

Mobile App Development 1

Study diary

by Dmitrii Bacherikov

Contents

1	Week exercises	3
1.1	Android Studio setup and Hello World	3
2	Week exercises	4
What is activity and what is the role of XML and Java files when implementing GUI?	4	
3	Week exercises	6
4	Week exercises	9
4.1	Address class.....	9
5	Week exercises	12
5.1	App GUI with 2 screens.....	12
5.2	Written exercises.....	14
6	Week exercises	15
6.1	Fetching HTTP data and parsing JSON.....	15
6.2	Fetching HTTP data and parsing JSON.....	15
7	Week exercises	17
7.1	Synchronous device API Example – Camera HW and “torch app”	17
7.2	Asynchronous API call example (listener) – Sensor API and “Level App” 18	
8	Week exercises	19
8.1	Android app permission workflow	19
8.2	Android app permission workflow	20
9	Week exercises	22
9.1	Written answers	22
9.2	Code	22
	Sources used with exercises	30

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

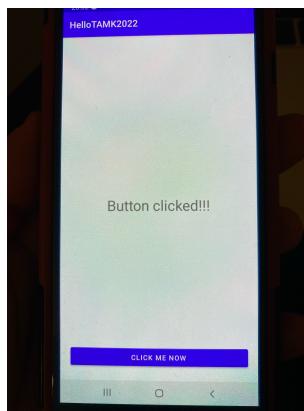
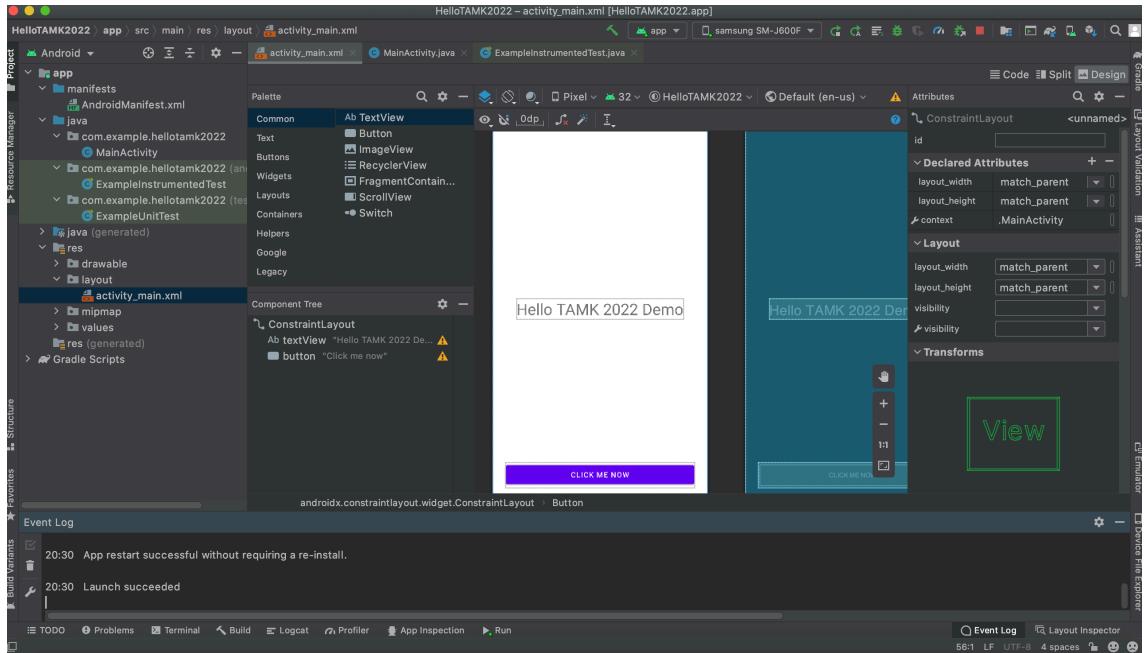
I have installed Android Studio in MacOS (Java runtime was installed previously already).

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?

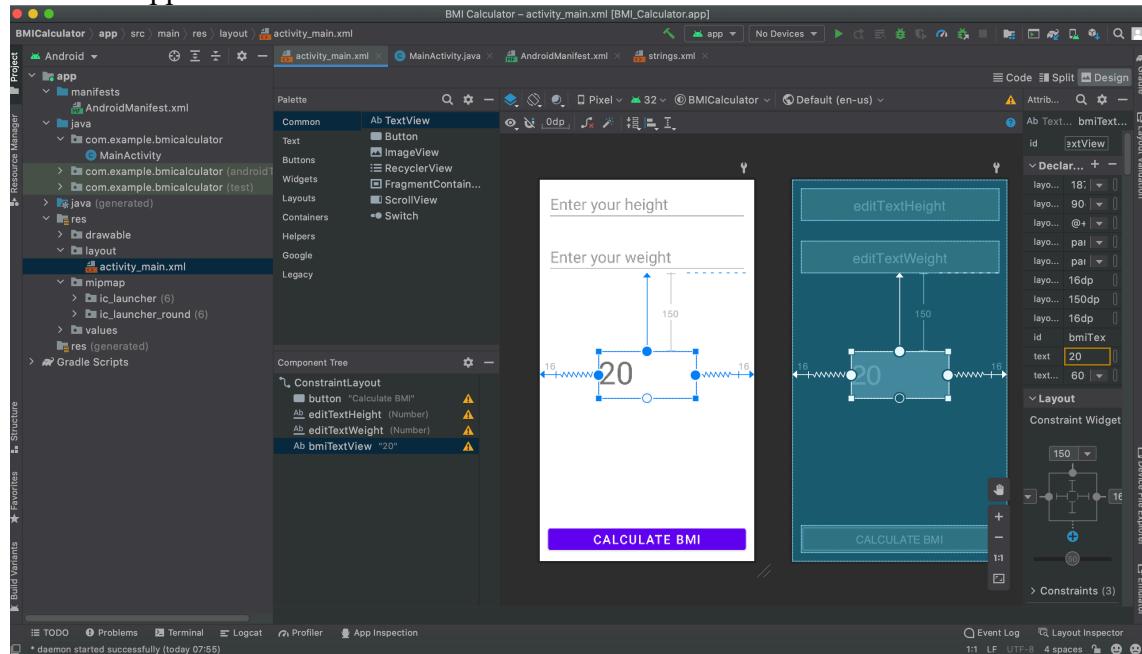
Activity is a specific screen of an application (can be multiple activities in one application) which consists of XML and Java files. XML is for GUI layout (design), and Java is for logic.

