## Layout’ы и View

### Цель работы

Познакомится с основными визуальными элементами приложения и приемами работы с ними

### Задания для выполнения

1. Внимательно прочтите и повторите в своем проекте методические указания.
2. Попробуйте креативно подойти к размещению и оформлению элементов. Экспериментируйте.

### Методические указания

#### Начальное состояние

Давайте познакомимся с содержимым основных файлов нашего проекта.

MainActivity.java сожержит исполняемый код нашего приложения. Сейчас он содержит класс основного активити и код метода, вызываемого при загрузке нашего приложения:

package com.example.helloandroid;

import android.support.v7.app.AppCompatActivity;

import android.os.Bundle;

public class MainActivity extends AppCompatActivity {

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

}

}

activiy\_main.xml содержит описание расположения элементов на нашем основном активити. Сейчас мы видим в нем два элемента - Layout и текстовое представление внутри него:

<?xml version="1.0" encoding="utf-8"?>

<android.support.constraint.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"

xmlns:app="http://schemas.android.com/apk/res-auto"

xmlns:tools="http://schemas.android.com/tools"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

tools:context=".MainActivity">

<TextView

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Hello World!"

app:layout\_constraintBottom\_toBottomOf="parent"

app:layout\_constraintLeft\_toLeftOf="parent"

app:layout\_constraintRight\_toRightOf="parent"

app:layout\_constraintTop\_toTopOf="parent" />

</android.support.constraint.ConstraintLayout>

Элемент TextView - это и есть строка текста. Подробнее про этот элемент читайте в документации: [TextView](https://developer.android.com/reference/android/widget/TextView)

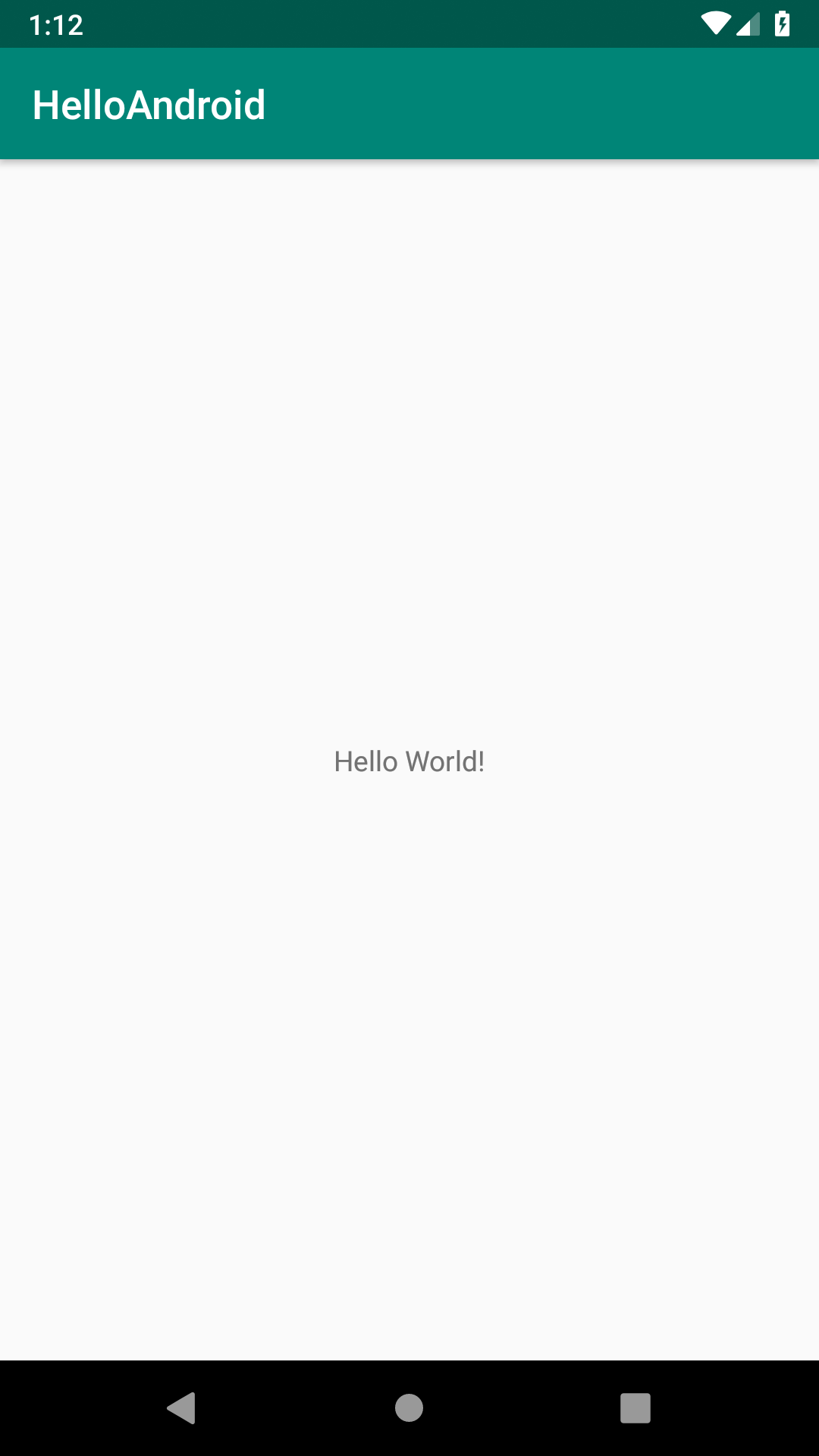
strings.xml содержит текстовые ресурсы - строки, которые использует приложение. Сейчас в нем одна строка - название приложения:

