Creating mobile applications in Android Studio

Creating mobile applications

1. Getting started

Introduction

Android is most popular mobile operating system developed by Google for phones, Tv's, tablets and more .

These materials were created for workshops for beginning users. They were written in the form of instructions.

What do we need?

Android Studio with the SDK bundle for your platform. You can freely download it here:

http://developer.android.com/sdk/index.html

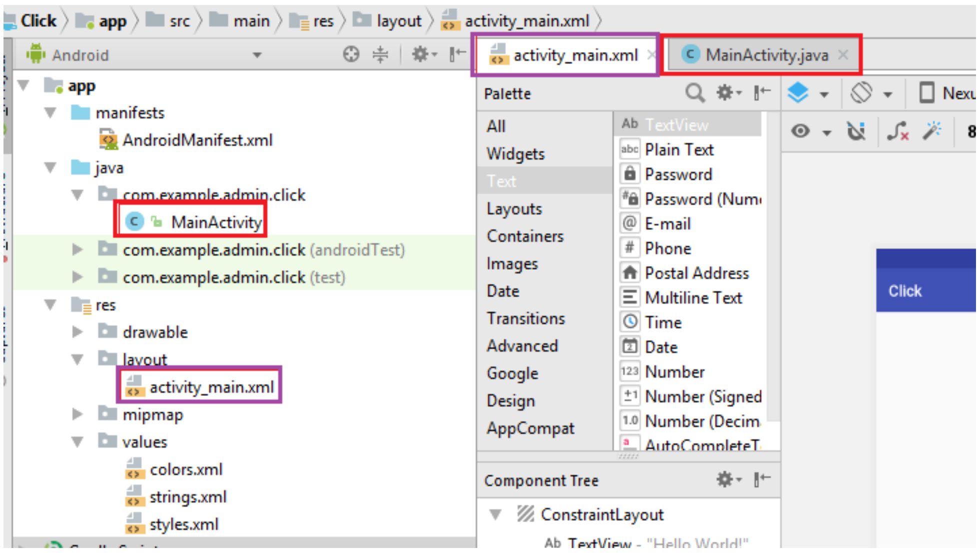
Goals of this exercice:

- ✔ setting up an Android Studio Project
- ✔ auto import of libraries
- ✔ running first app
- ✔ changing colors of application
- ✔ changing name of application

Project start:

1. Open Android Studio.
2. Select **File/New/NewProject** .
3. Follow the wizard.
    a) Put aplication name: **First** (Application name should start with uppercase letter)
    b) Make sure you select the **Phone and Tablet**
    c) A minimum SDK level: **API 23**
    d) Select the **Blank Activity** template and click on **Finish**.
    e) After a brief pause, the IDE will open up.

Main files:



1. You can see two tabs on your main window :
   a) **activity_main.xml**- xml file witch includes information about aplication layout of our Main Activity.

      We can view this file in ***design*** or ***text mode***.

   b) **MainActivity.java**. - java class for Main activity, responsable for app behavior

      You can also find this two files in Project structure in the left side of our project.

Auto import of library:

While programming you can see red underlined instruction. You can push **Alt+Enter** everytime you will see it or you can turn on Auto Import of Libraries at the begining of your work.

How to set up Auto Import of Libraries?:

- ✔ Push buttons **Ctrl+Alt+S.**
- ✔ In Setting Window click **Editor/General/Auto Import**
- ✔ Turn on option **Optimalize imports on the fly**
- ✔ Click button **Apply** and then **OK** button

Running the app:

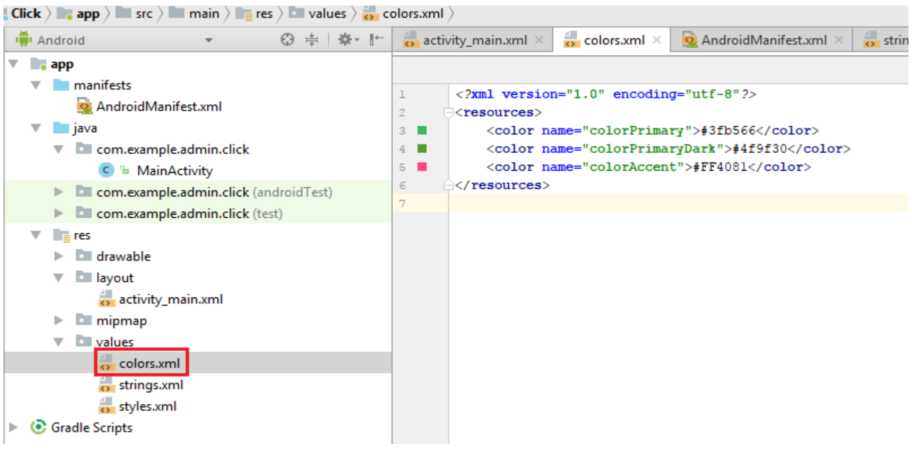
1. Select **Run/Run app**
2. Pick **Connected** or **Virtual Device.**

   If there is no device, click **Create New Virtual Device.**

   Its faster to use **Connected device**. To do that you must turn on **Programing options** and **USB debbuging** on your phone
3. Push **ok** button
4. After a while you will see our app running

Changing colors of the layout:

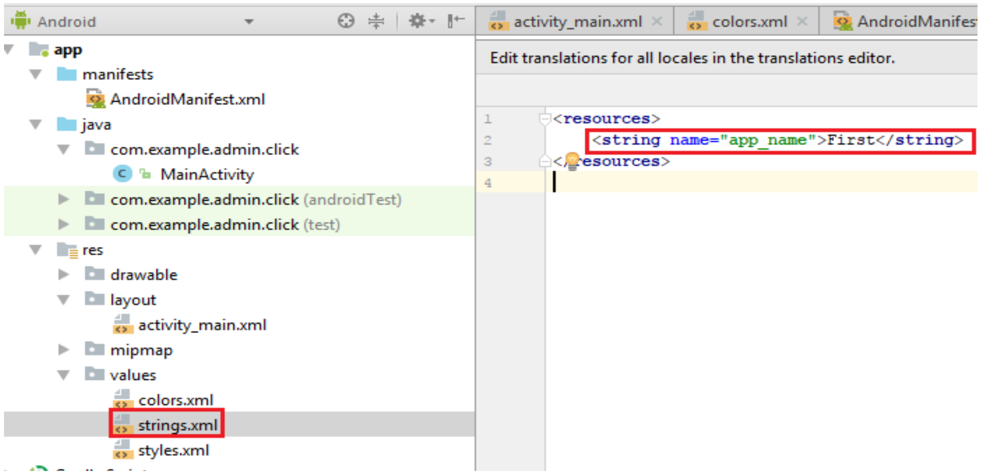
✔ Click Android/**app/res/values/colors.xml** in Project structure window on the left side of our project.



```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="colorPrimary">#3fb566</color>
    <color name="colorPrimaryDark">#4f9f30</color>
    <color name="colorAccent">#FF4081</color>
</resources>
```

Changing label of the project:

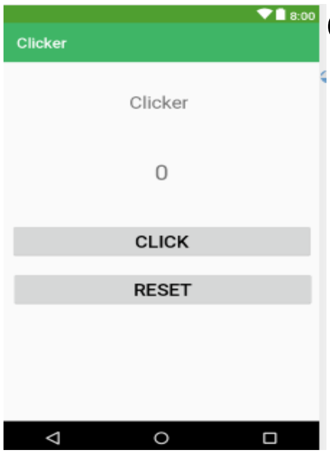
✔ *Click **Android/app/res/values/string.xml** in Project Structure window on the left side of our project.*



Edit translations for all locales in the translations editor.

```
<resources>
    <string name="app_name">First</string>
</resources>
```

2. TextView Component



Clicker app

- ✔ After pushing CLICK button, counter will increase its values and show it on the screen
- ✔ After pushing RESET button, counter will be reset and it will start counting from 0

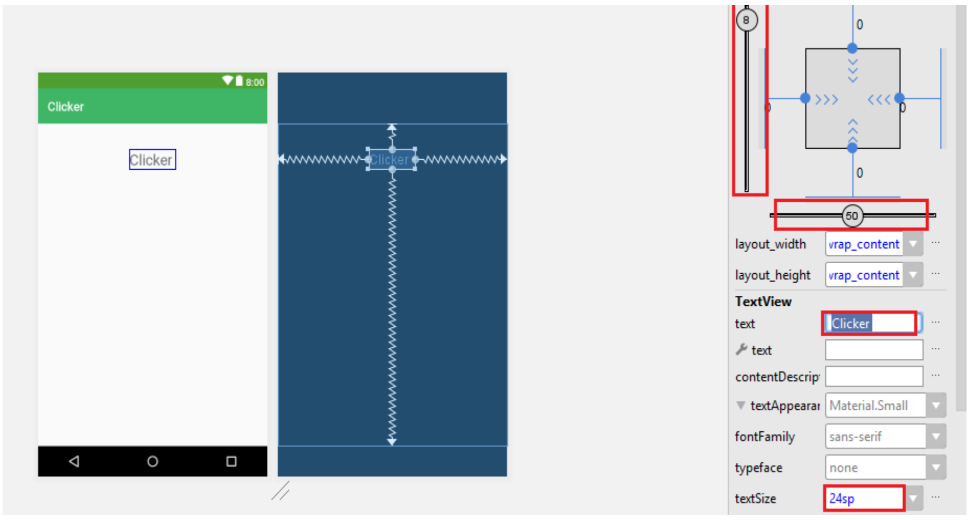
Goals of this exercice:

- ✔ Make an Clicker app
- ✔ Using TextView component and intger variables
- ✔ Learn how to sign method to the button

Clicker App Project:

1. Start a new Android Studio Project:

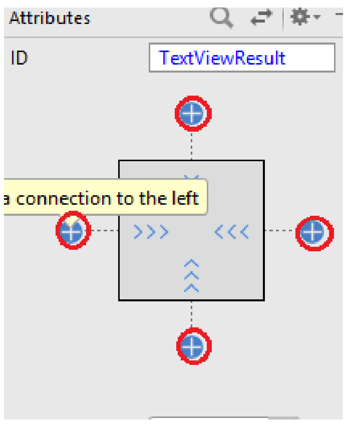
a) name:	**Clicker**
b) device:	**Phone and Tablet**
c) API:	**23**
d) Activity:	**Empty**


layout_width: wrap_content
layout_height: wrap_content
TextView
text: Clicker
text:
contentDescrip:
textAppearance: Material.Small
fontFamily: sans-serif
typeface: none
textSize: 24sp

2. Make a layout of your app as shown at picture below. Open ***design mode*** of the

activity_main.xml file:

a) Change label of the TextView „Hello World” to „Clicker”, text size 24.

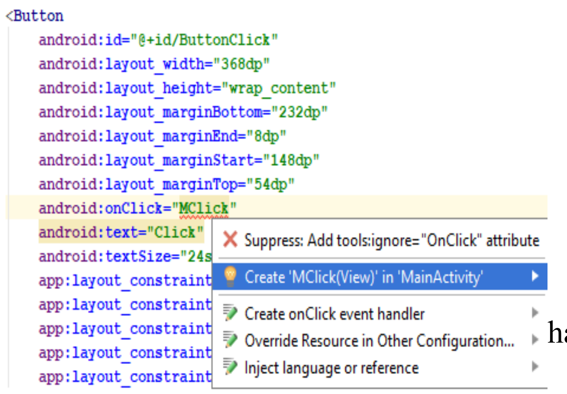


b) Put another TextView component:

- ID: **TextViewResult**
- name: **„0”**
- textSize: **24**
- Click spots showed on picture, it will create ConstraintLayout. If you don't do it every component of your layout can be shown in the left top side, not on the place we put it in our project.

c) Put button to our project:

- ID: **ButtonClick**
- name: **Click**
- textSize: **24**
- create ConstraintLayout like before
- onClick: **MClick** (The name of the metod signed to this button)
- Change mode in activity_main.xml from **design** to **text** .
- Look for the button Click section.
- Push **Alt+Enter** on the name of the method **MClick**
- Pick **Create MClick in Main Activity** from context menu. You have just created method **MClick** in **MainActivity.java** file



```
<Button
    android:id="@+id/ButtonClick"
    android:layout_width="368dp"
    android:layout_height="wrap_content"
    android:layout_marginBottom="232dp"
    android:layout_marginEnd="8dp"
    android:layout_marginStart="148dp"
    android:layout_marginTop="54dp"
    android:onClick="MClick"
    android:text="Click"
    android:textSize="24s
    app:layout_constraint
    app:layout_constraint
    app:layout_constraint
    app:layout_constraint
    app:layout_constraint
```

Suppress: Add tools:ignore="OnClick" attribute
Create 'MClick(View)' in 'MainActivity'
Create onClick event handler
Override Resource in Other Configuration...
Inject language or reference

d) Put another button to our project :

- ID: **ButtonReset**
- name: **Reset**
- textSize: **24**
- create ConstraintLayout like before

- onClick: **MReset**
- Create **MReset** in Main Activity.java like before

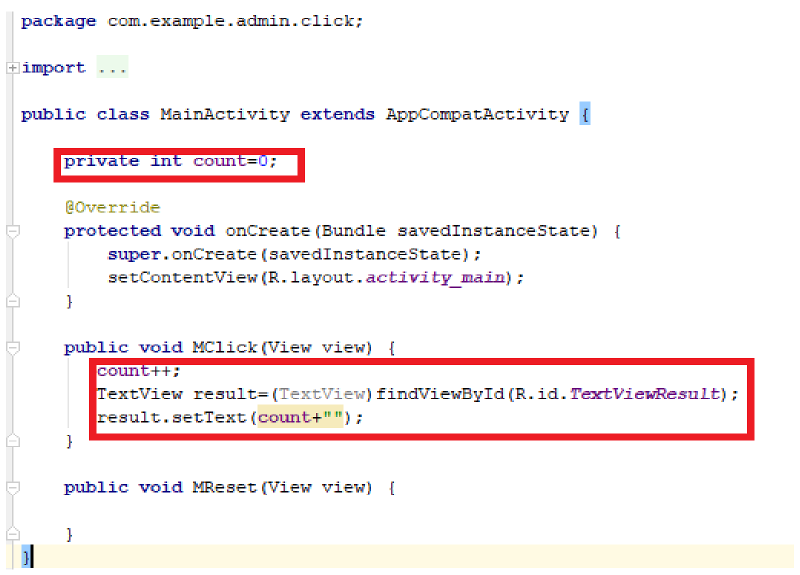
3. Run your app to see if layout looks the same as the layout shown in the picture
4. Write a code of our app :

a) You will need variable for store a counter status. You will call it **count,** it will store integer numberes and it should start from 0, at the start of the app.

