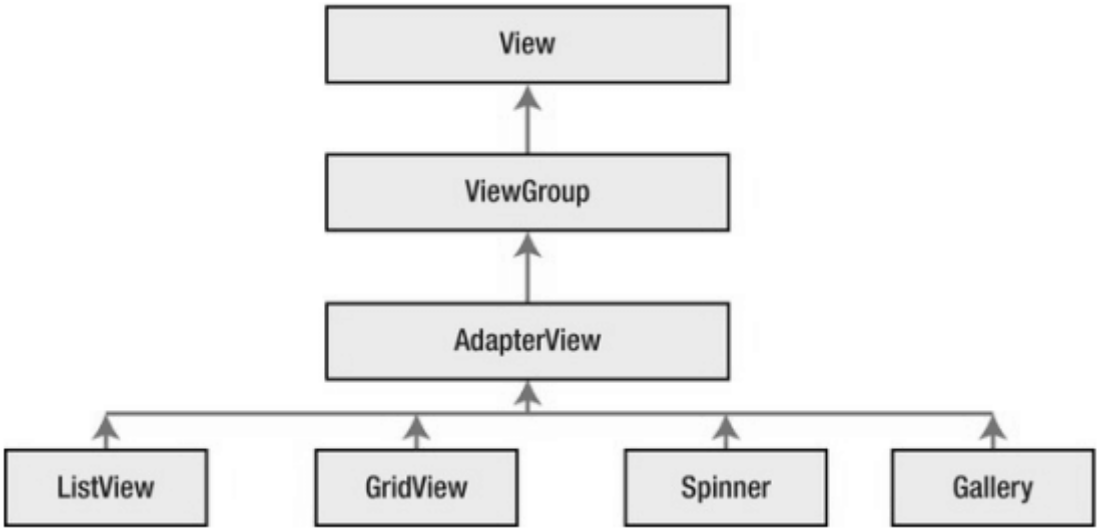
Практическая работа 9

Методические материалы

Адаптеры и списки

ListView и ArrayAdapter

Android представляет широкую палитру элементов, которые представляют списки. Все они является наследниками класса android.widget.AdapterView. Это такие виджеты как ListView, GridView, Spinner. Они могут выступать контейнерами для других элементов управления



Адаптеры в Android

При работе со списками мы имеем дело с тремя компонентами. Во-первых, это визуальный элемент или виджет, который на экране представляет список (ListView, GridView) и который отображает данные. Во-вторых, это источник данных - массив, объект ArrayList, база данных и т.д., в котором находятся сами отображаемые данные. И в-третьих, это адаптер - специальный компонент, который связывает источник данных с виджетом списка.

Одним из самых простых и распространенных элементов списка является виджет **ListView.** Рассмотрим связь элемента **ListView** с источником данных с помощью одного из таких адаптеров - класса **ArrayAdapter.**

Класс ArrayAdapter представляет собой простейший адаптер, который связывает массив данных с набором элементов TextView, из которых, к примеру, может состоять ListView. То есть в данном случае источником данных выступает массив объектов. ArrayAdapter вызывает у каждого объекта метод toString() для приведения к строковому виду и полученную строку устанавливает в элемент TextView.

Посмотрим на примере. Итак, разметка приложения может выглядеть так:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <ListView
        android:id="@+id/countriesList"
        android:layout_width="0dp"
        android:layout_height="0dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent">
    </ListView>

</androidx.constraintlayout.widget.ConstraintLayout>
```

Здесь также определен элемент ListView, который будет выводить список объектов. Теперь перейдем к коду activity и свяжем ListView через ArrayAdapter с некоторыми данными:

```
package com.example.listapp;

import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;
import android.widget.ArrayAdapter;
import android.widget.ListView;

public class MainActivity extends AppCompatActivity {

    // набор данных, которые свяжем со списком
    String[] countries = { "Бразилия", "Аргентина", "Колумбия", "Чили",
"Уругвай"};

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

        // получаем элемент ListView
        ListView countriesList = findViewById(R.id.countriesList);

        // создаем адаптер
        ArrayAdapter<String> adapter = new ArrayAdapter(this,
                android.R.layout.simple_list_item_1, countries);

        // устанавливаем для списка адаптер
        countriesList.setAdapter(adapter);
    }
```

}

Здесь вначале получаем по id элемент ListView и затем создаем для него адаптер.

Для создания адаптера использовался следующий конструктор ArrayAdapter<String>(this,android.R.layout.simple_list_item_1, countries), где

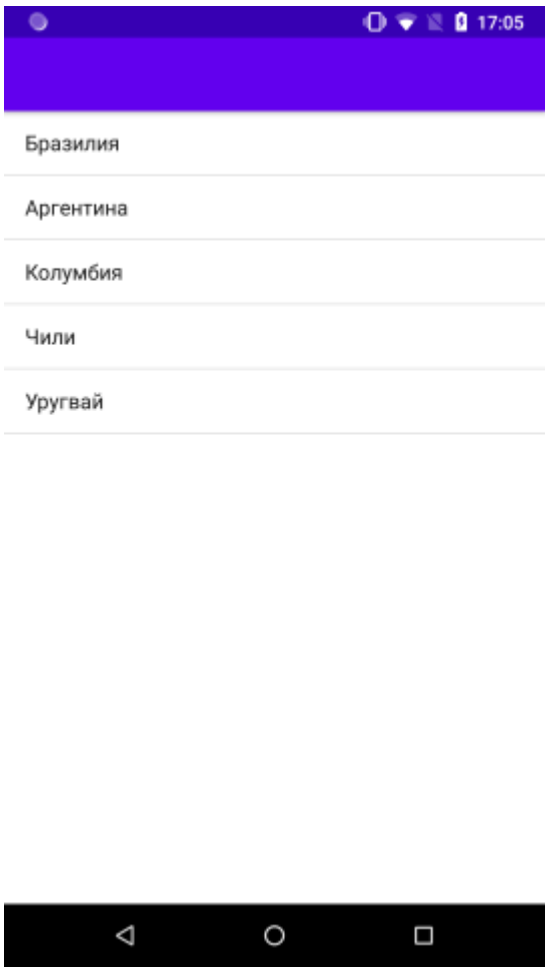
- **this** : текущий объект activity

- **android.R.layout.simple_list_item_1** : файл разметки списка, который фреймворк представляет по умолчанию. Он находится в папке Android SDK по пути platforms/[android-номер_версии]/data/res/layout. Если нас не удовлетворяет стандартная разметка списка, мы можем создать свою и потом в коде изменить id на id нужной нам разметки

- **countries** : массив данных. Здесь необязательно указывать именно массив, это может быть список ArrayList<T>.

В конце неоходимо установить для ListView адаптер с помощью метода setAdapter().

В итоге мы получим следующее отображение:



Адаптер ArrayAdapter в Android и Java

Ресурс string-array и ListView

В прошлом материале было рассмотрено, как выводить массив строк с помощью ArrayAdapter в ListView. При этом массив строк определялся программно в коде java. Однако подобный массив строк гораздо удобнее было бы хранить в файле xml в виде ресурса.

Ресурсы массивов строк представляют элемент типа string-array. Они могут находится в каталоге res/values в xml-файле с произвольным именем.

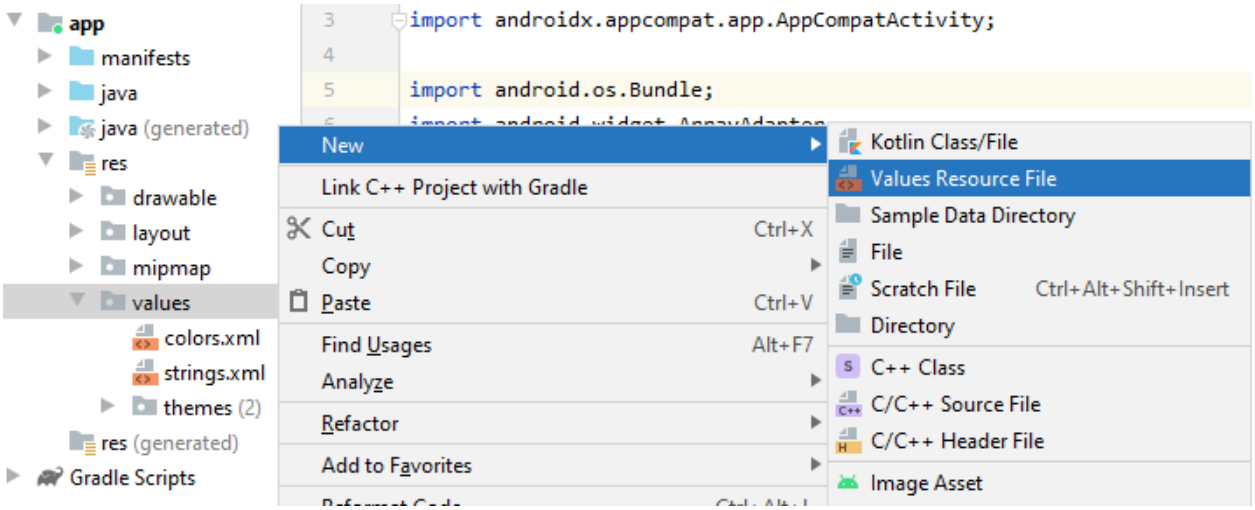
Определения массивов строк имеют следующий синтаксис:

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

```
<resources>
    <string-array name="имя_массива_строк">
        <item>элемент</item>
    </string-array>
</resources>
```

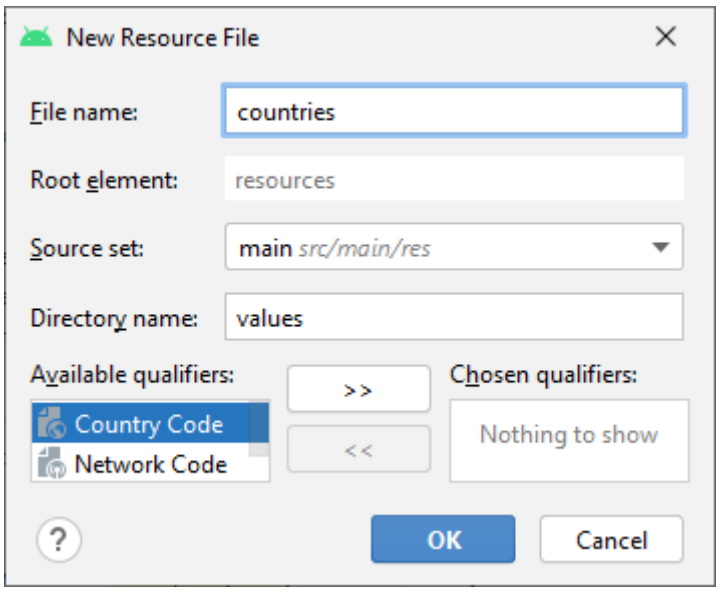
Массив строк задается с помощью элемента <string-array>, атрибут name которого может иметь произвольное значение, по которому затем будут ссылаться на этот массив. Все элементы массива представляют набор значений <item>

Например, добавим в папку res/values новый файл. Для этого нажмем правой кнопкой мыши на данный каталог и появившемся меню выберем пункт **New -> Value Resource file:**



Ресурс String-Array в Android Studio

В появившемся окне назовем файл как countries:



String-Array в Android Studio

После добавления файла в папку res/values изменим его содержимое следующим образом:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string-array name="countries">
        <item>Бразилия</item>
        <item>Аргентина</item>
        <item>Колумбия</item>
        <item>Чили</item>
        <item>Уругвай</item>
    </string-array>
</resources>
```