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





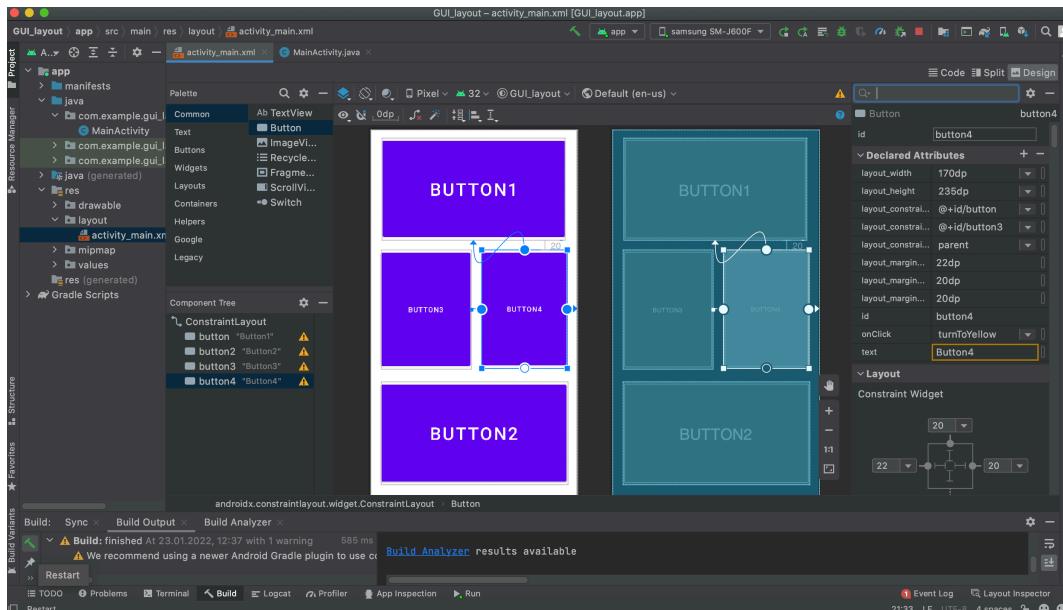
179

72

22,5



2.3 Create GUI layout with Constraint layout



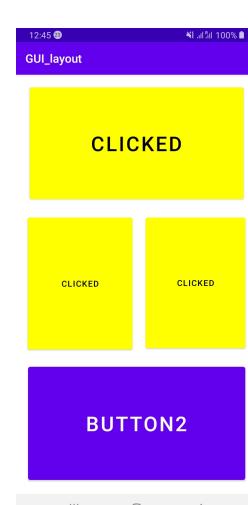
```

    package com.example.gui_layout;
    import androidx.appcompat.app.AppCompatActivity;
    import android.graphics.Color;
    import android.os.Bundle;
    import android.view.View;
    import android.widget.Button;

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

        public void turnToYellow(View view) {
            Button button1 = (Button) view;
            button1.setBackgroundColor(Color.YELLOW);
            button1.setText("CLICKED");
            button1.setTextColor(Color.BLACK);
        }
    }

```



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

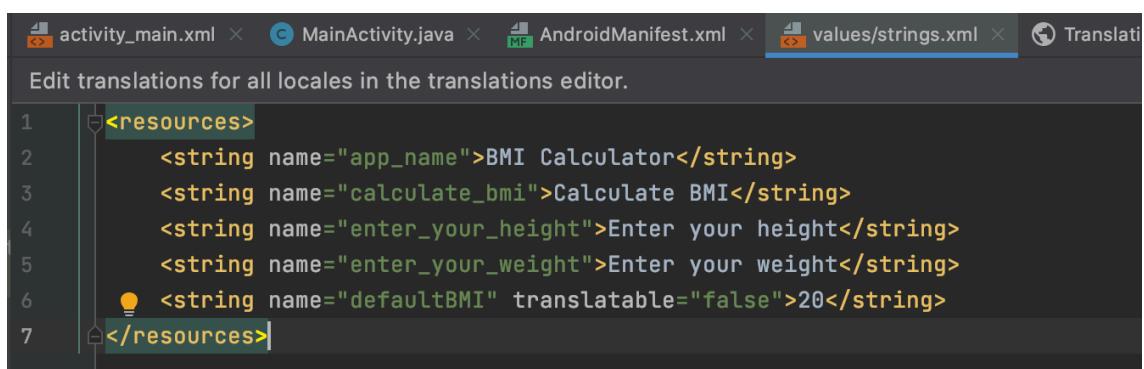
- Less code is needed and no hardcoded
- Easy to design layout using XML in Android Studio
- Easy to localize for different devices
- Easier to maintain
- It is faster

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

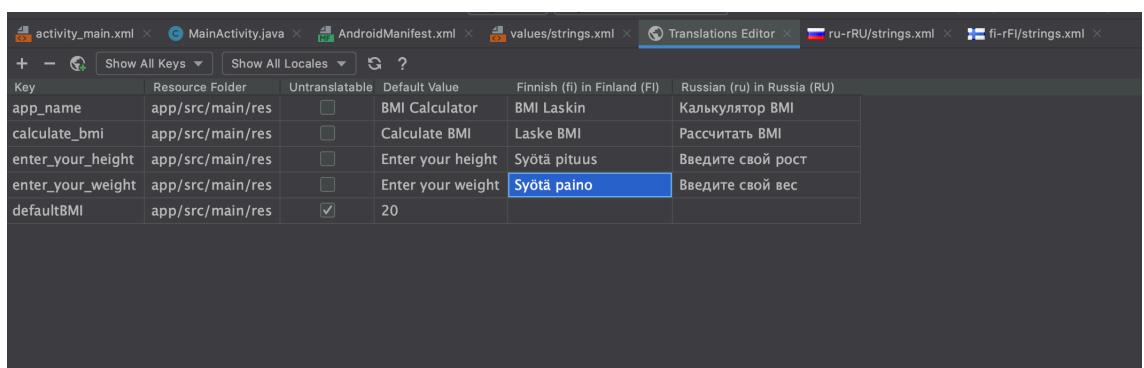


```

<resources>
    <string name="app_name">BMI Calculator</string>
    <string name="calculate_bmi">Calculate BMI</string>
    <string name="enter_your_height">Enter your height</string>
    <string name="enter_your_weight">Enter your weight</string>
    <string name="defaultBMI" translatable="false">20</string>
</resources>

```

Added 2 more languages (Russian and Finnish)



Key	Resource Folder	Untranslatable	Default Value	Finnish (fi) in Finland (FI)	Russian (ru) in Russia (RU)
app_name	app/src/main/res	<input type="checkbox"/>	BMI Calculator	BMI Laskin	Калькулятор BMI
calculate_bmi	app/src/main/res	<input type="checkbox"/>	Calculate BMI	Laske BMI	Рассчитать BMI
enter_your_height	app/src/main/res	<input type="checkbox"/>	Enter your height	Syötä pituus	Введите свой рост
enter_your_weight	app/src/main/res	<input type="checkbox"/>	Enter your weight	Syötä paino	Введите свой вес
defaultBMI	app/src/main/res	<input checked="" type="checkbox"/>	20		



