

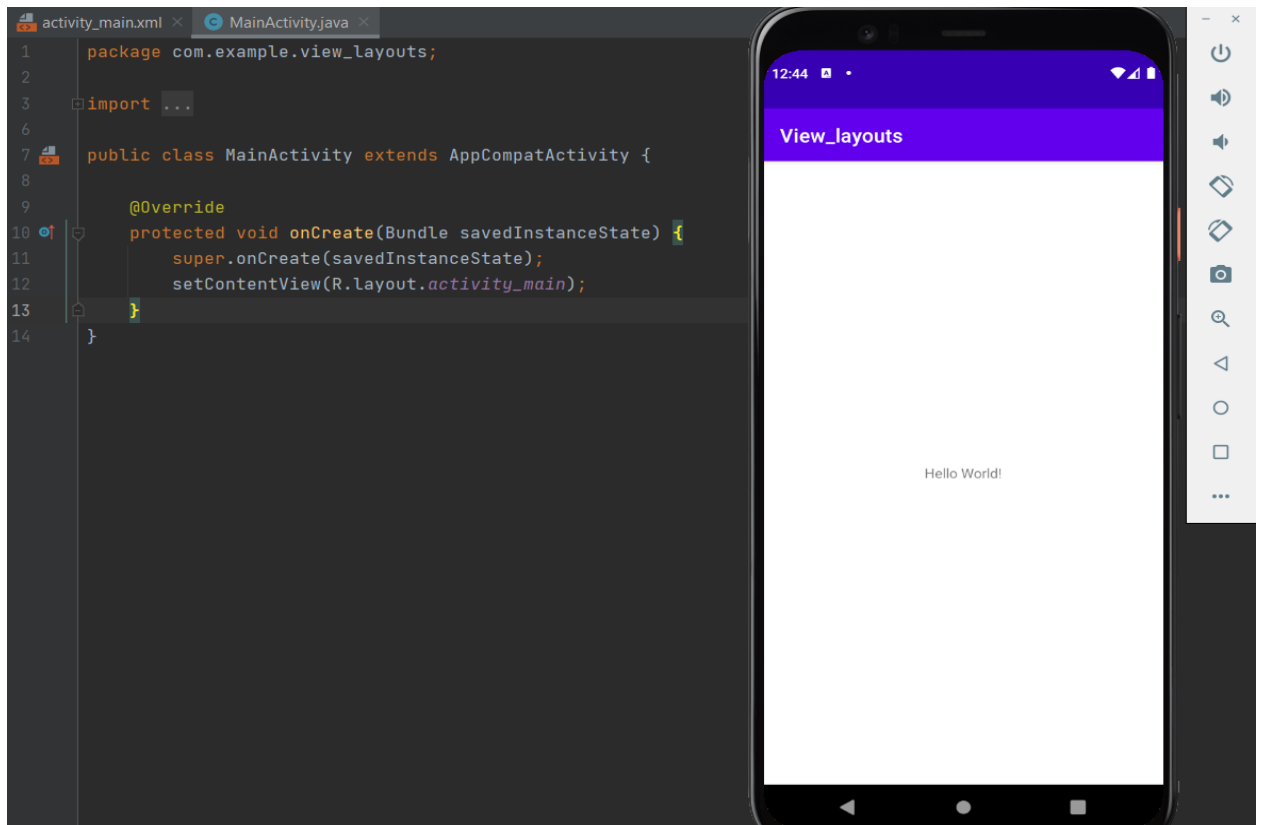
Отчёт по проделанной работе на семинаре по мобильной разработке (08.10.2021).

(выполнил Валяев Георгий Анатольевич)

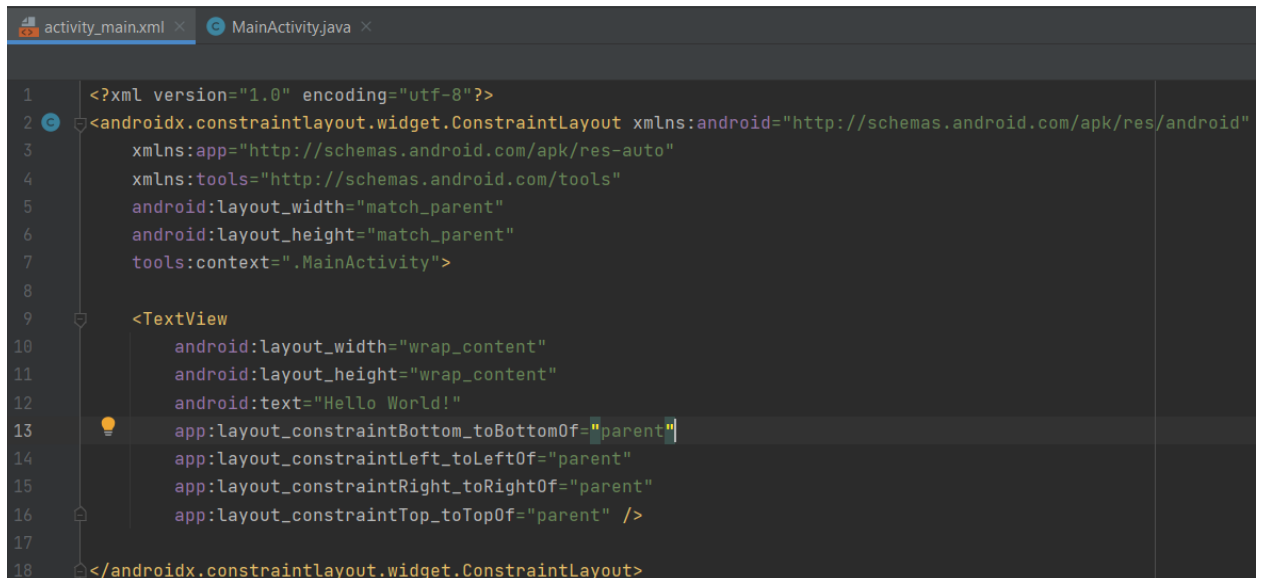
В этом отчёте будут отражены описания работ всех основных элементов активности приложения и детализация исходных файлов проекта. Собственно, с этого и стоит начать отчёт.

В новом, только что созданном проекте, мы видим 2 главных файла активности: MainActivity.java и activity_main.xml, каждый из которых выполняет конкретную роль в работоспособности приложения – поддержка и функционирование всех активностей и главного экрана, а также грамотное визуальное отображение в приложении всех элементов.

А именно, изначально в java-коде создан класс главной активности приложения и метод, который постоянно вызывается при запуске всего проекта.

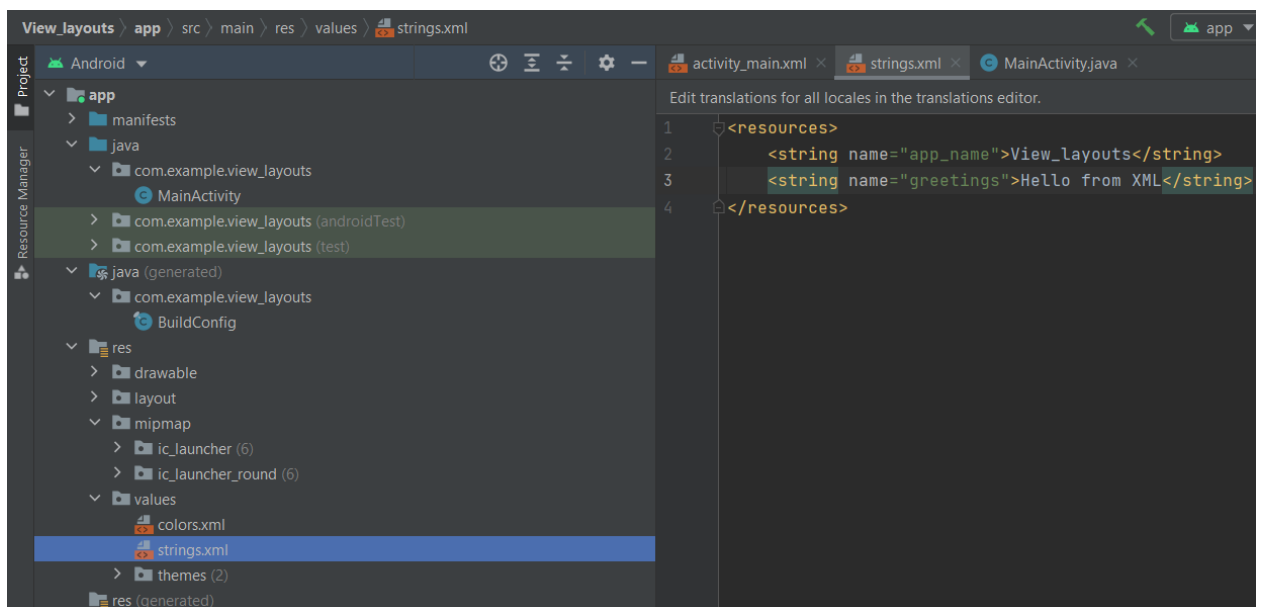


Если говорить про `activity_main.xml`, то в нём создан `ConstraintLayout`, задающий расположение элементов, и текстовое окно с текстом «Hello, World!»:



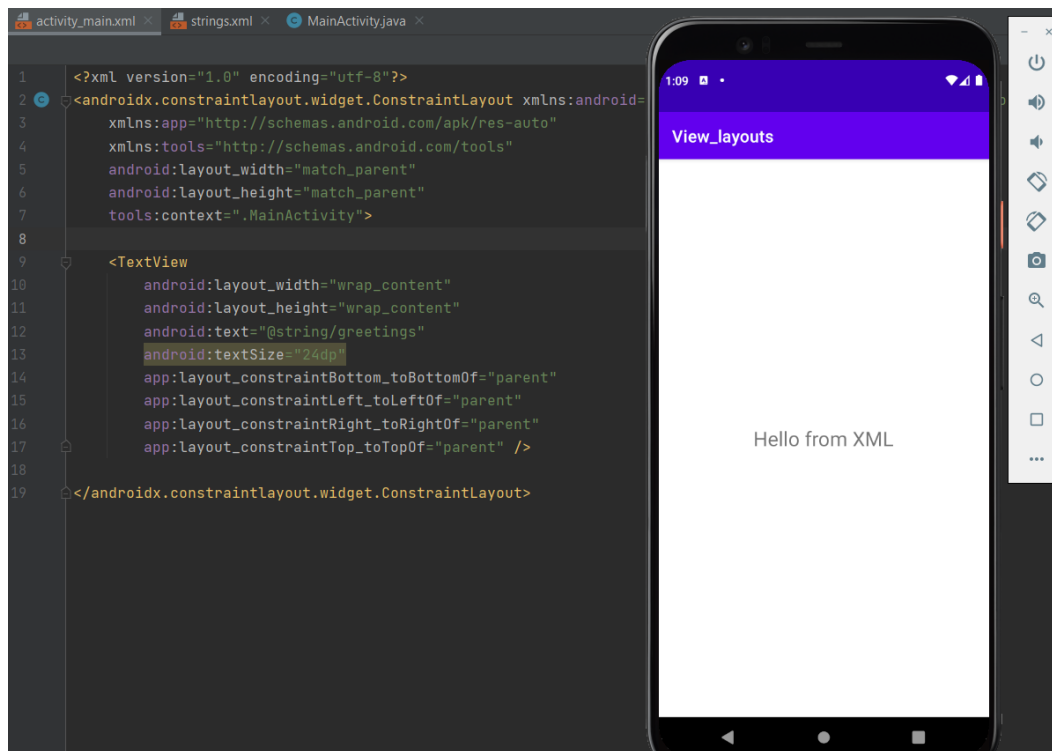
```
1  <?xml version="1.0" encoding="utf-8"?>
2  <androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
3      xmlns:app="http://schemas.android.com/apk/res-auto"
4      xmlns:tools="http://schemas.android.com/tools"
5      android:layout_width="match_parent"
6      android:layout_height="match_parent"
7      tools:context=".MainActivity">
8
9      <TextView
10         android:layout_width="wrap_content"
11         android:layout_height="wrap_content"
12         android:text="Hello World!"
13         app:layout_constraintBottom_toBottomOf="parent"
14         app:layout_constraintLeft_toLeftOf="parent"
15         app:layout_constraintRight_toRightOf="parent"
16         app:layout_constraintTop_toTopOf="parent" />
17
18  </androidx.constraintlayout.widget.ConstraintLayout>
```

Однако, ручное прописывание содержимого текстового окна считается дурным тоном в разработке... поэтому лучше стоит записывать весь контент и содержимое в ресурсный файл проекта – `strings.xml`:



```
1  <resources>
2      <string name="app_name">View_layouts</string>
3      <string name="greetings">Hello from XML</string>
4  </resources>
```

После этого в XML-файле активности заменим атрибут `text` на ссылку, ведущую к строке в ресурсном файле (“`@string/greetings`”) и прописываем дополнительно новый атрибут – `android:textSize=“24dp”`:

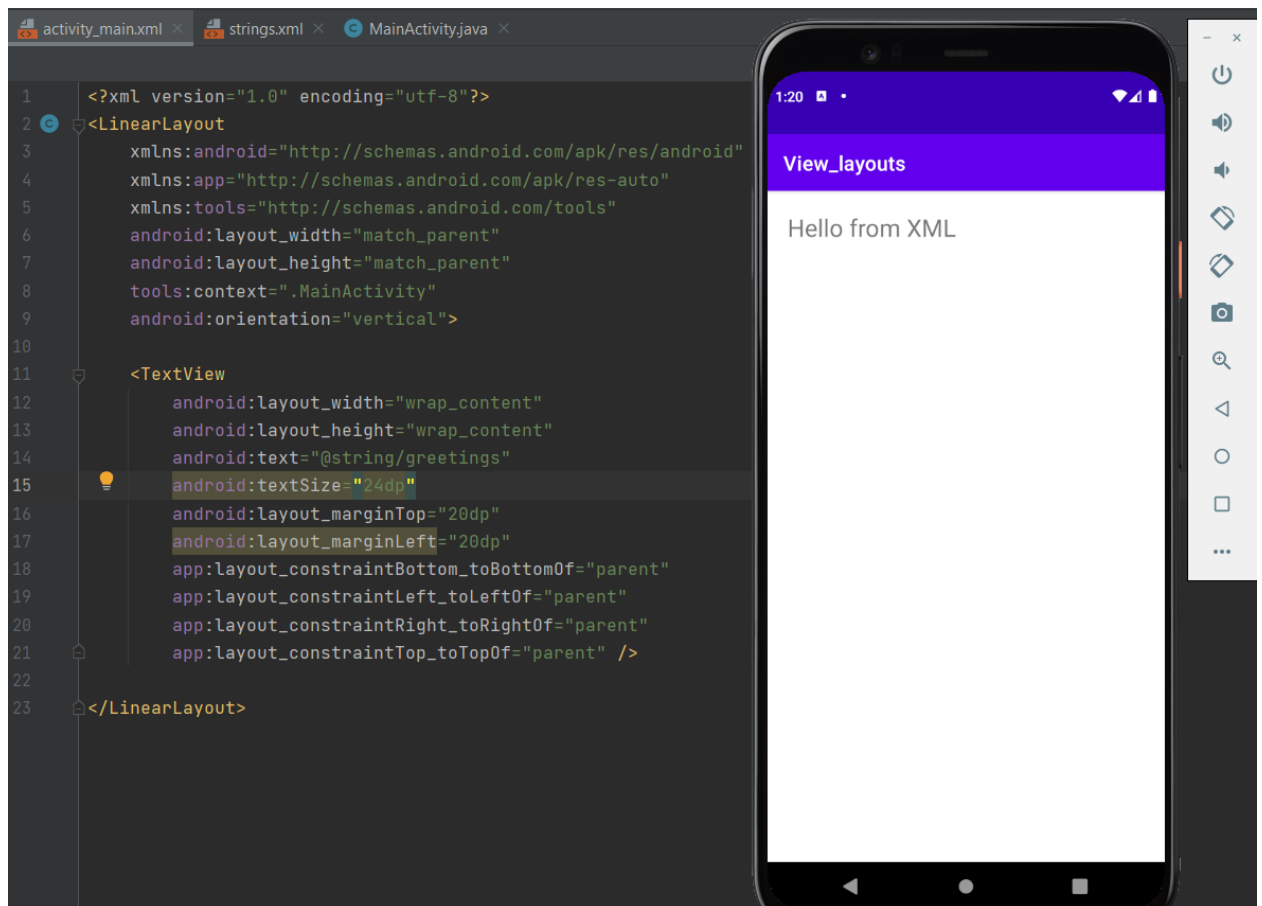


Теперь обратим внимание на вид расположения элементов приложения (ConstraintLayout) и сначала зададим более простой вид расположения – LinearLayout.

Также обратим внимание на такую особенность, что в таком случае текстовое окно будет вплотную к левому верхнему углу расположено, что имеет неправильное расположение.

Чтобы это исправить, добавим в activity_main.xml атрибуты (android:layout_marginTop="20dp" и android:layout_marginLeft="20dp"), которые помогут сместить текстовое окно приложения.

Смотрим, что получилось:



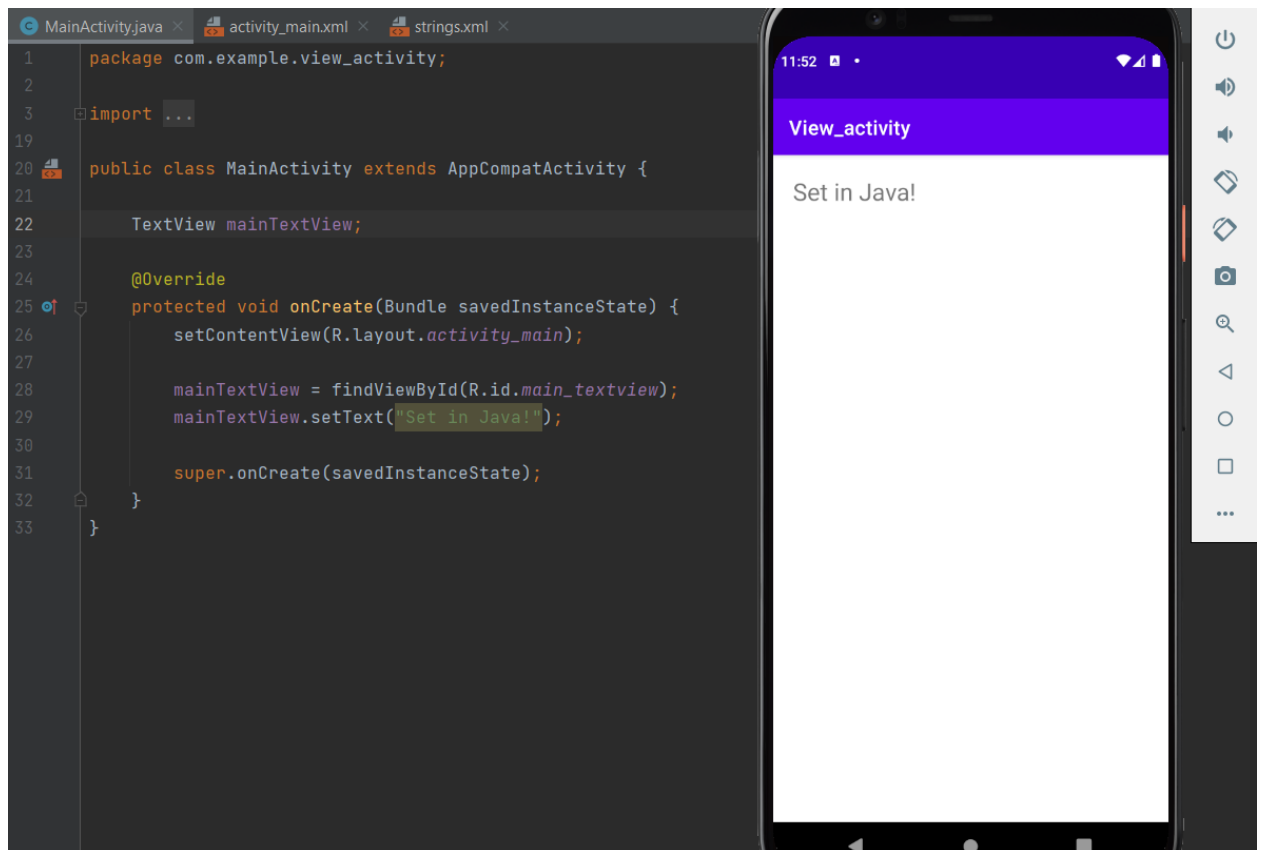
Перейдём теперь к возможности манипуляций с данными, которые выводятся на экран приложения при помощи файла MainActivity.java.