В файле разметки activity_main.xml остается определение элемента ListView:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
```

```
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <ListView
        android:id="@+id/countriesList"
        android:layout_width="0dp"
        android:layout_height="0dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent">
    </ListView>

</androidx.constraintlayout.widget.ConstraintLayout>
```

Теперь свяжем ресурс и ListView в коде MainActivity:

```
package com.example.listapp;

import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;
import android.widget.ArrayAdapter;
import android.widget.ListView;

public class MainActivity extends AppCompatActivity {

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

```
        setContentView(R.layout.activity_main);

        // получаем элемент ListView
        ListView countriesList = findViewById(R.id.countriesList);

        // получаем ресурс
        String[] countries = getResources().getStringArray(R.array.countries);

        // создаем адаптер
        ArrayAdapter<String> adapter = new ArrayAdapter(this,
                android.R.layout.simple_list_item_1, countries);

        // устанавливаем для списка адаптер
        countriesList.setAdapter(adapter);
    }
}
```

Для получения ресурса в коде java применяется выражение R.array.название_ресурса.

Но нам необязательно добавлять список строк в ListView программно. У этого элемента есть атрибут entries, который в качестве значения может принимать ресурс string-array:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
```

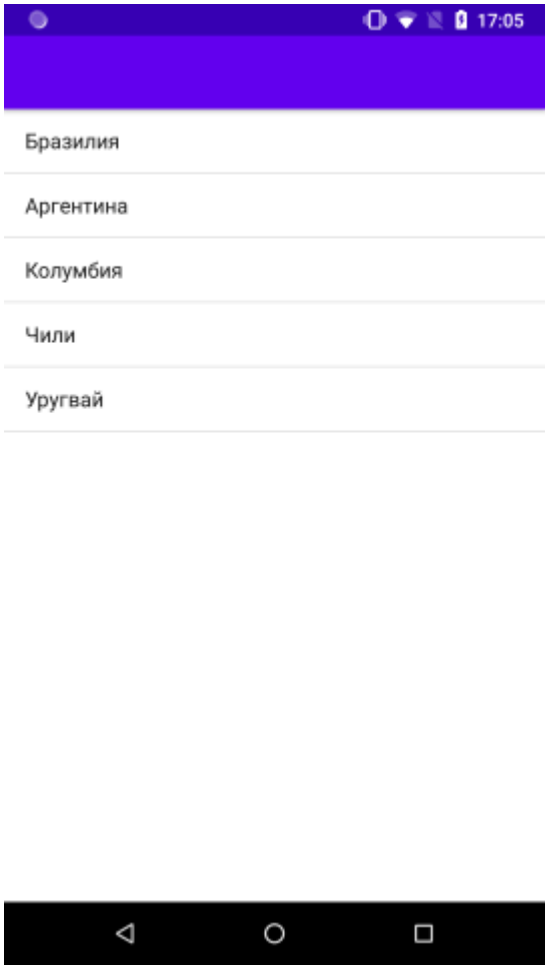
```
    <ListView
        android:id="@+id/countriesList"

        android:entries="@array/countries"

        android:layout_width="0dp"
        android:layout_height="0dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent">
    </ListView>

</androidx.constraintlayout.widget.ConstraintLayout>
```

В этом случае код MainActivity мы можем сократить до стандартного:

```
package com.example.listapp;

import androidx.appcompat.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);
    }
```

}

А результат будет тот же самый:



Адаптер ArrayAdapter в Android и Java

Выбор элемента в ListView

В прошлых материалах было рассмотрено, как можно загружать данные в ListView, связывать его с источником данных. Но кроме простого вывода списка элементов ListView позволяет выбирать элемент и обрабатывать его выбор. Рассмотрим, как это сделать. Определим следующую разметку в файле activity_main.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
```

```
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <TextView
        android:id="@+id/selection"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:textSize="22sp"

        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
    <ListView
        android:id="@+id/countriesList"
        android:layout_width="0dp"
        android:layout_height="0dp"
        app:layout_constraintTop_toBottomOf="@+id/selection"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent">
    </ListView>

</androidx.constraintlayout.widget.ConstraintLayout>
```

Теперь свяжем список ListView с источником данных и закрепим за ним слушатель нажатия на элемент списка:

```
package com.example.listapp;

import androidx.appcompat.app.AppCompatActivity;
```

```
import android.os.Bundle;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.ListView;
import android.widget.TextView;

public class MainActivity extends AppCompatActivity {

    String[] countries = { "Бразилия", "Аргентина", "Колумбия", "Чили",
"Уругвай"};
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        // получаем элемент TextView
        TextView selection = findViewById(R.id.selection);
        // получаем элемент ListView
        ListView countriesList = findViewById(R.id.countriesList);
        // создаем адаптер
        ArrayAdapter<String> adapter = new ArrayAdapter(this,
                android.R.layout.simple_list_item_1, countries);
        // устанавливаем для списка адаптер
        countriesList.setAdapter(adapter);
        // добавляем для списка слушатель
        countriesList.setOnItemClickListener(new
AdapterView.OnItemClickListener(){
```

```
        @Override
        public void onItemClick(AdapterView<?> parent, View v, int position,
long id)
        {
            // по позиции получаем выбранный элемент
            String selectedItem = countries[position];
            // установка текста элемента TextView
            selection.setText(selectedItem);
        }
    });
  }
}
```

Итак, метод setAdapter связывает элемент ListView с определенным адаптером. Далее для обработки выбора элемента списка устанавливается слушатель **OnItemClickListener**. Этот слушатель имеет один метод **onItemClick,** через параметры которого мы можем получить выделенный элемент и сопутствующие данные. Так, он принимает следующие параметры:

- **parent** : нажатый элемент AdapterView (в роли которого в данном случае выступает наш элемент ListView)
- **view :** нажатый виджет внутри AdapterView
- **position** : индекс нажатого виждета внутри AdapterView
- **id** : идентификатор строки нажатого элемента

Используя эти параметры, мы можем разными способами получить выделенный элемент.

Например, в данном случае получая индекс нажатого виджета, который соответствует индексу элемента в массиве строк, мы можем устанавить соответствующий элемент в массиве строк и таким образом получить его текст:

```
countriesList.setOnItemClickListener(new AdapterView.OnItemClickListener(){
    @Override
    public void onItemClick(AdapterView<?> parent, View v, int position, long id)
    {
        // по позиции получаем выбранный элемент
        String selectedItem = countries[position];
        // установка текста элемента TextView
        selection.setText(selectedItem);
    }
});
```

элемент ListView в Android

Также мы можем получить выделенный элемент из AdapterView, который передается в качестве первого параметра - AdapterView<?> parent. Так, в данном случае мы знаем, что каждый элемент в AdapterView фактически представляет строку или объект String, поэтому в данном случае можно получить выделенный элемент так:

```
countriesList.setOnItemClickListener(new AdapterView.OnItemClickListener(){
    @Override
    public void onItemClick(AdapterView<?> parent, View v, int position, long id)
    {
        // получаем выбранный элемент
        String selectedItem = (String)parent.getItemAtPosition(position);
        // установка текста элемента TextView
        selection.setText(selectedItem);
    }
});
```

Метод getItemAtPosition возвращает выделенный элемент по индексу. Это может актуально, если мы используем в качестве источника данных не массив строк, созданный в коде Java, а, например, ресурс <string-array>, заданный в файле xml.

В-третьих, мы можем использовать выделенный элемент, который передается в качестве второго параметра - View v. Так, в данном случае адаптер использует в качестве типа разметки ресурс - **android.R.layout.simple_list_item_1**, а это значит, что выделенный элемент представляет элемент **TextView**, в котором выводится данный текст. Поэтому в данном случае мы также могли бы получить выделенный элемент так:

```
countriesList.setOnItemClickListener(new AdapterView.OnItemClickListener(){
    @Override
    public void onItemClick(AdapterView<?> parent, View v, int position, long id)
    {
```

```
        // получаем выбранный элемент
        TextView textView = (TextView) v;
        String selectedItem = (String)textView.getText();
        // установка текста элемента TextView
        selection.setText(selectedItem);
        // или так
        // selection.setText(textView.getText());
    }
});
```

Множественный выбор в списке

Иногда требуется выбрать не один элемент, как по умолчанию, а несколько. Для этого, во-первых, в разметке списка надо установить атрибут android:choiceMode="multipleChoice":

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <TextView
        android:id="@+id/selection"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:textSize="22sp"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
```

```
        app:layout_constraintTop_toTopOf="parent" />
    <ListView
        android:id="@+id/countriesList"

        android:choiceMode="multipleChoice"

        android:layout_width="0dp"
        android:layout_height="0dp"
        app:layout_constraintTop_toBottomOf="@+id/selection"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent">
    </ListView>

</androidx.constraintlayout.widget.ConstraintLayout>
```

Теперь определим в коде MainActivity обработку выбора элементов списка:

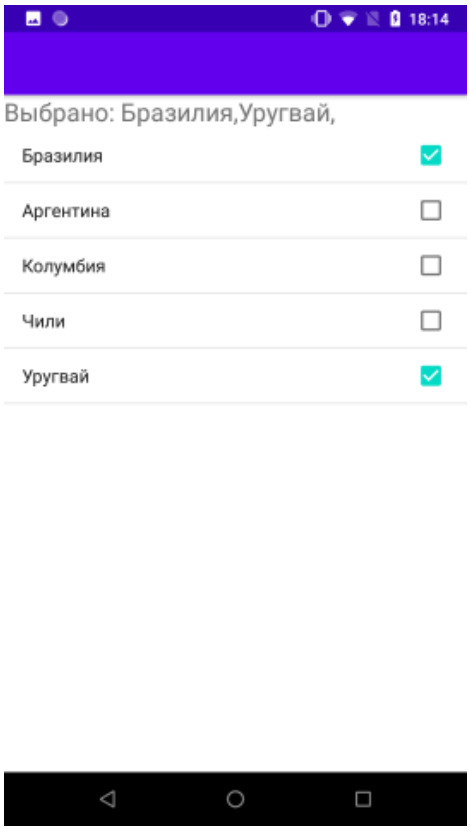
```
package com.example.listapp;

import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;
import android.util.SparseBooleanArray;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.ListView;
import android.widget.TextView;
```