It will be use in both metods: MClick and MReset, but only in MainActivity

b) Metod **MClick** should:

- increase value of variable count,
- sign TextViewResult to variable. We will call it result.
- display variable count in the TextViewResult component



```
package com.example.admin.click;

import ...

public class MainActivity extends AppCompatActivity {

    private int count=0;

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

    public void MClick(View view) {
        count++;
        TextView result=(TextView)findViewById(R.id.TextViewResult);
        result.setText(count+"");
    }

    public void MReset(View view) {

    }
}
```

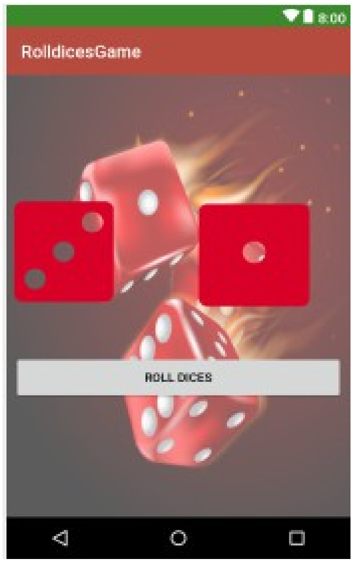
c) Run an app to check if counter is increasing its value after clicking the button

d) Do metod **MReset** yourself :

- it sign 0 to variable count
- sign TextViewResult to result variable.
- display variable count in the TextViewResult component

e) Run an app

3. Resourses folder



Roll Dices App

- ✔ Click on button and photoes of dices will randomly change

Goals of the exercise

- ✔ Preparing multilanguage app
- ✔ Changing background of app
- ✔ Using text and design mode of activity_main.xml to make an layout of the app
- ✔ Using random variables to randomly change of dices
- ✔ Creating extra methods needed in the app

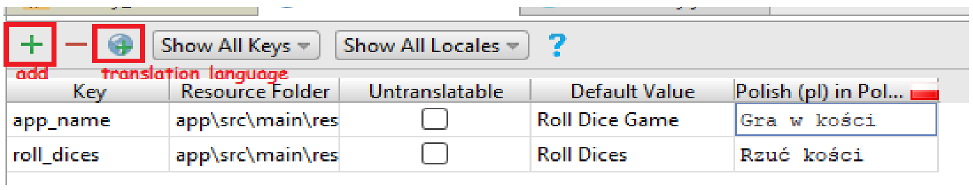
Starting Project

1.	Aplication name:	**RollDicesGame**
2.	Company domain**:**	**zslp.edu.pl**
3.	Device**:**	**Phone and Tablet**
4.	API**:**	**21**
5.	Activity**:**	**Empty**

Preparing multilanguage app

If you want your app to be multilanguage app, you must add all strings visible in your application and their translations into the ***strings.xml*** file:

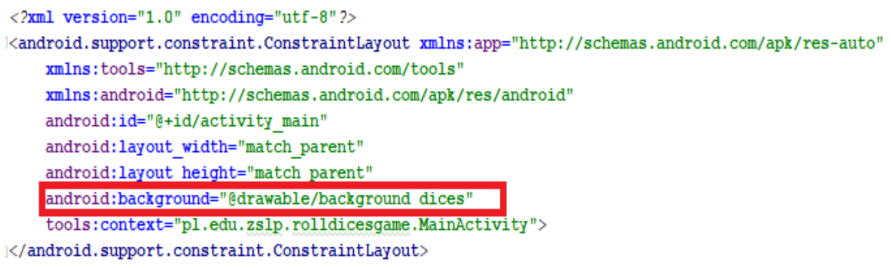
1. Open **strings.xml** file (***app/res/values/strings.xml***)
2. Click on right top corner **Open editor**
3. Add here string „**Roll Dices”** and its translation into your language. We will use this string on our app's button.



Key	Resource Folder	Untranslatable	Default Value	Polish (pl) in Pol...
app_name	app\src\main\res	☐	Roll Dice Game	Gra w kości
roll_dices	app\src\main\res	☐	Roll Dices	Rzuć kości

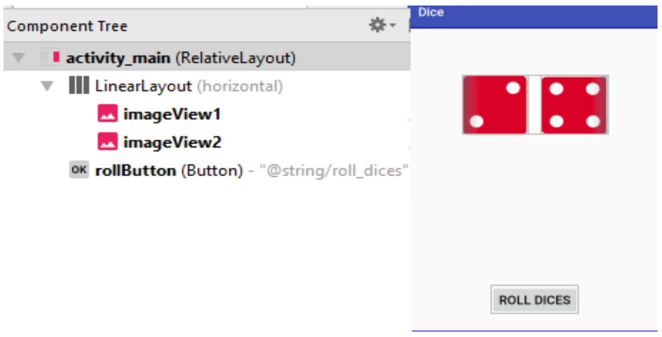
Making layout of your app

1. Change colors of your app:
   a) Open **colors.xml** (app/res/values/colors.xml) and change app colors as you like
2. Set the background
   a) Copy all pictures you will use in the app from **folder dice_photo** and past it into **drawable** folder
   b) Open **text mode** of **acitvity_main.xml** (app/layout/activity_main.xml)
   c) Write a command to set background_dices.jpg file as a background of the app

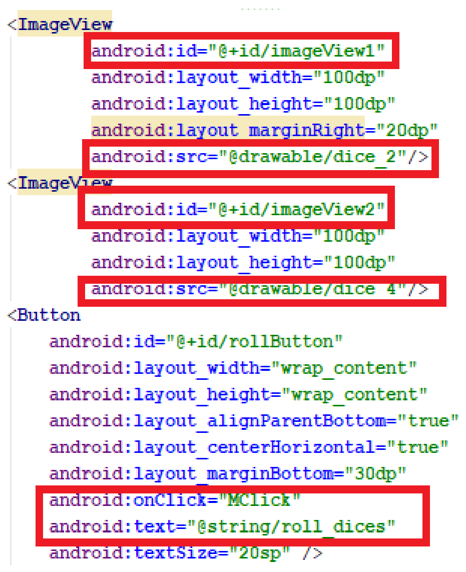


```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/activity_main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@drawable/background_dices"
    tools:context="pl.edu.zslp.rolldicesgame.MainActivity">
</android.support.constraint.ConstraintLayout>
```

3. Open ***design mode*** of **activity_main.xml,** and prepare the layout shown in the picture below


Component Tree
activity_main (RelativeLayout)
LinearLayout (horizontal)
imageView1
imageView2
rollButton (Button) - "@string/roll_dices"
Dice
ROLL DICES

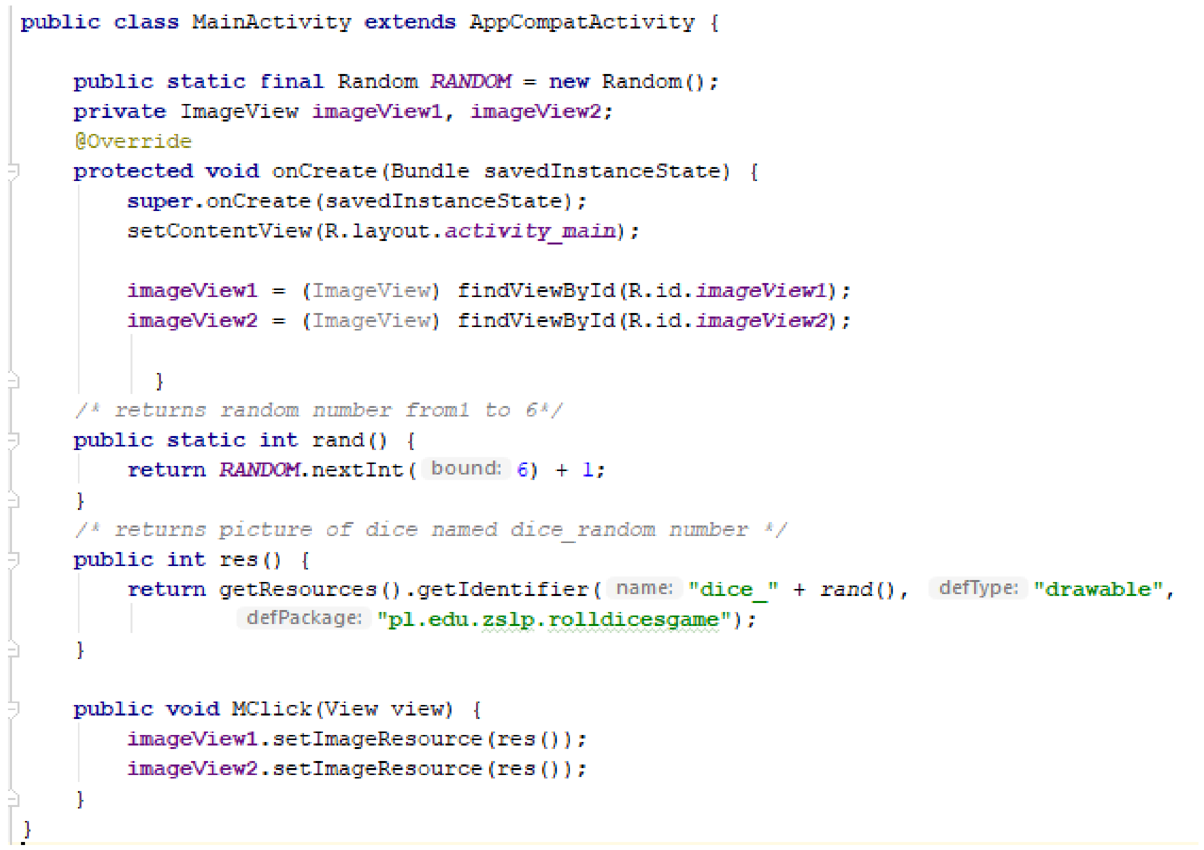
4. Check in ***text mode*** of **activity_main.xml** if all is like on the picture below:



```
<ImageView
        android:id="@+id/imageView1"
        android:layout_width="100dp"
        android:layout_height="100dp"
        android:layout_marginRight="20dp"
        android:src="@drawable/dice_2"/>
<ImageView
        android:id="@+id/imageView2"
        android:layout_width="100dp"
        android:layout_height="100dp"
        android:src="@drawable/dice_4"/>
<Button
    android:id="@+id/rollButton"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentBottom="true"
    android:layout_centerHorizontal="true"
    android:layout_marginBottom="30dp"
    android:onClick="MClick"
    android:text="@string/roll_dices"
    android:textSize="20sp" />
```

Programming the button to randomly change the dice image

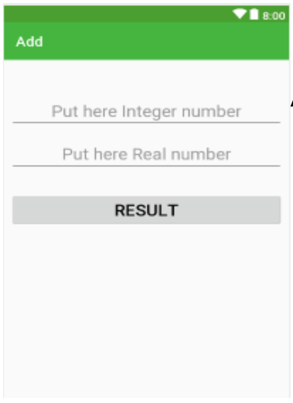
1. Open **MainActivity.java** file:
   a) creat method named **rand** to generate random number from 1 to 6
   b) create metod **res** returning photo with dice with number random by method rand
   c) set up appropriate dices after clicking the button



```java
public class MainActivity extends AppCompatActivity {

    public static final Random RANDOM = new Random();
    private ImageView imageView1, imageView2;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        imageView1 = (ImageView) findViewById(R.id.imageView1);
        imageView2 = (ImageView) findViewById(R.id.imageView2);

            }
    /* returns random number from1 to 6*/
    public static int rand() {
        return RANDOM.nextInt( bound: 6) + 1;
    }
    /* returns picture of dice named dice_random number */
    public int res() {
        return getResources().getIdentifier( name: "dice_" + rand(),  defType: "drawable",
                 defPackage: "pl.edu.zslp.rolldicesgame");
    }

    public void MClick(View view) {
        imageView1.setImageResource(res());
        imageView2.setImageResource(res());
    }
}
```

2. Run an app

4. EditText Component, Data conversion, Toast message



Add app

- User will put two numbers, one Integer and one Real
- After pushing RESULT button user will see Toast message with the result of adding one number to another

Goals of this exercice:

- Make an aplication witch will display a result of adding to numbers : integer and real number
- Use EditText component
- Data conversion
- Use Toast message

Add App Project:

1. Start a new Android Studio Project:

a) name:	**Add**
b) device**:**	**Phone and Tablet**
c) API**:**	**23**
d) Activity**:**	**Empty**

2. Make layout of your project. Open ***Design mode*** of the **activity_main.xml** file:
   a) Delete TextView „Hello World”
   b) Put EditText (Number) component:

• ID:	**EditTextInt**
• name:	**(no name)**
• hint:	**„Put here Integer number”**
• textSize:	**24**

- Create ConstraintLayout (Click the spots)

c) Put another EditText, this time Decimal:

• ID:	**EditTextReal**
• name:	**(no name)**
• hint:	**„Put here Real number"**
• textSize:	**24**

