  minDistanceM: 0,  listener: this);
```

8.2 Android app permission workflow

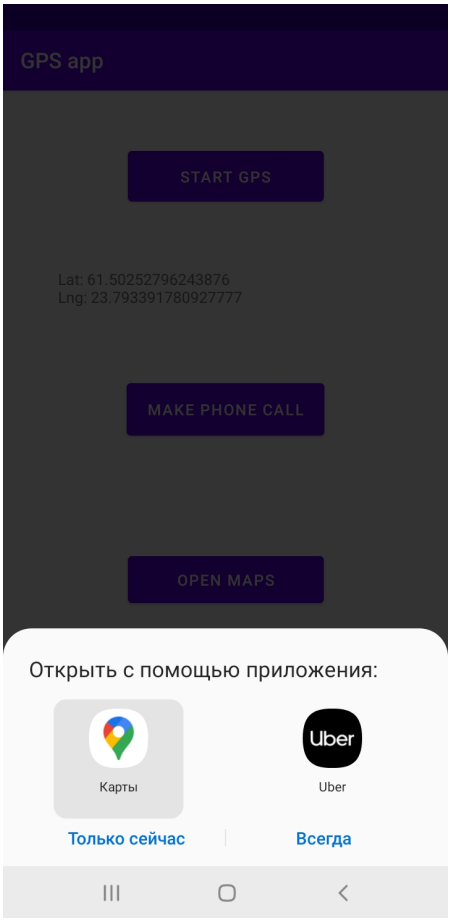
GPS feature:

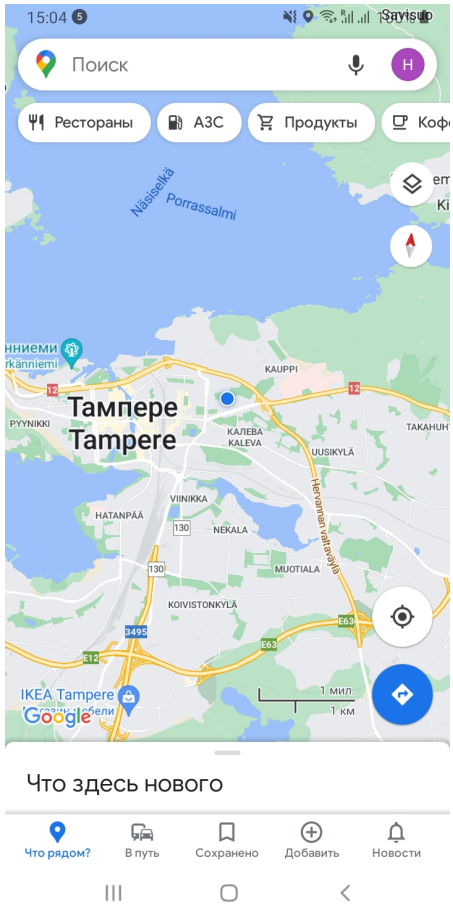


Calling feature after pushing the button:



Maps feature using GPS location data (asking which app to use and then another activity)





showMap method in MainActivity.java:

```
public void showMap(View view) {
    Intent intent = new Intent(Intent.ACTION_VIEW);
    Uri geoLocation = Uri.parse("geo:" + latitude + "," + longitude + "?z=12");