```
public class MainActivity extends AppCompatActivity {


    String[] countries = { "Бразилия", "Аргентина", "Колумбия", "Чили",
"Уругвай"};
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);


        // получаем элемент TextView
        TextView selection = findViewById(R.id.selection);
        // получаем элемент ListView
        ListView countriesList = findViewById(R.id.countriesList);
        // создаем адаптер
        // создаем адаптер
        ArrayAdapter<String> adapter = new ArrayAdapter(this,
                android.R.layout.simple_list_item_multiple_choice, countries);
        // устанавливаем для списка адаптер
        countriesList.setAdapter(adapter);
        // добавляем для списка слушатель
        countriesList.setOnItemClickListener(new
AdapterView.OnItemClickListener(){
            @Override
            public void onItemClick(AdapterView<?> parent, View v, int position,
long id)
            {
                SparseBooleanArray selected=countriesList.getCheckedItemPositions();


                String selectedItems="";
                for(int i=0;i < countries.length;i++)
```

```
            {
                if(selected.get(i))
                    selectedItems+=countries[i]+",";
            }
            // установка текста элемента TextView
            selection.setText("Выбрано: " + selectedItems);
        }
    });
  }
}
```

Ресурс android.R.layout.simple_list_item_multiple_choice представляет стандартную разметку, предоставляемую фреймворком, для создания списка с множественным выбором.

А при выборе элементов мы получаем все выбранные позиции в объект **SparseBooleanArray**, затем пробегаемся по всему массиву, и если позиция элемента в массиве есть в SparseBooleanArray, то есть она отмечена, то добавляем отмеченный элемент в строку.



Множественный выбор в ListView в Android и Java

Добавление и удаление в ArrayAdapter и ListView

После привязки ListView к источнику данных через адаптер мы можем работать с данными - добавлять, удалять, изменять только через адаптер. ListView служит только для отображения данных.

Для управления данными мы можем использовать методы адаптера или напрямую источника данных. Например, класс ArrayAdapter предоставляет следующие методы для управления данными:

- **void add(T object)**: добавляет элемент object в конец массива
- **void addAll(T... items)**: добавляет все элементы items в конец массива
- **void addAll(Collection<? extends T>** collection): добавляет коллекцию элементов collection в конец массива
- **void clear()**: удаляет все элементы из списка
- **void insert(T object, int index):** добавляет элемент object в массив по индексу index
- **void remove(T object):** удаляет элемент object из массива

Однако после применения вышеуказанных методов изменения коснутся только массива, выступающего источником данных. Чтобы синхронизировать изменения с элементом ListView, надо вызвать у адаптера метод notifyDataSetChanged().

Например, определим в файле **activity_main.xml** следующие элементы:

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

```
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
        <EditText
            android:id="@+id/userName"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            app:layout_constraintHorizontal_weight="4"
            app:layout_constraintLeft_toLeftOf="parent"
            app:layout_constraintRight_toLeftOf="@+id/add"
            app:layout_constraintTop_toTopOf="parent" />
        <Button
            android:id="@+id/add"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            app:layout_constraintHorizontal_weight="1"
            android:text="+"
            android:onClick="add"
            app:layout_constraintRight_toLeftOf="@+id/remove"
            app:layout_constraintLeft_toRightOf="@+id/userName"
            app:layout_constraintTop_toTopOf="parent"/>
        <Button
            android:id="@+id/remove"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            app:layout_constraintHorizontal_weight="1"
            android:text="-"
```

```
        android:onClick="remove"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintLeft_toRightOf="@+id/add"
        app:layout_constraintRight_toRightOf="parent" />
    <ListView
        android:id="@+id/usersList"
        android:layout_width="0dp"
        android:layout_height="0dp"
        android:choiceMode="multipleChoice"
        app:layout_constraintTop_toBottomOf="@+id/userName"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent">
    </ListView>

</androidx.constraintlayout.widget.ConstraintLayout>
```

Для вывода списка предназначен ListView с возможностью множественного выбора элементов. Для добавления и удаления определены две кнопки. Для ввода нового объекта в список предназначено поле EditText.

Теперь изменим класс **MainActivity**:

```
package com.example.listapp;
import androidx.appcompat.app.AppCompatActivity;import android.os.Bundle;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.EditText;
import android.widget.ListView;
import java.util.ArrayList;
```

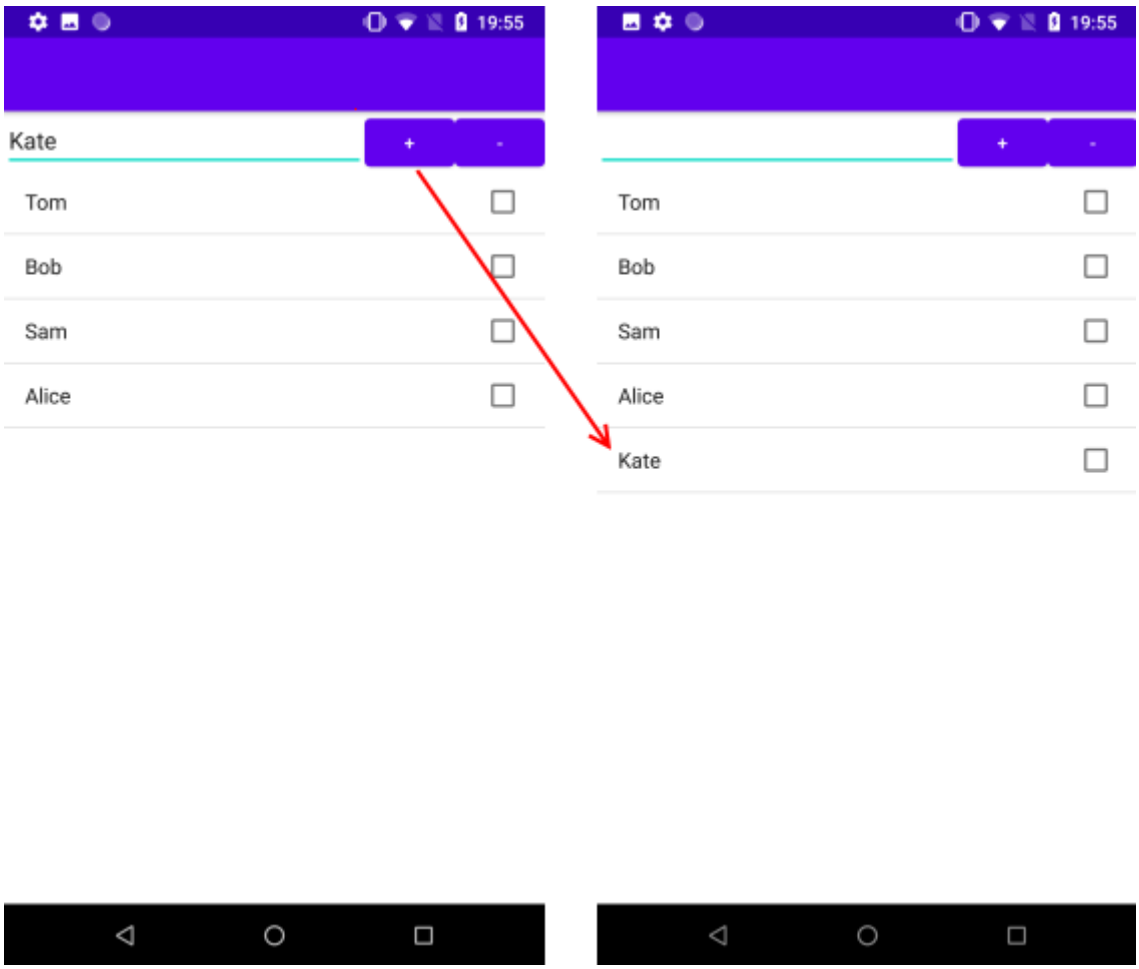
```
import java.util.Collections;
public class MainActivity extends AppCompatActivity {

    ArrayList<String> users = new ArrayList<String>();
    ArrayList<String> selectedUsers = new ArrayList<String>();
    ArrayAdapter<String> adapter;
    ListView usersList;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
         // добавляем начальные элементы
        Collections.addAll(users, "Tom", "Bob", "Sam", "Alice");
        // получаем элемент ListView
        usersList = findViewById(R.id.usersList);
        // создаем адаптер
        adapter = new ArrayAdapter(this,
android.R.layout.simple_list_item_multiple_choice, users);
        // устанавливаем для списка адаптер
        usersList.setAdapter(adapter);


        // обработка установки и снятия отметки в списке
        usersList.setOnItemClickListener(new AdapterView.OnItemClickListener(){
            @Override
            public void onItemClick(AdapterView<?> parent, View v, int position,
long id)
            {
                // получаем нажатый элемент
                String user = adapter.getItem(position);
```

```
            if(usersList.isItemChecked(position))
                selectedUsers.add(user);
            else
                selectedUsers.remove(user);
        }
    });
}
 public void add(View view){
     EditText userName = findViewById(R.id.userName);
    String user = userName.getText().toString();
    if(!user.isEmpty()){
        adapter.add(user);
        userName.setText("");
        adapter.notifyDataSetChanged();
    }
}
public void remove(View view){
    // получаем и удаляем выделенные элементы
    for(int i=0; i< selectedUsers.size();i++){
        adapter.remove(selectedUsers.get(i));
    }
    // снимаем все ранее установленные отметки
    usersList.clearChoices();
    // очищаем массив выбраных объектов
    selectedUsers.clear();