20



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

```

1  public class Student {
2      private String sName;
3      private int sAge;
4      private String sID;
5      private int sCredits;
6
7      public Student() {
8          System.out.println("Default constructor was activated");
9      }
10
11     public Student(String aName, int aAge, String aID) {
12         sName = aName;
13         sAge = aAge;
14         sID = aID;
15         sCredits = 0;
16     }
17
18     public void printStudentData() {
19         System.out.println("Name: " + this.sName);
20         System.out.println("Age: " + this.sAge);
21         System.out.println("ID: " + this.sID);
22         System.out.println("Credit points: " + this.sCredits);
23     }
24
25     public void addCreditPoints(int addedCredits) {
26         this.sCredits += addedCredits;
27         System.out.println("Student " + this.sName + " received " + addedCredits + " credit points");
28     }
29 }

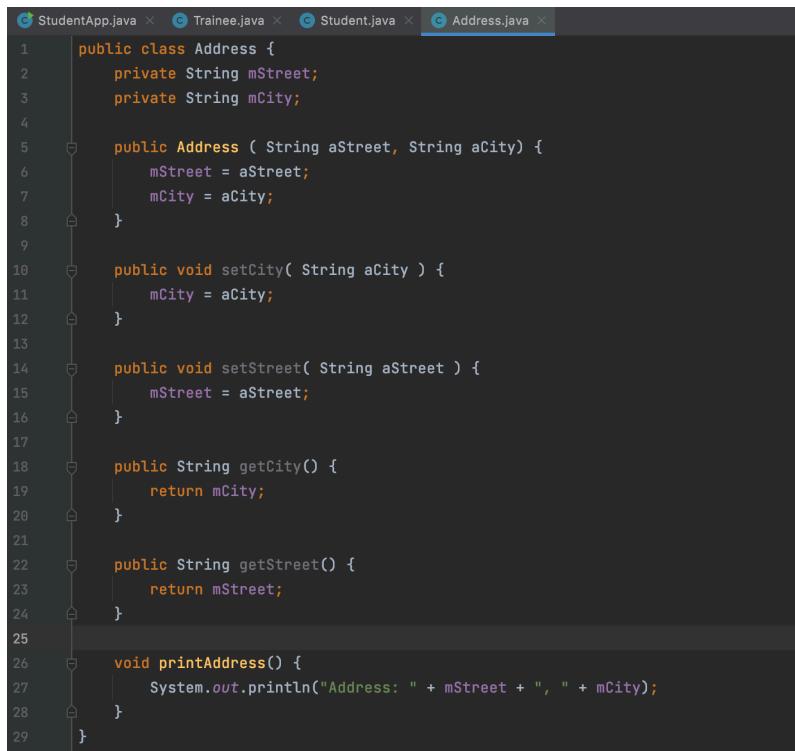
```

```
1  public class Main {  
2      Run | Debug  
3      public static void main( String args[] ) {  
4          System.out.println("Test of default constructor:");  
5          Student st1 = new Student();  
6          System.out.println();  
7  
8          Student st2 = new Student ("Jack", 25, "ID1");  
9          Student st3 = new Student ("Mike", 21, "ID2");  
10         Student st4 = new Student ("Angela", 18, "ID3");  
11  
12         st2.printStudentData();  
13         System.out.println();  
14         st3.printStudentData();  
15         System.out.println();  
16         st4.printStudentData();  
17         System.out.println();  
18         st2.addCreditPoints(5);  
19         System.out.println();  
20         st2.printStudentData();  
21     }  
22 }
```

```
[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



```

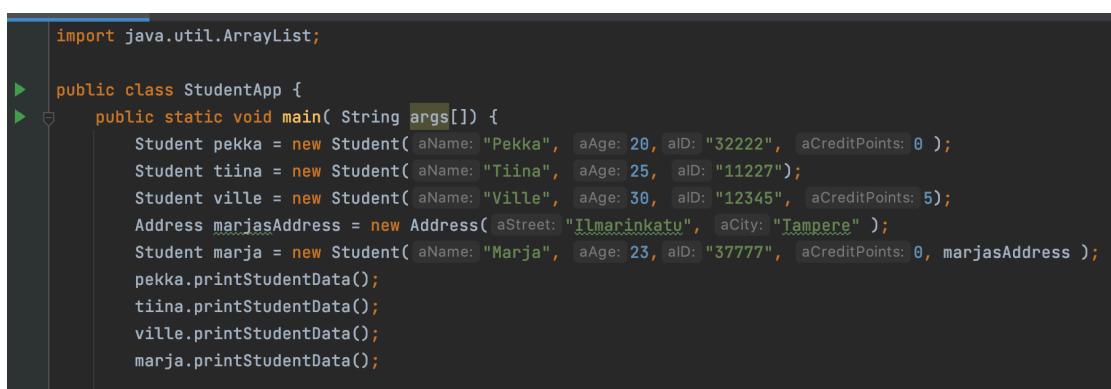
1  public class Address {
2      private String mStreet;
3      private String mCity;
4
5      public Address ( String aStreet, String aCity) {
6          mStreet = aStreet;
7          mCity = aCity;
8      }
9
10     public void setCity( String aCity ) {
11         mCity = aCity;
12     }
13
14     public void setStreet( String aStreet ) {
15         mStreet = aStreet;
16     }
17
18     public String getCity() {
19         return mCity;
20     }
21
22     public String getStreet() {
23         return mStreet;
24     }
25
26     void printAddress() {
27         System.out.println("Address: " + mStreet + ", " + mCity);
28     }
29 }
```

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:

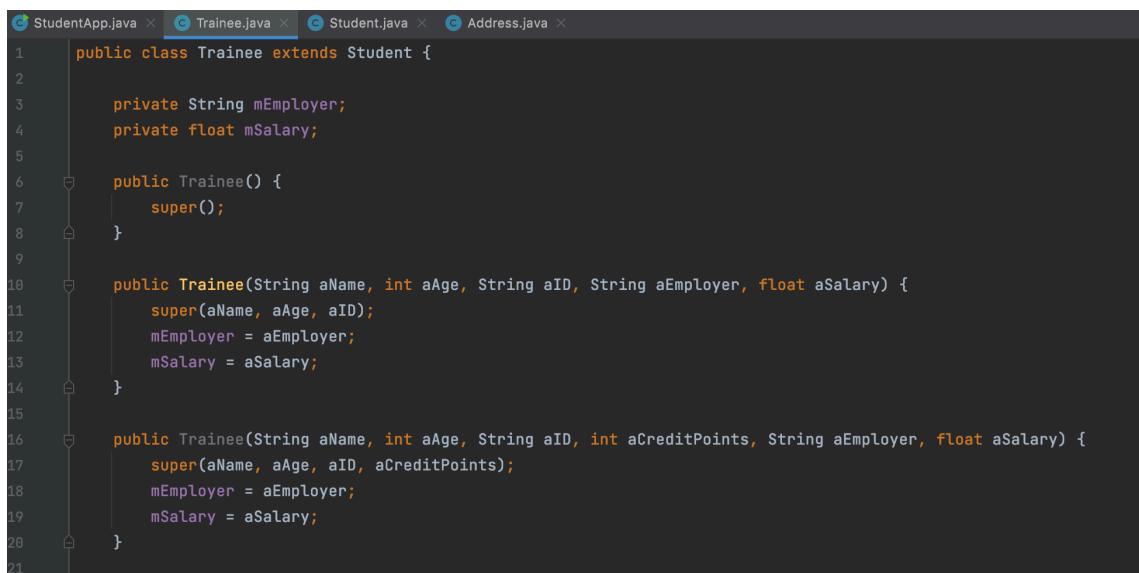