Для начала внутри главного класса (вне всех методов) инициализируем переменную `mainTextView` типа `TextView`, которая будет отвечать как раз таки за всё содержимое текстового окна.

Однако, в `activity_main.xml` нужно в теге `TextView` указать уникальный идентификатор, который поможет выводить изменения, перенесённые из Java-файла на текстовое окно приложения:

```
activity_main.xml × MainActivity.java × strings.xml ×
1      <?xml version="1.0" encoding="utf-8"?>
2      <LinearLayout
3          xmlns:android="http://schemas.android.com/apk/res/android"
4          xmlns:app="http://schemas.android.com/apk/res-auto"
5          xmlns:tools="http://schemas.android.com/tools"
6          android:layout_width="match_parent"
7          android:layout_height="match_parent"
8          tools:context=".MainActivity"
9          android:orientation="vertical">
10
11      <TextView
12          android:id="@+id/main_textview"
13          android:layout_width="wrap_content"
14          android:layout_height="wrap_content"
15          android:text="@string/greetings"
16          android:textSize="24dp"
17          android:layout_marginTop="20dp"
18          android:layout_marginLeft="20dp"
19          app:layout_constraintBottom_toBottomOf="parent"
20          app:layout_constraintLeft_toLeftOf="parent"
21          app:layout_constraintRight_toRightOf="parent"
22          app:layout_constraintTop_toTopOf="parent" />
23
24  </LinearLayout>
```

Далее переходим в метод `onCreate()` и дополняем его 2 строками кода, которые отвечают за нахождение данных текстового окна при помощи его ID в `activity_main.xml` и отправку нового текста в это окно для последующего отображения на экране приложения:

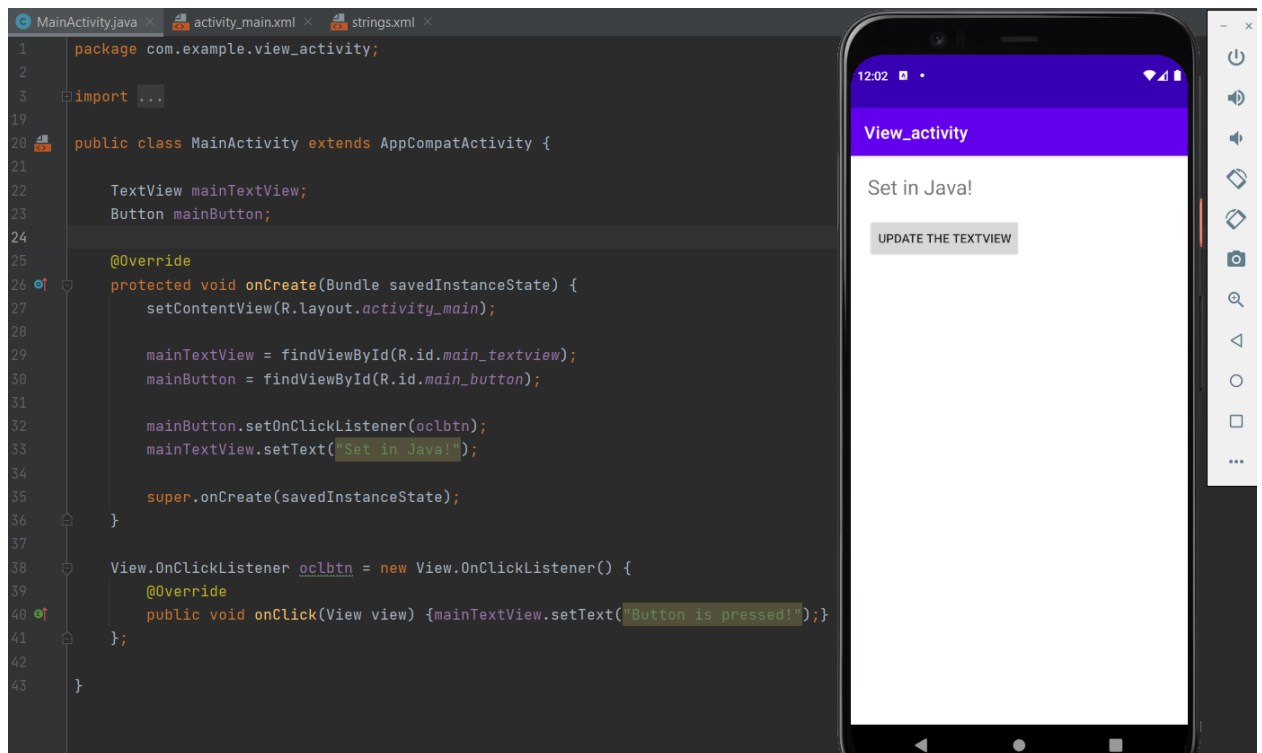


Следующим, не менее важным, элементом экрана приложения является кнопка, которую мы попытаемся прикрутить к нынешней программе.

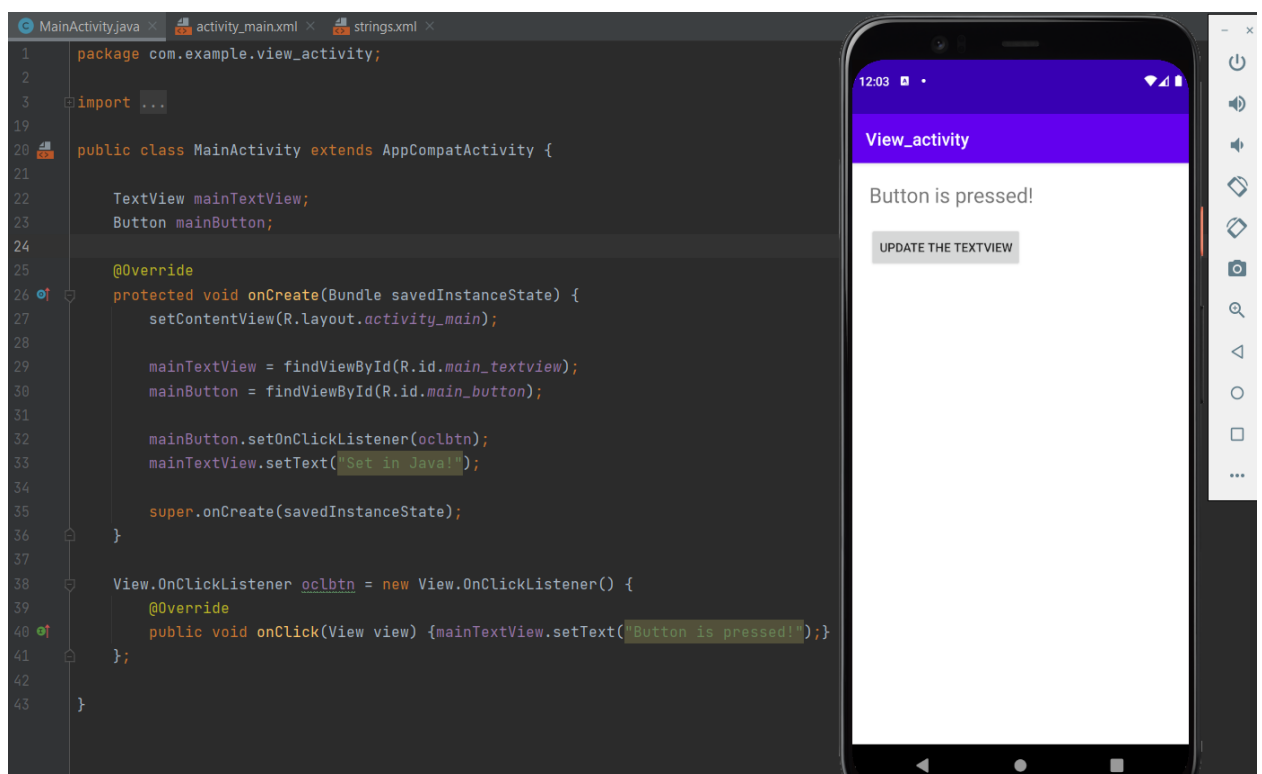
Сначала добавим элемент Button в activity_main.xml.