    intent.setData(geoLocation);
    if (intent.resolveActivity(getPackageManager()) != null) {
        startActivity(intent);
    }
}
```

9 Week exercises

9.1 Written answers

Android Activity lifecycle states:

An activity lifecycle is based on 7 methods: onCreate(), onStart(), onResume(), onPause(), onStop(), onRestart(), onDestroy(). Thus it can be created by onCreate() method and closed/finished by onDestroy() method.

Main activity is created when the application is launched (can be other activities which are created when the user switches to another view), and closed when the app is stopped or another activity is started.

onPause() method is implemented when the activity is going to the background (still visible), onStop() is implemented when the activity is not visible anymore. If the the acitivy is back on the foreground, then onResume() and onStart() methods are impelemented. These methods are callbacks.

Activity data bundle means all data that is stored during the activity lifecycle. For example, if an activity is destroyed at some point, data can be saved and restored later.

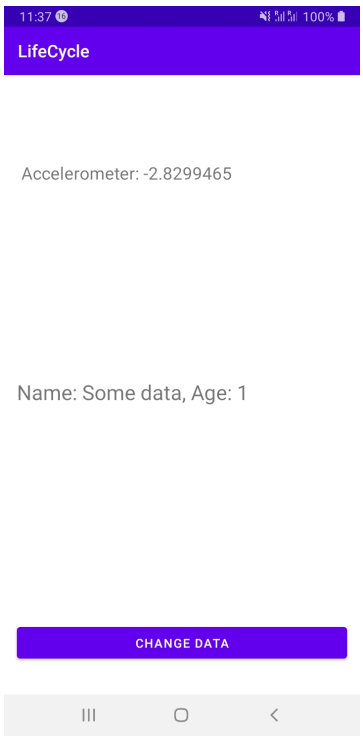
If we change the device orientation, then the activity is stopped, destroyed and recreated again, so data can be lost if it is not saved before destroying the activity.

9.2 Code

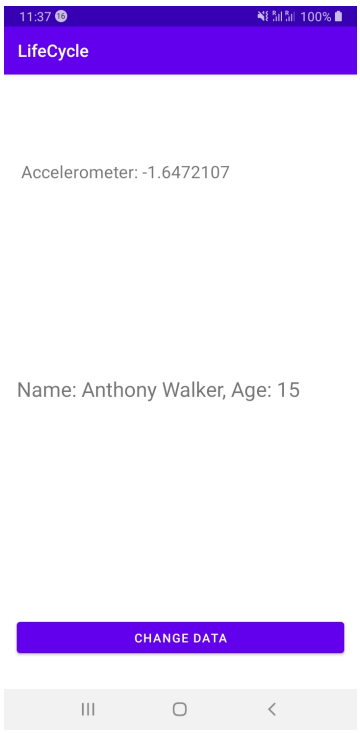
Implementation of saving/restoring local activity data whenever the device is turned and activity is destroyed and recreated. Here is the code:

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

    setContentView(R.layout.activity_main);

    TextView activityDataTextView = (TextView)findViewById(R.id.activityDataTextView);
    activityDataTextView.setText("Name: " + name + ", Age: " + age);
    sensorManager = (SensorManager) getSystemService(Context.SENSOR_SERVICE);
    accelerometer = sensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER);
    sensorManager.registerListener( listener: this, accelerometer, SensorManager.SENSOR_DELAY_NORMAL);

}

@Override
protected void onSaveInstanceState( Bundle savedInstanceState ) {
    super.onSaveInstanceState(savedInstanceState);
    savedInstanceState.putString("NAME", name);
    savedInstanceState.putInt("AGE", age);
}

@Override
protected void onRestoreInstanceState( Bundle savedInstanceState ) {
    super.onRestoreInstanceState(savedInstanceState);
    name = savedInstanceState.getString( key: "NAME");
    age = savedInstanceState.getInt( key: "AGE");
    TextView activityDataTextView = (TextView)findViewById(R.id.activityDataTextView);
    activityDataTextView.setText("Name: " + name + ", Age: " + age);

}
```

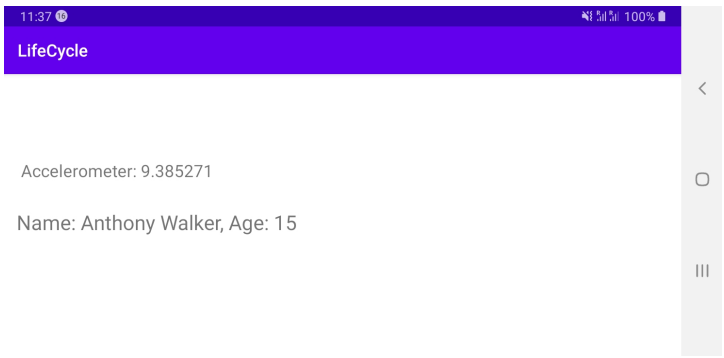
Initial screen:



After pushing the button:



After changing portrait orientation to landscape:



Implementation of unregistering and registering to sensors as a listener:

```
59
60      @Override
61      protected void onPause() {
62          super.onPause();
63          //stopping listening
64          sensorManager.unregisterListener( listener: this, accelerometer);
65      }
66
67      @Override
68      protected void onResume() {
69          super.onResume();
70          //resuming listening to accelerometer
71          sensorManager.registerListener( listener: this, accelerometer, SensorManager.SENSOR_DELAY_NORMAL);
72      }
73
74      @Override
75      public void onAccuracyChanged(Sensor sensor, int i) {
76      }
77
78      @Override
79      public void onSensorChanged(SensorEvent sensorEvent) {
80          float xSensor = sensorEvent.values[0];
81          TextView accelerometerTextView = (TextView)findViewById(R.id.accelerometerTextView);
82          accelerometerTextView.setText("Accelerometer: " + xSensor);
83
84      }
85  }
```