- Create ConstraintLayout (Click the spots)

d) Put button to our project:

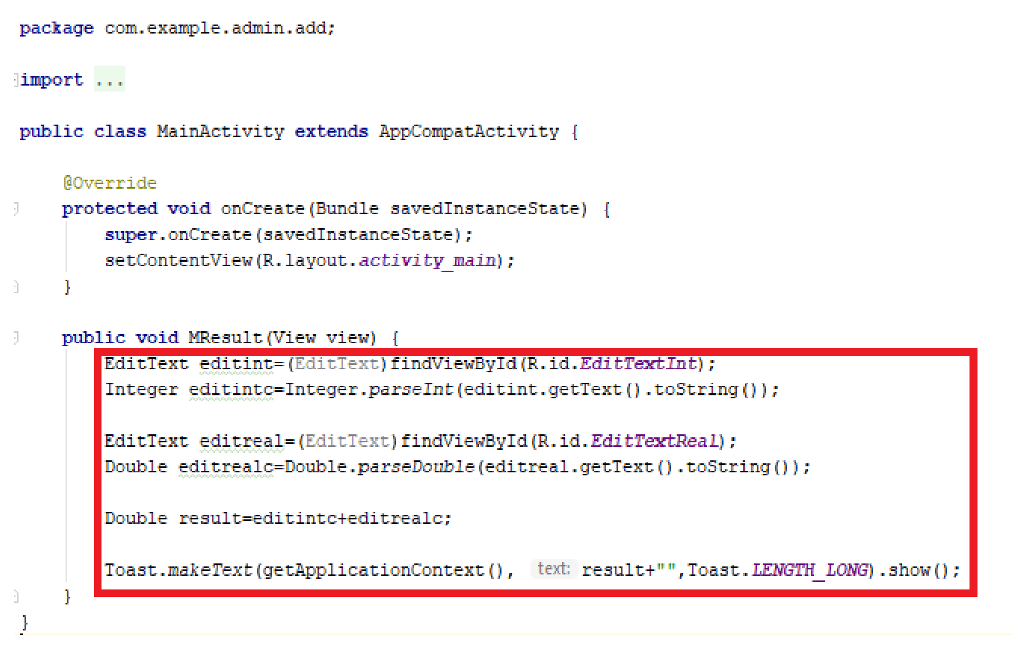
• ID:	**ButtonResult**
• name:	**RESULT**
• textSize:	**24**

- create ConstraintLayout like before
- onClick: **MResult** (The name of the metod signed to this button)
- Change mode in activity_main.xml from **design** to **text** .
  - ✔ Look for the button Result section.
  - ✔ Push **Alt+Enter** on the name of the method **MResult**
  - ✔ Pick **Create MResult in Main Activity** from context menu. You have just created method **MResult** in **MainActivity.java** file

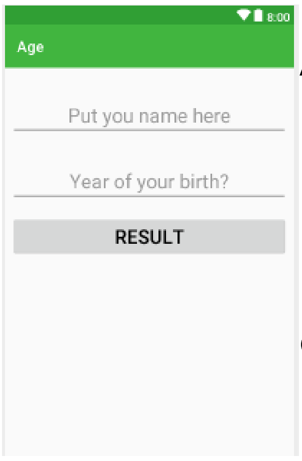
4. Run your app to see if layout look the same as the layout shown in the picture
5. Now write a code for your app :

a) Metod **MResult** should:

- ✔ sign EditTextInt component to variable. Call it editint
- ✔ convert editint to Integer number and call it editintc
- ✔ sign EditTextReal component to variable.Call it editreal
- ✔ convert editreal to Real number and call it editrealc
- ✔ add edititintc and edittextrealc and sign it to variable result
- ✔ display result of adding in Toast message



```
package com.example.admin.add;

import ...

public class MainActivity extends AppCompatActivity {

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

    public void MResult(View view) {
        EditText editint=(EditText)findViewById(R.id.EditTextInt);
        Integer editintc=Integer.parseInt(editint.getText().toString());

        EditText editreal=(EditText)findViewById(R.id.EditTextReal);
        Double editrealc=Double.parseDouble(editreal.getText().toString());

        Double result=editintc+editrealc;

        Toast.makeText(getApplicationContext(), text: result+"",Toast.LENGTH_LONG).show();
    }
}
```



Age app

- ✔ User will put his name and year of his birth
- ✔ After pushing RESULT button user will see Toast message „Hello name. You are age years old"

Goals of this exercice:

- ✔ Make an age app
- ✔ Practise using TextView, EditText, data conversion, Toast message

Age App Project:

1. Start a new Android Studio Project, named **Age**
2. Make a layout similar to picture above.
3. Run our app to see if layout look the same as the layout shown on the picture
4. Make an **MResult** method:
   a) sign EditText component with name variable, and change it into String variable

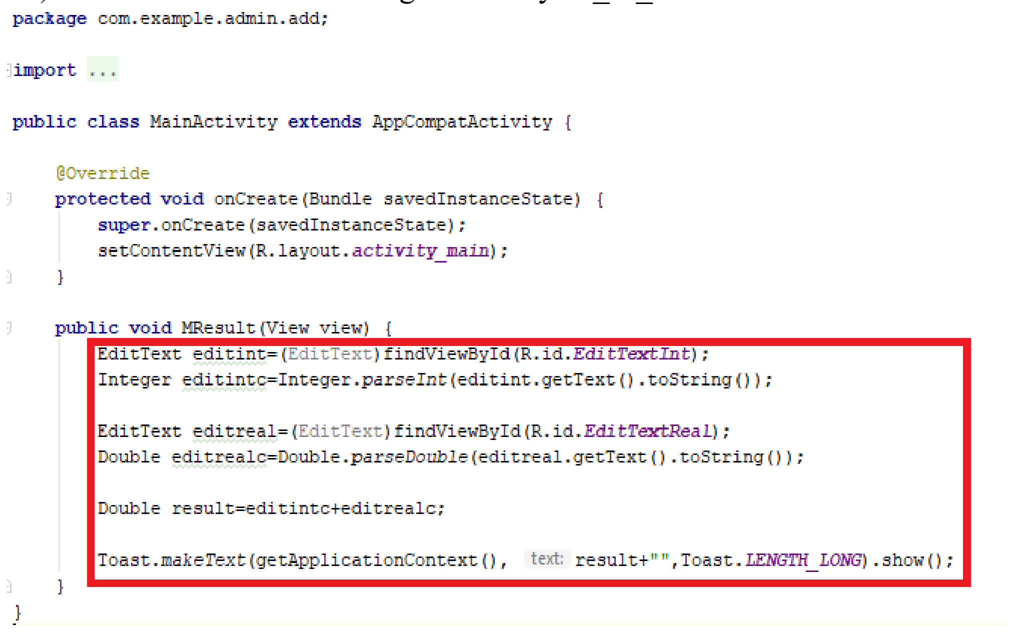
      EditText name=(EditText)findViewById(R.id.***EditTextName***);

      String names=name.getText().toString();

   b) sign EditText component with year of the birth to variable.

c) convert text from EditText component to Integer number

d) make an variable named age = 2018-year_of_birth



```
package com.example.admin.add;

import ...

public class MainActivity extends AppCompatActivity {

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

    public void MResult(View view) {
        EditText editint=(EditText)findViewById(R.id.EditTextInt);
        Integer editintc=Integer.parseInt(editint.getText().toString());

        EditText editreal=(EditText)findViewById(R.id.EditTextReal);
        Double editrealc=Double.parseDouble(editreal.getText().toString());

        Double result=editintc+editrealc;

        Toast.makeText(getApplicationContext(),  text: result+"",Toast.LENGTH_LONG).show();
    }
}
```

e) display „*Hello* **name**. *You are* **age** *years old”* in Toast message

5. Run an app

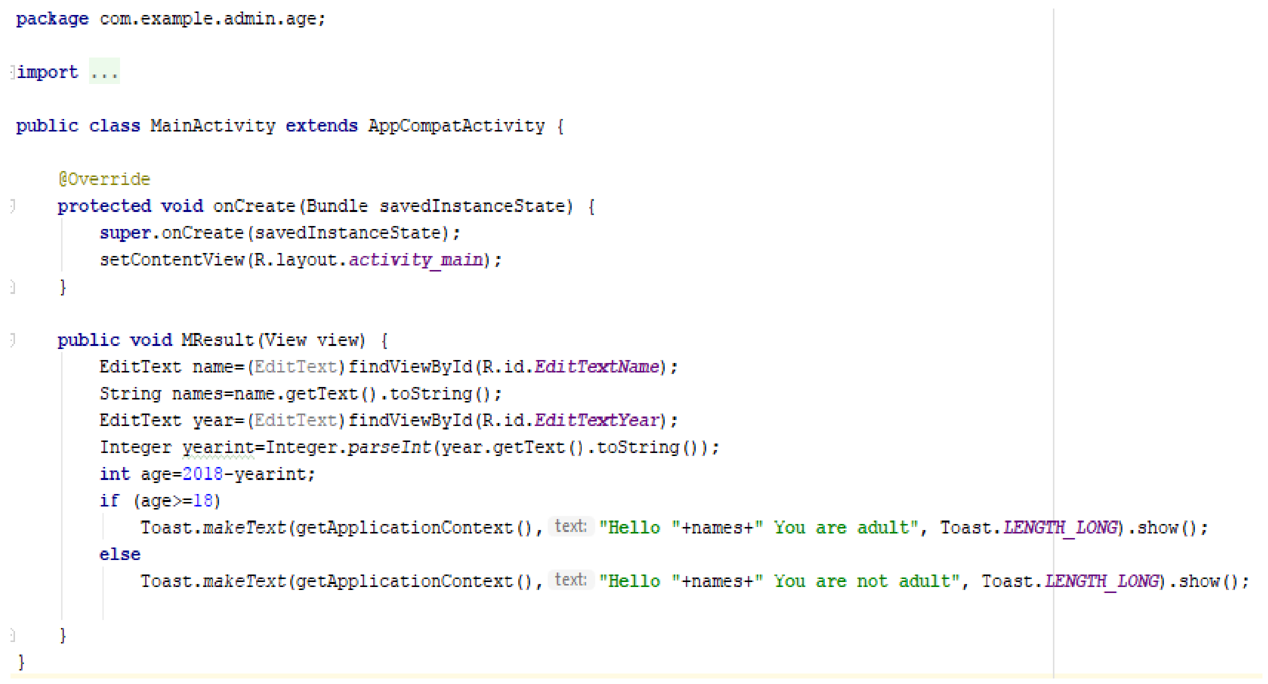
Extra exercise-Age app*:*

- ✔ User will put his name and year of his birth
- ✔ After pushing RESULT button user will see Toast message: „Hello name. You are adult” or „Hello name. You are not adult”

You must use if statment:

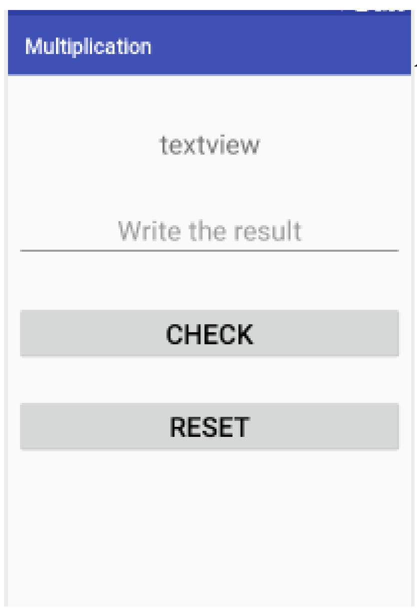
```
if (condision) {
// codes if condition is true
            }
else {
 // code if condition is false
      }
```

Hint:



```java
package com.example.admin.age;

import ...

public class MainActivity extends AppCompatActivity {

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

    public void MResult(View view) {
        EditText name=(EditText)findViewById(R.id.EditTextName);
        String names=name.getText().toString();
        EditText year=(EditText)findViewById(R.id.EditTextYear);
        Integer yearint=Integer.parseInt(year.getText().toString());
        int age=2018-yearint;
        if (age>=18)
            Toast.makeText(getApplicationContext(), text: "Hello "+names+" You are adult", Toast.LENGTH_LONG).show();
        else
            Toast.makeText(getApplicationContext(), text: "Hello "+names+" You are not adult", Toast.LENGTH_LONG).show();

    }
}
```

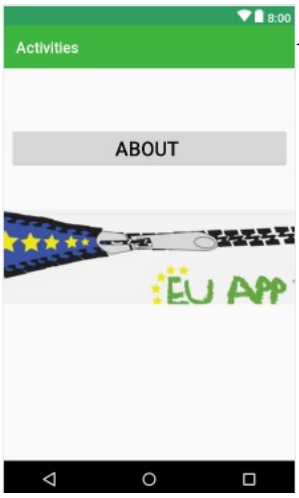
Extra exercise Multiplication app



Multiplication app

- App will show 2 random numbers in range [0,10]
- User will put result of multiplications this two numbers into EditText component
- After pushing Check button he will get a Toast message „:)" or „:(„ , and counter with count the points
- user gets 1 point for good answear and -1 for the bad one

5. Intents, multiactivities applications



Activities app

- ✔ Main Activity of the app will have two buttons,
- ✔ First button will transfer user to Second Activity, and will pass a text to be shown from MainActivity to Second Activity
- ✔ Second image button will transfer user outside the app to webside: http://eu-app16.eu/

Goals of this exercice:

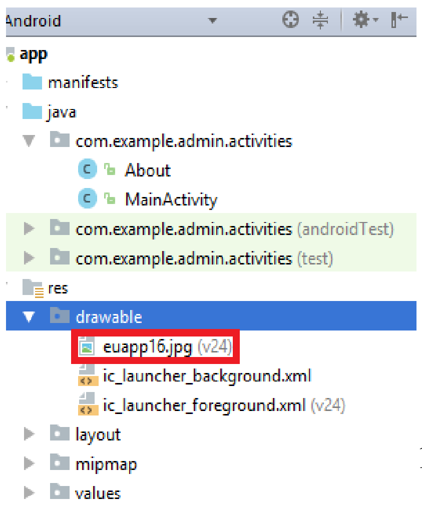
- ✔ Make the Activities app
- ✔ Use Image Button
- ✔ Use Intent to switch between activities,
- ✔ Pass informations between activities
- ✔ Use Intent to open webside outside our app

An Intent is an "intention" to perform an action. We can use Intent for example to open webside or to open another activity

Activities App Project:

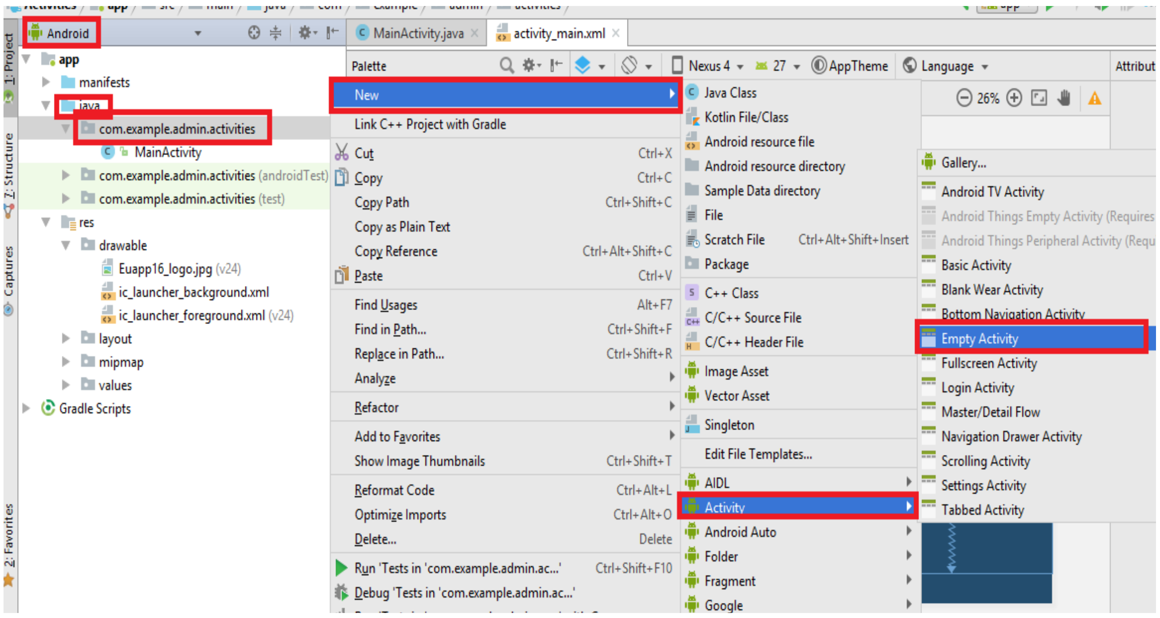
1. Start a new Android Studio Project, named **Activities**
2. Past ***euapp16_logo.jpg*** file to **res/drawable**
3. Make a layout of Main Activity, similar to picture above:
   a) Put image button
      - name: **ImButton,**
      - onClick: **MWeb**

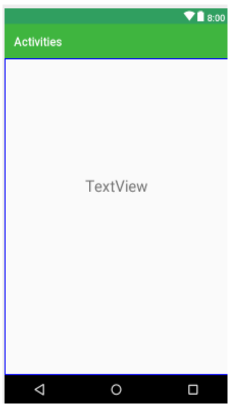
   b) Put button
      - name: **AboutButton,**


Android
app
manifests
java
com.example.admin.activities
About
MainActivity
com.example.admin.activities (androidTest)
com.example.admin.activities (test)
res
drawable
euapp16.jpg (v24)
ic_launcher_background.xml
ic_launcher_foreground.xml (v24)
layout
mipmap
values

- onClick: **Mabout**

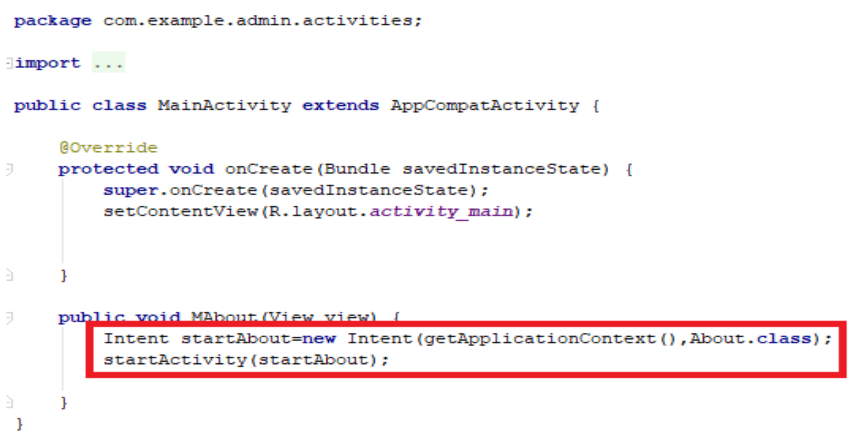
4. Create a new Empty Activity named About





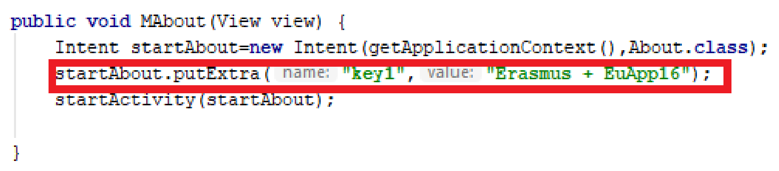
5. Open **Design mode** of **activity_about.xml**, put there TextView component and name it TextViewAbout (Don't forget to make Constraint Layout)
6. Create method **Mabout**. It will have an Intent.

   Intent is an action being requested, that device should try to perform.



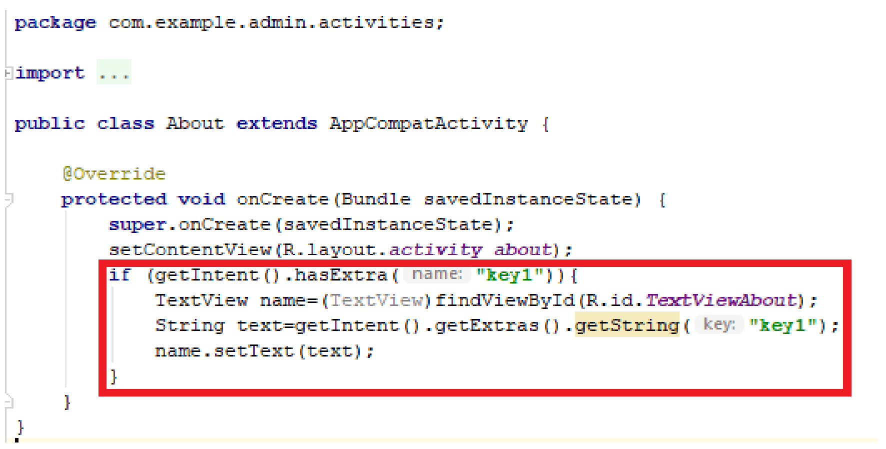
```
package com.example.admin.activities;

import ...

public class MainActivity extends AppCompatActivity {

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

    }

    public void MAbout(View view) {
        Intent startAbout=new Intent(getApplicationContext(),About.class);
        startActivity(startAbout);

    }
}
```

7. Run ou app and click About button
8. Now you will try to pass something from Main Activity to About Activity
9. Make a change in ***MainActivity.java***



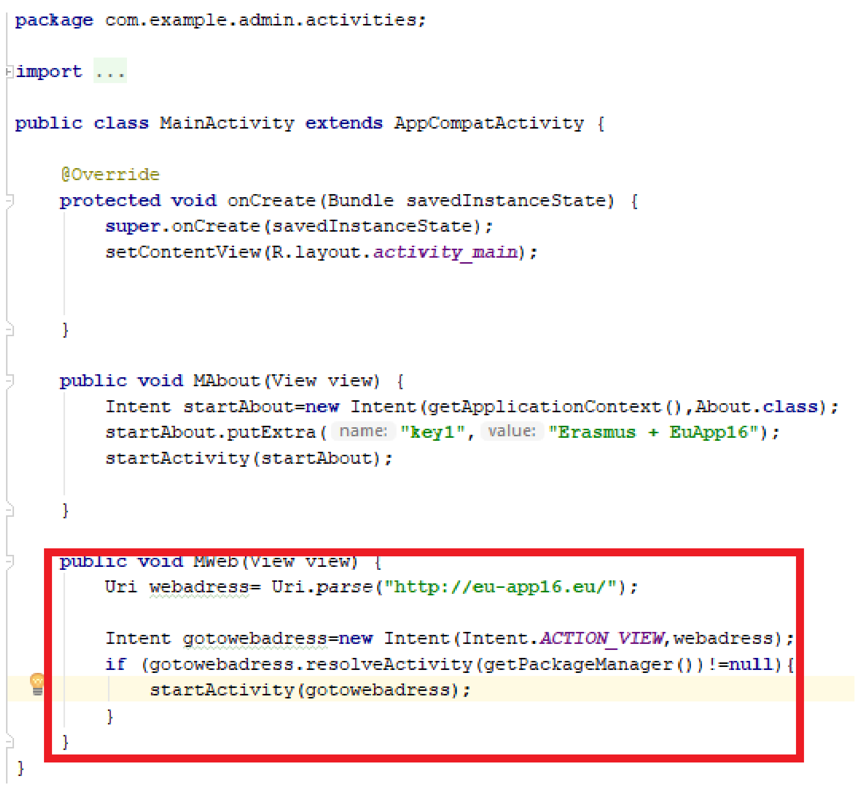
```
public void MAbout(View view) {
    Intent startAbout=new Intent(getApplicationContext(),About.class);
    startAbout.putExtra( name: "key1", value: "Erasmus + EuApp16");
    startActivity(startAbout);

}
```

10. Make change in ***About.java***



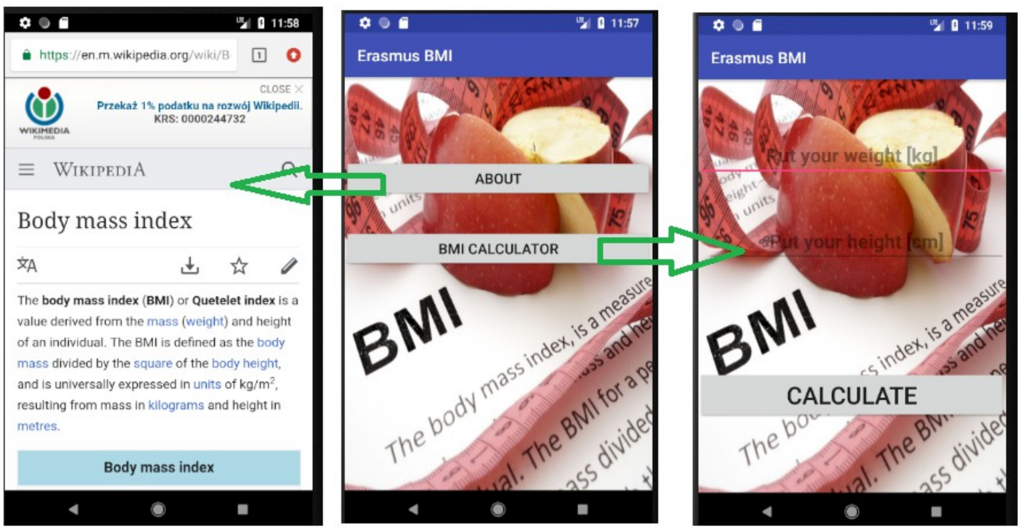
```
package com.example.admin.activities;

import ...

public class About extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_about);
        if (getIntent().hasExtra( name: "key1")){
            TextView name=(TextView)findViewById(R.id.TextViewAbout);
            String text=getIntent().getExtras().getString( key: "key1");
            name.setText(text);
        }
    }
}
```

11. Make **MWeb** method:



```
package com.example.admin.activities;

import ...

public class MainActivity extends AppCompatActivity {

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


    }

    public void MAbout(View view) {
        Intent startAbout=new Intent(getApplicationContext(),About.class);
        startAbout.putExtra( name: "key1", value: "Erasmus + EuApp16");
        startActivity(startAbout);

    }

    public void MWeb(View view) {
        Uri webadress= Uri.parse("http://eu-app16.eu/");

        Intent gotowebadress=new Intent(Intent.ACTION_VIEW,webadress);
        if (gotowebadress.resolveActivity(getPackageManager())!=null){
            startActivity(gotowebadress);
        }
    }
}
```

BMICalculator app



The body mass index (BMI) or Quetelet index is a value derived from the mass (weight) and height of an individual. The BMI is defined as the body mass divided by the square of the body height, and is universally expressed in units of kg/m^2, resulting from mass in kilograms and height in metres.