А в MainActivity.java при этом создадим новую переменную типа View.OnClickListener, в которую занесём одноимённую функцию, выполняющую алгоритм отправки нового содержимого в переменную типа текстового окна при как раз-таки нажатии самой кнопки на экране приложения:

```
6      android:layout_width="match_parent"
7      android:layout_height="match_parent"
8      tools:context=".MainActivity"
9      android:orientation="vertical">
10
11      <TextView
12          android:id="@+id/main_textview"
13          android:layout_width="wrap_content"
14          android:layout_height="wrap_content"
15          android:text="Hello from XML"
16          android:textSize="24dp"
17          android:layout_marginTop="20dp"
18          android:layout_marginLeft="20dp"
19          app:layout_constraintBottom_toBottomOf="parent"
20          app:layout_constraintLeft_toLeftOf="parent"
21          app:layout_constraintRight_toRightOf="parent"
22          app:layout_constraintTop_toTopOf="parent">
23      </TextView>
24
25      <Button
26          android:id="@+id/main_button"
27          android:layout_width="wrap_content"
28          android:layout_height="wrap_content"
29          android:layout_marginStart="20dp"
30          android:layout_marginLeft="20dp"
31          android:layout_marginTop="20dp"
32          android:text="Update the TextView">
33      </Button>
34
35
36  </LinearLayout>
```

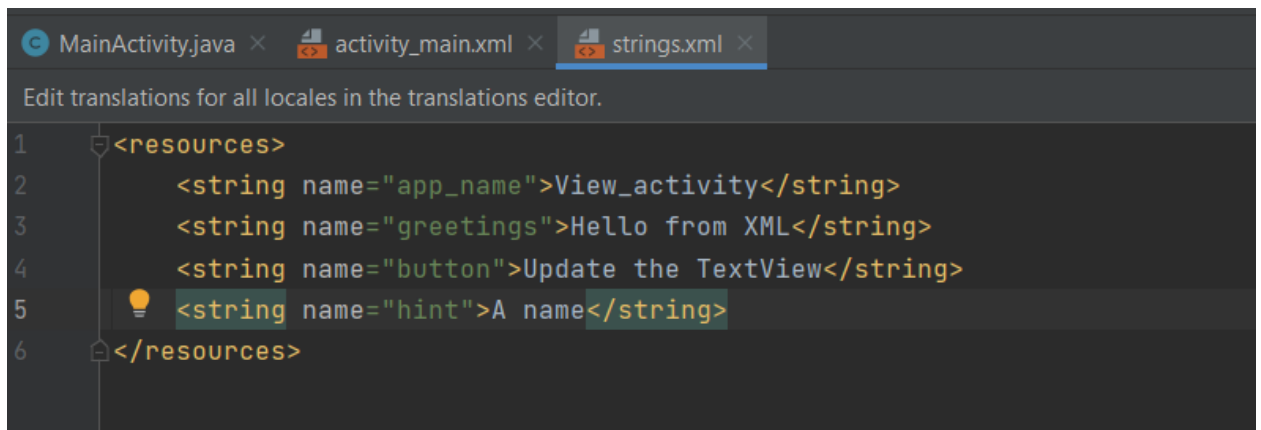


И в итоге при нажатии на кнопку произойдёт считывание этого процесса в главном классе и переменная `mainTextView` сменит своё содержимое на «Button is pressed!»:

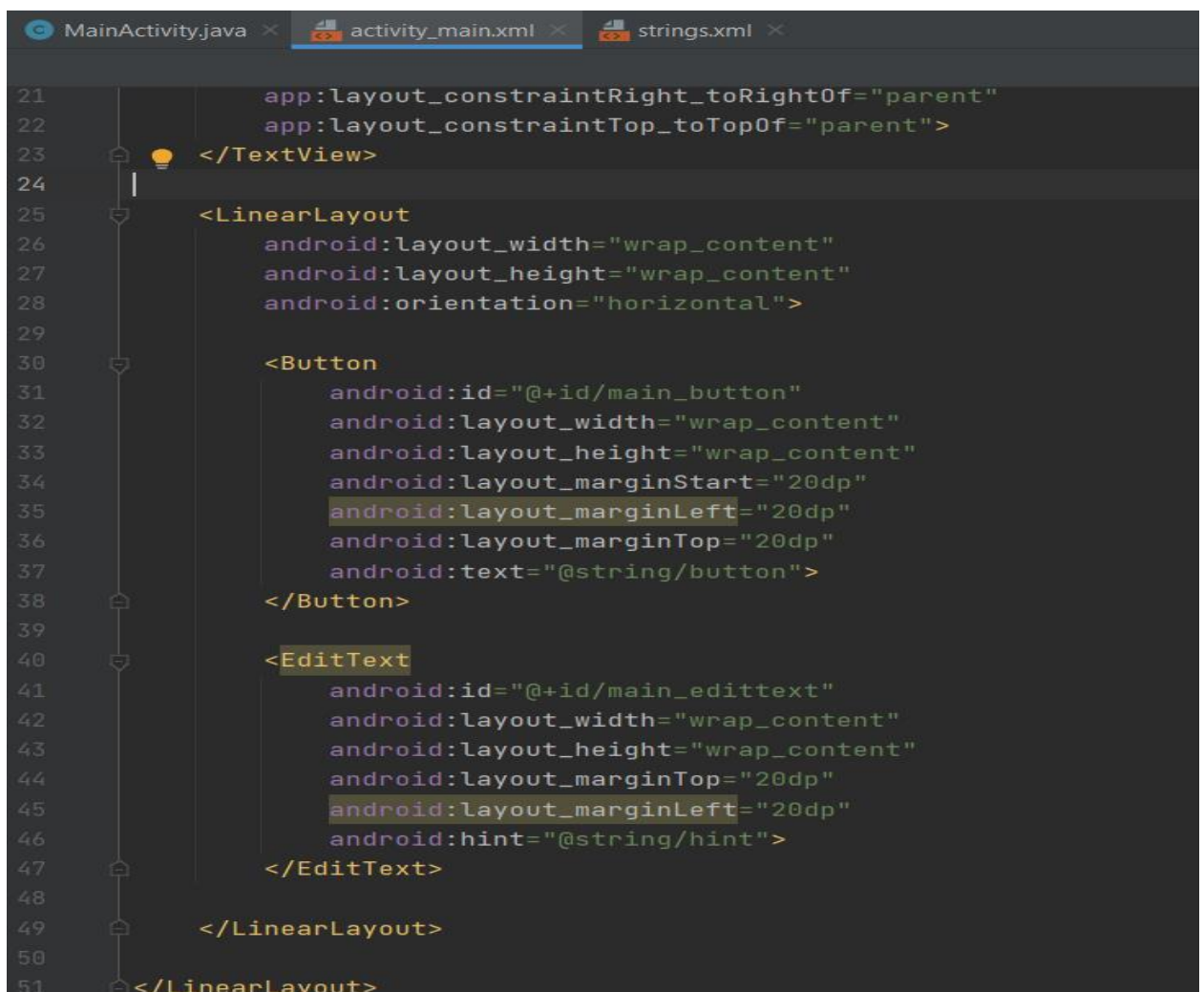


Переходим к важнейшему элементу при составлении и разработке форм регистрации и прочему – поле для ввода текста пользователем.

Однако, сделаем это более нестандартным образом: построим новый `LinearLayout` с горизонтальной ориентацией уже после тега поля с ознакомительным текстом и при этом добавим туда нашу кнопку и новый элемент `EditText` со своим `id`, расположением относительно кнопки и фоновым текстом, наполнение которого есть в файле `strings.xml`.

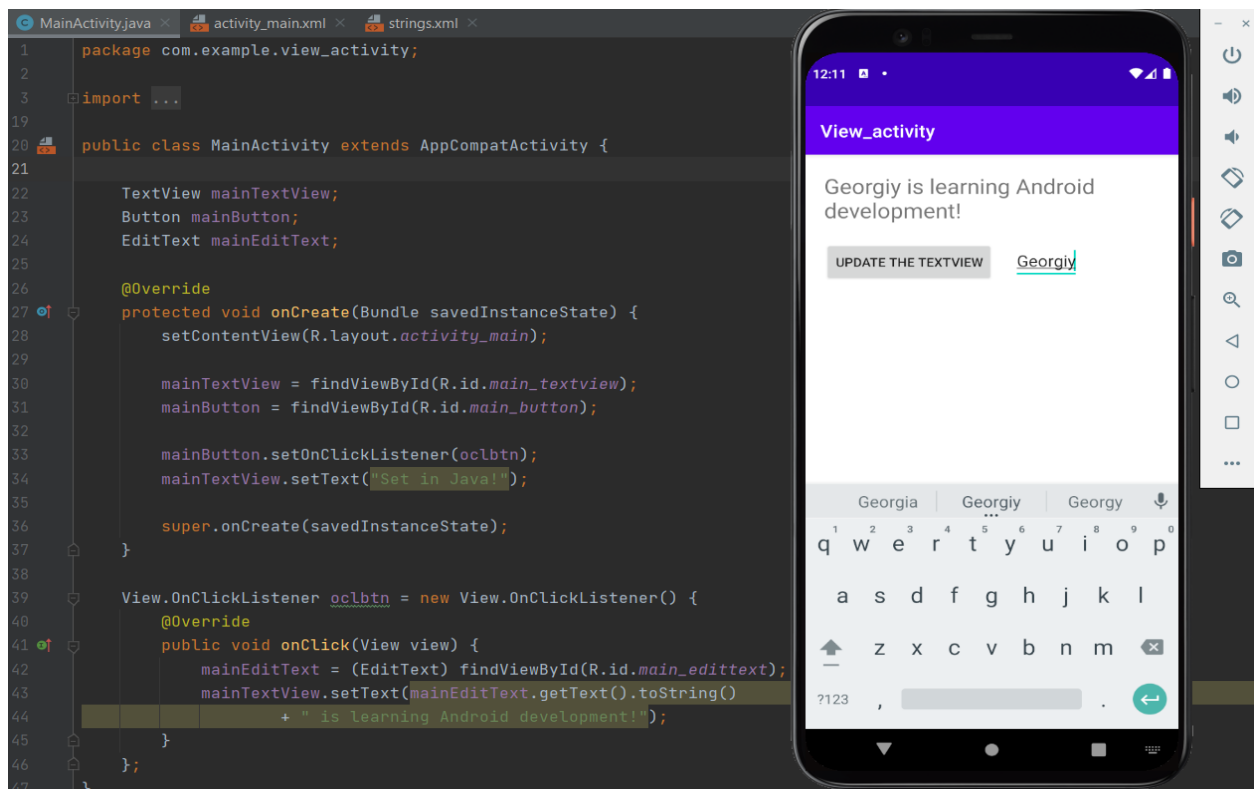
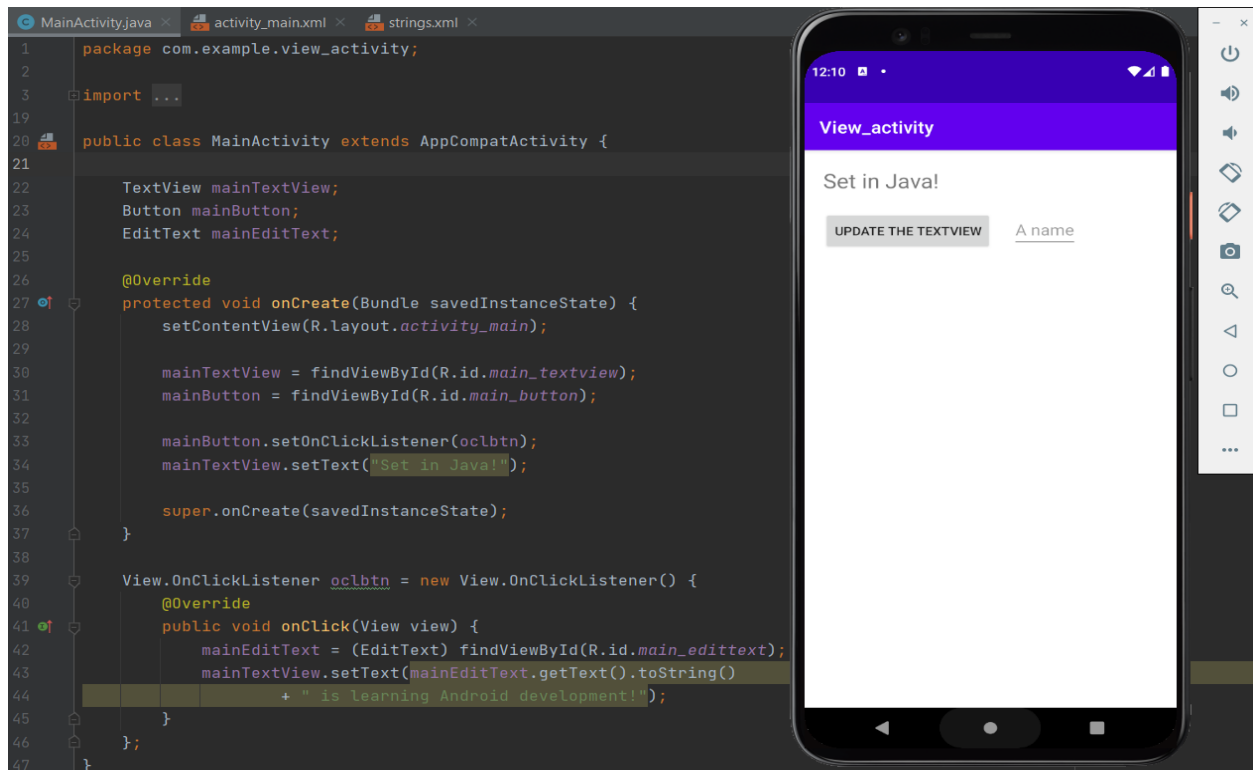


```
1 <resources>
2     <string name="app_name">View_activity</string>
3     <string name="greetings">Hello from XML</string>
4     <string name="button">Update the TextView</string>
5     <string name="hint">A name</string>
6 </resources>
```