<resources>

<string name="app\_name">HelloAndroid</string>

</resources>

Давайте запустим это приложение в эмуляторе. На экране виртуального устройства мы должны увидеть следующее:



#### Модификация строки

Сейчас строка, отображаемая на экране забита в коде layout’а. Обычно это считается плохим тоном при разработке поддерживаемых приложений. Поэтому давайте выделим эту строку в текстовый ресурс:

<resources>

<string name="app\_name">HelloAndroid</string>

<string name="greetings">Hello from XML</string>

</resources>

В файле activiy\_main.xml заменим саму строку на ссылку на текстовый ресурс:

android:text="@string/greetings"

Заодно добавим новый атрибут, задающий размер шрифта.

android:textSize="24dp"

В общем, наша текстовая строка принимает такой вид:

<TextView

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="@string/greetings"

app:layout\_constraintBottom\_toBottomOf="parent"

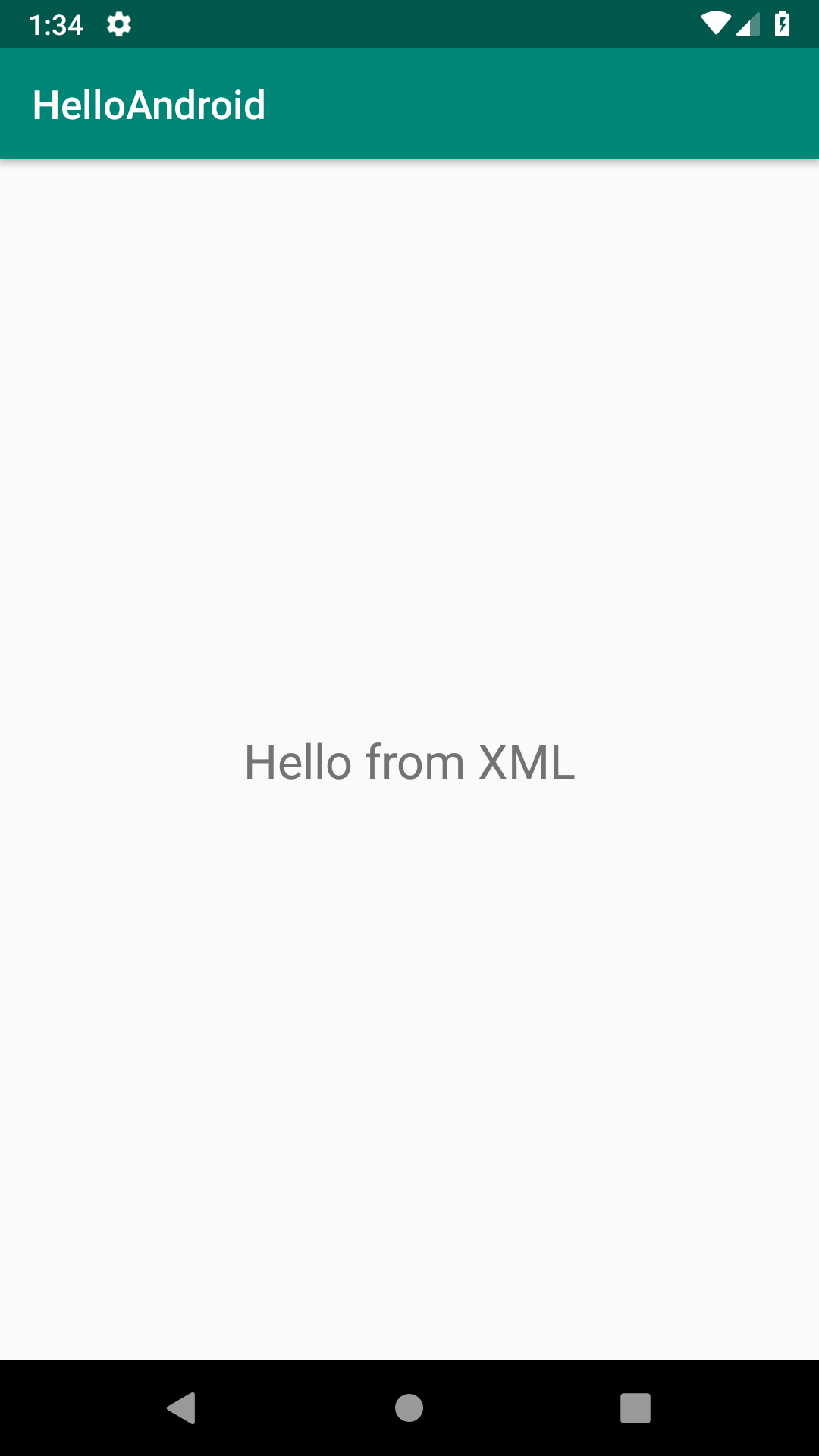
app:layout\_constraintLeft\_toLeftOf="parent"