Final project

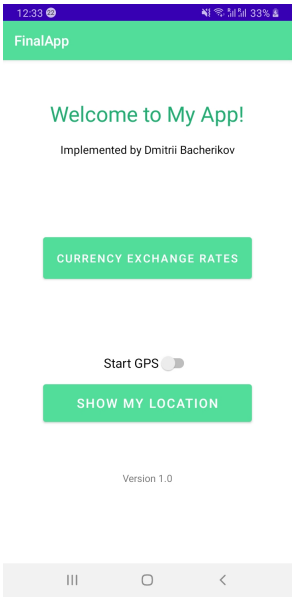
Description of the project: the Android application consists of 2 activities (main screen and Currency exchange rates activity) + using GPS sensor with opportunity to open one more activity – showing the current location of the user in the Google Maps application. Currency exchange rates activity will use REST API to parse rates as of current date.

Current version: 1.0

Programming language: Java/XML

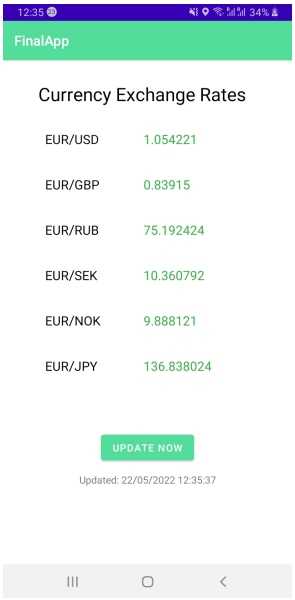
Github link: https://github.com/Dialex2006/AndroidAppDevelopment

The main screen activity looks like this:



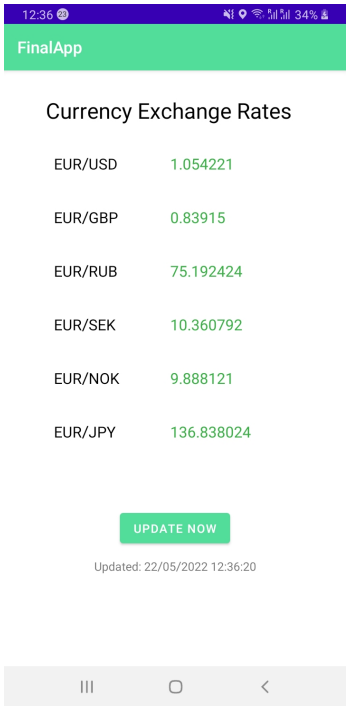
On this page the user has 2 opportunities:

1. Start another activity (Currency Exchange Rates), Intent is used:



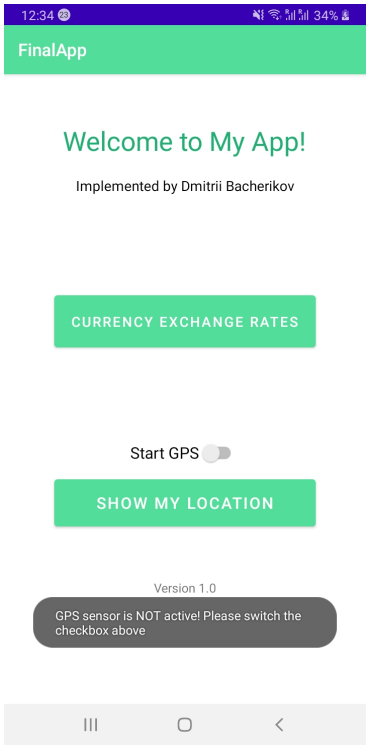
On this page the user has possibility to request update manually, data is parsed using REST API in JSON (URL: https://api.exchangerate.host/latest)

On this screen after clicking “Update now” button we can see that time of update has been changed:

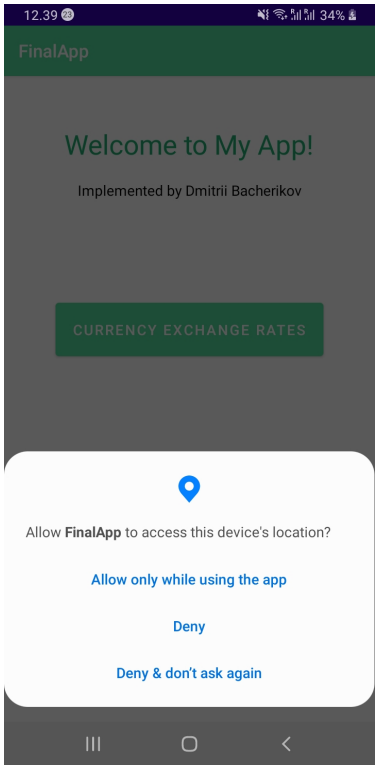


2. Show My Location

If the user wants to show one’s location, first one needs to turn on GPS switch “Start GPS” to start the sensor and to make sure the user granted required permission. If the switch is not turned on, then the app won’t allow to open Google Maps app.