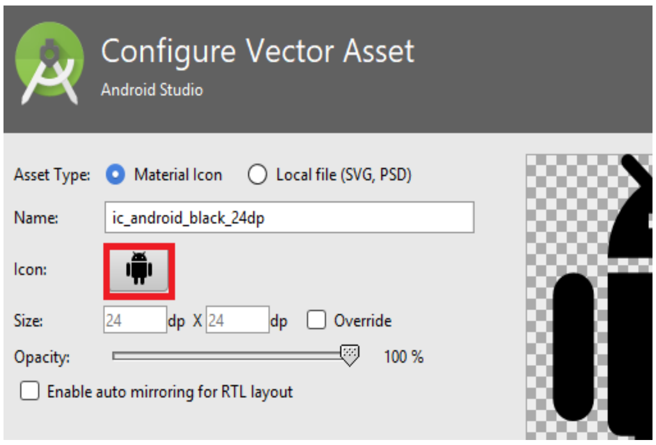
Goals of the exercise

- ✔ Checking knowledge from previous exercise
- ✔ Creating icon of the app
- ✔ Difference between ConstrainLayout and RelativeLayout
- ✔ Creating new Activities
- ✔ Different use of the Intent in app (opening webside or another activity)
- ✔ Using different data types, making data conversion

Starting Project

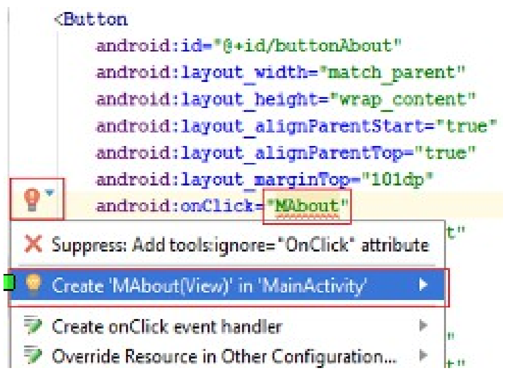
1. Open Android Studio Projekt **BMI_EUAPP16**
2. Preparing multilanguage app
   a) **Open strings.xml** file (app/res/values/strings.xml)
   a) Click on right top corner **Open editor**
   b) Make translation of strings
   c) Change colors of your app
   d) Set bmi.jpg picture as a background of this activity
   e) Set **picture_200_200.jpg** file as an icon of your app
      - Right click on picture_200_200.jpg file in drawable file and **Copy the path**

- Right click on ***drawable*** folder (app/res/drawable) pick ***New/Image Asset***
- Click icon
- Name your icon **ic_my_icon,** you can change color of your icon in ic_my_icon.xml file



Asset Type: [✓ Material Icon] Local file (SVG, PSD)
Name: ic_android_black_24dp
Icon:
Size: 24 dp X 24 dp Override
Opacity: 100 %
Enable auto mirroring for RTL layout

2. Programe buttons in MainActivity
   a) Create method **MAbout** and **MCalc** in MainActivity.java by clicking on the red bulb in the activity_main.xml, look at the picture below

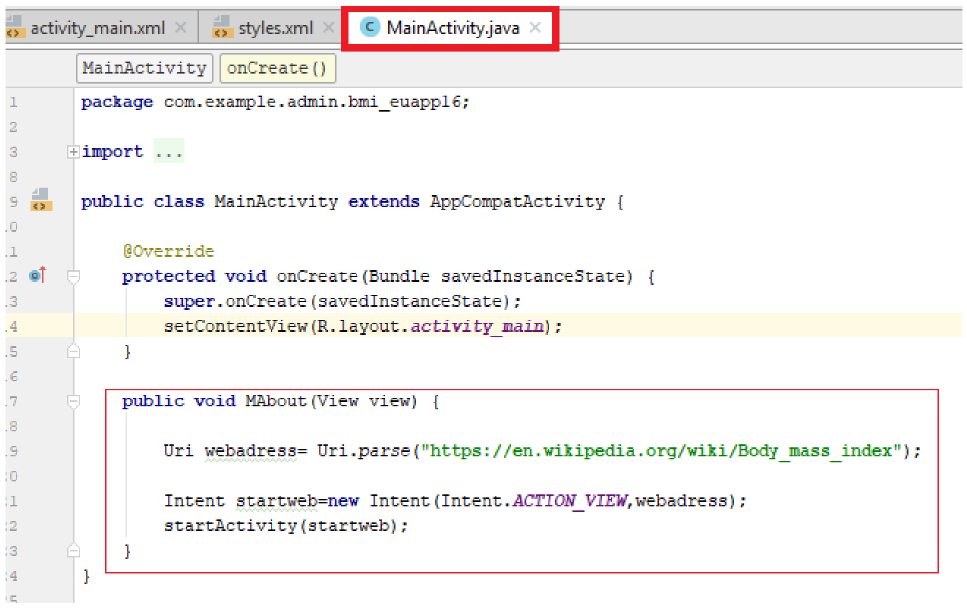


```
<Button
    android:id="@+id/buttonAbout"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_alignParentStart="true"
    android:layout_alignParentTop="true"
    android:layout_marginTop="101dp"
    android:onClick="MAbout"
```

Suppress: Add tools:ignore="OnClick" attribute
Create 'MAbout(View)' in 'MainActivity'
Create onClick event handler
Override Resource in Other Configuration...

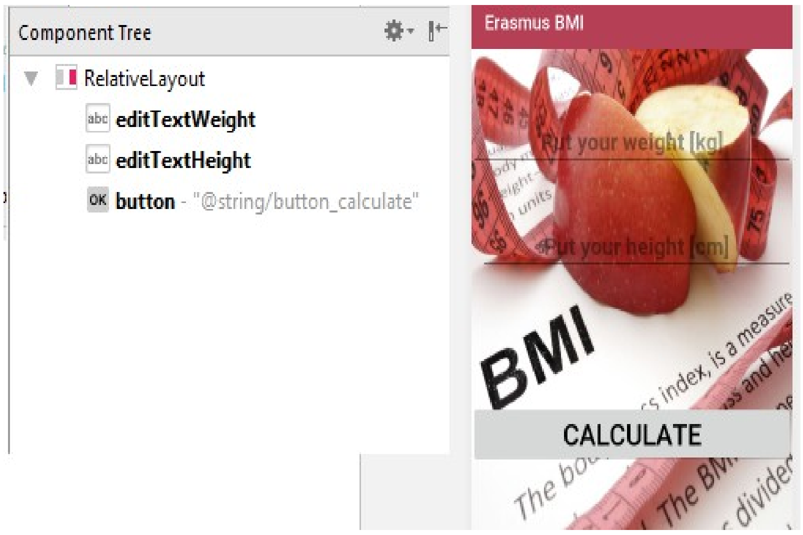
   b) Programme onClick behavior of button About. Open MainActivity.java (app/java/packet name)

You will use Intent to open webside https://en.wikipedia.org/wiki/Body_mass_index



```
package com.example.admin.bmi_euappl6;

import ...

public class MainActivity extends AppCompatActivity {

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

    public void MAbout(View view) {

        Uri webadress= Uri.parse("https://en.wikipedia.org/wiki/Body_mass_index");

        Intent startweb=new Intent(Intent.ACTION_VIEW,webadress);
        startActivity(startweb);
    }
}
```

3. Run an App to check if About button works
4. Create second empty activty named ***BmiCalculator***
5. Create a method **Mcalc** in MainActivity.java file
   a) Write an Intent to open activity with BMI Calculator
6. Creating BMI Calculator Layout
   a) Open file **activity bmi_calculator.xml,** change ConstrainLayout to RelativeLayout and use file bmi.jpg as a background



   b) Create layout according to picture above
      - edithTextWeight:
         - **android:id="@+id/editTextWeight"**
         - **android:hint="@string/hint_weight"**
         - **android:inputType="numberDecimal"**
      - editTextHeight:
         - **android:id="@+id/editTextHeight"**
         - **android:layout_below="@+id/editTextWeight"**
         - **android:inputType="numberDecimal"**
         - **android:hint="@string/hint_hight"**
      - buttonCalculate
         - **android:id="@+id/buttonCalculate"**
         - **android:onClick="MBMI"**
         - **android:text="@string/button_calculate"**

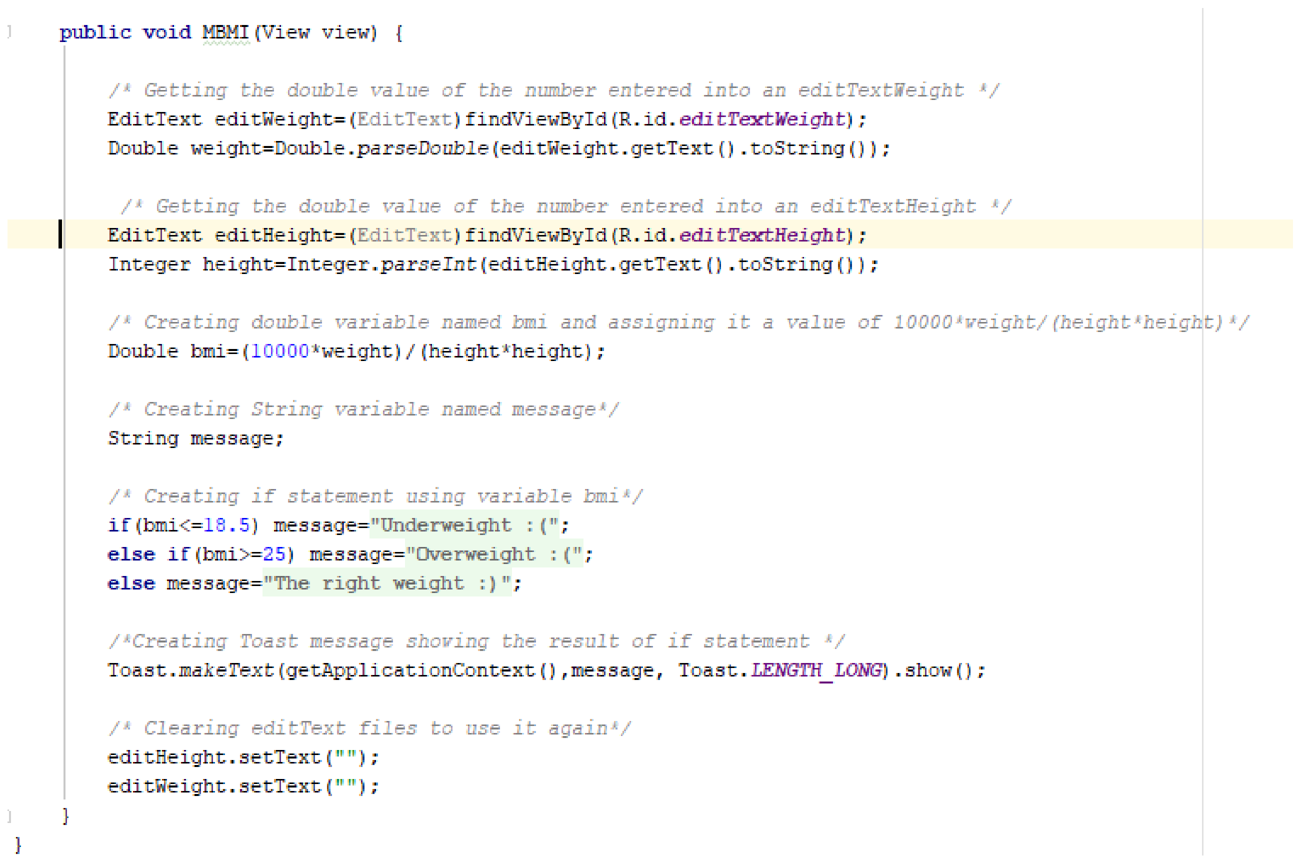
   c) Create method **MBMI** in **BmiCalculator.java** file

BMI formula:

$$BMI = \frac{10000 * weight[kg]}{height[cm] * height[cm]}$$

BMI range:

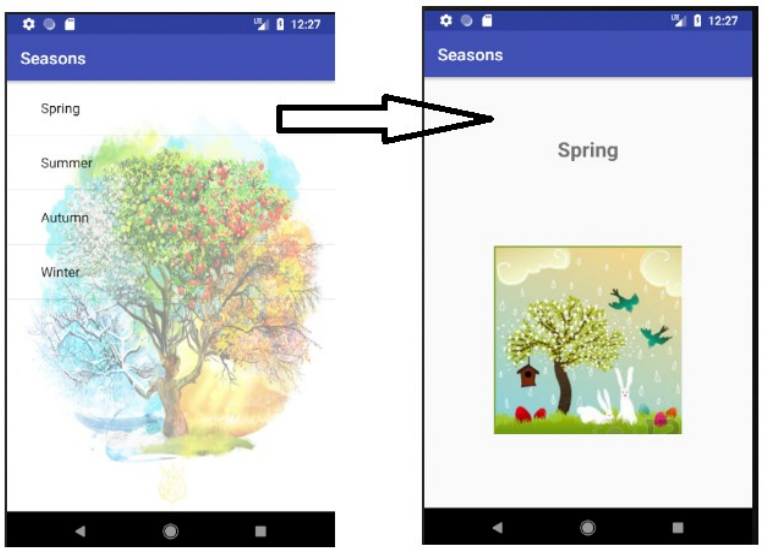
BMI	Message	Example	
		weight[kg]	height[cm]
<=18.5	**Underweight :(**	50.5	180
(18.5; 25)	**Normal Range :)**	60	160
>=25	**Overweight :(**	180.2	150



```
    public void MBMI(View view) {

        /* Getting the double value of the number entered into an editTextWeight */
        EditText editWeight=(EditText)findViewById(R.id.editTextWeight);
        Double weight=Double.parseDouble(editWeight.getText().toString());

         /* Getting the double value of the number entered into an editTextHeight */
        EditText editHeight=(EditText)findViewById(R.id.editTextHeight);
        Integer height=Integer.parseInt(editHeight.getText().toString());

        /* Creating double variable named bmi and assigning it a value of 10000*weight/(height*height)*/
        Double bmi=(10000*weight)/(height*height);

        /* Creating String variable named message*/
        String message;

        /* Creating if statement using variable bmi*/
        if(bmi<=18.5) message="Underweight :(";
        else if(bmi>=25) message="Overweight :(";
        else message="The right weight :)";

        /*Creating Toast message showing the result of if statement */
        Toast.makeText(getApplicationContext(),message, Toast.LENGTH_LONG).show();

        /* Clearing editText files to use it again*/
        editHeight.setText("");
        editWeight.setText("");
    }
}
```