app:layout\_constraintRight\_toRightOf="parent"

app:layout\_constraintTop\_toTopOf="parent"

android:textSize="24dp"/>

Проверим правильность внесенных правок, запустив приложение на эмуляторе. Вы должны увидеть следующее:



#### Другое расположение

Обратите внимание на первую строку в файле activiy\_main.xml:

<android.support.constraint.ConstraintLayout

Он задает вид расположения элементов. Давайте воспользуемся более простым представлением - LinearLayout. Подробнее про это расположение [документация](https://developer.android.com/reference/android/widget/LinearLayout). Исправим эту строку на следующую:

<LinearLayout

Зададим вертикальное расположение элементов. Таким образом, элементы, входящие в этот layout будут расположены один под другим:

android:orientation="vertical"

Вы можете заметить, что теперь строка расположена вплотную к левому верхнему углу экрана. Чтобы добавить отступ модифицируем элемент TextView, добавив два следующих атрибута:

android:layout\_marginTop="20dp"

android:layout\_marginLeft="20dp"

Проверим расположение элемента в эмуляторе:

### 

#### Доступ к ресурсам из кода

Для того, чтобы приложение было динамическим, нужно иметь возможность изменять визуальные элементы и их свойства из программного кода. Для этого хотя бы нужно уметь обращаться к ним из файла с кодом активити. Для этого у каждого элемента должен быть уникальный идентификатор, по которому его можно найти. Добавим атрибут id к нашему текстовому полю:

android:id="@+id/main\_textview"

Теперь перейдем в файл MainActivity.java. Внутри основного класса, но вне методов введем новый член класса - переменную, которая будет содержать ссылку на наше текстовое поле:

TextView mainTextView;

Скорее всего на этом моменте студия предупредит, что присутствует неизвестный идентификатор TextView. Этот класс объявлен в одном из пакетов Android SDK. Студия предложит автоматические его импортировать:

import android.widget.TextView;

Дальнейшие действия будем производить внутри метода onCreate().

Теперь найдем нужный элемент по его идентификатору:

mainTextView = findViewById(R.id.*main\_textview*);

Теперь мы можем изменить его свойства из программного кода:

mainTextView.setText("Set in Java!");

Так как метод onCreate() выполняется сразу при запуске приложения, запустив его в эмуляторе мы должны сразу увидеть модифицированную строку.



Выполнение задания:



#### Простая кнопка

Добавим кнопку к нашему приложению. Для того, чтобы на кнопке отображалась надпись, заранее создадим для нее текстовый ресурс:

<string name="button">Update The TextView</string>

Теперь создадим непосредственно кнопку. Этот элемент называется button. Подробнее про работу с этим элементов читайте [документацию](https://developer.android.com/reference/android/widget/Button). Расположим ее внутри нашего линейного layout’а после текстового поля. Так она расположится непосредственно под ним.

<Button

android:id="@+id/main\_button"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_marginStart="20dp"

android:layout\_marginLeft="20dp"

android:layout\_marginTop="20dp"

android:text="@string/button" />

Теперь добавим обработчик события нажатия на кнопку. Для этого, нам нужно указать, что наш класс реализует интерфейс обработчика события тапа. Для этого изменим строку объявления класса:

public class MainActivity extends AppCompatActivity {

Добавив объявление реализации интерфейса:

public class MainActivity extends AppCompatActivity implements View.OnClickListener {

Теперь создадим переменную-ссылку на кнопку:

Button mainButton;

Найдем кнопку по идентификатору:

mainButton = findViewById(R.id.*main\_button*);

Теперь зададим ей обработчик события, указав, какой класс будет его обрабатывать (в данном случае, текущий):

mainButton.setOnClickListener(this);

Создадим метод-обработчик события и в нем изменим значение текстового поля:

@Override

public void onClick(View v) {

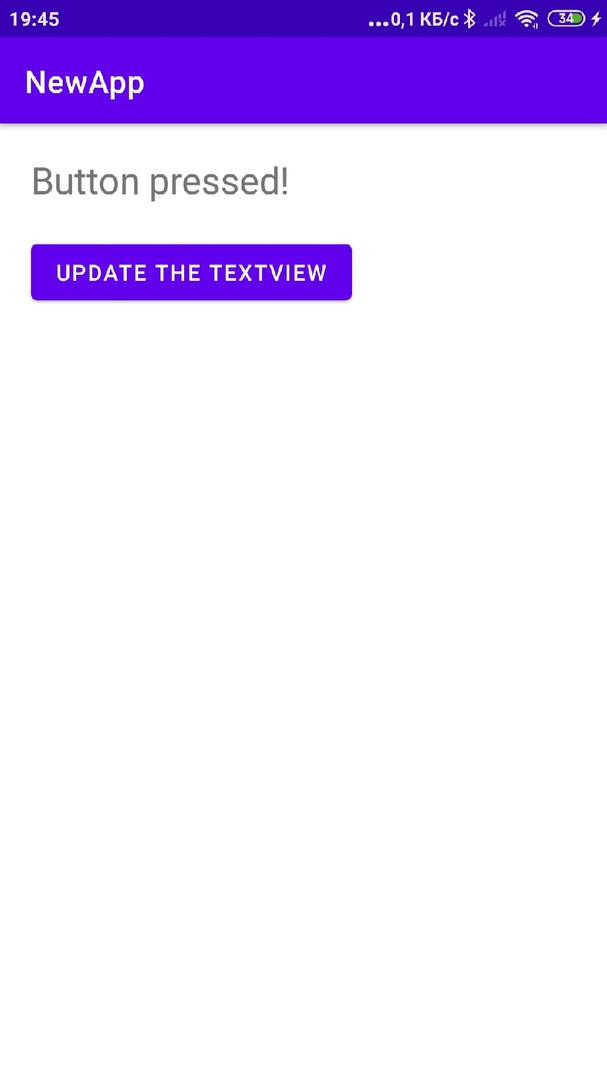
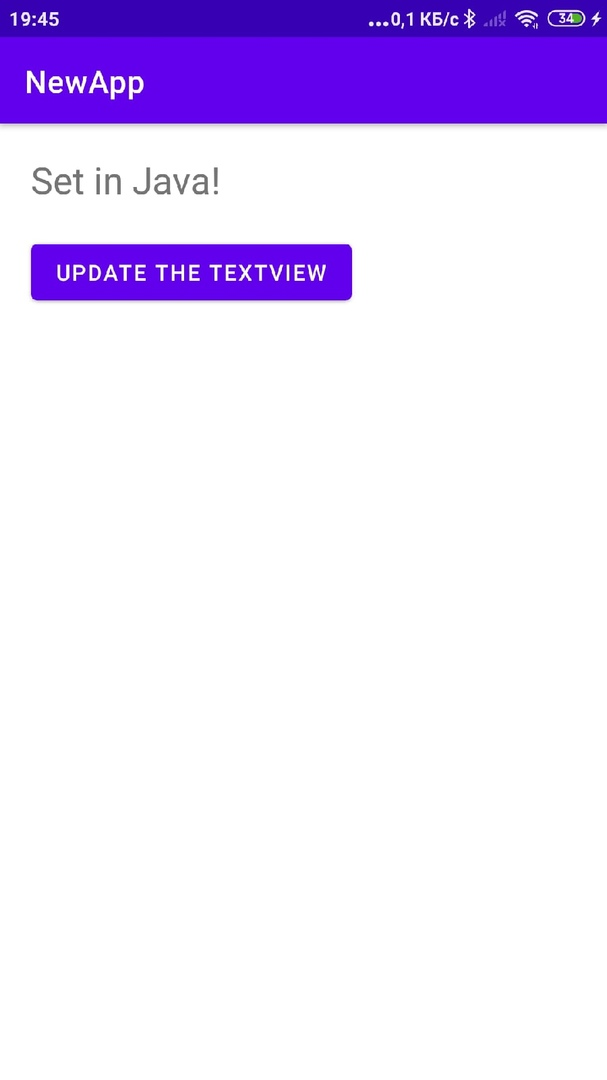
mainTextView.setText("Button pressed!");

}

Теперь при нажатии на кнопку надпись в текстовом поле должна измениться:

|  |  |
| --- | --- |
|  |  |

#### Выполненение задания:



#### Текстовый ввод

Теперь мы будем добавлять поле текстового ввода. Для этого добавим еще один вложенный линейный layout после текстового поля и поместим нашу кнопку внутрь него:

<LinearLayout

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:orientation="horizontal">

Теперь добавим элемент, содержащий поле текстового ввода (элемент [EditText](https://developer.android.com/reference/android/widget/EditText)):

<EditText

android:id="@+id/main\_edittext"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_marginLeft="20dp"

android:layout\_marginTop="20dp"

android:hint="@string/hint" />

Так как этот элемент использует текстовый ресурс, создадим и его:

<string name="hint">A name</string>

Теперь попробуем работать с этим текстовым полем из кода Java. Также, как и раньше создадим для него переменную:

EditText mainEditText;

Находим по идентификатору:

mainEditText = (EditText) findViewById(R.id.*main\_edittext*);