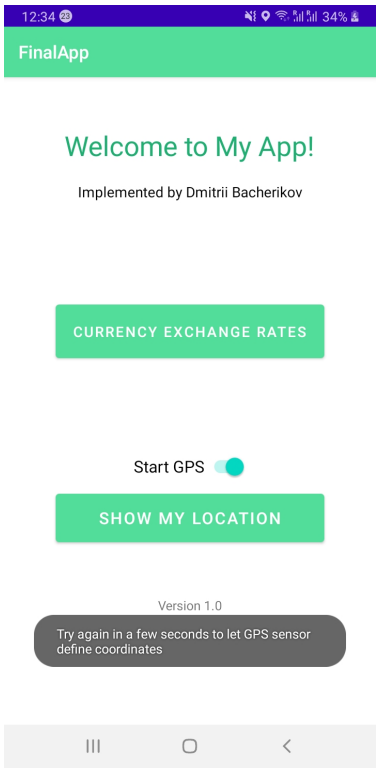


If the user turns on the switch, the app will ask for permission from the user because this permission is ‘dangerous’:



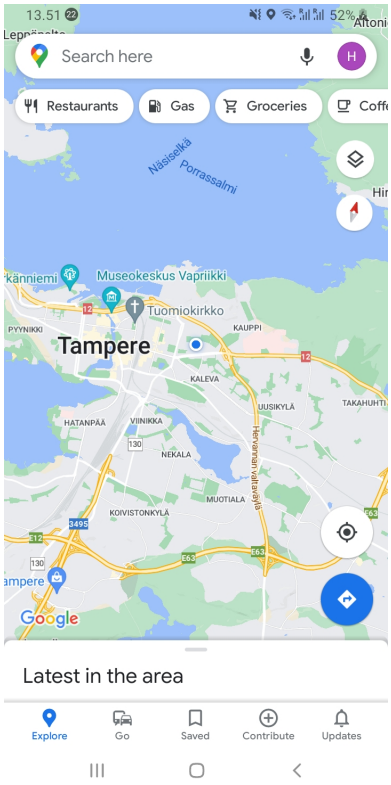
After the user allows to access the device’s location, the user can click the button “Show My Location”, but here 2 options are possible:

If GPS hasn’t defined location yet, then it will show this message:

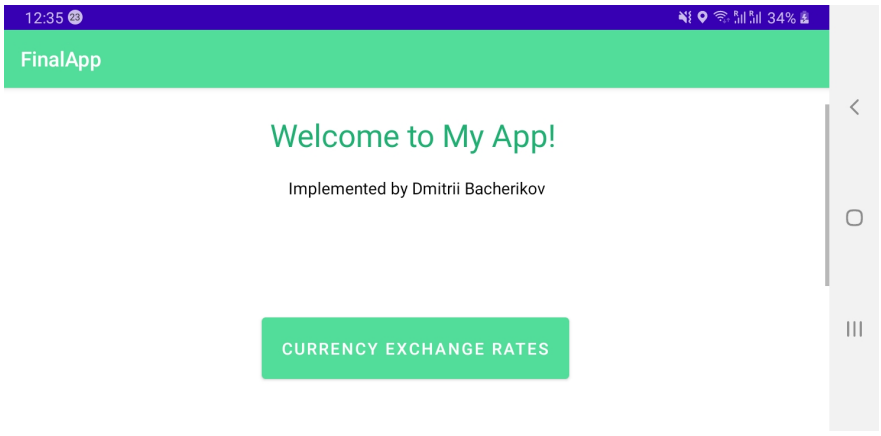


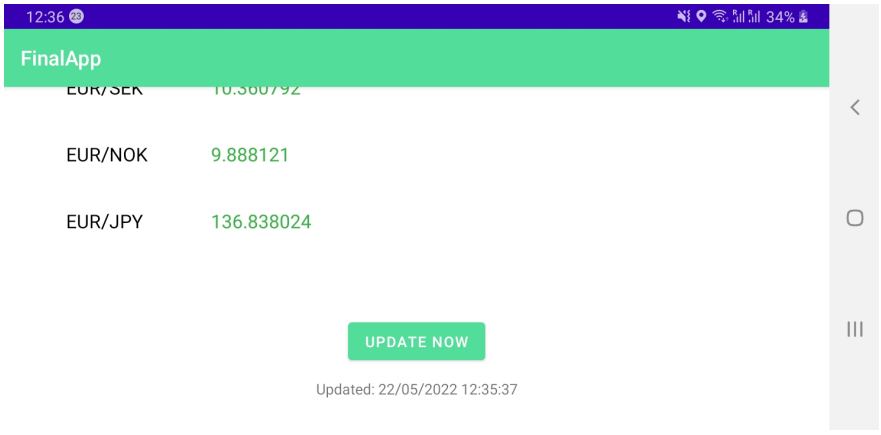
It means that the sensor needs a bit more time to define location or the device is located inside the building where GPS is not accessible. If the sensor has defined coordinates,