    adapter.notifyDataSetChanged();
}
}
```

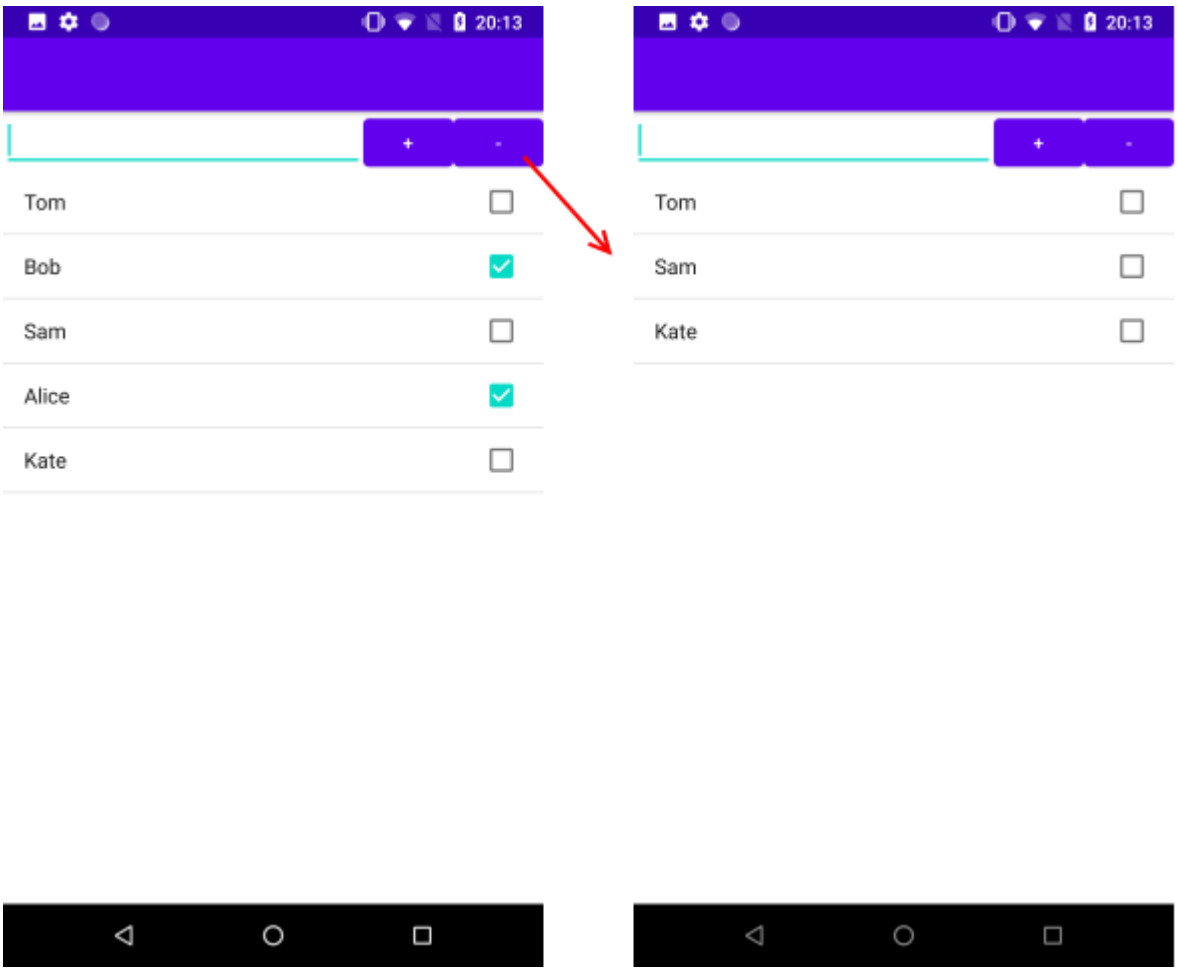
С добавлением все относительно просто: получаем введенную строку и добавляем в список с помощью метода **adapter.add()**. Чтобы обновить ListView после добавления вызывается метод **adapter.notifyDataSetChanged().**

А для удаления создается дополнительный список selectedUsers, который будет содержать выделенные элементы. Для получения выделенных элементов и добавления их в список используется слушатель **AdapterView.OnItemClickListener**, метод onItemClick() которого вызывается при установке или снятия отметки с элемента, то есть при любом нажатии на элемент.

По нажатию на кнопку удаления пробегаемся по списку выделенных элементов и вызываем для каждого из них метод **adapter.remove().**



Добавление в ListView и ArrayAdapter в Android и Java



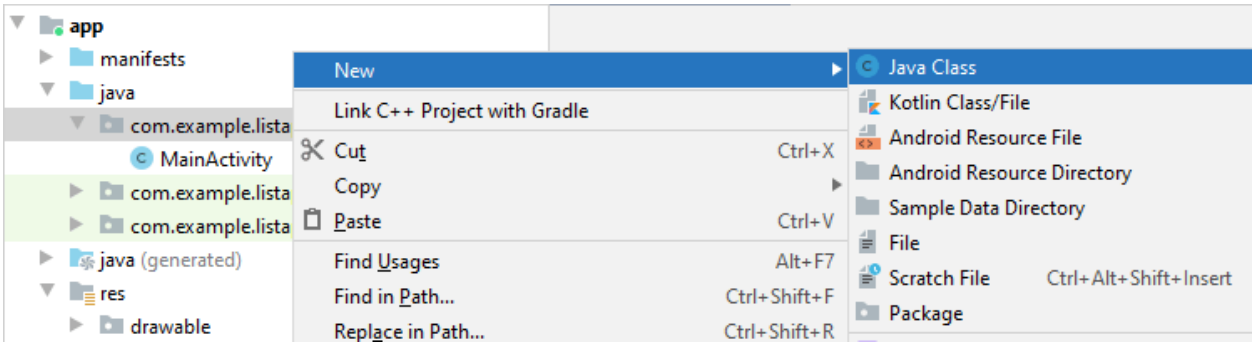
Удаление в ListView и ArrayAdapter в Android и Java

Расширение списков и создание адаптера

Традиционные списки ListView, использующие стандартные адаптеры ArrayAdapter, прекрасно работают с массивами строк. Однако чаще мы будем сталкиваться с более сложными по структуре списками, где один элемент представляет не одну строку, а несколько строк, картинок и других компонентов.

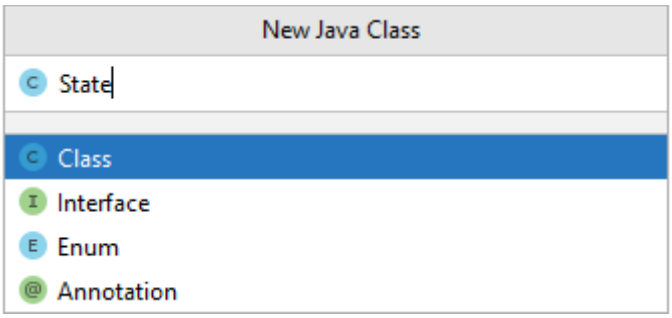
Для создания сложного списка нам надо переопределить один из используемых адаптеров. Поскольку, как правило, используется ArrayAdapter, то именно его мы и переопределим.

Но вначале определим модель, данные которой будут отображаться в списке. Для этого добавим в тот же каталог, где находится класс MainActivity, новый класс. Для этого нажмем на данный каталог правой кнопкой мыши и в меню выберем **New -> Java Class**:



Добавление класса java в Android Studio

В появившемся окне укажем для добавляемого класса имя State



Создание класса java в Android Studio

После добавления изменим класс State следующим образом:

```
package com.example.listapp;

public class State {

    private String name; // название
    private String capital;  // столица
    private int flagResource; // ресурс флага

    public State(String name, String capital, int flag){

        this.name=name;
        this.capital=capital;
```

```
        this.flagResource=flag;
    }

    public String getName() {
        return this.name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getCapital() {
        return this.capital;
    }

    public void setCapital(String capital) {
        this.capital = capital;
    }

    public int getFlagResource() {
        return this.flagResource;
    }

    public void setFlagResource(int flagResource) {
        this.flagResource = flagResource;
    }
}
```

Данный класс хранит два строковых поля - название государства и его столицу, а также числовое поле, которое будет указывать на ресурс изображения из папки drawable, которое будет отображать флаг государства.

Далее добавим в папку **res/layout** новый файл **list_item.xml**, который будет представлять разметку одного элемента в списке:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="wrap_content">
    <ImageView
        android:id="@+id/flag"
        android:layout_width="70dp"
        android:layout_height="50dp"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toLeftOf="@+id/name"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintBottom_toBottomOf="parent" />

    <TextView
        android:id="@+id/name"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginLeft="16dp"
        android:text="Название"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintLeft_toRightOf="@+id/flag"
        app:layout_constraintTop_toTopOf="parent"
```

```
        app:layout_constraintBottom_toTopOf="@+id/capital" />

    <TextView
        android:id="@+id/capital"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginLeft="16dp"
        android:text="Столица"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintLeft_toRightOf="@+id/flag"
        app:layout_constraintTop_toBottomOf="@+id/name"
        app:layout_constraintBottom_toBottomOf="parent" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

Каждый элемент будет иметь изображение в виде ImageView и два компонента TextView для отображения названия и столицы государства.

После этого добавим в каталог, где находятся классы MainActivity и State, новый класс, который назовем **StateAdapter**:

```
package com.example.listapp;

import android.content.Context;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ArrayAdapter;
import android.widget.ImageView;
import android.widget.TextView;
```

```
import java.util.List;

public class StateAdapter extends ArrayAdapter<State> {

    private LayoutInflater inflater;
    private int layout;
    private List<State> states;

    public StateAdapter(Context context, int resource, List<State> states) {
        super(context, resource, states);
        this.states = states;
        this.layout = resource;
        this.inflater = LayoutInflater.from(context);
    }
    public View getView(int position, View convertView, ViewGroup parent) {

        View view=inflater.inflate(this.layout, parent, false);

        ImageView flagView = view.findViewById(R.id.flag);
        TextView nameView = view.findViewById(R.id.name);
        TextView capitalView = view.findViewById(R.id.capital);

        State state = states.get(position);

        flagView.setImageResource(state.getFlagResource());
        nameView.setText(state.getName());
        capitalView.setText(state.getCapital());
```