И модифицируем

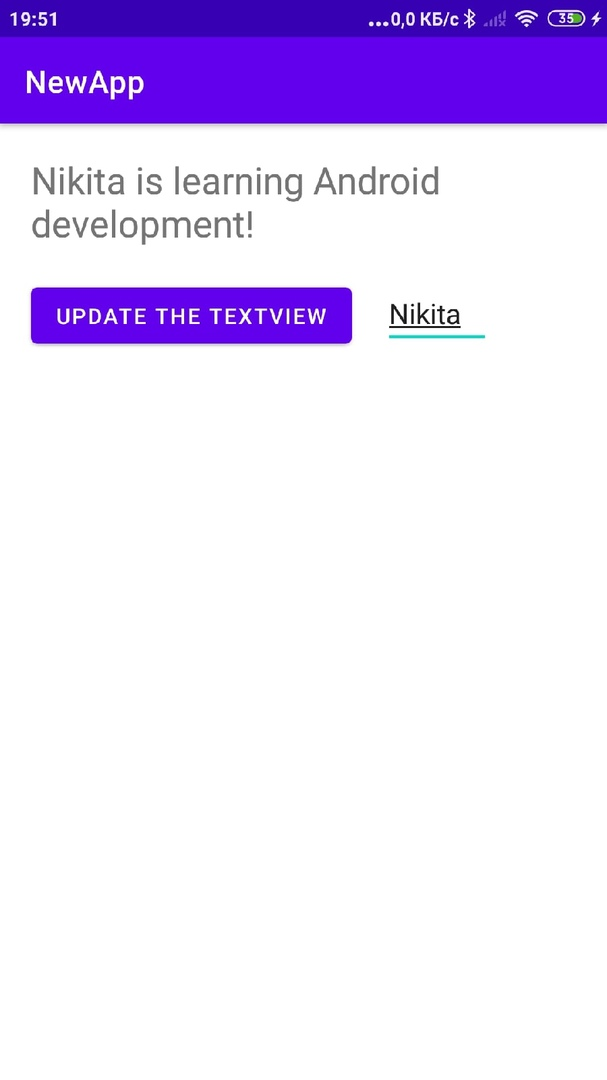
public void onClick(View v) {  mainTextView.setText(mainEditText.getText().toString()

+ " is learning Android development!");

}

|  |  |  |
| --- | --- | --- |
|  |  |  |

#### Выполненение задания:



#### Списки

Теперь изучим еще один визуальный элемент - список ([ListView](https://developer.android.com/reference/android/widget/ListView)). Создадим его под кнопкой с полем:

<ListView

android:id="@+id/main\_listview"

android:layout\_width="match\_parent"

android:layout\_height="0dp"

android:layout\_marginTop="20dp"

android:layout\_weight="1"/>

Теперь добавим необходимые переменные. Для списка нам понадобится целых три: ссылка на элемент, адаптер списка ([подробнее](https://developer.android.com/reference/android/widget/ArrayAdapter)) и динамический массив, содержащий элементы списка:

ListView mainListView;

ArrayAdapter mArrayAdapter;

ArrayList mNameList = new Arrx`ayList();

Добавим код в основной метод:

mainListView = findViewById(R.id.*main\_listview*);

mArrayAdapter = new ArrayAdapter(this,

android.R.layout.*simple\_list\_item\_1*,

mNameList);