```

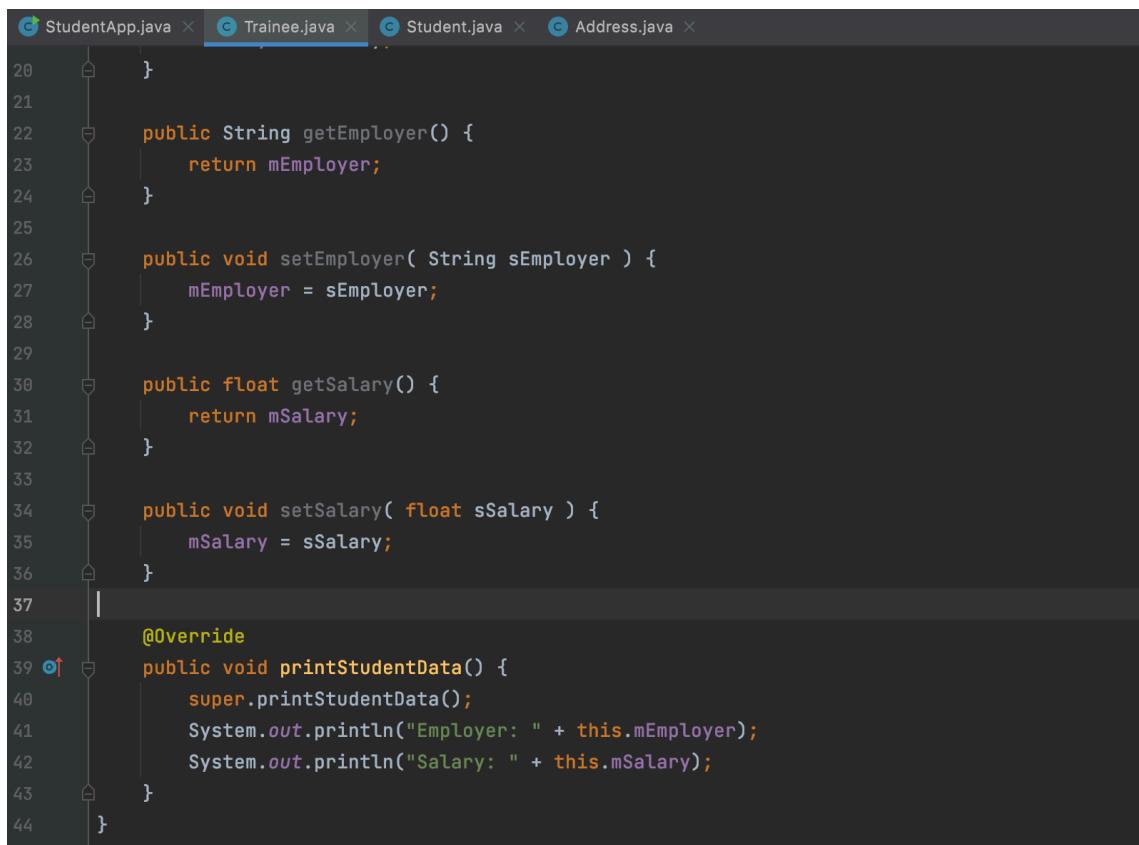
import java.util.ArrayList;

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



```
1  public class Trainee extends Student {
2
3      private String mEmployer;
4      private float mSalary;
5
6      public Trainee() {
7          super();
8      }
9
10     public Trainee(String aName, int aAge, String aID, String aEmployer, float aSalary) {
11         super(aName, aAge, aID);
12         mEmployer = aEmployer;
13         mSalary = aSalary;
14     }
15
16     public Trainee(String aName, int aAge, String aID, int aCreditPoints, String aEmployer, float aSalary) {
17         super(aName, aAge, aID, aCreditPoints);
18         mEmployer = aEmployer;
19         mSalary = aSalary;
20     }
21 }
```



```
20 }
21
22     public String getEmployer() {
23         return mEmployer;
24     }
25
26     public void setEmployer( String sEmployer ) {
27         mEmployer = sEmployer;
28     }
29
30     public float getSalary() {
31         return mSalary;
32     }
33
34     public void setSalary( float sSalary ) {
35         mSalary = sSalary;
36     }
37
38     @Override
39     public void printStudentData() {
40         super.printStudentData();
41         System.out.println("Employer: " + this.mEmployer);
42         System.out.println("Salary: " + this.mSalary);
43     }
44 }
```

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));

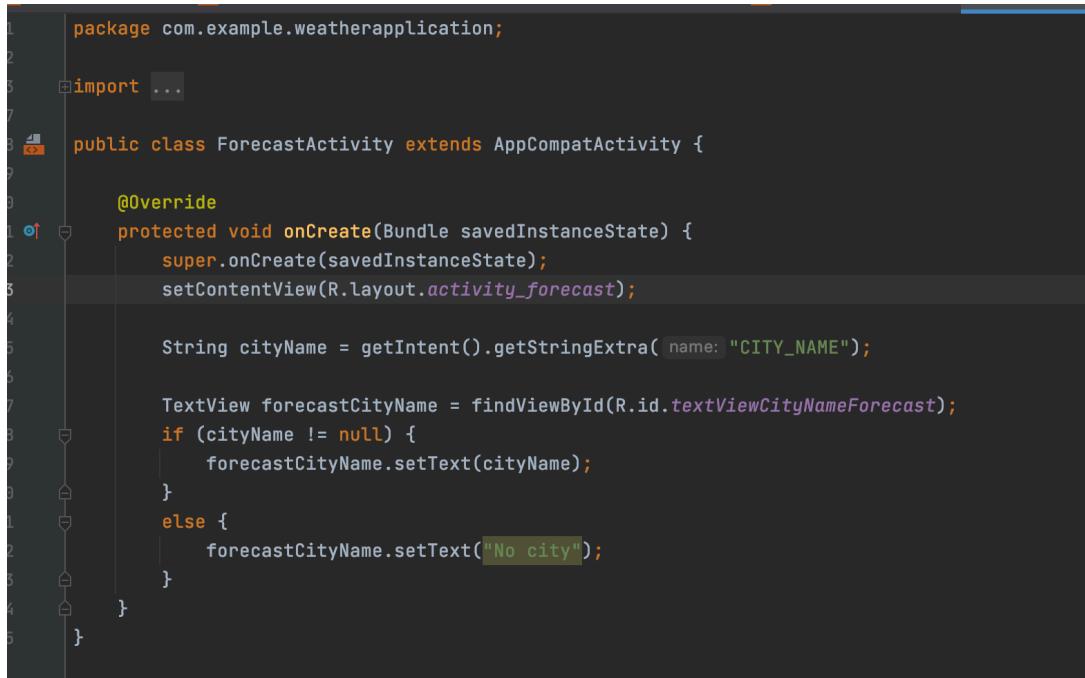
for (Student s:students) {
    s.printStudentData();
}

}
```

5 Week exercises

5.1 App GUI with 2 screens

Creating the second activity

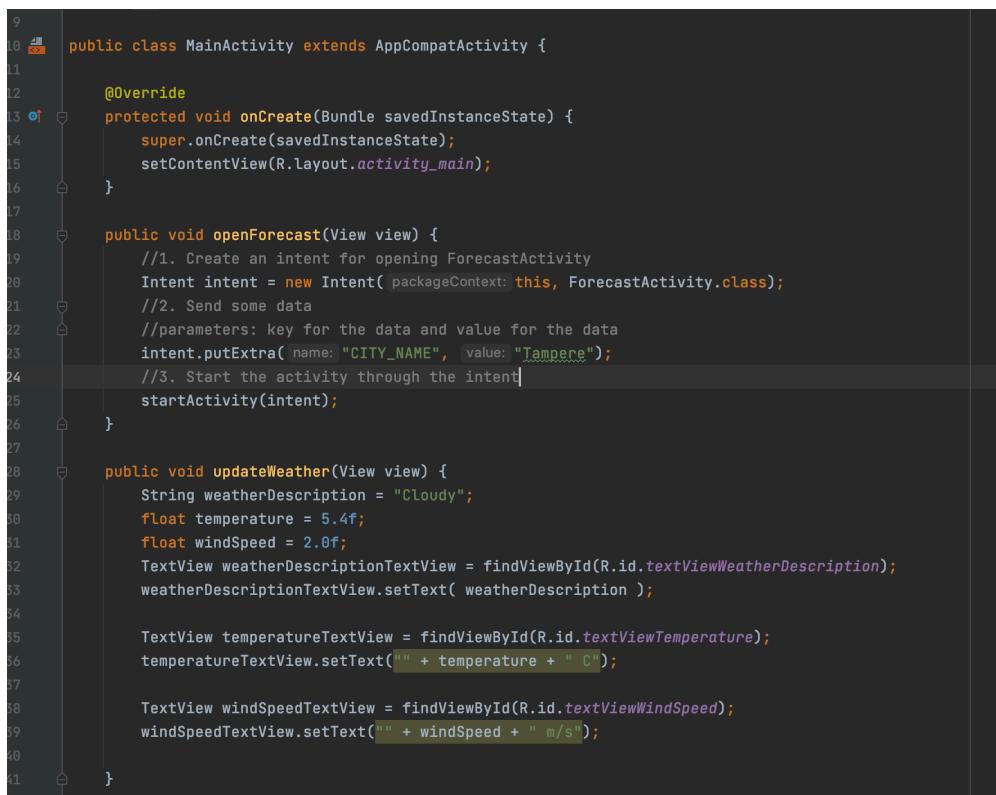