6. ListView Component

Seasons app

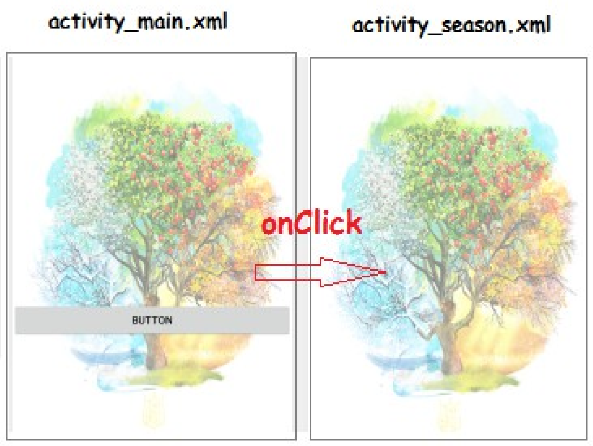


Goals of the exercise

- ✔ Using ListView Component
- ✔ Passing information from one activity to another activity
- ✔ Using arrays of strings and photoes

Starting Project

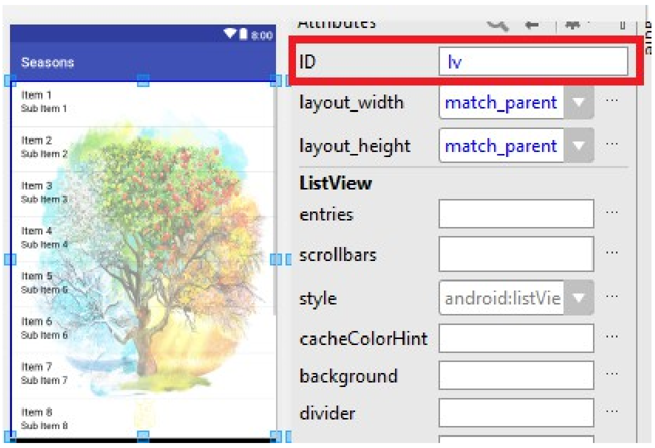
1. Open Android Studio Project **Sesons**
2. Add new emptyActivity named Season
3. Change both layouts from Constraint Layout to Relative Layout, and change background to seasons.jpg file
4. Put button in Main layout and program on click action to transfer you to season activity layout



5. Run your app
6. **Open strings.xml** file (app/res/values/strings.xml) and define strings in our app
7. Change icon of the app to **icon_seasons.png**
8. Run your app

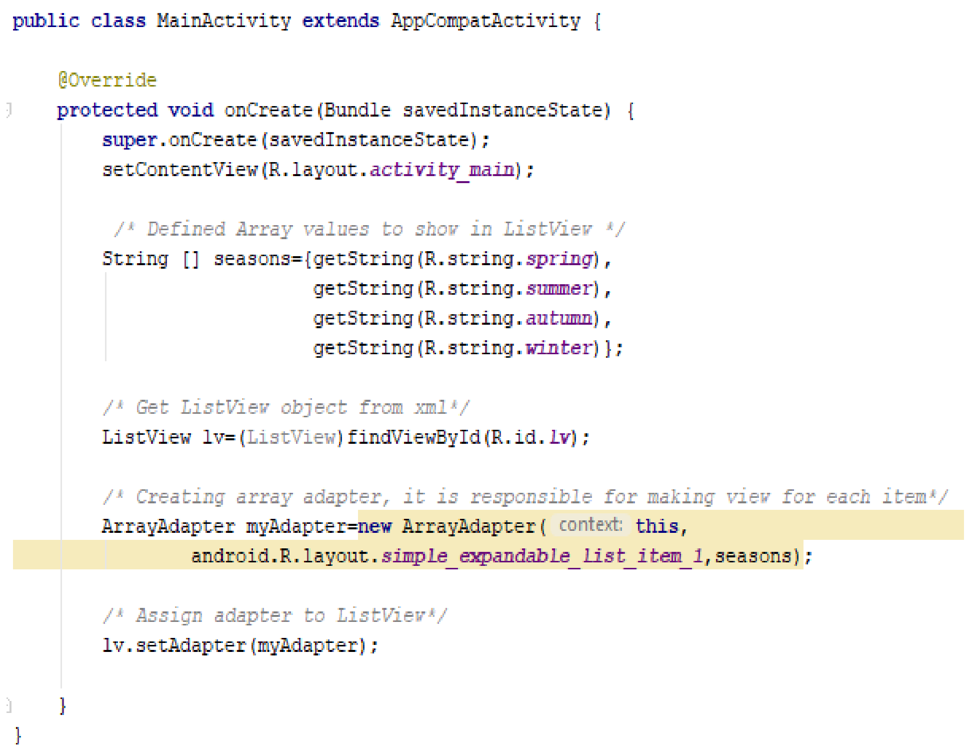
Creating ListView with seasons' name

1. Delete Button from activity_main.xml, and metod you made in MainActivity.java programing the button
2. Put listView component in design mode activity_main.xml (id: **listview)**



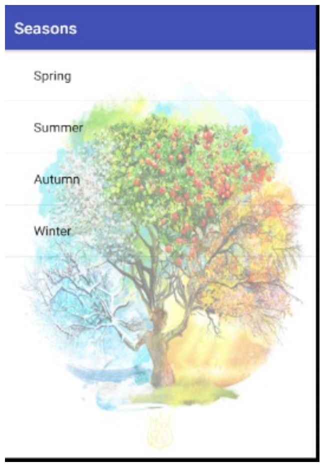
ID: lv
layout_width: match_parent
layout_height: match_parent
ListView
entries
scrollbars
style: android:listVie
cacheColorHint
background
divider

3. Open ***MainActivity.java*** file, and rewrite programme below without comments lines



```
public class MainActivity extends AppCompatActivity {

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

         /* Defined Array values to show in ListView */
        String [] seasons={getString(R.string.spring),
                           getString(R.string.summer),
                           getString(R.string.autumn),
                           getString(R.string.winter)};

        /* Get ListView object from xml*/
        ListView lv=(ListView)findViewById(R.id.lv);

        /* Creating array adapter, it is responsible for making view for each item*/
        ArrayAdapter myAdapter=new ArrayAdapter( context: this,
                android.R.layout.simple_expandable_list_item_1,seasons);

        /* Assign adapter to ListView*/
        lv.setAdapter(myAdapter);

    }
}
```

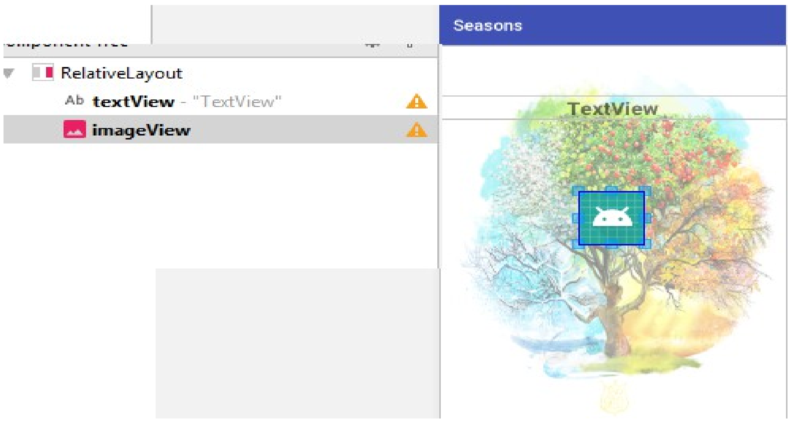
4. Run your app- you should see a listView with seasons' name



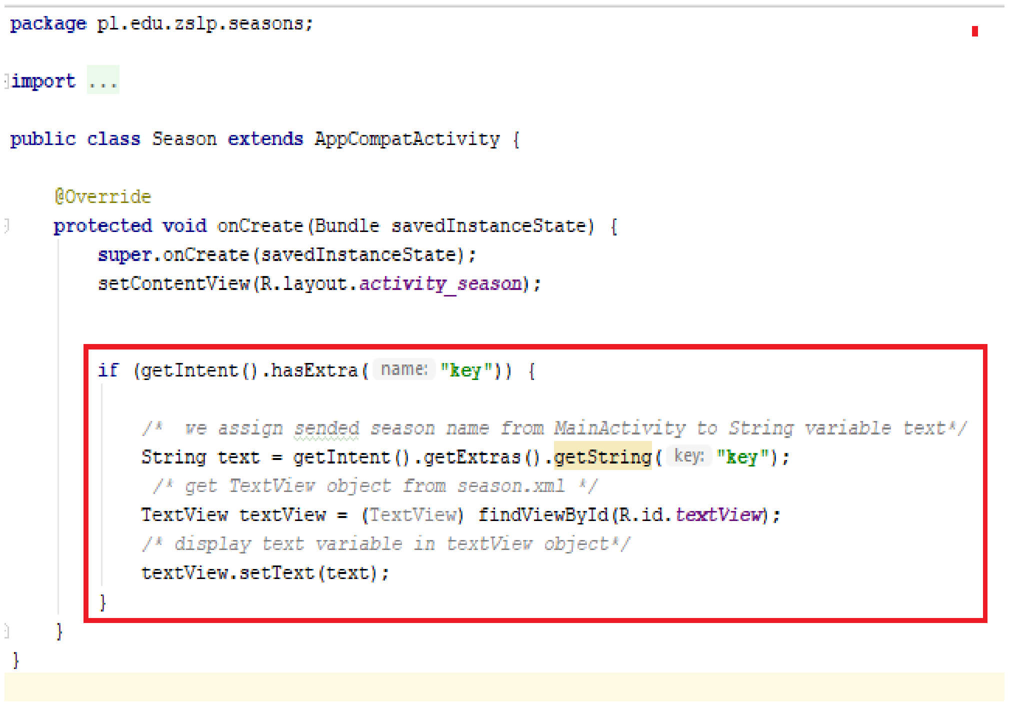
5. Copy code from MainActiviy(java).odt and replace with it your MainActivity.class.
6. Now after clicking in listView element Intent should open new activity and pass there name of clicked season (putExtras statment)
7. Run your app, and check what will happend
8. Our app opens SeasonActivity but we don't se season name in second activity

Creating Seasons activity

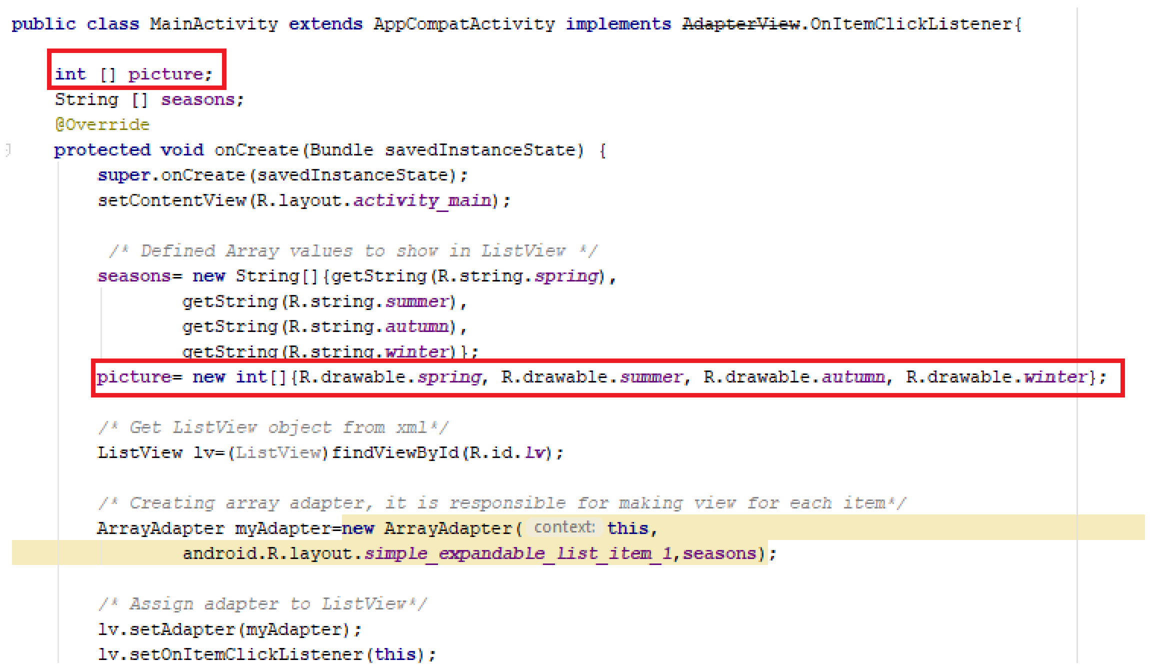
1. Create layout of Second Activity as on the picture below



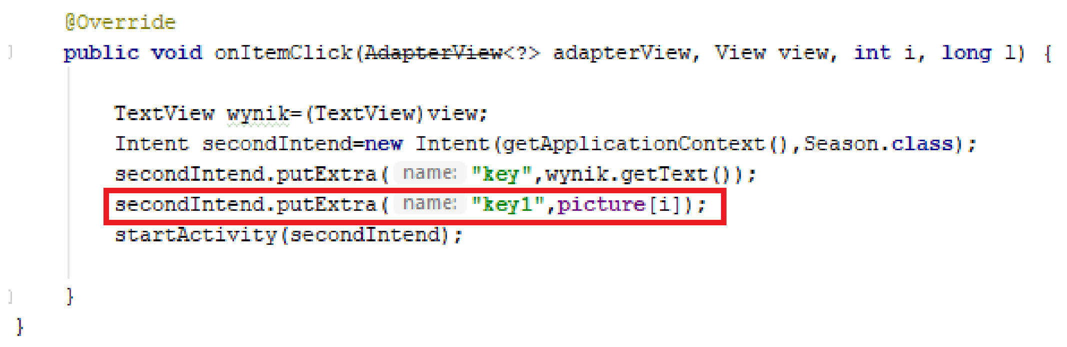
2. Copy code from Season(java).odt and past it in right place
3. Run an app and see what will happen
4. You will make **int array picture** in xml it will contain pictures of seasons. You will try to send it to the second activity and display it there.
5. Past pictures spring.jpg,summer.jpg,autumn.jpg,winter.jpg to folder drawable(app/res)
6. Make changes in MainActivity.java as in the picture belowe



```
package pl.edu.zslp.seasons;

import ...

public class Season extends AppCompatActivity {

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

        if (getIntent().hasExtra( name: "key")) {

            /*  we assign sended season name from MainActivity to String variable text*/
            String text = getIntent().getExtras().getString( key: "key");
             /* get TextView object from season.xml */
            TextView textView = (TextView) findViewById(R.id.textView);
            /* display text variable in textView object*/
            textView.setText(text);
        }
    }
}
```