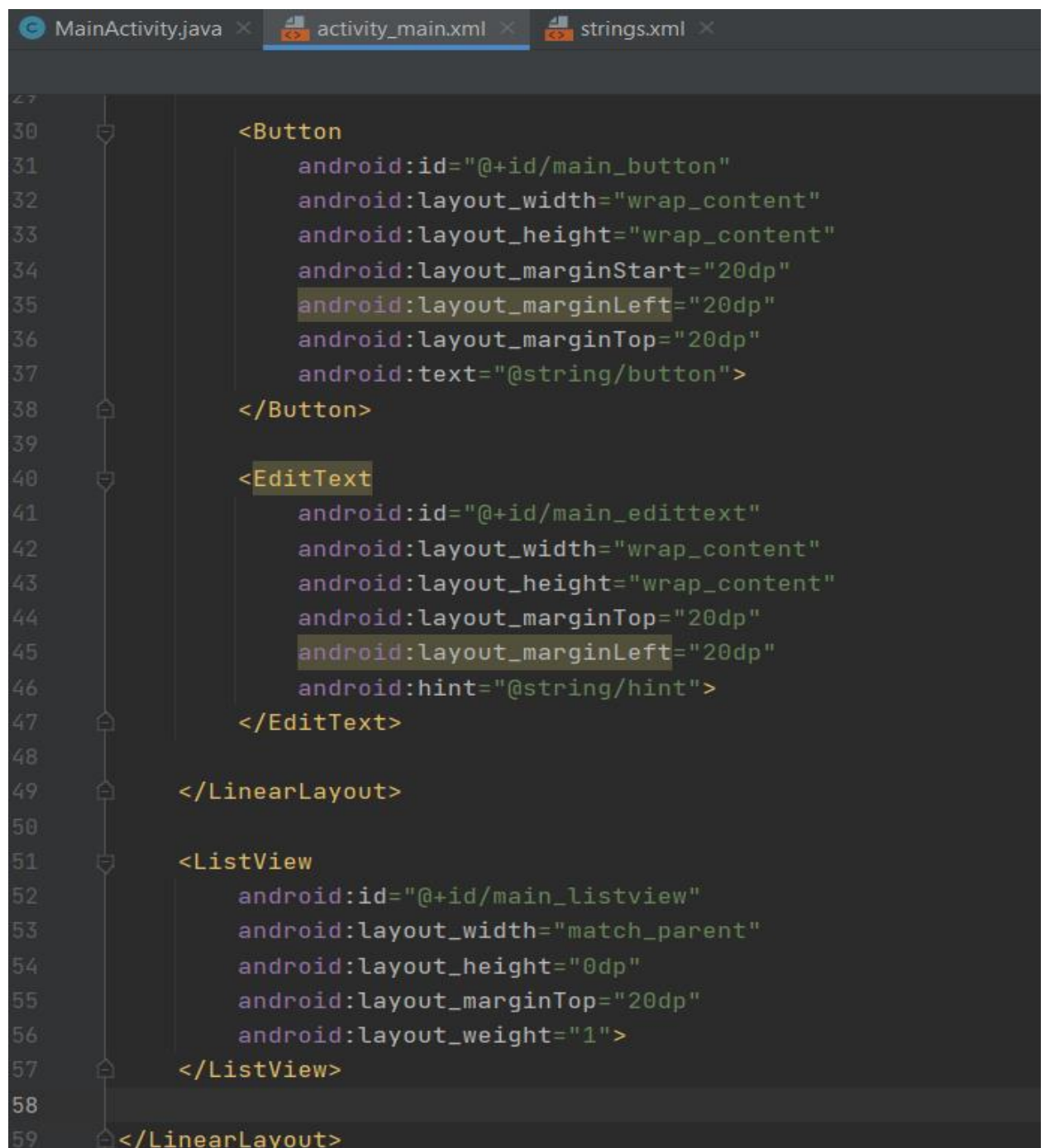
```
21         app:layout_constraintRight_toRightOf="parent"
22         app:layout_constraintTop_toTopOf="parent">
23     </TextView>
24
25     <LinearLayout
26         android:layout_width="wrap_content"
27         android:layout_height="wrap_content"
28         android:orientation="horizontal">
29
30         <Button
31             android:id="@+id/main_button"
32             android:layout_width="wrap_content"
33             android:layout_height="wrap_content"
34             android:layout_marginStart="20dp"
35             android:layout_marginLeft="20dp"
36             android:layout_marginTop="20dp"
37             android:text="@string/button">
38         </Button>
39
40         <EditText
41             android:id="@+id/main_edittext"
42             android:layout_width="wrap_content"
43             android:layout_height="wrap_content"
44             android:layout_marginTop="20dp"
45             android:layout_marginLeft="20dp"
46             android:hint="@string/hint">
47         </EditText>
48
49     </LinearLayout>
50
51 </LinearLayout>
```

Соответственно, нужно связать с этим и MainActivity.java. Поэтому пропишем новую переменную с типом TextEdit, пропишем в функции нажатия на кнопку код считывания содержимого с текстового поля и вывода на текстовое окно результат ввода и конкатенированную строку « is learning Android development!»:»:



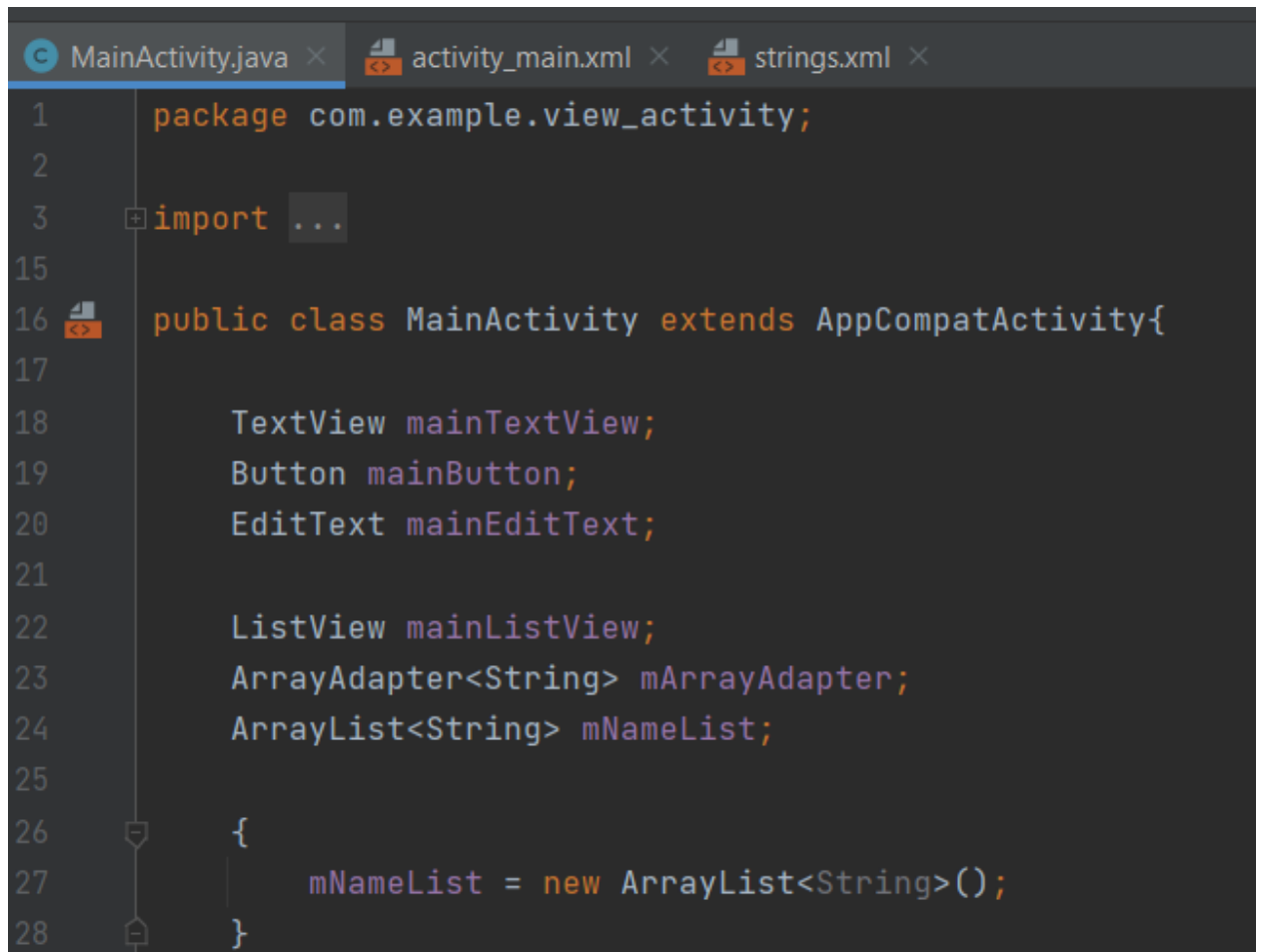
Сейчас стоит затронуть тему списков в разработке приложения, начав при этом с создания простого динамического списка вводимого пользователем текста, который будет отображён ниже кнопки в приложении.