```

1 package com.example.weatherapplication;
2
3 import ...
4
5 public class ForecastActivity extends AppCompatActivity {
6
7     @Override
8     protected void onCreate(Bundle savedInstanceState) {
9         super.onCreate(savedInstanceState);
10        setContentView(R.layout.activity_forecast);
11
12        String cityName = getIntent().getStringExtra("CITY_NAME");
13
14        TextView forecastCityName = findViewById(R.id.textViewCityNameForecast);
15        if (cityName != null) {
16            forecastCityName.setText(cityName);
17        } else {
18            forecastCityName.setText("No city");
19        }
20    }
21}

```

Creating the intent

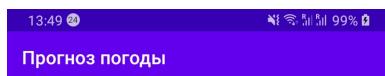


```

9
10 public class MainActivity extends AppCompatActivity {
11
12     @Override
13     protected void onCreate(Bundle savedInstanceState) {
14         super.onCreate(savedInstanceState);
15         setContentView(R.layout.activity_main);
16     }
17
18     public void openForecast(View view) {
19         //1. Create an intent for opening ForecastActivity
20         Intent intent = new Intent(getApplicationContext(), ForecastActivity.class);
21         //2. Send some data
22         //parameters: key for the data and value for the data
23         intent.putExtra("CITY_NAME", "Tampere");
24         //3. Start the activity through the intent
25         startActivity(intent);
26     }
27
28     public void updateWeather(View view) {
29         String weatherDescription = "Cloudy";
30         float temperature = 5.4f;
31         float windSpeed = 2.0f;
32         TextView weatherDescriptionTextView = findViewById(R.id.textViewWeatherDescription);
33         weatherDescriptionTextView.setText(weatherDescription);
34
35         TextView temperatureTextView = findViewById(R.id.textViewTemperature);
36         temperatureTextView.setText(" " + temperature + " C");
37
38         TextView windSpeedTextView = findViewById(R.id.textViewWindSpeed);
39         windSpeedTextView.setText(" " + windSpeed + " m/s");
40     }
41 }

```

Application screens (localized Russian version):



Тампере

Солнечно

6 С

3 м/с

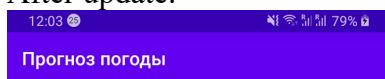
Тампере

ОБНОВИТЬ

ПРОГНОЗ

||| □ <

After update:



Тампере

Пасмурно

5.4 С

2.0 м/с

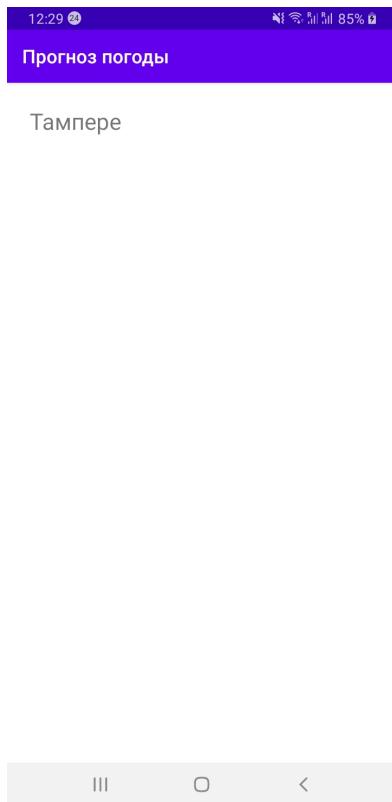
Тампере

ОБНОВИТЬ

ПРОГНОЗ

||| □ <

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, response, Toast.LENGTH_LONG).show();
51
52                 //parsing the data from the JSON Object
53                 parseJsonAndUpdateUI(response);
54
55             }, error -> {
56                 Toast.makeText(context, "Error!", Toast.LENGTH_LONG).show();
57             }
58         );
59         //Sending the request by adding it to the queue
60         queue.add(stringRequest);
61

```

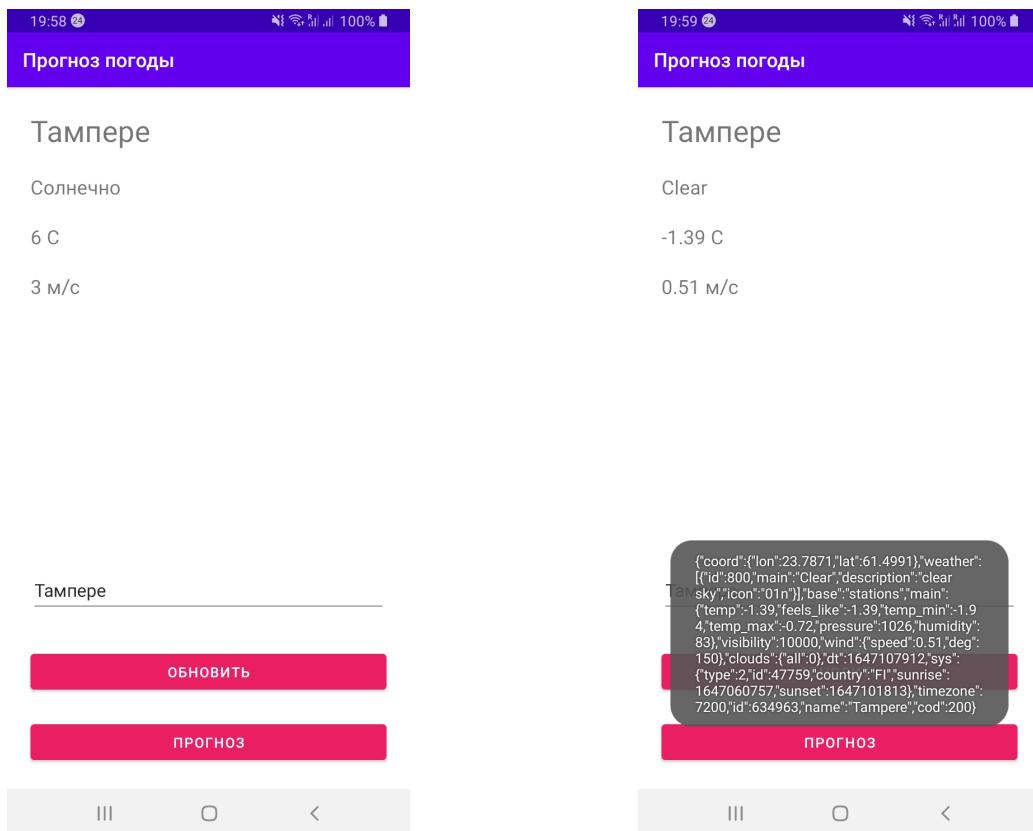
Then we parse data from JSON object:

```

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("temp");
86             weatherDescription = weather.getJSONArray("weather").getJSONObject(0).getString("main");
87             windSpeed = weather.getJSONObject("wind").getDouble("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));