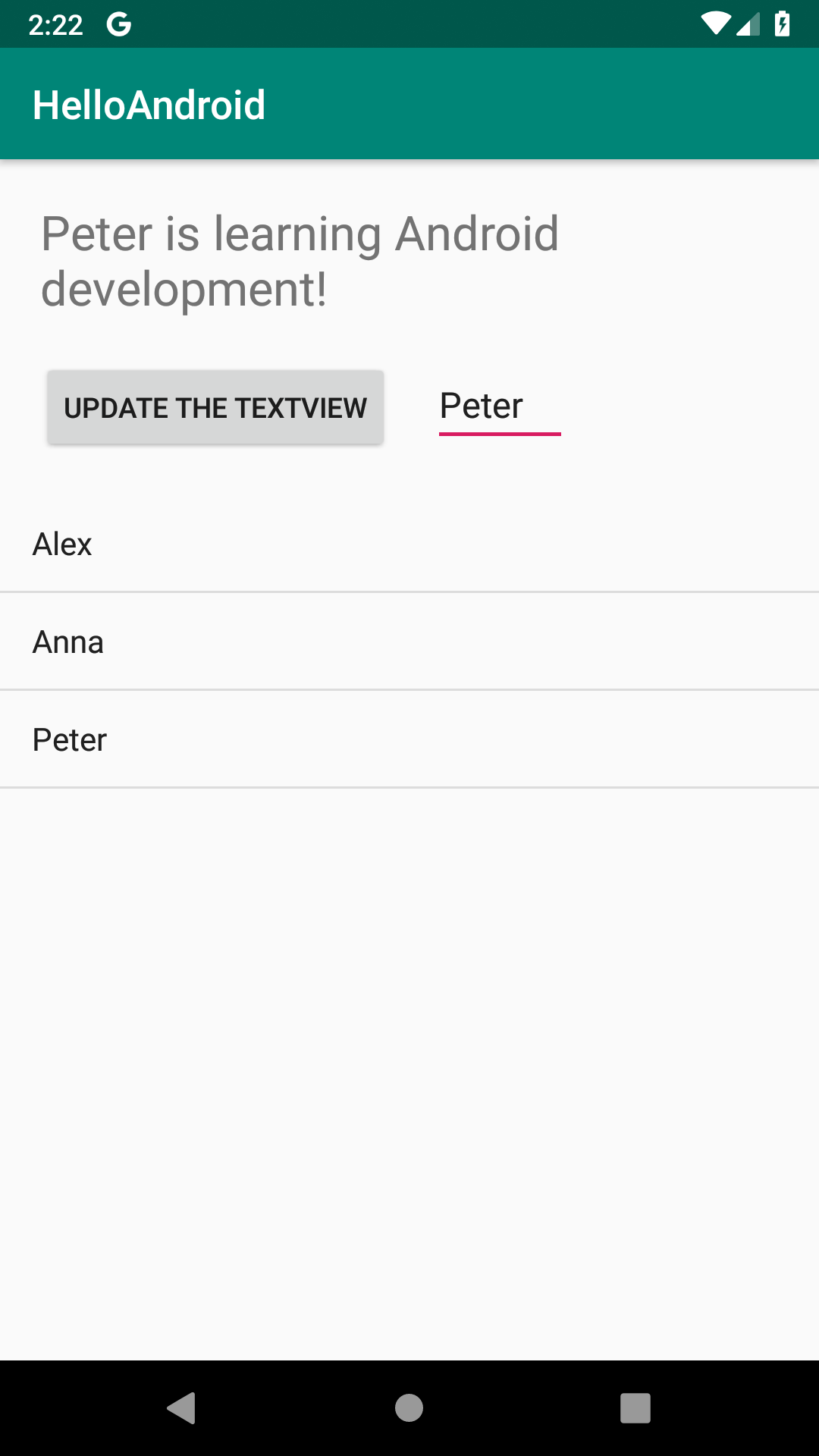
mainListView.setAdapter(mArrayAdapter);

И, наконец, модифицируем код обработчика нажатия на кнопку так, чтобы при нажатии на нее в список добавлялось текущее значение текстового поля:

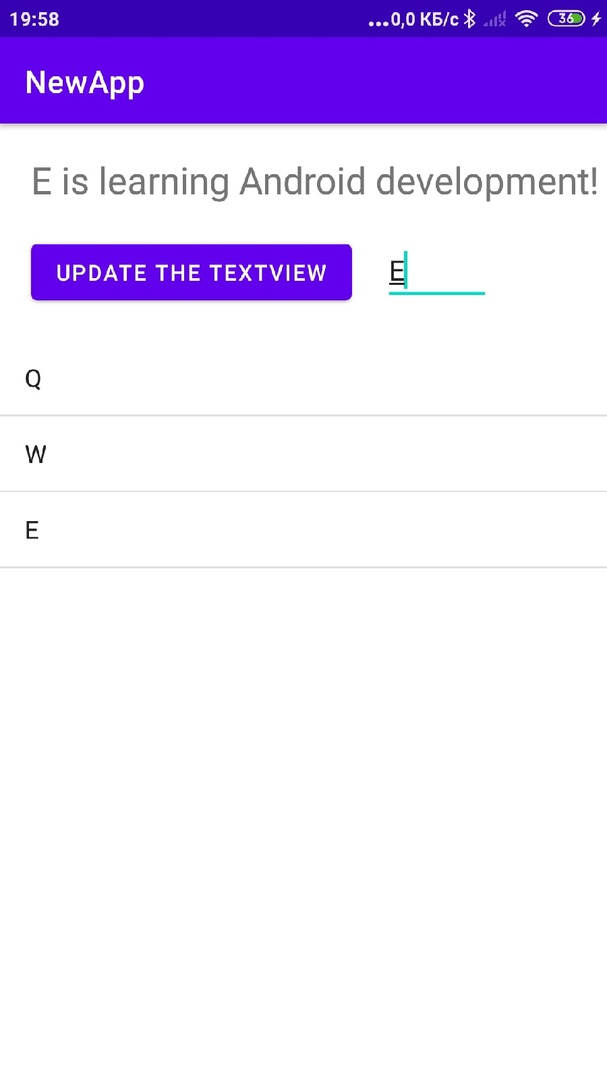
mNameList.add(mainEditText.getText().toString());

mArrayAdapter.notifyDataSetChanged();

Теперь мы можем добавлять элементы к списку:



#### Выполненение задания:



#### Интерактивный список

Иметь простой список в приложении не очень интересно, ведь он не умеет реагировать на нажатия пользователя.Добавим же ему такой возможности. Вся работа сейчас будет производиться в программном коде.