Первым шагом создадим в `activity_main.xml` элемент списка `ListView`, у которого будет полноэкранное (ниже кнопки, соответственно) разрешение на отображение данных в приложении, также указав единый размер показа каждого нового элемента динамического списка данных:



```
27
30     <Button
31         android:id="@+id/main_button"
32         android:layout_width="wrap_content"
33         android:layout_height="wrap_content"
34         android:layout_marginStart="20dp"
35         android:layout_marginLeft="20dp"
36         android:layout_marginTop="20dp"
37         android:text="@string/button">
38     </Button>
39
40     <EditText
41         android:id="@+id/main_edittext"
42         android:layout_width="wrap_content"
43         android:layout_height="wrap_content"
44         android:layout_marginTop="20dp"
45         android:layout_marginLeft="20dp"
46         android:hint="@string/hint">
47     </EditText>
48
49 </LinearLayout>
50
51 <ListView
52     android:id="@+id/main_listview"
53     android:layout_width="match_parent"
54     android:layout_height="0dp"
55     android:layout_marginTop="20dp"
56     android:layout_weight="1">
57 </ListView>
58
59 </LinearLayout>
```

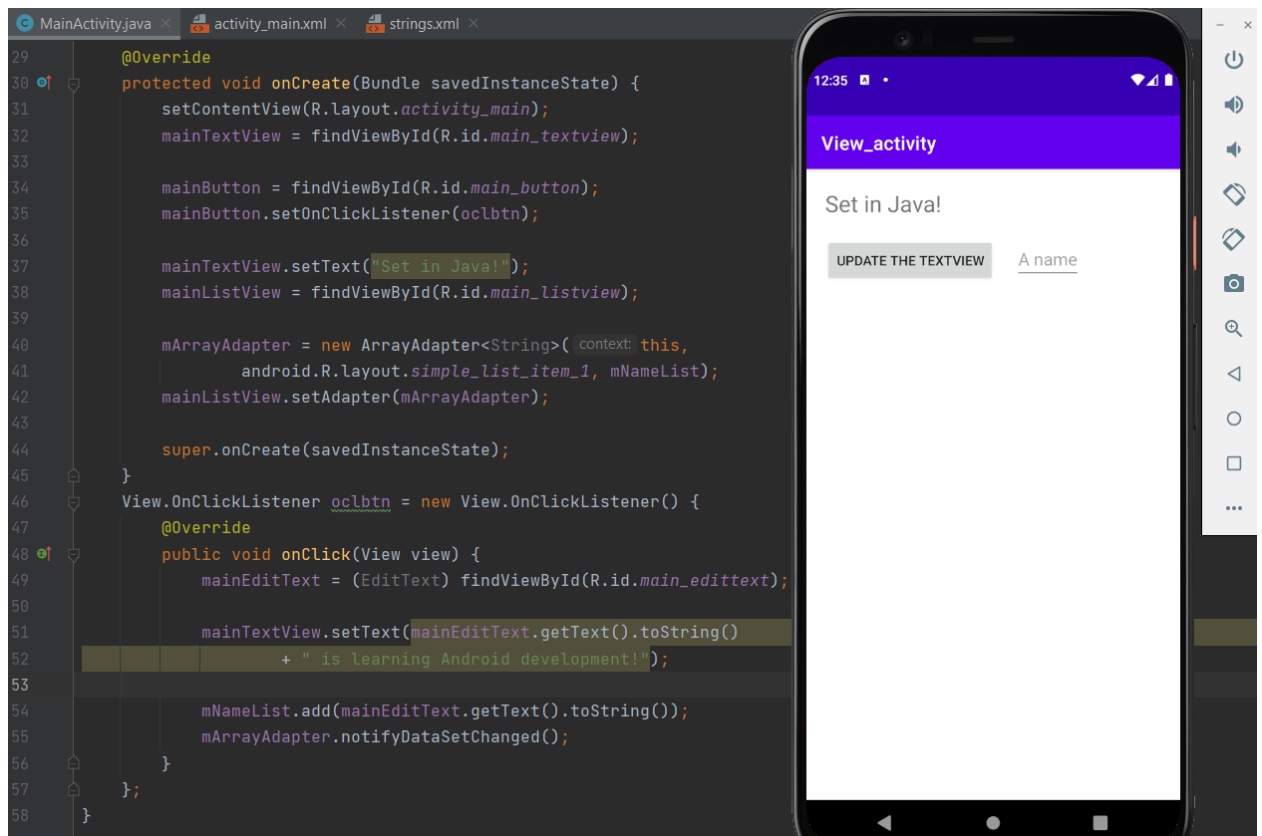
Теперь в MainActivity.java инициализируем три переменные – сам ListView, адаптер списка и динамический список с введенными пользователем данными через кнопку, написав при этом небольшой код создания нашего динамического списка:



```
1 package com.example.view_activity;
2
3 import ...
4
15
16 public class MainActivity extends AppCompatActivity{
17
18     TextView mainTextView;
19     Button mainButton;
20     EditText mainEditText;
21
22     ListView mainListView;
23     ArrayAdapter<String> mAdapter;
24     ArrayList<String> mNameList;
25
26     {
27         mNameList = new ArrayList<String>();
28     }
```

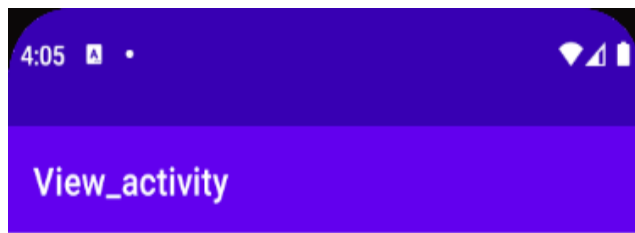
Затем в главном методе создания считываем данные с ListView, создаём новый адаптер массива в виде простого списка элементов через строчку и отправляем данные адаптеру.

И уже в методе обработчика нажатия мы добавляем в список новые данные в строковом типе и говорим адаптеру, что нужно после нажатия кнопки добавить эти данные списка на экран (при каждом нажатия будут добавляться новые данные):



Смотрим, что получилось:

(на след. странице)

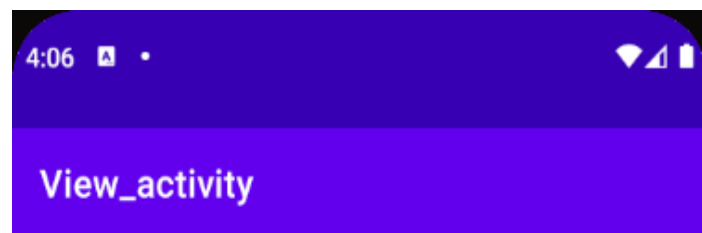


Masha is learning Android development!