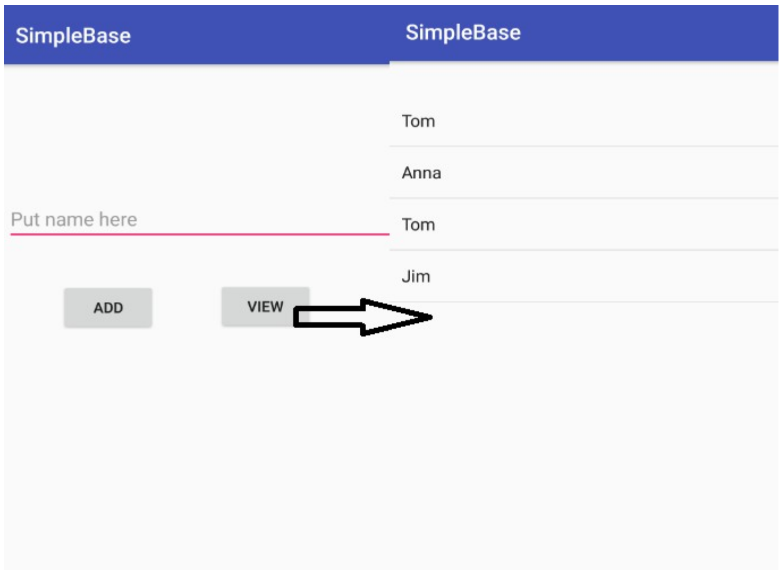
```
public class MainActivity extends AppCompatActivity implements ~~AdapterView~~.OnItemClickListener{

    int [] picture;
    String [] seasons;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

         /* Defined Array values to show in ListView */
        seasons= new String[]{getString(R.string.spring),
                getString(R.string.summer),
                getString(R.string.autumn),
                getString(R.string.winter)};
        picture= new int[]{R.drawable.spring, R.drawable.summer, R.drawable.autumn, R.drawable.winter};

        /* Get ListView object from xml*/
        ListView lv=(ListView)findViewById(R.id.lv);

        /* Creating array adapter, it is responsible for making view for each item*/
        ArrayAdapter myAdapter=new ArrayAdapter( context: this,
                android.R.layout.simple_expandable_list_item_1,seasons);

        /* Assign adapter to ListView*/
        lv.setAdapter(myAdapter);
        lv.setOnItemClickListener(this);
```



```
    @Override
    public void onItemClick(~~AdapterView~~<?> adapterView, View view, int i, long l) {

        TextView wynik=(TextView)view;
        Intent secondIntend=new Intent(getApplicationContext(),Season.class);
        secondIntend.putExtra( name: "key",wynik.getText());
        secondIntend.putExtra( name: "key1",picture[i]);
        startActivity(secondIntend);

    }
}
```

5. Try to make changes in SeasonActivity to recive and display picture
6. *Run your app*

7. Using Simple DataBase in ListView Component



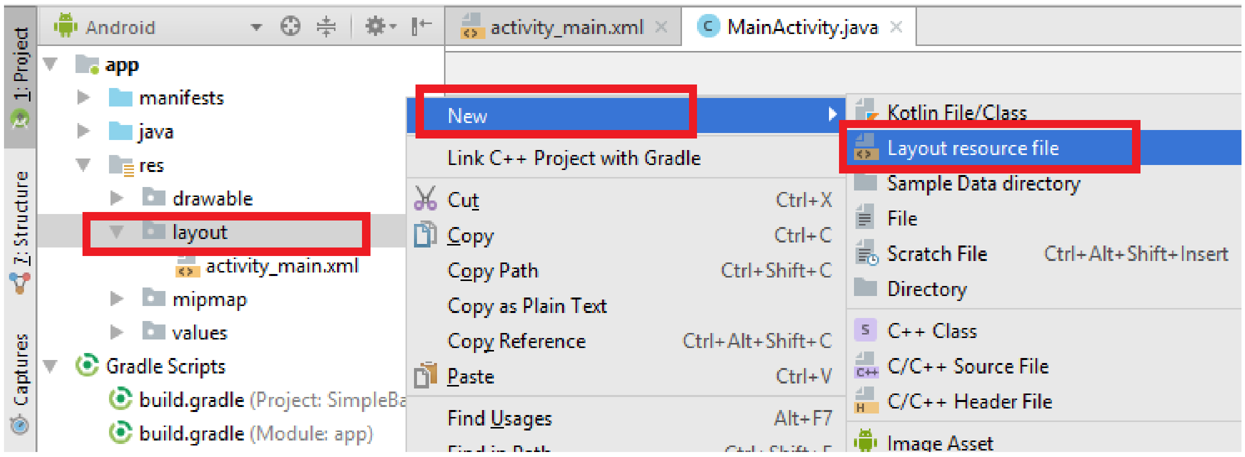
SimpleBase app:

- ✔ Creating simple database
- ✔ Adding elements to database
- ✔ Showing data from database in list view

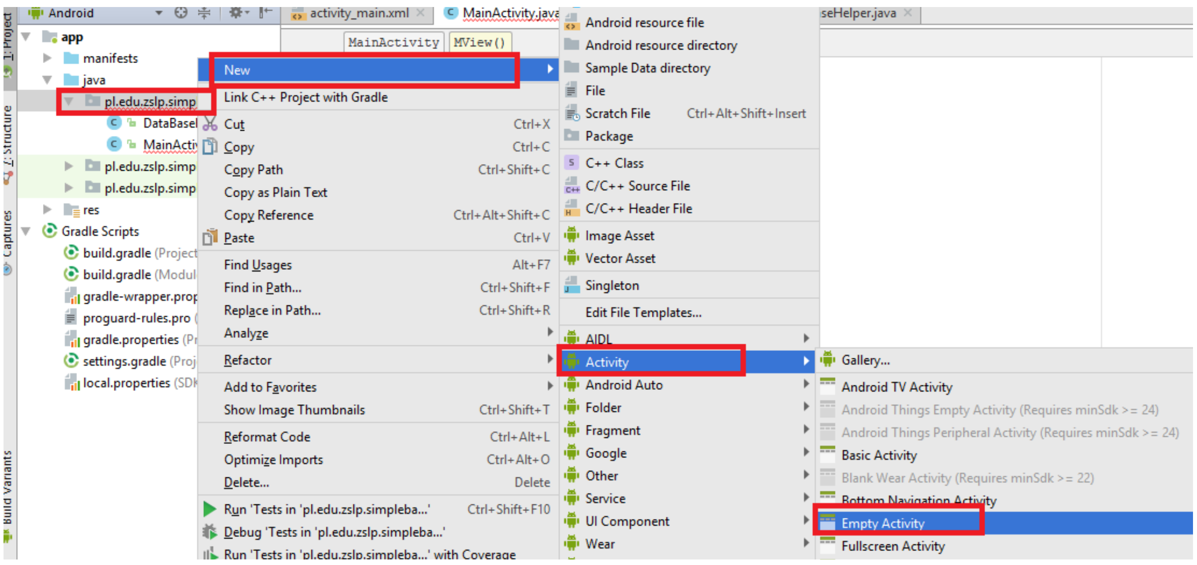
Starting Project

1. File/New/ **New Project**:

a) Aplication name:	**SimpleBase**
b) Company domain:	**zslp.edu.pl**
c) Device:	**Phone and Tablet**
d) API:	**21**
e) Activity:	**Empty**

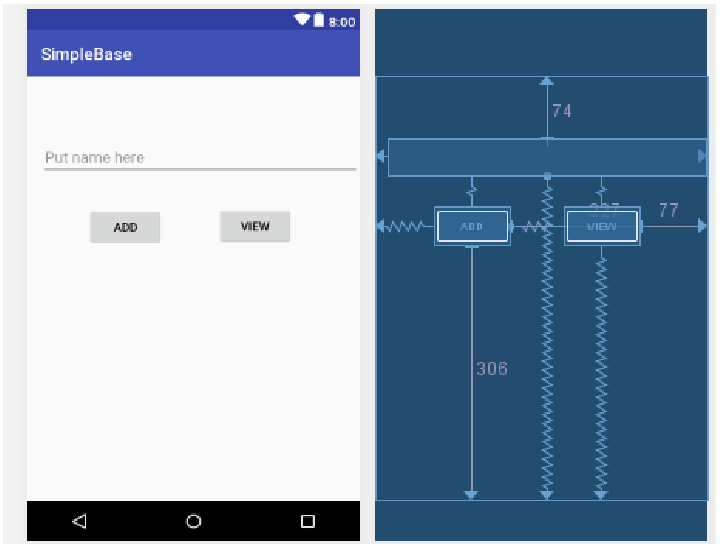


2. Right click on app/layout then New/ Layout resource file:
   a) New/Activity/Empty Activity name it **Names**

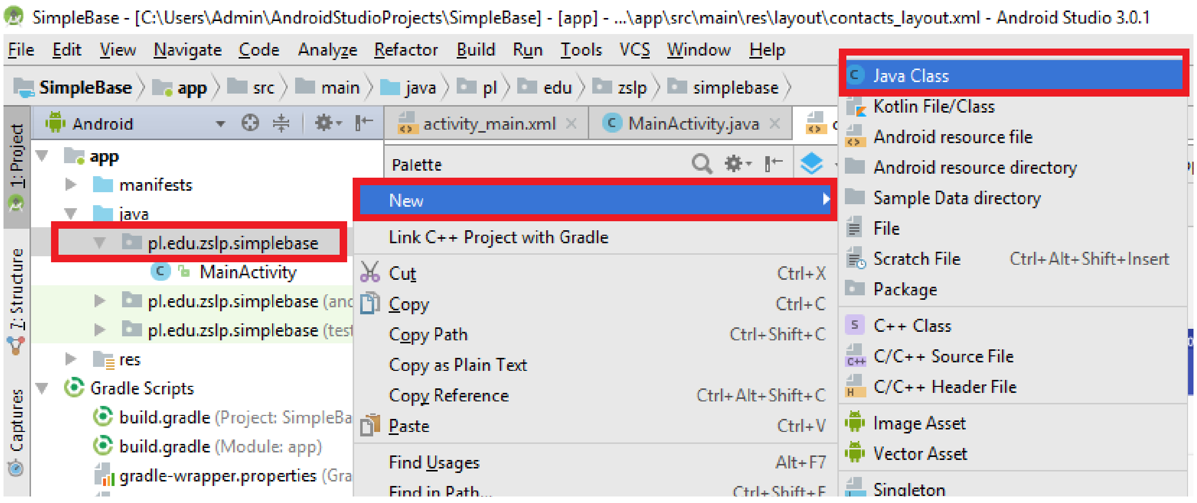


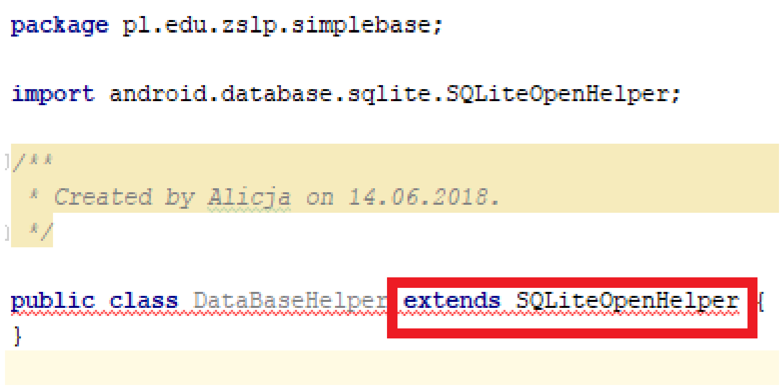
b) Put listView component to Names_activity layout