Для начала, укажем, что наш основной класс будет обрабатывать нажатия на список, реализовав соответствующий интерфейс:

public class MainActivity extends AppCompatActivity implements View.OnClickListener, AdapterView.OnItemClickListener {

Зарегистрируем обработчик события:

mainListView.setOnItemClickListener(this);

И добавим новый метод, в котором при тапе на элемент списка он отображался в текстовом поле:

@Override

public void onItemClick(AdapterView<?> parent, View view, int position, long id) {

Log.*d*("omg android", position + ": " + mNameList.get(position));

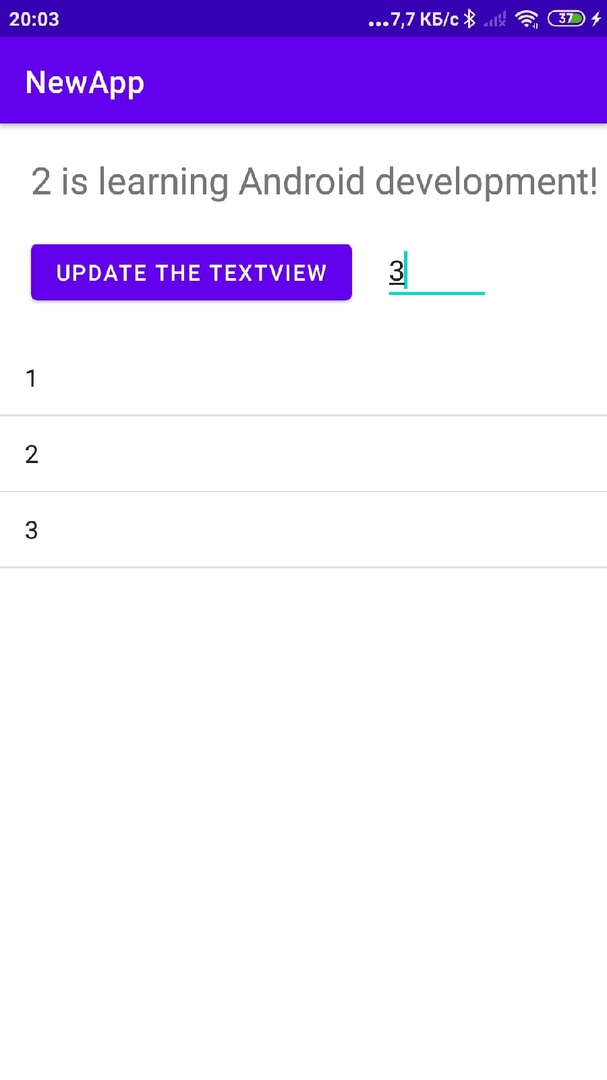
mainTextView.setText(mNameList.get(position).toString()

+ " is learning Android development!");

}

Самостоятельно проверьте работоспособность программы.

#### Выполненение задания:



### Контрольные вопросы

1. Какие основные элементы управления Вы изучили?

Я изучил текстовое поле(TextView), кнопку(Button) и список(ListView)

1. Как происходит группировка элементов управления на странице?

Они располагаются друг под другом и зависят от того, где находится код, который их описывает.

1. Зачем разделять XML и .java файлы?

В .java файлах описываются фукции, переменные, происходит выполнение основных действий, с него начинается работа всей программы. А XML файлы служат для описания элементов, управления их расположения на на странице, по факту здесь описываеся элемент, к которому потом можно обращаться в файле .java.

1. Где в проекте размещаются строковые ресурсы?

Во вкладке res->values->strings.xml

1. Что такое событийно-ориентированное программирование?

Это способ построения компьютерной программы, при котором в коде (как правило, в головной функции программы) явным образом выделяется главный цикл приложения, тело которого состоит из двух частей: выборки события и обработки события.

1. Какие события вы использовали в своем приложении?

Нажатие на кнопку и нажатие на элемент интерактивного списка.

1. Какие события еще Вы знаете?