UPDATE THE TEXTVIEW

Masha

Masha



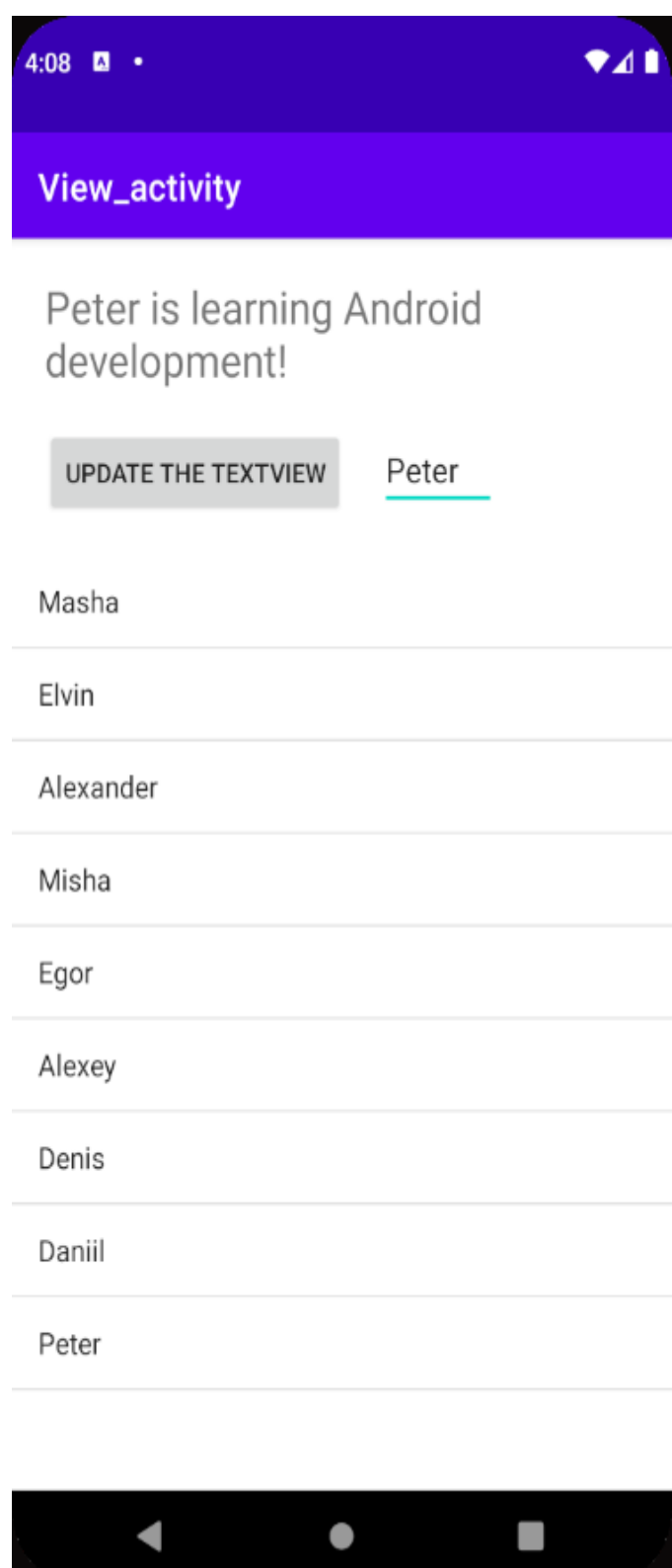
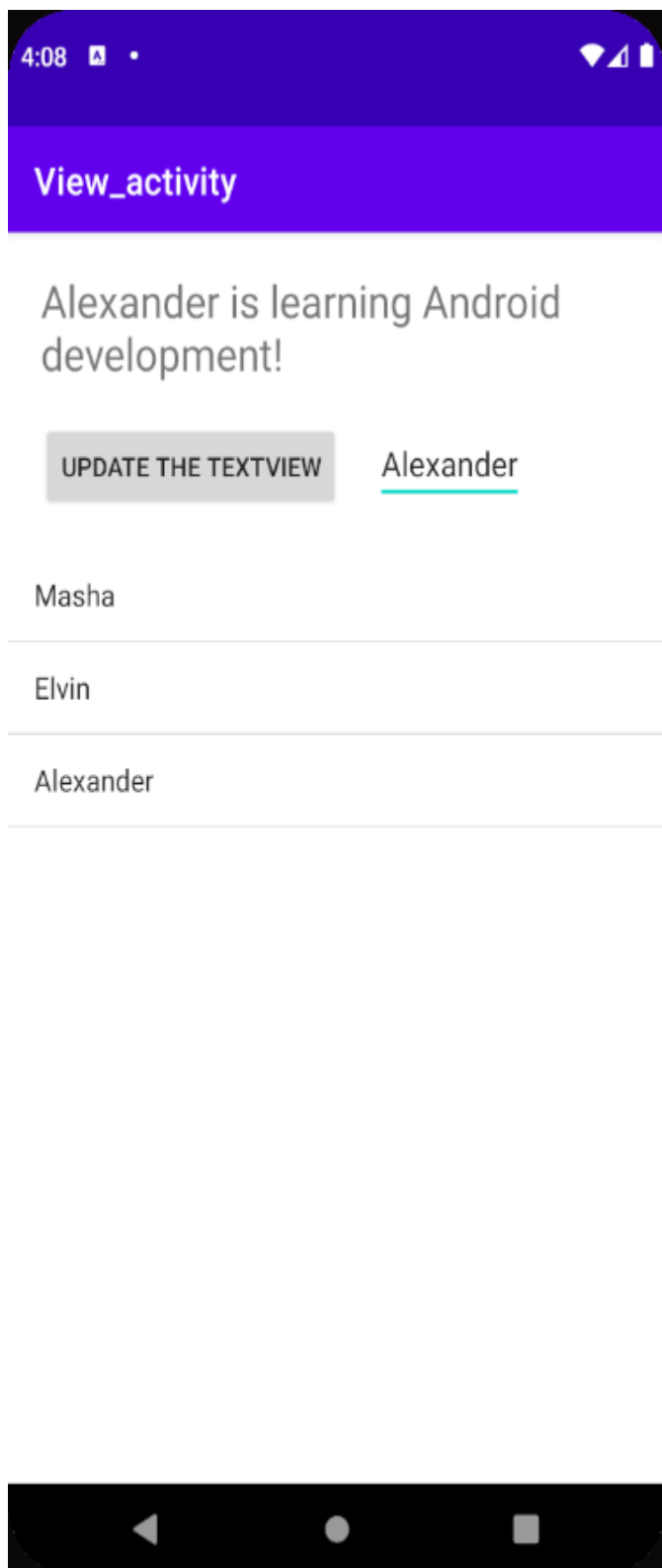
Elvin is learning Android development!