c) Open activity_main.xml file and make layout as shown at picture above



d) Create new class named **DataBaseHelper**

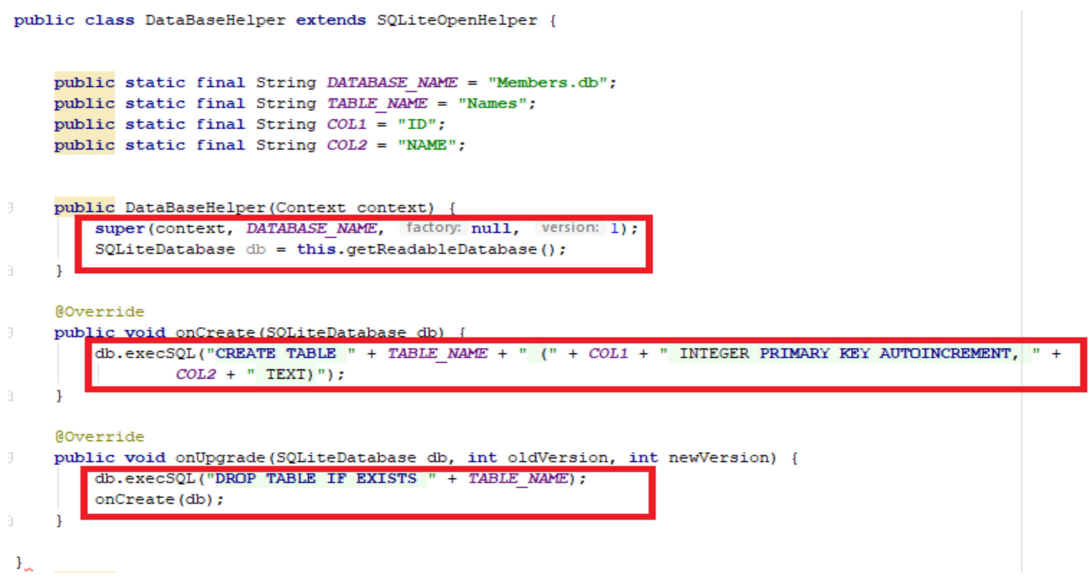

SimpleBase - [C:\Users\Admin\AndroidStudioProjects\SimpleBase] - [app] - ...\app\src\main\res\layout\contacts_layout.xml - Android Studio 3.0.1
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
SimpleBase app src main java pl edu zslp simplebase
Android
app
manifests
java
pl.edu.zslp.simplebase
MainActivity
pl.edu.zslp.simplebase (and
pl.edu.zslp.simplebase (test
res
Gradle Scripts
build.gradle (Project: SimpleBa
build.gradle (Module: app)
gradle-wrapper.properties (Gra
activity_main.xml
MainActivity.java
Palette
New
Link C++ Project with Gradle
Cut Ctrl+X
Copy Ctrl+C
Copy Path Ctrl+Shift+C
Copy as Plain Text
Copy Reference Ctrl+Alt+Shift+C
Paste Ctrl+V
Find Usages Alt+F7
Find in Path... Ctrl+Shift+F
Java Class
Kotlin File/Class
Android resource file
Android resource directory
Sample Data directory
File
Scratch File Ctrl+Alt+Shift+Insert
Package
C++ Class
C/C++ Source File
C/C++ Header File
Image Asset
Vector Asset
Singleton



```
package pl.edu.zslp.simplebase;

import android.database.sqlite.SQLiteOpenHelper;

/**
 * Created by Alicja on 14.06.2018.
 */

public class DataBaseHelper extends SQLiteOpenHelper {
}
```

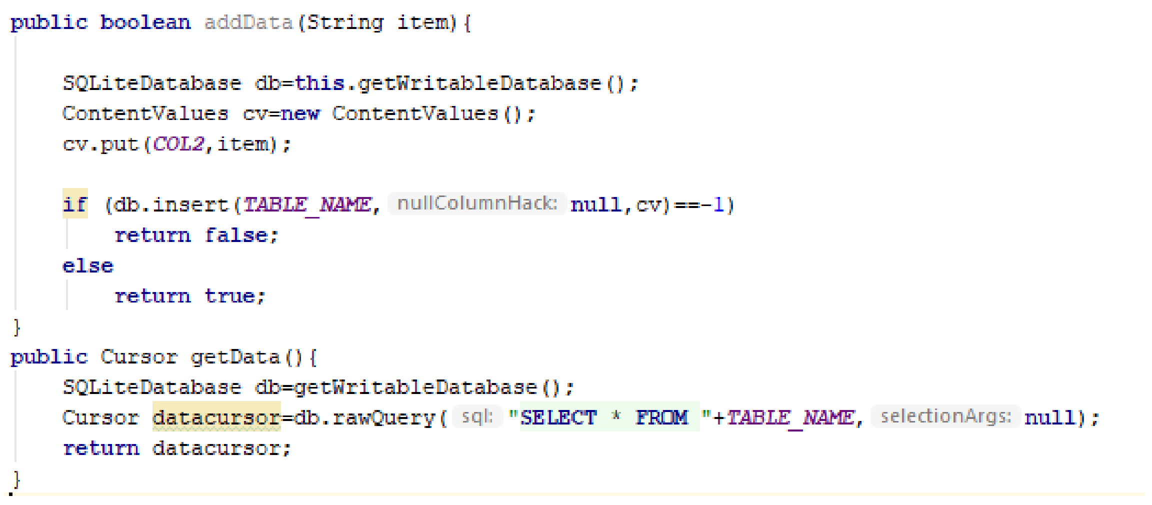
- After name of class write it extands SQLiteOpenHelper
- Click **alt+Enter** on underlined text and import all methods,
- Repeat it to create constructor

6. Rewrite metods responsible for creating table and upgrading table



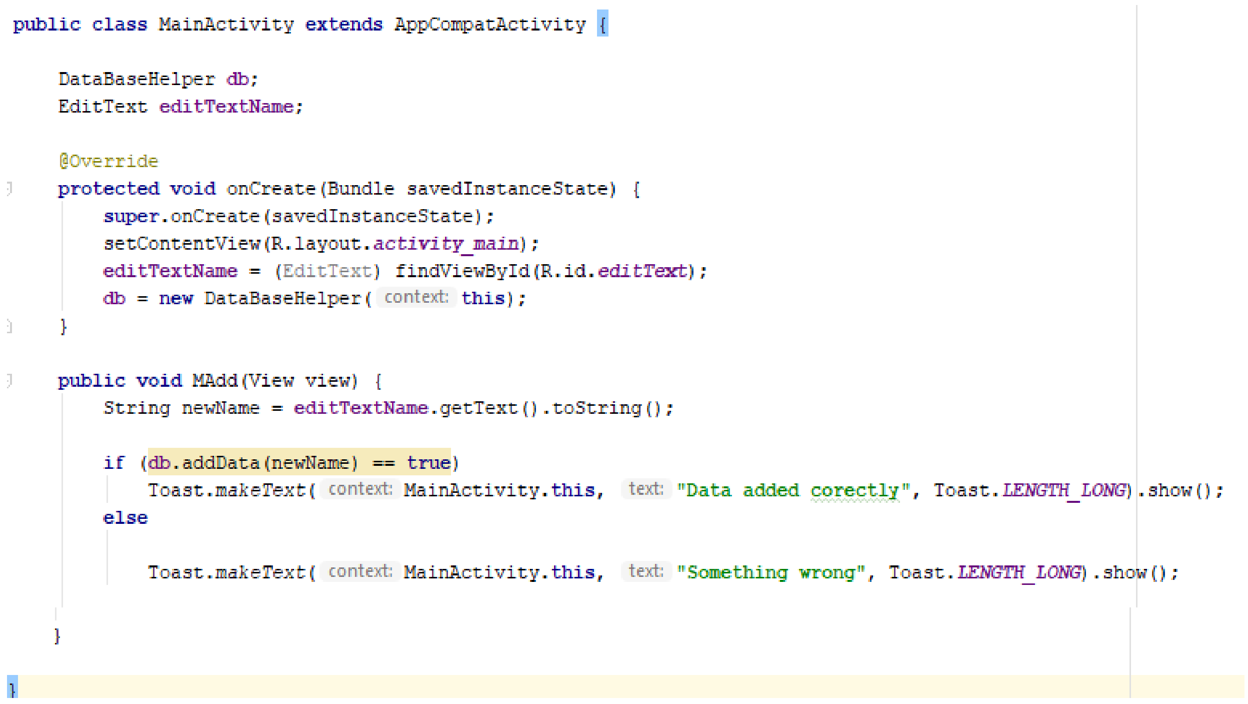
```
public class DataBaseHelper extends SQLiteOpenHelper {


    public static final String DATABASE_NAME = "Members.db";
    public static final String TABLE_NAME = "Names";
    public static final String COL1 = "ID";
    public static final String COL2 = "NAME";


    public DataBaseHelper(Context context) {
        super(context, DATABASE_NAME,  factory: null,  version: 1);
        SQLiteDatabase db = this.getReadableDatabase();
    }

    @Override
    public void onCreate(SQLiteDatabase db) {
        db.execSQL("CREATE TABLE " + TABLE_NAME + " (" + COL1 + " INTEGER PRIMARY KEY AUTOINCREMENT, " +
                COL2 + " TEXT)");
    }

    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
        db.execSQL("DROP TABLE IF EXISTS " + TABLE_NAME);
        onCreate(db);
    }

}
```

7. Create method addData and getData in DataBaseHelper class



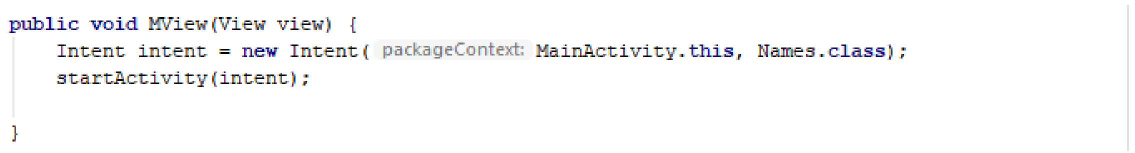
```
public boolean addData(String item){

    SQLiteDatabase db=this.getWritableDatabase();
    ContentValues cv=new ContentValues();
    cv.put(COL2,item);

    if (db.insert(TABLE_NAME, nullColumnHack: null,cv)==-1)
        return false;
    else
        return true;
}
public Cursor getData(){
    SQLiteDatabase db=getWritableDatabase();
    Cursor datacursor=db.rawQuery( sql: "SELECT * FROM "+TABLE_NAME, selectionArgs: null);
    return datacursor;
}
```

8. Open MainActivity.java
    a) Program addButton to add new name written in editText component to DataBase
    b) Run app and see if you get correct toast message



```
public class MainActivity extends AppCompatActivity {

    DataBaseHelper db;
    EditText editTextName;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        editTextName = (EditText) findViewById(R.id.editText);
        db = new DataBaseHelper( context: this);
    }

    public void MAdd(View view) {
        String newName = editTextName.getText().toString();

        if (db.addData(newName) == true)
            Toast.makeText( context: MainActivity.this,  text: "Data added corectly", Toast.LENGTH_LONG).show();
        else

            Toast.makeText( context: MainActivity.this,  text: "Something wrong", Toast.LENGTH_LONG).show();

    }

}
```

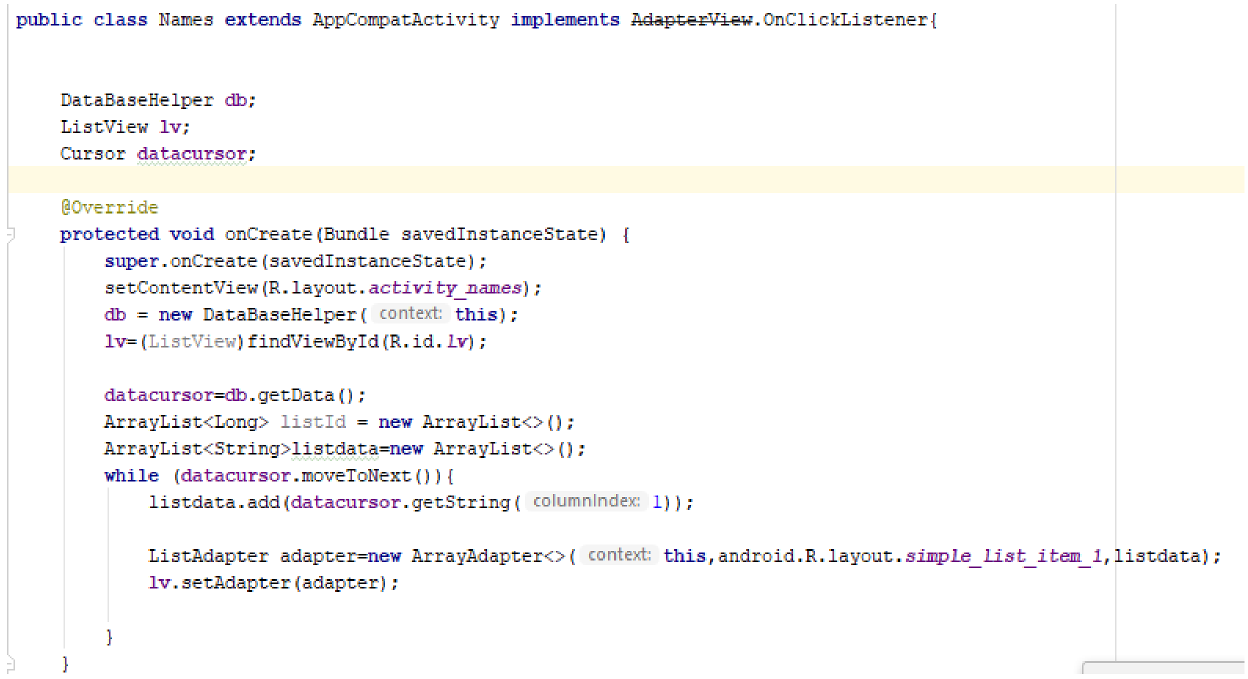
9. Now program ViewButton to show activity with listView



```
public void MView(View view) {
    Intent intent = new Intent( packageContext: MainActivity.this, Names.class);
    startActivity(intent);

}
```

10. Run app and check if button works

11. Open Names.java file



```
public class Names extends AppCompatActivity implements AdapterView.OnClickListener{


    DataBaseHelper db;
    ListView lv;
    Cursor datacursor;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_names);
        db = new DataBaseHelper( context: this);
        lv=(ListView)findViewById(R.id.lv);

        datacursor=db.getData();
        ArrayList<Long> listId = new ArrayList<>();
        ArrayList<String>listdata=new ArrayList<>();
        while (datacursor.moveToNext()){
            listdata.add(datacursor.getString( columnIndex: 1));

            ListAdapter adapter=new ArrayAdapter<>( context: this,android.R.layout.simple_list_item_1,listdata);
            lv.setAdapter(adapter);

        }
    }
```

12. Run an ap