then the app will send location coordinates and show location in the Google Maps app (Intent is used here):



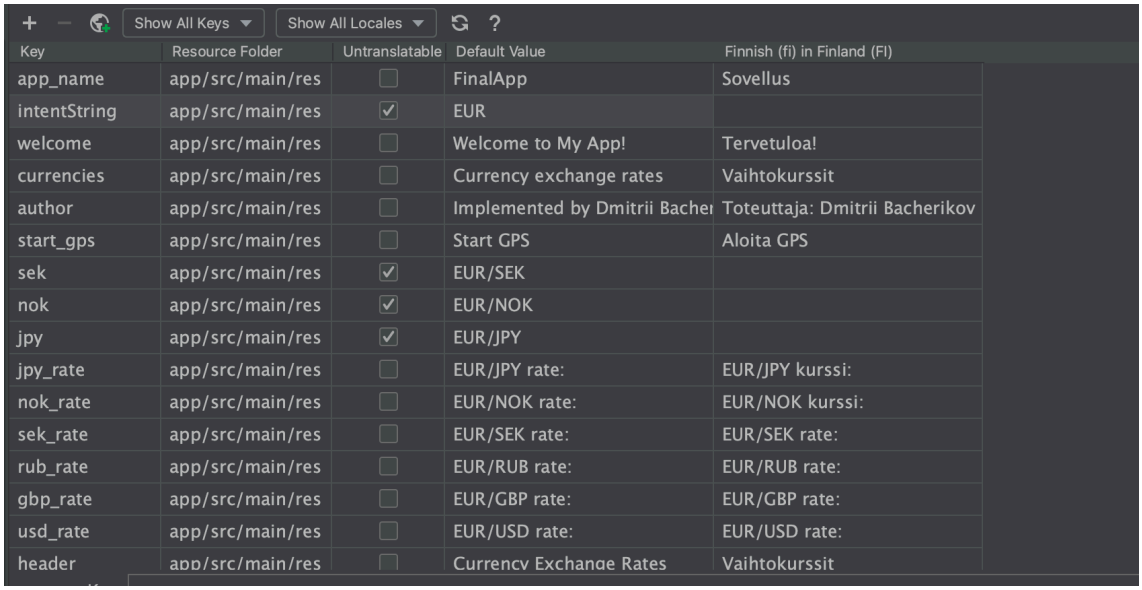
In my application opportunity of changing orientation (e.g. from portrait to landscape) is taken into account. In this case the user can scroll the screen:





In case of changing orientation activities are stopped, destroyed and re-created, therefore **onCreate(), onResume(), onSaveInstanceState(), onRestoreInstanceState()** are overridden to restore our data.

In version 1.0 all strings, colors and themes use resource files instead of hard coding, which is the right way make the app. Besides the English version (by default) there is the Finnish localization using Translations Editor:



Sources used with exercises

1. https://stackoverflow.com/questions/54273412/failed-to-install-the-following-android-sdk-packages-as-some-licences-have-not

2. https://stackoverflow.com/questions/16596877/android-studio-doesnt-see-device
3. https://developer.android.com/guide/components/activities/activity-lifecycle
4. https://java-programming.mooc.fi
5. https://google.github.io/volley/
6. https://api.openweathermap.org
7. https://developer.android.com/reference/android/hardware/camera2/CameraManager.TorchCallback
8. https://developer.android.com/guide/components/intents-common
9. https://stackoverflow.com/questions/14274815/calling-maps-intent-with-geolocation-is-it-possible-to-have-a-pushpin-on-it
10. https://developer.android.com/guide/topics/sensors/sensors_overview#java
11. https://www.section.io/engineering-education/understanding-and-implementing-the-android-lifecycle/
12. https://developer.android.com/reference/android/widget/Switch
13. https://developer.android.com/reference/android/widget/ScrollView
14. https://developer.android.com/topic/libraries/architecture/saving-states
15. https://stackoverflow.com/questions/151777/how-can-i-save-an-activity-state-using-the-save-instance-state