```

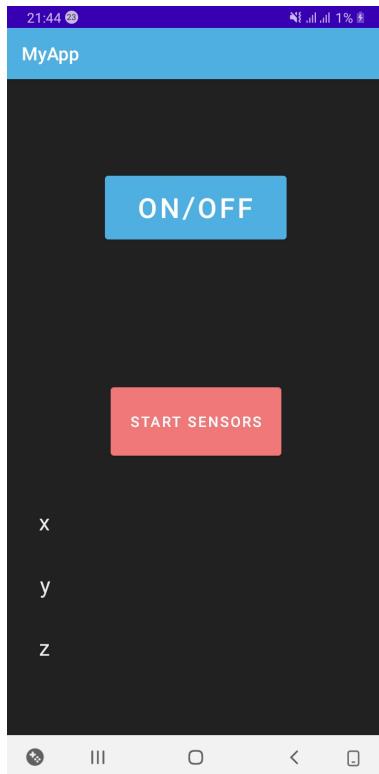
Here are screenshots of the app:



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 {

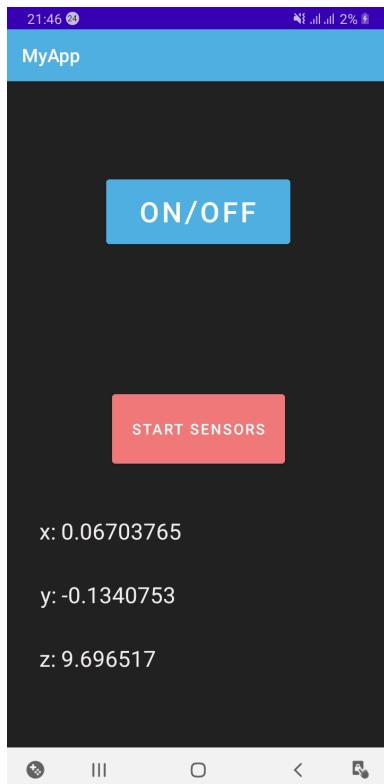
        //Todo: ask if there is a camera, which has flashlight
        for (String id : cameraManager.getCameraIdList()) {
            CameraCharacteristics cameraChar = cameraManager.getCameraCharacteristics(id);
            if (cameraChar.get(CameraCharacteristics.FLASH_INFO_AVAILABLE)) {

                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



```

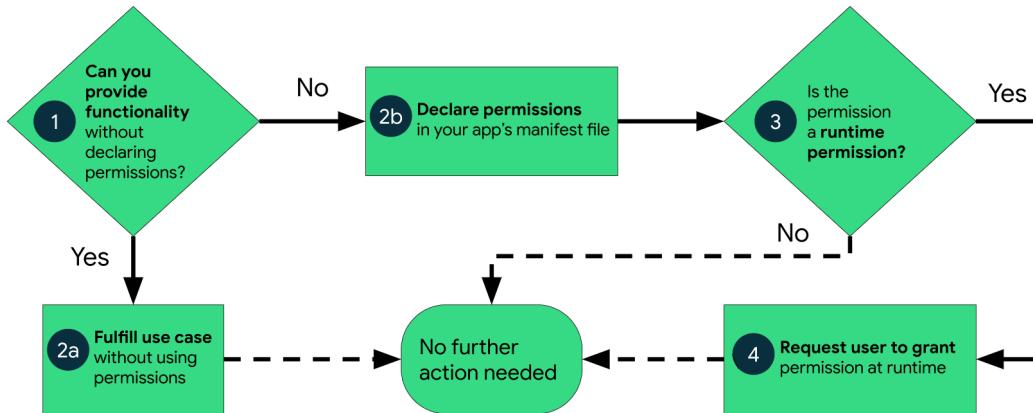
58     public void startSensors(View view) {
59         //Start listening
60
61         //Todo: connect sensor Manager
62         SensorManager sensorManager = (SensorManager) getSystemService(Context.SENSOR_SERVICE);
63         List<Sensor> sensorList = sensorManager.getSensorList(Sensor.TYPE_ALL);
64         for ( Sensor s : sensorList) {
65             Toast.makeText( context: this, s.getName(), Toast.LENGTH_SHORT);
66         }
67
68         //Todo: Register to Listen to accelerometer sensor events from the HW
69         Sensor accelerometer = sensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER);
70
71         //Todo: React to sensor events by updating the text on the activity screen
72         sensorManager.registerListener( listener: this, accelerometer, SensorManager.SENSOR_DELAY_NORMAL);
73     }
74
75     @Override
76     public void onSensorChanged(SensorEvent sensorEvent) {
77         //here we get sensor
78         float x = sensorEvent.values[0];
79         float y = sensorEvent.values[1];
80         float z = sensorEvent.values[2];
81
82         TextView sensorTextViewX = (TextView) findViewById(R.id.sensorTextView);
83         TextView sensorTextViewY = (TextView) findViewById(R.id.sensorTextView2);
84         TextView sensorTextViewZ = (TextView) findViewById(R.id.sensorTextView3);
85         sensorTextViewX.setText("x: " + x);
86         sensorTextViewY.setText("y: " + y);
87         sensorTextViewZ.setText("z: " + z);
88     }

```

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, Manifest.permission.ACCESS_FINE_LOCATION) != PackageManager.PERMISSION_GRANTED && ActivityCompat.checkSelfPermission(context, Manifest.permission.ACCESS_COARSE_LOCATION) != PackageManager.PERMISSION_GRANTED) {
    ActivityCompat.requestPermissions(activity, new String[]{Manifest.permission.ACCESS_FINE_LOCATION, Manifest.permission.ACCESS_COARSE_LOCATION}, requestCode);
}