```
        return view;
    }
}
```

Все взаимодействие со списком здесь будет идти через класс StateAdapter. В конструкторе StateAdapter нам надо передать в конструктор базового класса три параметра:

- контекст, в котором используется класс. В его роли кк правило выступает класс Activity
- ресурс разметки интерфейса, который будет использоваться для создания одного элемента в ListView
- набор объектов, которые будут выводиться в ListView

В конструкторе StateAdapter мы получаем ресурс разметки и набор объекто и сохраняем их в отдельные переменные. Кроме того, для создания объекта View по полученному ресурсу разметки потребуется объект LayoutInflater, который также сохраняется в переменную.

В методе getView() устанавливается отображение элемента списка. Данный метод принимает три параметра:

- position: передает позицию элемента внутри адаптера, для которого создается представление
- convertView: старое представление элемента, которое при наличии используется ListView в целях оптимизации
- parent: родительский компонент для представления элемента

В данном случае с помощью объекта LayoutInflater создаем объект View для каждого отдельного элемента в списке:

View view=inflater.inflate(this.layout, parent, false);

Из созданного объекта View получаем элементы ImageView и TextView по id:

ImageView flagView = (ImageView) view.findViewById(R.id.flag);

TextView nameView = (TextView) view.findViewById(R.id.name);

TextView capitalView = (TextView) view.findViewById(R.id.capital);

Это те элементы, которые определены в файле list_item.xml. Здесь же мы их получаем.

Далее используя параметр position, получаем объект State, для которого создается разметка:

State state = states.get(position);

Затем полученные элементы ImageView и TextView наполняем из полученного по позиции объекта State:

flagView.setImageResource(state.getFlagResource());

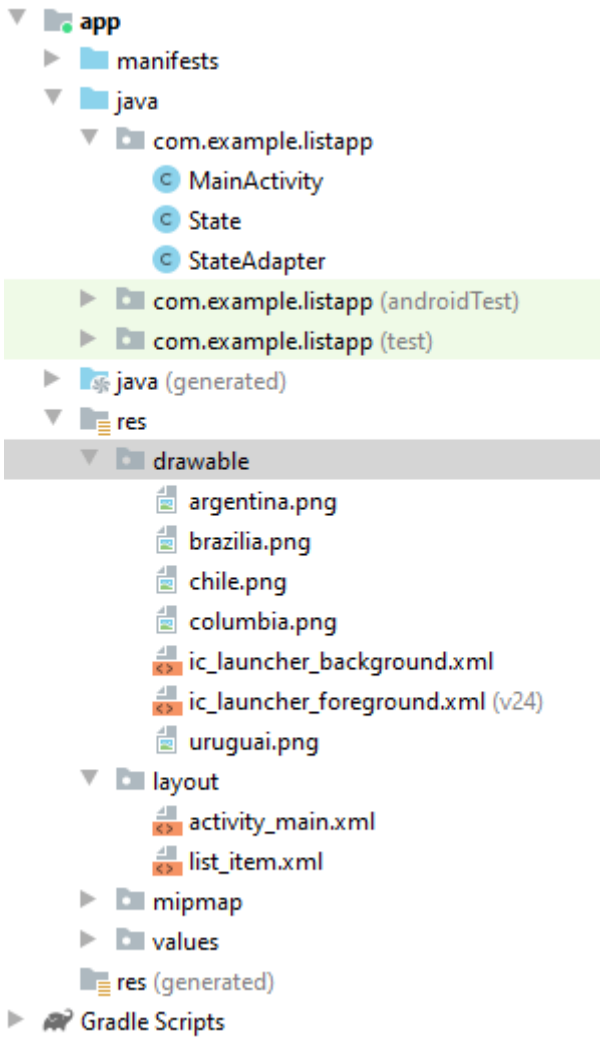
nameView.setText(state.getName());

capitalView.setText(state.getCapital());

И в конце созданный для отображения объекта State элемент View возвращается из метода:

return view;

Для использования изображений добавим в папку res/drawable несколько изображений, в моем случае это пять изображений флагов государств. В итоге проект будет выглядеть следующим образом:



Изображения в ListView в Android и Java

В файле **activity_main.xml** определим ListView, в который будут загружатся данные:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <ListView
        android:id="@+id/countriesList"
        android:layout_width="0dp"
```

```
        android:layout_height="0dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

А в файле MainActivity соединим StateAdapter с ListView:

```
package com.example.listapp;

import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ListView;
import android.widget.Toast;
import java.util.ArrayList;

public class MainActivity extends AppCompatActivity {

    ArrayList<State> states = new ArrayList<State>();
    ListView countriesList;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        // начальная инициализация списка
        setInitialData();
```

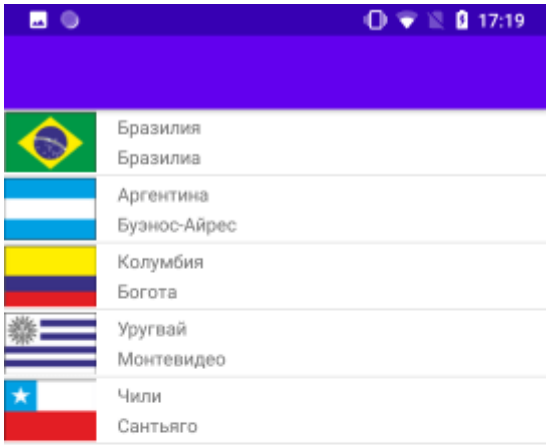
```
        // получаем элемент ListView
        countriesList = findViewById(R.id.countriesList);
        // создаем адаптер
        StateAdapter stateAdapter = new StateAdapter(this, R.layout.list_item,
states);
        // устанавливаем адаптер
        countriesList.setAdapter(stateAdapter);
        // слушатель выбора в списке
        AdapterView.OnItemClickListener itemListener = new
AdapterView.OnItemClickListener() {
            @Override
            public void onItemClick(AdapterView<?> parent, View v, int position,
long id) {

                // получаем выбранный пункт
                State selectedState = (State)parent.getItemAtPosition(position);
                Toast.makeText(getApplicationContext(), "Был выбран пункт " +
selectedState.getName(),
                        Toast.LENGTH_SHORT).show();
            }
        };
        countriesList.setOnItemClickListener(itemListener);
    }
    private void setInitialData(){

        states.add(new State ("Бразилия", "Бразилиа", R.drawable.brazilia));
        states.add(new State ("Аргентина", "Буэнос-Айрес", R.drawable.argentina));
        states.add(new State ("Колумбия", "Богота", R.drawable.columbia));
        states.add(new State ("Уругвай", "Монтевидео", R.drawable.uruguai));
        states.add(new State ("Чили", "Сантьяго", R.drawable.chile));
```

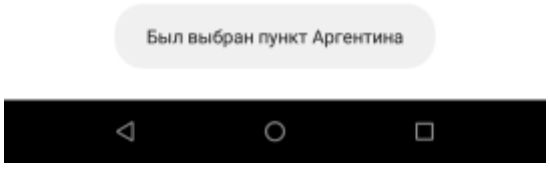
```
    }
}
```

В качестве источника данных здесь выступает класс ArrayList, который получает данные в методе setInitialData. Каждому добавляемому объекту State в списке передается название государства, его столица и ресурс изображения из папки res/drawable, который представляет флаг государства.

При создании адаптера ему передается ранее созданный ресурс разметки list_item.xml и список states:

```
StateAdapter stateAdapter = new StateAdapter(this, R.layout.list_item, states);
```





Переопределение ArrayAdapter в Android и Java

Оптимизация адаптера и View Holder

В прошлом материале был создан кастомный адаптер, который позволял работать со сложными списками объектов:

```
import android.content.Context;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ArrayAdapter;
import android.widget.ImageView;
import android.widget.TextView;

import java.util.List;

public class StateAdapter extends ArrayAdapter<State> {

    private LayoutInflater inflater;
    private int layout;
    private List<State> states;

    public StateAdapter(Context context, int resource, List<State> states) {
        super(context, resource, states);
        this.states = states;
        this.layout = resource;
        this.inflater = LayoutInflater.from(context);
    }
    public View getView(int position, View convertView, ViewGroup parent) {

        View view=inflater.inflate(this.layout, parent, false);
```

```
        ImageView flagView = view.findViewById(R.id.flag);
        TextView nameView = view.findViewById(R.id.name);
        TextView capitalView = view.findViewById(R.id.capital);

        State state = states.get(position);

        flagView.setImageResource(state.getFlagResource());
        nameView.setText(state.getName());
        capitalView.setText(state.getCapital());

        return view;
    }
}
```

Но этот адаптер имеет один очень большой минус - при прокрутке в ListView, если в списке очень много объектов, то для каждого элемента, когда он попадет в зону видимости, будет повторно вызываться метод getView, в котором будет заново создаваться новый объект View. Соответственно будет увеличиваться потребление памяти и снижаться производительность. Поэтому оптимизируем код метода **getView:**

```
public View getView(int position, View convertView, ViewGroup parent) {

    if(convertView==null){
        convertView = inflater.inflate(this.layout, parent, false);
    }

    ImageView flagView = convertView.findViewById(R.id.flag);
    TextView nameView = convertView.findViewById(R.id.name);
    TextView capitalView = convertView.findViewById(R.id.capital);
```

```
    State state = states.get(position);

    flagView.setImageResource(state.getFlagResource());
    nameView.setText(state.getName());
    capitalView.setText(state.getCapital());

    return convertView;
}
```

Параметр convertView указывает на элемент View, который используется для объекта в списке по позиции position. Если ранее уже создавался View для этого объекта, то параметр convertView уже содержит некоторое значение, которое мы можем использовать.

В этом случае мы будем повторно использовать уже созданные объекты и увеличим производительность, однако этот код можно еще больше оптимизировать. Дело в том, что получение элементов по id тоже относительно затратная операция. Поэтому дальше оптимизируем код **StateAdapter**, изменив его следующим образом:

```
import android.content.Context;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ArrayAdapter;
import android.widget.ImageView;
import android.widget.TextView;

import java.util.List;

public class StateAdapter extends ArrayAdapter<State> {
```

```
private LayoutInflater inflater;
private int layout;
private List<State> states;

public StateAdapter(Context context, int resource, List<State> states) {
    super(context, resource, states);
    this.states = states;
    this.layout = resource;
    this.inflater = LayoutInflater.from(context);
}
public View getView(int position, View convertView, ViewGroup parent) {

    ViewHolder viewHolder;
    if(convertView==null){
        convertView = inflater.inflate(this.layout, parent, false);
        viewHolder = new ViewHolder(convertView);
        convertView.setTag(viewHolder);
    }
    else{
        viewHolder = (ViewHolder) convertView.getTag();
    }
    State state = states.get(position);

    viewHolder.imageView.setImageResource(state.getFlagResource());
    viewHolder.nameView.setText(state.getName());
    viewHolder.capitalView.setText(state.getCapital());

    return convertView;
}
```

```
    private class ViewHolder {
        final ImageView imageView;
        final TextView nameView, capitalView;
        ViewHolder(View view){
            imageView = view.findViewById(R.id.flag);
            nameView = view.findViewById(R.id.name);
            capitalView = view.findViewById(R.id.capital);
        }
    }
}
```

Для хранения ссылок на используемые элементы **ImageView** и **TextView** определен внутренний приватный класс **ViewHolder**, который в конструкторе получает объект View, содержащий ImageView и TextView.

В методе getView, если convertView равен null (то есть если ранее для объекта не создана разметка) создаем объект ViewHolder, который сохраняем в тег в convertView:

```
convertView.setTag(viewHolder);
```

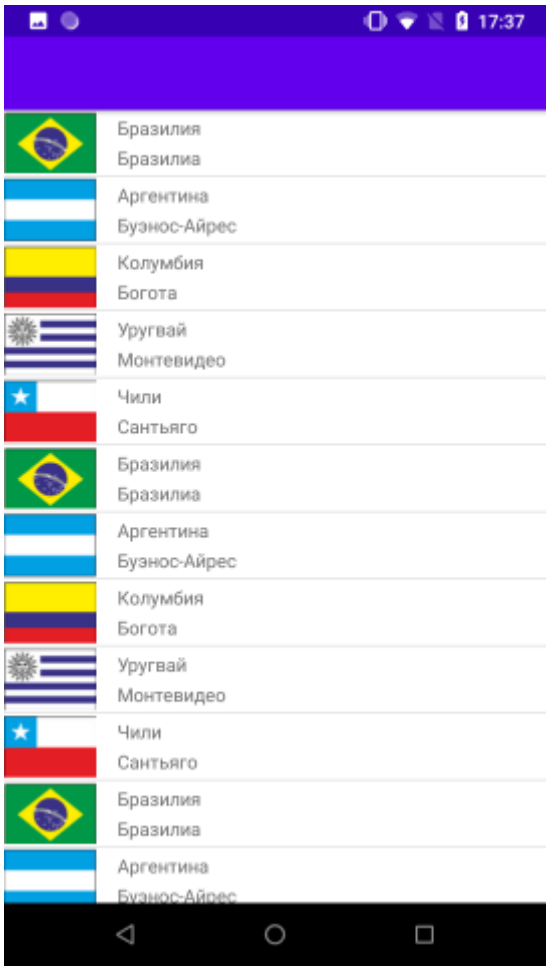
Если же разметка для объекта в ListView уже ранее была создана, то обратно получаем ViewHolder из тега:

```
viewHolder = (ViewHolder) convertView.getTag();
```

Затем также для ImageView и TextView во ViewHolder устанавливаются значения из объекта State:

```
viewHolder.imageView.setImageResource(state.getFlagResource());
viewHolder.nameView.setText(state.getName());
viewHolder.capitalView.setText(state.getCapital());
```

И теперь ListView особенно при больших списках будет работать плавнее и производительнее, чем в предыдущих материалах:



Переопределение ArrayAdapter в Android и Java и ViewHolder

Сложный список с кнопками

Ранее были расмотрены кастомные адаптеры, которые позволяют выводить в списки сложные данные. Теперь пойдем дальше и рассмотрим, как мы можем добавить в списки другие элементы, например, кнопки, и обрабатывать их события.

Для этого вначале определим следующий класс **Product:**

```
package com.example.listapp;
```

```
public class Product {
    private String name;
    private int count;
    private String unit;

    Product(String name, String unit){
        this.name = name;
        this.count=0;
        this.unit = unit;
    }
    public String getUnit() {
        return this.unit;
    }
    public void setCount(int count) {
        this.count = count;
    }

    public int getCount() {
        return count;
    }
    public void setName(String name){
        this.name = name;
    }
    public String getName(){
        return this.name;
    }
}
```

Данный класс хранит название, количество продукта, а также единицу измерения. И объекты этого классы будем выводить в список.

Для этого в папку **res/layout** добавим новый файл **list_item.xml**:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:padding="16dp" >

    <TextView
        android:id="@+id/nameView"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:textSize="18sp"
        app:layout_constraintHorizontal_weight="2"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toLeftOf="@+id/countView"
        app:layout_constraintTop_toTopOf="parent"/>
    <TextView
        android:id="@+id/countView"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:textSize="18sp"
        app:layout_constraintHorizontal_weight="2"
        app:layout_constraintBottom_toBottomOf="parent"
```

```
        app:layout_constraintLeft_toRightOf="@+id/nameView"
        app:layout_constraintRight_toLeftOf="@+id/addButton"
        app:layout_constraintTop_toTopOf="parent" />
    <Button
        android:id="@+id/addButton"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:text="+"
        app:layout_constraintHorizontal_weight="1"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toRightOf="@+id/countView"
        app:layout_constraintRight_toLeftOf="@+id/removeButton"
        app:layout_constraintTop_toTopOf="parent" />
    <Button
        android:id="@+id/removeButton"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:text="-"
        app:layout_constraintHorizontal_weight="1"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toRightOf="@+id/addButton"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent"/>

</androidx.constraintlayout.widget.ConstraintLayout>
```

Здесь определены два текстовых поля для вывода названия и количества продукта и две кнопки для добавления и удаления однйо единицы продукта.

Теперь добавим класс адаптера, который назовем **ProductAdapter**:

```
package com.example.listapp;

import android.content.Context;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.TextView;

import java.util.ArrayList;

class ProductAdapter extends ArrayAdapter<Product> {
    private LayoutInflater inflater;
    private int layout;
    private ArrayList<Product> productList;

    ProductAdapter(Context context, int resource, ArrayList<Product> products) {
        super(context, resource, products);
        this.productList = products;
        this.layout = resource;
        this.inflater = LayoutInflater.from(context);
    }
    public View getView(int position, View convertView, ViewGroup parent) {

        final ViewHolder viewHolder;
        if(convertView==null){
            convertView = inflater.inflate(this.layout, parent, false);
```

```
            viewHolder = new ViewHolder(convertView);
            convertView.setTag(viewHolder);
        }
        else{
            viewHolder = (ViewHolder) convertView.getTag();
        }
        final Product product = productList.get(position);

        viewHolder.nameView.setText(product.getName());
        viewHolder.countView.setText(product.getCount() + " " + product.getUnit());

        viewHolder.removeButton.setOnClickListener(new View.OnClickListener()
{
            @Override
            public void onClick(View v) {
                int count = product.getCount()-1;
                if(count<0) count=0;
                product.setCount(count);
                viewHolder.countView.setText(count + " " + product.getUnit());
            }
        });
        viewHolder.addButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                int count = product.getCount()+1;
                product.setCount(count);
                viewHolder.countView.setText(count + " " + product.getUnit());
            }
        });
```

```
        return convertView;
    }
    private class ViewHolder {
        final Button addButton, removeButton;
        final TextView nameView, countView;
        ViewHolder(View view){
            addButton = view.findViewById(R.id.addButton);
            removeButton = view.findViewById(R.id.removeButton);
            nameView = view.findViewById(R.id.nameView);
            countView = view.findViewById(R.id.countView);
        }
    }
}
```

Для каждой кнопки здесь определен обработчик нажатия, в котором мы уменьшаем, либо увеличиваем количество продукта на единицу и затем переустанавливаем текст в сооветствующем текстовом поле.

Далее в файле **activity_main.xml** определим элемент ListView:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <ListView
        android:id="@+id/productList"
        android:layout_width="0dp"
```

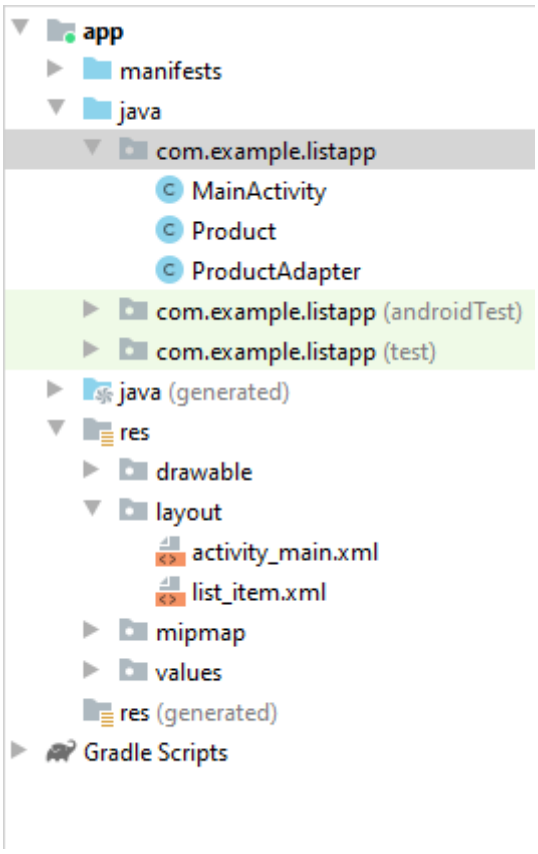
```
        android:layout_height="0dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

И изменим класс **MainActivity**:

```
package com.example.listapp;

import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.widget.ListView;
import java.util.ArrayList;

public class MainActivity extends AppCompatActivity {

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

        ArrayList<Product> products = new ArrayList<Product>();
        if(products.size()==0){
            products.add(new Product("Картофель", "кг."));
            products.add(new Product("Чай", "шт."));
            products.add(new Product("Яйца", "шт."));
            products.add(new Product("Молоко", "л."));
```

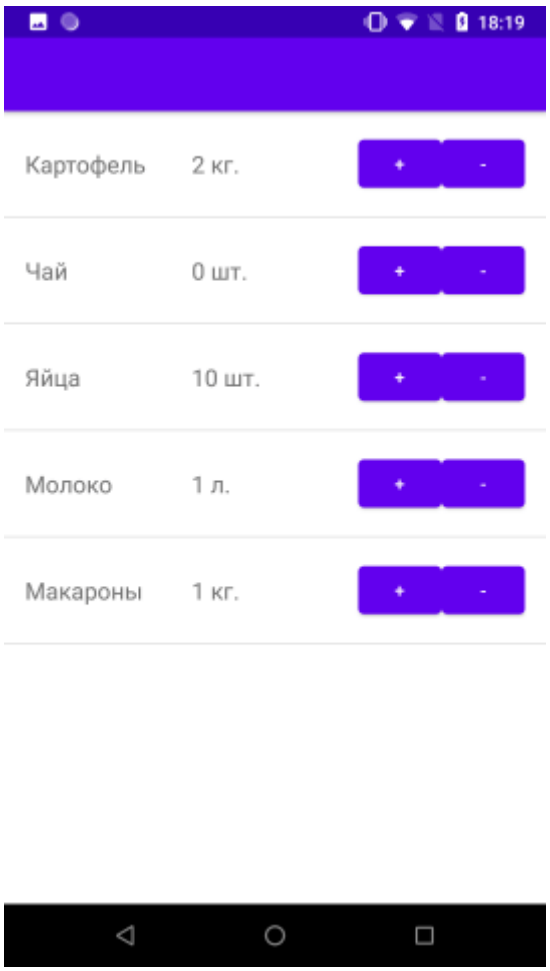
```
            products.add(new Product("Макароны", "кг."));
        }
        ListView productList = findViewById(R.id.productList);
        ProductAdapter adapter = new ProductAdapter(this, R.layout.list_item,
products);
        productList.setAdapter(adapter);
    }
}
```

В итоге получится следующий проект:



ListView с кнопками в Android и Java

И после запуска приложения мы сможем управлять количеством продуктов через кнопки:



ListView с кнопками в Android и Java

Выпадающий список Spinner

Spinner представляет собой выпадающий список. Определим в файле разметки activity_main.xml элемент Spinner:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:padding="16dp">
```

```
    <Spinner
        android:id="@+id/spinner"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