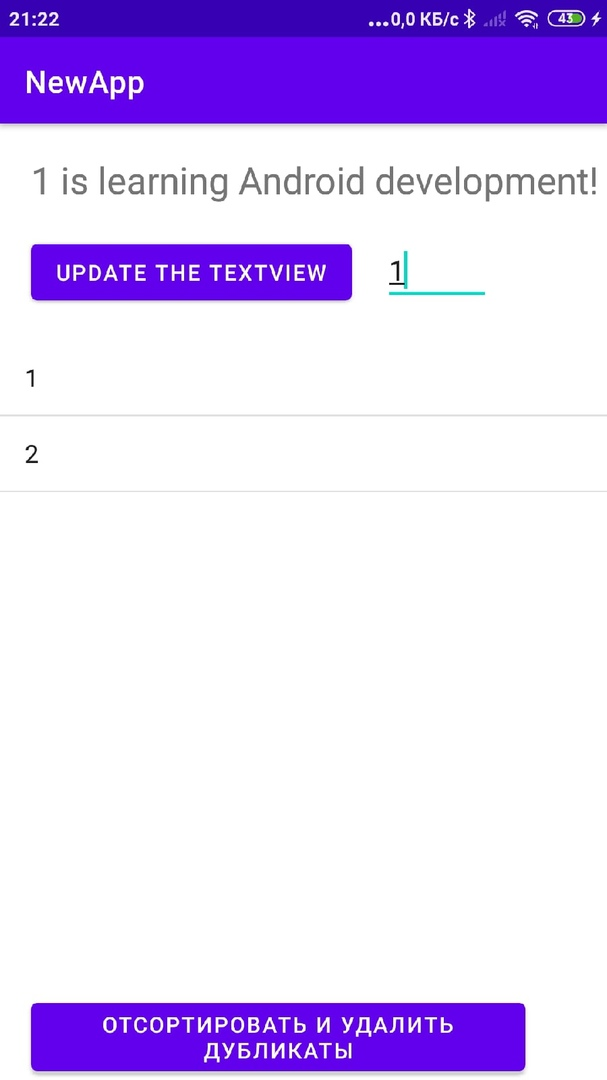
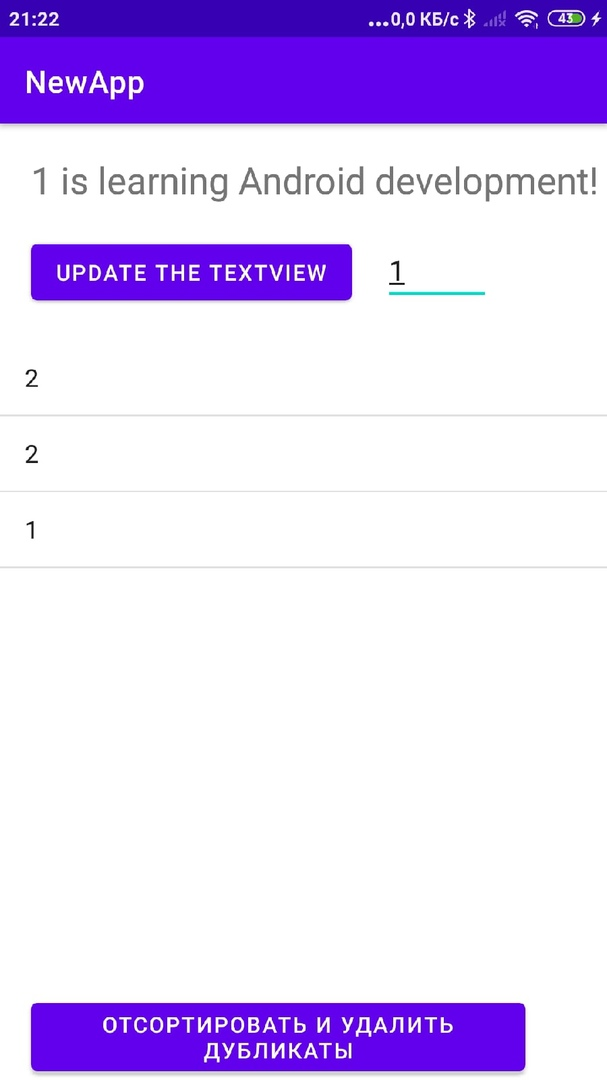
Еще существует множество событий, которые реагируют на движение пользователя по экрану, например нажатие, движение и отпускание, а также ращличные действия со списками, окнами и другими элементами, которые могут быть размещены на экране.

### Дополнительные задания

1. Реализуйте интерактивный список с автоматической сортировкой и удалением дубликатов элементов.

<Button  
 android:id="@+id/main\_button\_1"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginStart="20dp"  
 android:layout\_marginLeft="20dp"  
 android:layout\_marginTop="20dp"  
 android:text="отсортировать и удалить дубликаты" />

mainButton\_1 = findViewById(R.id.*main\_button\_1*);  
mainButton\_1.setOnClickListener(this);  
mainButton\_1.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View view) {  
 Set<String> s = new LinkedHashSet<>(mNameList);  
 mNameList.clear();  
 mNameList.addAll(s);  
 Collections.*sort*(mNameList);  
 mArrayAdapter.notifyDataSetChanged();  
 }  
});



1. Реализуйте возможность удаления выделенного элемента списка.

@Override  
 public void onItemClick(AdapterView<?> adapterView, View view, int i, long l) {  
 Log.*d*("omg android", i + ": " + mNameList.get(i));  
 mainTextView.setText(mNameList.get(i).toString()  
 + " is learning Android development!");  
 mNameList.remove(i);  
 mArrayAdapter.notifyDataSetChanged();  
 }  
}