return;
}

locationManager.requestLocationUpdates(LocationManager.GPS_PROVIDER, minTimeMs, minDistanceM, listener);
```

8.2 Android app permission workflow

GPS feature:



START GPS

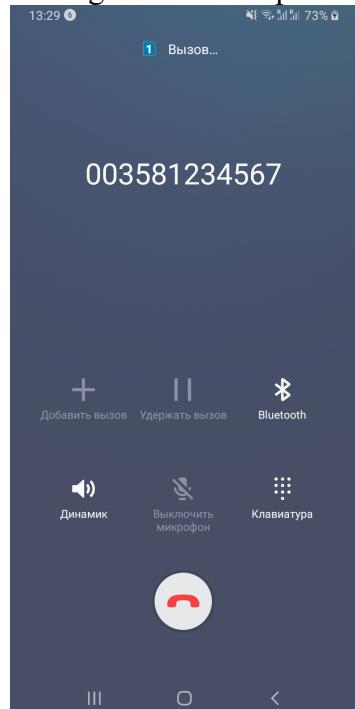
Lat: 61.502387397922575
Lng: 23.793741893023252

MAKE PHONE CALL

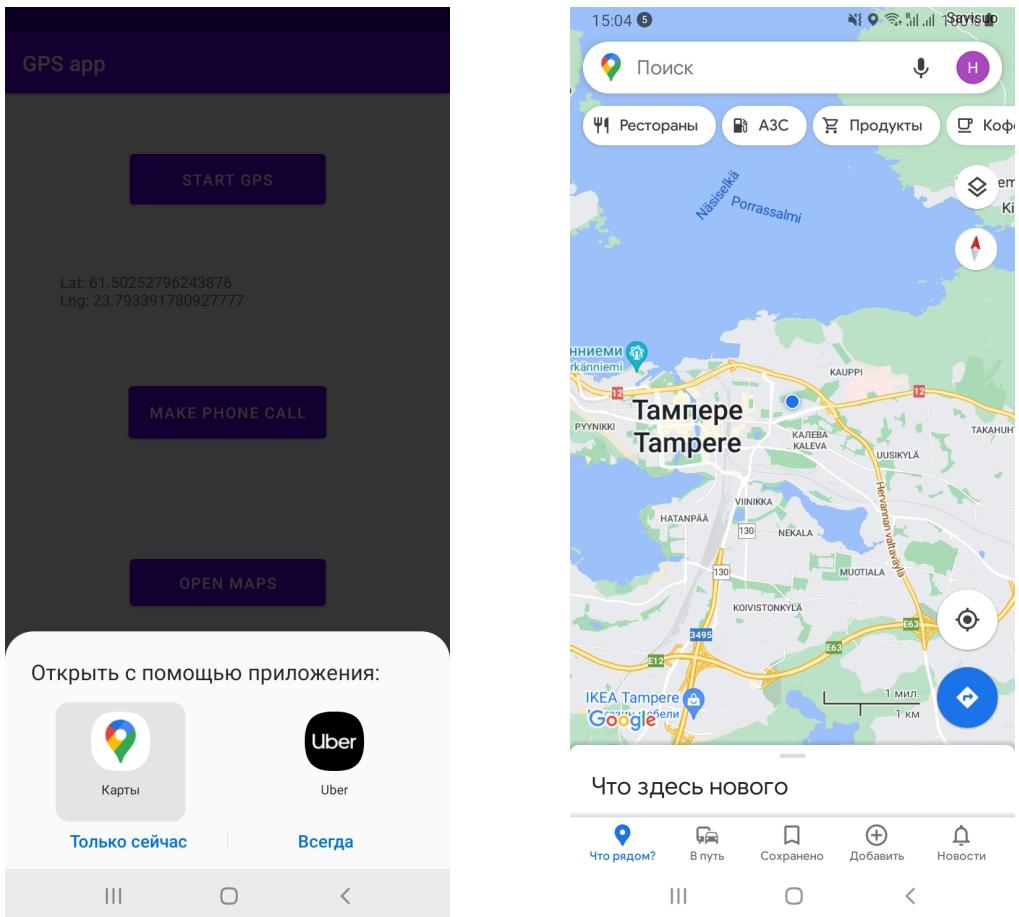
OPEN MAPS

III O <

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 activity is back on the foreground, then onResume() and onStart() methods are implemented. 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 activityTextView = (TextView) findViewById(R.id.activityTextView);
    activityTextView.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 activityTextView = (TextView) findViewById(R.id.activityTextView);
    activityTextView.setText("Name: " + name + ", Age: " + age);
}

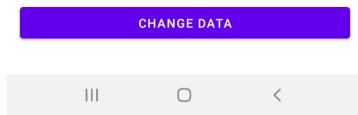
```

Initial screen:



Accelerometer: -2.8299465

Name: Some data, Age: 1

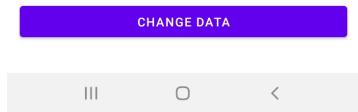


After pushing the button:

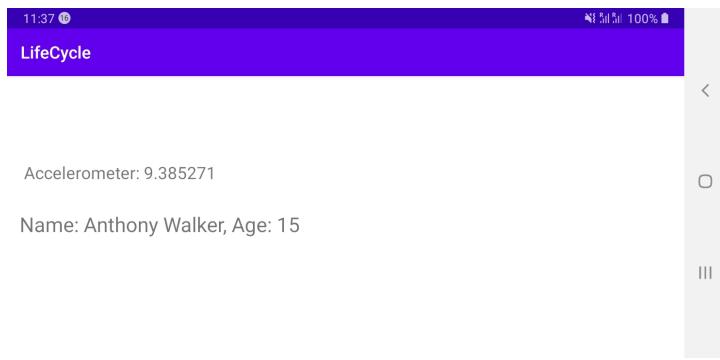


Accelerometer: -1.6472107

Name: Anthony Walker, Age: 15



After changing portrait orientation to landscape:



Implementation of unregistering and registering to sensors as a listener:

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

Gitlab link:

The main screen activity looks like this:



Welcome to My App!

Implemented by Dmitrii Bacherikov

CURRENCY EXCHANGE RATES

Start GPS

SHOW MY LOCATION

Version 1.0

III O <

On this page the user has 2 opportunities:

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



Currency Exchange Rates

EUR/USD 1.054221

EUR/GBP 0.83915

EUR/RUB 75.192424

EUR/SEK 10.360792

EUR/NOK 9.888121

EUR/JPY 136.838024

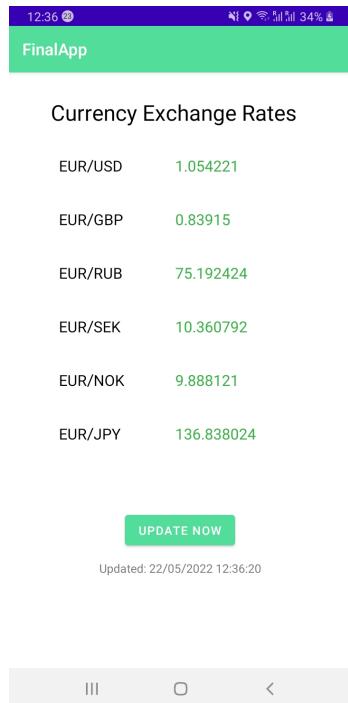
UPDATE NOW

Updated: 22/05/2022 12:35:37

III O <

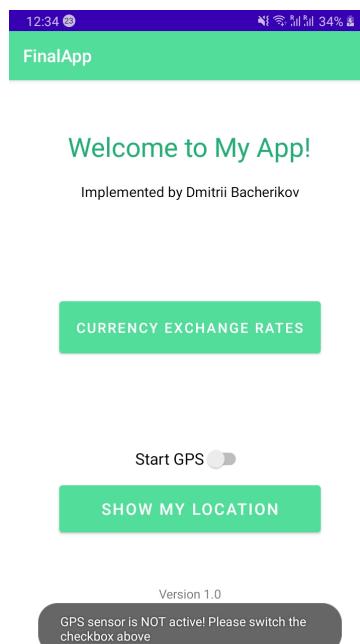
On this page the user has possibility to request update manually, data is parsed using REST API in JSON (URL: <https://api.exchangerate.host/latest>)

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

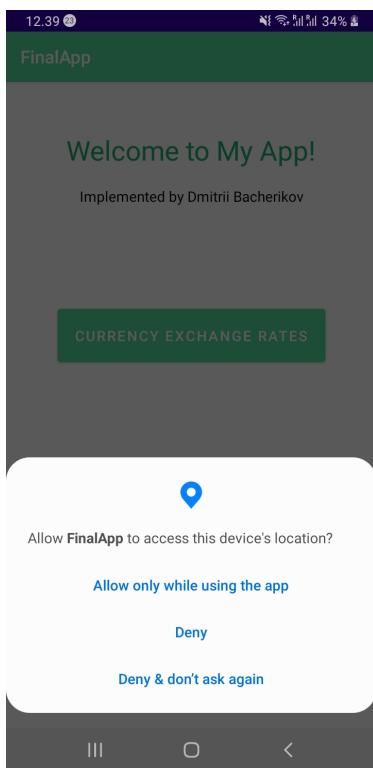


2. Show My Location

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.