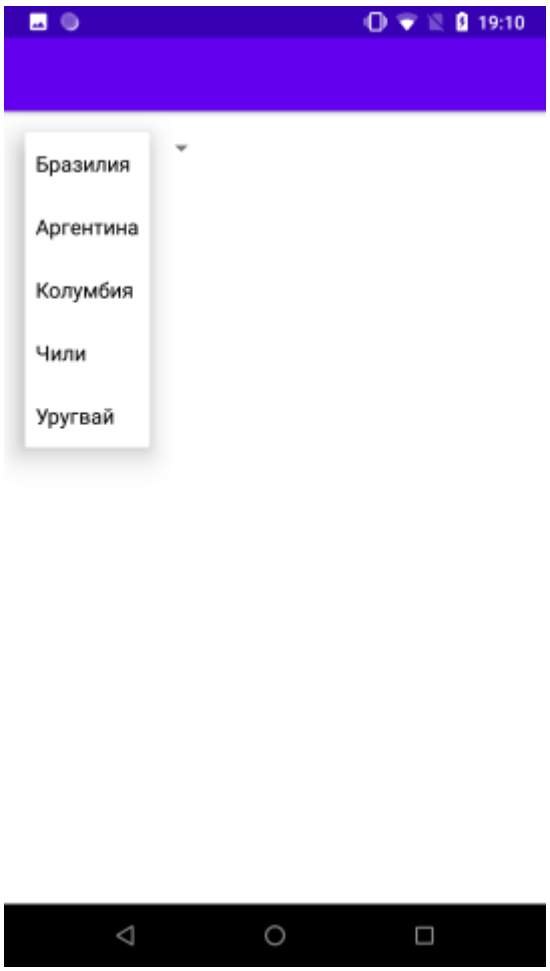
В качестве источника данных, как и для ListView, для Spinner может служить простой список или массив, соданный программно, либо ресурс string-array. Взаимодействие с источником данных также будет идти через адаптер. В данном случае определим источник программно в виде массива в коде MainActivity:

```
package com.example.listapp;

import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.widget.ArrayAdapter;
import android.widget.Spinner;

public class MainActivity extends AppCompatActivity {

    String[] countries = { "Бразилия", "Аргентина", "Колумбия", "Чили",
"Уругвай"};
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
```

```
        Spinner spinner = findViewById(R.id.spinner);
        // Создаем адаптер ArrayAdapter с помощью массива строк и стандартной разметки элемета spinner
        ArrayAdapter<String> adapter = new ArrayAdapter(this, android.R.layout.simple_spinner_item, countries);
        // Определяем разметку для использования при выборе элемента

adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
        // Применяем адаптер к элементу spinner
        spinner.setAdapter(adapter);
    }
}
```

Используемый при создании ArrayAdapter ресурс android.R.layout.simple_spinner_item предоставляется платформой и является стандартной разметкой для создания выпадающего списка.

С помощью метода **adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item**) устанавливаются дополнительные визуальные возможности списка. А передаваемый в метод ресурс **android.R.layout.simple_spinner_dropdown_item** используется для визуализации выпадающего списка и также предоставляется платформой.



Элемент Spinner в Android и Java

Обработка выбора элемента

Используя слушатель **OnItemSelectedListener**, в частности его метод onItemSelected(), мы можем обрабатывать выбор элемента из списка. Вначале добавим в разметку интерфейса текстовое поле, которое будет выводить выбранный элемент:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:padding="16dp">
```

```
    <TextView
        android:id="@+id/selection"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="26sp"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintTop_toTopOf="parent">
    </TextView>
    <Spinner
        android:id="@+id/spinner"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/selection" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

И изменим код MainActivity, определив для элемента Spinner слушатель **OnItemSelectedListener:**

```
package com.example.listapp;

import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.Spinner;
import android.widget.TextView;
```

```
public class MainActivity extends AppCompatActivity {


    String[] countries = { "Бразилия", "Аргентина", "Колумбия", "Чили",
"Уругвай"};
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);


        TextView selection = findViewById(R.id.selection);


        Spinner spinner = findViewById(R.id.spinner);
        // Создаем адаптер ArrayAdapter с помощью массива строк и
стандартной разметки элемета spinner
        ArrayAdapter<String> adapter = new ArrayAdapter(this,
android.R.layout.simple_spinner_item, countries);
        // Определяем разметку для использования при выборе элемента

adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_
item);
        // Применяем адаптер к элементу spinner
        spinner.setAdapter(adapter);


        AdapterView.OnItemSelectedListener itemSelectedListener = new
AdapterView.OnItemSelectedListener() {
            @Override
            public void onItemSelected(AdapterView<?> parent, View view, int
position, long id) {


                // Получаем выбранный объект
```

```
            String item = (String)parent.getItemAtPosition(position);
            selection.setText(item);
        }

        @Override
        public void onNothingSelected(AdapterView<?> parent) {
        }
    };
    spinner.setOnItemSelectedListener(itemSelectedListener);
  }
}
```

Метод onItemSelected слушателя OnItemSelectedListener получает четыре параметра:

- parent: объект Spinner, в котором произошло событие выбора элемента
- view: объект View внутри Spinnera, который представляет выбранный элемент
- position: индекс выбранного элемента в адаптере
- id: идентификатор строки того элемента, который был выбран

Получив позицию выбранного элемента, мы можем найти его в списке:

```
String item = (String)parent.getItemAtPosition(position);
```

Для установки слушателя OnItemSelectedListener в классе Spinner применяется метод **setOnItemSelectedListener.**

Выбор элемента в списке Spinner в Android и Java с помощью OnItemSelectedListener

Виджет автодополнения AutoCompleteTextView

AutoCompleteTextView представляет элемент, созданный на основе класса EditText и обладающий возможностью автодополнения

Во-первых, объявим в ресурсе разметке данный элемент:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
```

```
    android:layout_height="match_parent"
    android:padding="16dp">

    <AutoCompleteTextView
        android:id="@+id/autocomplete"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:completionHint="Введите город"
        android:completionThreshold="1"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        />

</androidx.constraintlayout.widget.ConstraintLayout>
```

Атрибут **android:completionHint** позволяет задать надпись, которая отображается внизу списка, а свойство **android:completionThreshold** устанавливает, какое количество символов надо ввести, чтобы начало работать автодополнение. То есть в данном случае уже после ввода одного символа должен появться список с подстановками.

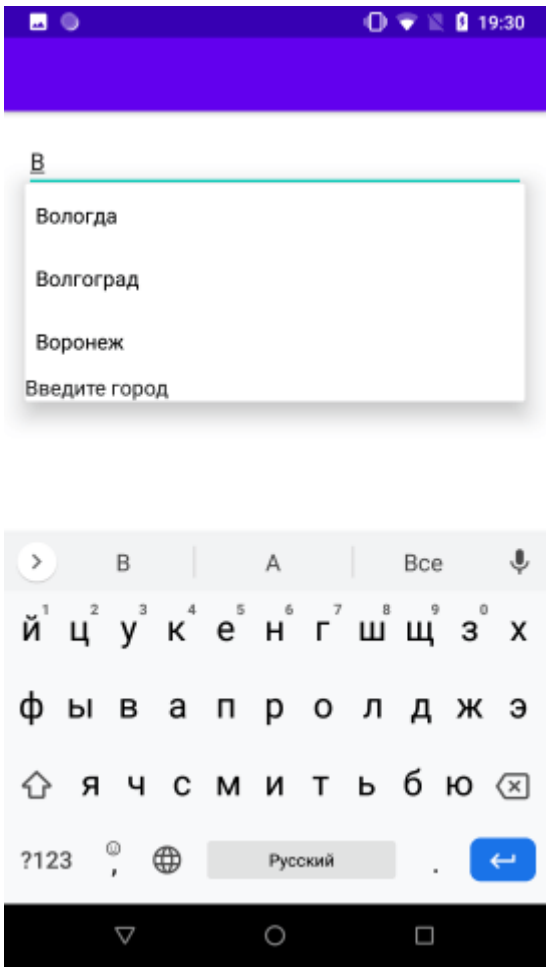
Как и в случае с элементами ListView и Spinner, AutoCompleteTextView подключается к источнику данных через адаптер. Источником данных опять же может служить массив или список объектов, либо ресурс string-array.

Теперь подключим к виджету массив строк в классе MainActivity:

```
package com.example.listapp;

import androidx.appcompat.app.AppCompatActivity;
```

```
import android.os.Bundle;
import android.widget.ArrayAdapter;
import android.widget.AutoCompleteTextView;

public class MainActivity extends AppCompatActivity {

   String[] cities = {"Москва", "Самара", "Вологда", "Волгоград", "Саратов", "Воронеж"};
   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);

      // Получаем ссылку на элемент AutoCompleteTextView в разметке
      AutoCompleteTextView autoCompleteTextView = findViewById(R.id.autocomplete);
      // Создаем адаптер для автозаполнения элемента AutoCompleteTextView
      ArrayAdapter<String> adapter = new ArrayAdapter (this, R.layout.support_simple_spinner_dropdown_item, cities);
      autoCompleteTextView.setAdapter(adapter);
   }
}
```

После ввода в текстовое поле одной буквы отобразится список с вариантами автодополнения, где можно выбрать предпочтительный:



Элемент AutoCompleteTextView в Android и Java

MultiAutoCompleteTextView

Этот виджет дополняет функциональность элемента AutoCompleteTextView. MultiAutoCompleteTextView позволяет использовать автодополнения не только для одной строки, но и для отдельных слов. Например, если вводится слово и после него ставится запятая, то автозаполнение все равно будет работать для вновь вводимых слов после запятой или другого разделителя.

MultiAutoCompleteTextView имеет такую же форму объявления, как и AutoCompleteTextView:

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

<androidx.constraintlayout.widget.ConstraintLayout

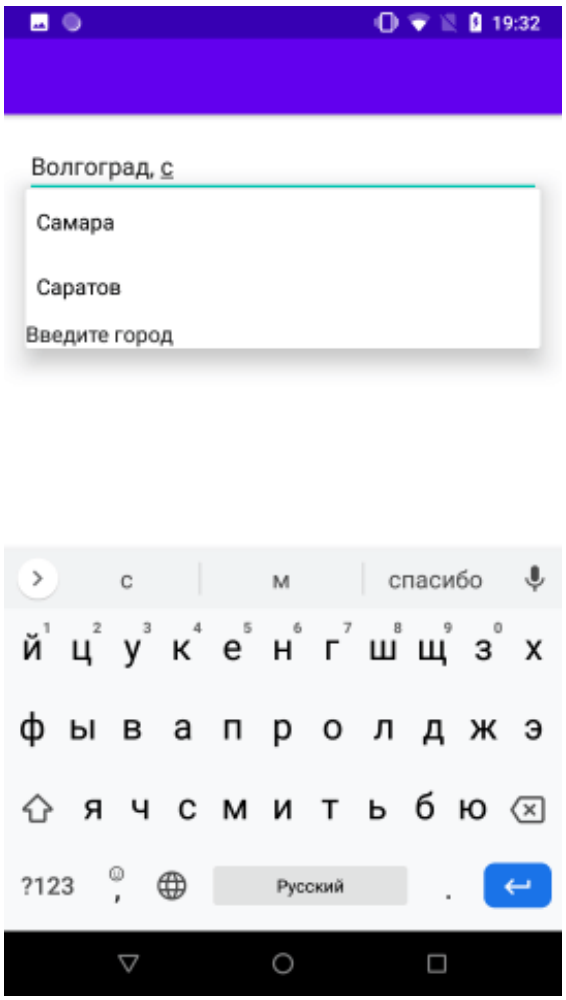
```
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:padding="16dp">