UPDATE THE TEXTVIEW

Elvin

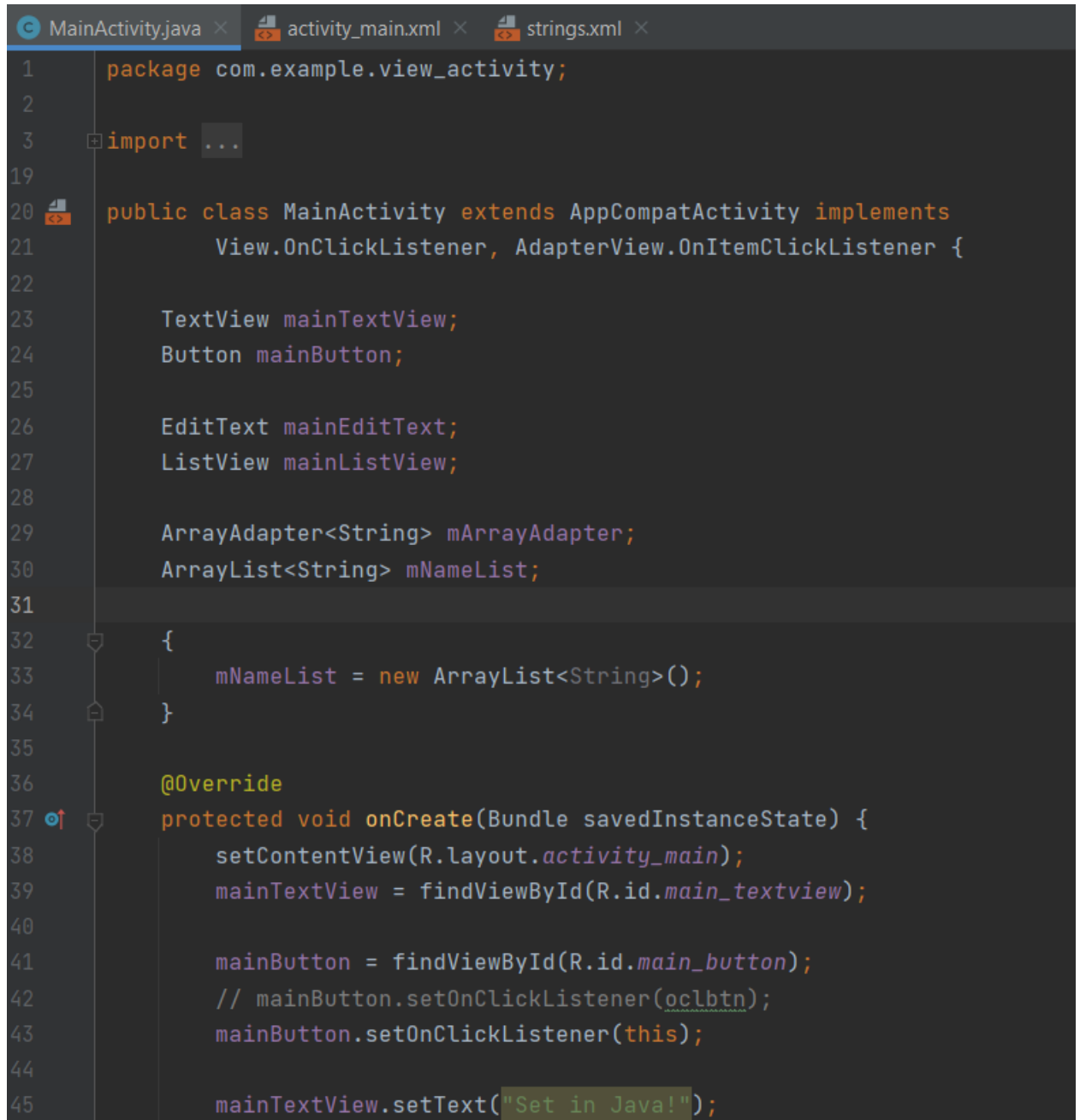
Masha

Elvin



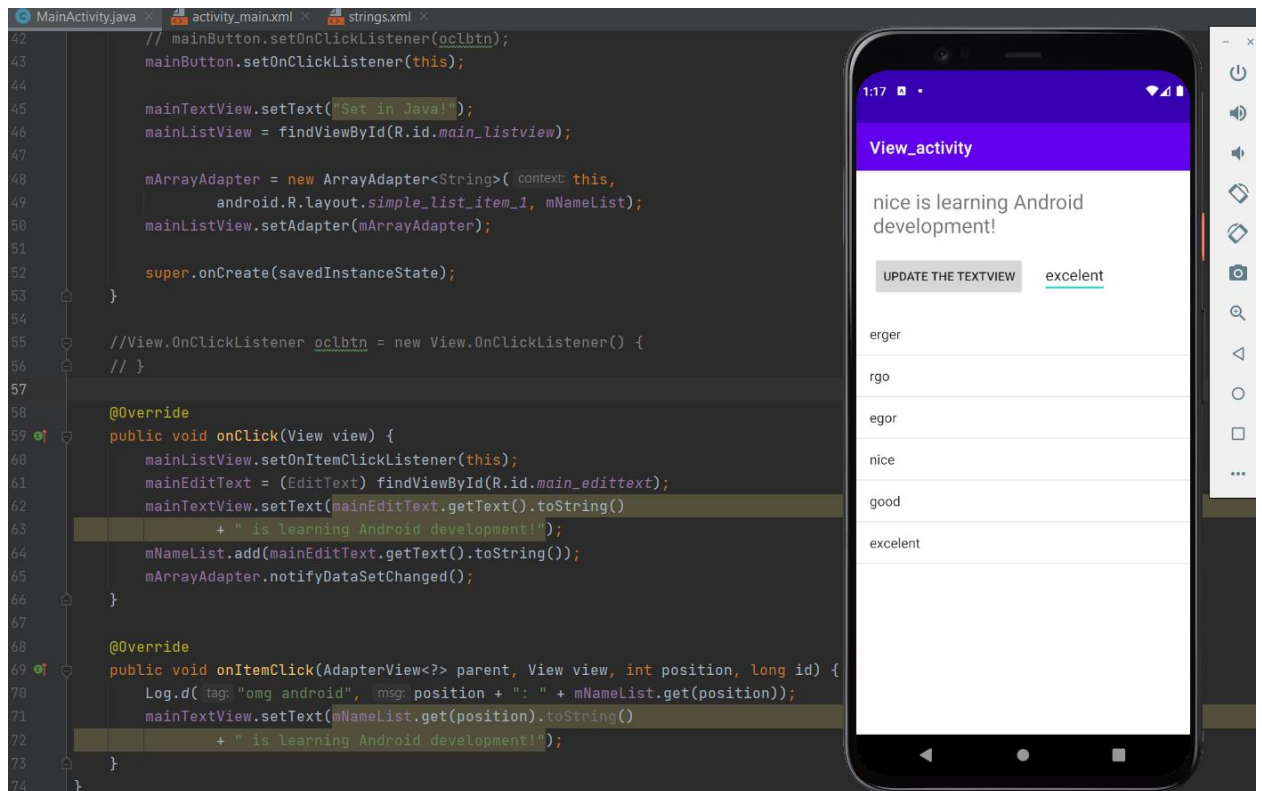
Но такой простой список используется не так часто, как интерактивный, который позволяет продемонстрировать реакцию на нажатия пользователем своих же собственных введённых данных списка. Попробуем реализовать эту возможность.

Сначала укажем в основном классе обработку нажатия на список:



```
1 package com.example.view_activity;
2
3 import ...
4
19
20 public class MainActivity extends AppCompatActivity implements
21     View.OnClickListener, AdapterView.OnItemClickListener {
22
23     TextView mainTextView;
24     Button mainButton;
25
26     EditText mainEditText;
27     ListView mainListView;
28
29     ArrayAdapter<String> mAdapter;
30     ArrayList<String> mNameList;
31
32     {
33         mNameList = new ArrayList<String>();
34     }
35
36     @Override
37     protected void onCreate(Bundle savedInstanceState) {
38         setContentView(R.layout.activity_main);
39         mainTextView = findViewById(R.id.main_textview);
40
41         mainButton = findViewById(R.id.main_button);
42         // mainButton.setOnClickListener(oclbtn);
43         mainButton.setOnClickListener(this);
44
45         mainTextView.setText("Set in Java!");
```

Далее в методе обработчика нажатия кнопки добавим регистрацию нажатий на уже ранее введённые данные и затем пропишем новый метод, в котором при тапе на элемент списка он отображался в текстовом поле:



Таким образом, при нажатии на какой-то пункт, введённый ранее, будет в текстовом окне выведена информация о нём, которая отображалась ранее при первом его вводе и входе в список.

Далее стоит разобраться с такой ситуацией, когда в обработчике нажатия отображалась информация о каком-то конкретном нажатии на определённую кнопку среди множества других кнопок.

При этом у нас есть только один обработчик событий с такого рода назначением.

Для реализации этого функционала сначала создадим LinearLayout с двумя кнопками – OK и Cancel.

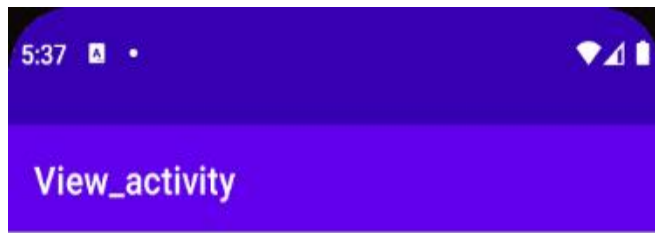
```
MainActivity.java × activity_main.xml × strings.xml ×
44         android:layout_marginTop="20dp"
45         android:layout_marginLeft="20dp"
46         android:hint="A name">
47     </EditText>
48
49 </LinearLayout>
50
51 <LinearLayout
52     android:layout_width="match_parent"
53     android:layout_height="match_parent"
54     android:orientation="horizontal">
55
56     <Button
57         android:id="@+id/ok_btn"
58         android:layout_width="wrap_content"
59         android:layout_height="wrap_content"
60         android:layout_weight="1"
61         android:text="@string/ok">
62     </Button>
63
64     <Button
65         android:id="@+id/cnc_btn"
66         android:layout_width="wrap_content"
67         android:layout_height="wrap_content"
68         android:layout_weight="1"
69         android:text="@string/cancel">
70     </Button>
71
72 </LinearLayout>
73
74 </LinearLayout>
```

Перейдя в MainActivity.java, инициализируем 2 переменные типа Button, назначаем далее созданный класс View.OnClickListener() как обработчик событий двум кнопкам, и в конце файла прописываем сам класс через его объект (oclbtn).

Наконец, внутри класса при помощи метода получения ID нажатой кнопки и сравнения его с ID кнопок в activity_main.xml мы по итогу отправляем нужные данные на текстовое окно вывода информации пользователю на экран приложения:

```
MainActivity.java x activity_main.xml x strings.xml x
1 package com.example.view_activity;
2
3 import ...
19
20 public class MainActivity extends AppCompatActivity {
21
22     TextView mainTextView;
23     Button mainButton;
24     EditText mainEditText;
25
26     Button ok_btn, cnc_btn;
27
28     @Override
29     protected void onCreate(Bundle savedInstanceState) {
30         setContentView(R.layout.activity_main);
31
32         ok_btn = findViewById(R.id.ok_btn);
33         cnc_btn = findViewById(R.id.cnc_btn);
34
35         mainTextView = findViewById(R.id.main_textview);
36         mainButton = findViewById(R.id.main_button);
37
38         ok_btn.setOnClickListener(oclbtn);
39         cnc_btn.setOnClickListener(oclbtn);
40
41         mainTextView.setText("Set in Java!");
42
43         super.onCreate(savedInstanceState);
44     }
```

```
MainActivity.java x activity_main.xml x strings.xml x
35
36     mainTextView = findViewById(R.id.main_textview);
37     mainButton = findViewById(R.id.main_button);
38
39     ok_btn.setOnClickListener(oclbtn);
40     cnc_btn.setOnClickListener(oclbtn);
41
42     mainTextView.setText("Set in Java!");
43
44     super.onCreate(savedInstanceState);
45 }
46
47 View.OnClickListener oclbtn = new View.OnClickListener() {
48     @Override
49     public void onClick(View view) {
50         switch (view.getId()) {
51             case R.id.ok_btn:
52                 mainTextView.setText("Нажата кнопка OK");
53                 break;
54
55             case R.id.cnc_btn:
56                 mainTextView.setText("Нажата кнопка Cancel");
57                 break;
58         }
59     }
60 };
61
62 }
```

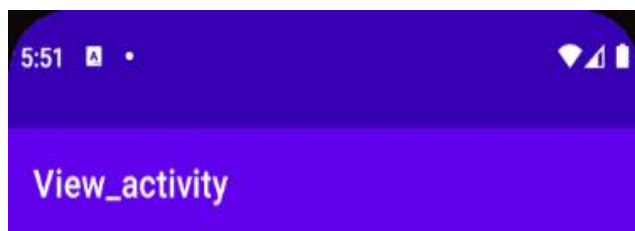


В конце также важно разобрать всплывающие сообщения в приложении. Здесь сильно поможет модуль Toast, в котором нам понадобится метод `makeText()` и `show()`:

```
MainActivity.java × activity_main.xml × strings.xml ×
39 ok_btn.setOnClickListener(oclbtn);
40 cnc_btn.setOnClickListener(oclbtn);
41
42 mainTextView.setText("Set in Java!");
43
44 super.onCreate(savedInstanceState);
45 }
46
47 View.OnClickListener oclbtn = new View.OnClickListener() {
48     @Override
49     public void onClick(View view) {
50         switch (view.getId()) {
51             case R.id.ok_btn:
52                 mainTextView.setText("Нажата кнопка OK");
53                 Toast.makeText(getApplicationContext(),
54                     text: "Нажата кнопка OK", Toast.LENGTH_LONG).show();
55                 break;
56
57             case R.id.cnc_btn:
58                 mainTextView.setText("Нажата кнопка Cancel");
59                 Toast.makeText(getApplicationContext(),
60                     text: "Нажата кнопка Cancel", Toast.LENGTH_LONG).show();
61                 break;
62         }
63     }
64 };
65
66 }
```

Получаемый результат:

(на след. странице)



Нажата кнопка OK

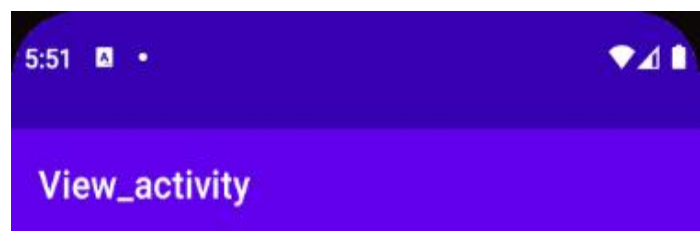
UPDATE THE TEXTVIEW

A name

OK

CANCEL

Нажата кнопка OK



Нажата кнопка Cancel

UPDATE THE TEXTVIEW

A name

OK

CANCEL

Нажата кнопка Cancel