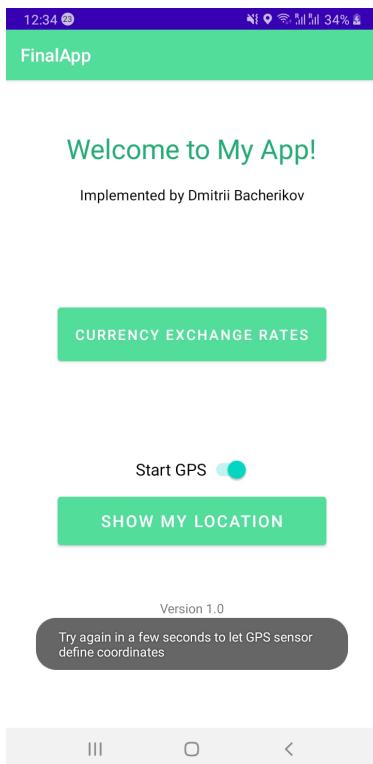


If the user turns on the switch, the app will ask for permission from the user because this permission is ‘dangerous’:



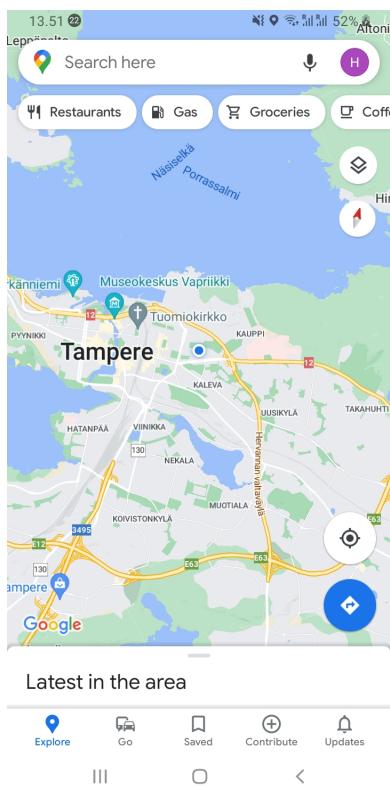
After the user allows to access the device’s location, the user can click the button “Show My Location”, but here 2 options are possible:

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

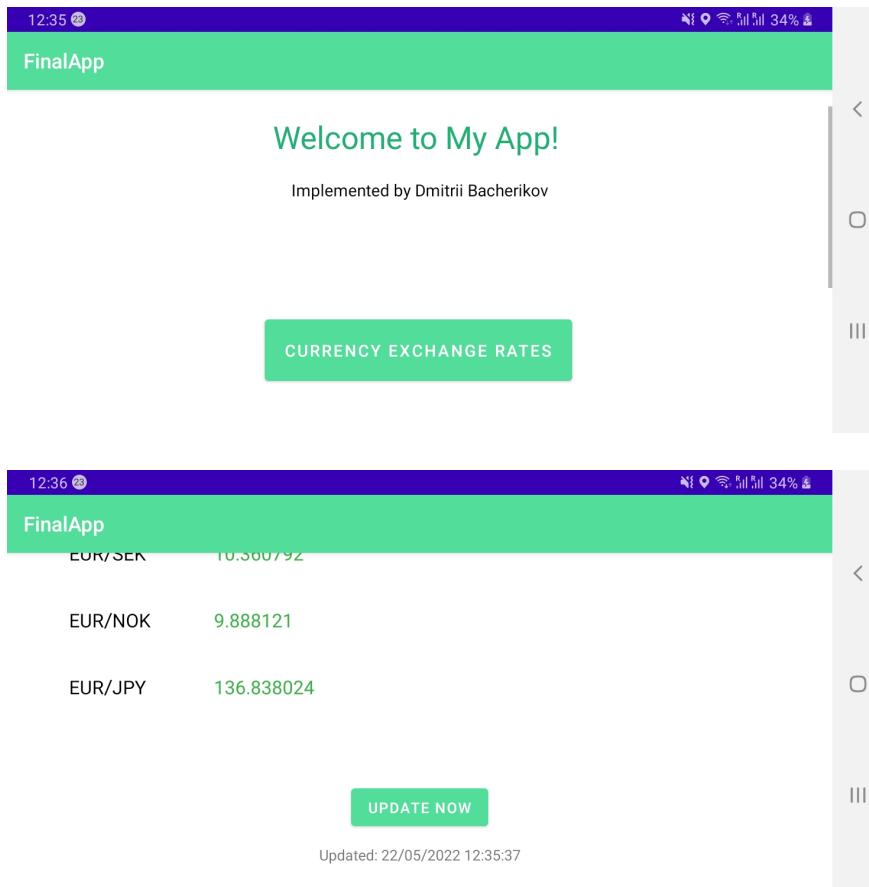


It means that the sensor needs a bit more time to define location or the device is located inside the building where GPS is not accessible. If the sensor has defined coordinates,

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:

Key	Resource Folder	Untranslatable	Default Value	Finnish (fi) in Finland (FI)
app_name	app/src/main/res	<input type="checkbox"/>	FinalApp	Sovellus
intentString	app/src/main/res	<input checked="" type="checkbox"/>	EUR	
welcome	app/src/main/res	<input type="checkbox"/>	Welcome to My App!	Tervetuloa!
currencies	app/src/main/res	<input type="checkbox"/>	Currency exchange rates	Vaihtokurssit
author	app/src/main/res	<input type="checkbox"/>	Implemented by Dmitrii Bacher	Toteuttaja: Dmitrii Bacherikov
start_gps	app/src/main/res	<input type="checkbox"/>	Start GPS	Aloita GPS
sek	app/src/main/res	<input checked="" type="checkbox"/>	EUR/SEK	
nok	app/src/main/res	<input checked="" type="checkbox"/>	EUR/NOK	
jpy	app/src/main/res	<input checked="" type="checkbox"/>	EUR/JPY	
jpy_rate	app/src/main/res	<input type="checkbox"/>	EUR/JPY rate:	EUR/JPY kurssi:
nok_rate	app/src/main/res	<input type="checkbox"/>	EUR/NOK rate:	EUR/NOK kurssi:
sek_rate	app/src/main/res	<input type="checkbox"/>	EUR/SEK rate:	EUR/SEK rate:
rub_rate	app/src/main/res	<input type="checkbox"/>	EUR/RUB rate:	EUR/RUB rate:
gbp_rate	app/src/main/res	<input type="checkbox"/>	EUR/GBP rate:	EUR/GBP rate:
usd_rate	app/src/main/res	<input type="checkbox"/>	EUR/USD rate:	EUR/USD rate:
header	app/src/main/res	<input type="checkbox"/>	Currency Exchange Rates	Vaihtokurssit

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>