    <MultiAutoCompleteTextView
        android:id="@+id/autocomplete"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:completionHint="Введите город"
        android:completionThreshold="1"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        />
 </androidx.constraintlayout.widget.ConstraintLayout>
```

Чтобы включить MultiAutoCompleteTextView в коде, надо установить токен разделителя:

```
package com.example.listapp;
import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.widget.ArrayAdapter;
import android.widget.MultiAutoCompleteTextView;
public class MainActivity extends AppCompatActivity {
String[] cities = {"Москва", "Самара", "Вологда", "Волгоград", "Саратов", "Воронеж"};
    @Override
```

```
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        // Получаем ссылку на элемент AutoCompleteTextView в разметке
        MultiAutoCompleteTextView autoCompleteTextView =
findViewById(R.id.autocomplete);
        // Создаем адаптер для автозаполнения элемента
MultiAutoCompleteTextView
        ArrayAdapter<String> adapter = new ArrayAdapter(this,
R.layout.support_simple_spinner_dropdown_item, cities);
        autoCompleteTextView.setAdapter(adapter);
        // установка запятой в качестве разделителя
        autoCompleteTextView.setTokenizer(new
MultiAutoCompleteTextView.CommaTokenizer());
    }
}
```

Здесь в качестве разделителя используется встроенный разделитель на основе запятой **CommaTokenizer()**. При необходимости мы можем создать свои разделители.



Элемент MultiAutoCompleteTextView в Android и Java

GridView

Элемент **GridView** представляет отображение в виде таблицы - набора строк и столбцов.

Основные атрибуты GridView:

- **android:columnWidth**: устанавливает фиксированную ширину столбцов
- **android:gravity**: устанавливает выравнивание содержимого внутри каждой ячейки

- **android:horizontalSpacing**: устанавливает горизональные отступы между столбцами

- **android:numColumns**: устанавливает количество столбцов

- **android:stretchMode**: устанавливает, как столбцы будут растягиваться и занимать пространство контейнера. Может принимать следующие значения:

columnWidth: каждый столбец растягивается равномерно по всей ширине. Эквивалентно значению 2

none: столбцы не растягиваются. Эквивалентно значению 0

spacingWidth: между столбцами образуются отступы. Эквивалентно значению 1

spacingWidthUniform: между столбцами образуются равномерные отступы. Эквивалентно значению 3

- **android:verticalSpacing**: устанавливает вертикальные отступы между строками

Основные методы класса GridView:

int getColumnWidth(): возвращает ширину столбцов

int getHorizontalSpacing(): возвращает размер горизонтального отступа

int getNumColumns(): возвращает количество столбцов

int getStretchMode(): возвращает режим растяжения пространства внутри грида

int getVerticalSpacing(): возвращает размер вертикального отступа

void setAdapter(ListAdapter adapter): устанавливает адаптер для подключения к источнику данных

void setColumnWidth(int columnWidth): устанавливает ширину столбцов

void setHorizontalSpacing(int horizontalSpacing): устанавливает размер горизонтального отступа

void setNumColumns(int numColumns): устанавливает количество столбцов

void setStretchMode(int stretchMode): устанавливает режим растяжения пространства внутри грида

void setVerticalSpacing(int verticalSpacing): устанавливает размер вертикального отступа

void setSelection(int position): устанавливает текущий выделенный элемент

Определим **GridView** в **activity_main.xml:**

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:padding="16dp">
    <GridView
```

```
    android:id="@+id/gridview"
    android:layout_width="0dp"
    android:layout_height="0dp"
    android:numColumns="2"
    android:verticalSpacing="16dp"
    android:horizontalSpacing="16dp"
    android:stretchMode="columnWidth"

    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintBottom_toBottomOf="parent" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

В данном случае указываем, что грид будет иметь 2 столбца, которые растягиваются равномерно по всей ширине грида, а между ячейками будут отступы по горизонтали и вертикали в 16 dp.

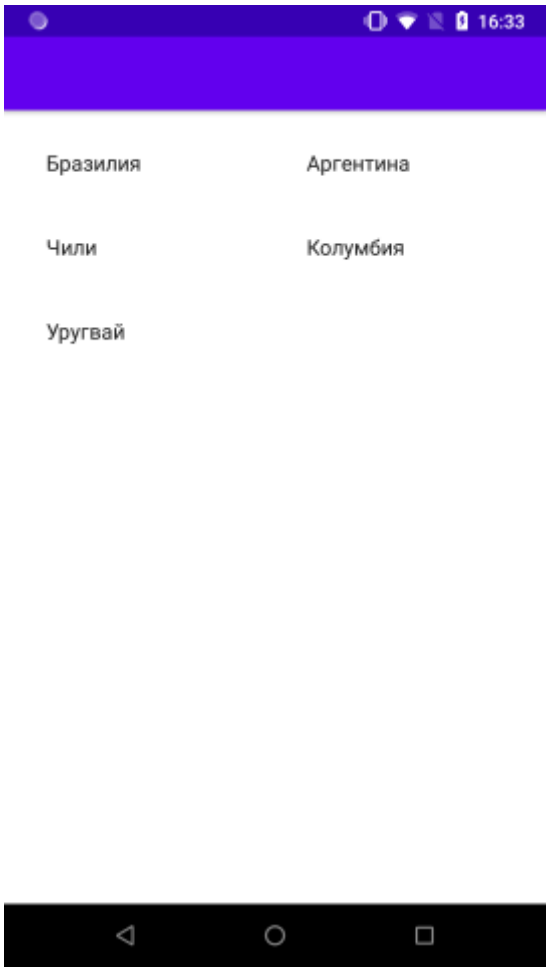
Теперь, как и в случае с ListView, надо установить связь с адаптером:

```
package com.example.listapp;

import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.GridView;
import android.widget.Toast;
```

```
public class MainActivity extends AppCompatActivity {


    String[] countries = { "Бразилия", "Аргентина", "Чили", "Колумбия",
"Уругвай"};
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);


        // получаем элемент GridView
        GridView countriesList = findViewById(R.id.gridview);
        // создаем адаптер
        ArrayAdapter<String> adapter = new ArrayAdapter(this,
android.R.layout.simple_list_item_1, countries);
        countriesList.setAdapter(adapter);


        AdapterView.OnItemClickListener itemListener = new
AdapterView.OnItemClickListener() {


            @Override
            public void onItemClick(AdapterView<?> parent, View view, int position,
long id) {
                Toast.makeText(getApplicationContext(),"Вы выбрали "
                        + parent.getItemAtPosition(position).toString(),
                    Toast.LENGTH_SHORT).show();
            }
        };
        countriesList.setOnItemClickListener(itemListener);
    }
```

}

Для обработки нажатия в GridView применяется слушатель **AdapterView.OnItemClickListener**.



Элемент GridView в Android и Java

RecyclerView

Элемент RecyclerView предназначен для оптимизации работы со списками и во многом позволяет повысить производительность по сравнению со стандартным ListView.

Для представления данных добавим в проект в ту же папку, где расположен класс MainActivity, новый класс Java, который назовем State:

package com.example.listapp;

```
public class State {

    private String name; // название
    private String capital;  // столица
    private int flagResource; // ресурс флага

    public State(String name, String capital, int flag){

        this.name=name;
        this.capital=capital;
        this.flagResource=flag;
    }

    public String getName() {
        return this.name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getCapital() {
        return this.capital;
    }

    public void setCapital(String capital) {
        this.capital = capital;
    }
```

```
    public int getFlagResource() {
        return this.flagResource;
    }

    public void setFlagResource(int flagResource) {
        this.flagResource = flagResource;
    }
}
```

Класс State содержит поля для хранения названия и столицы страны, а также ссылку на ресурс изображения флага страны. В данном случае предполагается, что в папке **res/drawable** будут располагаться файлы изображений флагов для используемых стран.

Допустим, мы хотим вывести список объектов State с помощью **RecyclerView**. Для этого добавим в папку **res/layout** новый файл **list_item.xml**:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="wrap_content">
    <ImageView
        android:id="@+id/flag"
        android:layout_width="70dp"
        android:layout_height="50dp"
        android:layout_marginTop="16dp"
        android:layout_marginBottom="16dp"
        app:layout_constraintLeft_toLeftOf="parent"
```

```
        app:layout_constraintRight_toLeftOf="@+id/name"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintBottom_toBottomOf="parent" />

    <TextView
        android:id="@+id/name"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginLeft="16dp"
        android:text="Название"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintLeft_toRightOf="@+id/flag"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintBottom_toTopOf="@+id/capital" />

    <TextView
        android:id="@+id/capital"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginLeft="16dp"
        android:text="Столица"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintLeft_toRightOf="@+id/flag"
        app:layout_constraintTop_toBottomOf="@+id/name"
        app:layout_constraintBottom_toBottomOf="parent" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

Этот файл определяет разметку для вывода одного объекта State.

Как и в случае с ListView, для вывода сложных объектов в **RecyclerView** необходимо определить свой адаптер. Поэтому добавим в ту же папку, где расположен класс MainActivity и State, новый класс Java, который назовем **StateAdapter**:

```
package com.example.listapp;


import android.content.Context;

import android.view.LayoutInflater;

import android.view.View;

import android.view.ViewGroup;

import android.widget.ImageView;

import android.widget.TextView;


import androidx.recyclerview.widget.RecyclerView;


import java.util.List;


public class StateAdapter  extends
RecyclerView.Adapter<StateAdapter.ViewHolder>{


    private final LayoutInflater inflater;

    private final List<State> states;


    StateAdapter(Context context, List<State> states) {

        this.states = states;

        this.inflater = LayoutInflater.from(context);

    }

    @Override
```

```
    public StateAdapter.ViewHolder onCreateViewHolder(ViewGroup parent, int
viewType) {

        View view = inflater.inflate(R.layout.list_item, parent, false);
        return new ViewHolder(view);
    }

    @Override
    public void onBindViewHolder(StateAdapter.ViewHolder holder, int position) {
        State state = states.get(position);
        holder.flagView.setImageResource(state.getFlagResource());
        holder.nameView.setText(state.getName());
        holder.capitalView.setText(state.getCapital());
    }

    @Override
    public int getItemCount() {
        return states.size();
    }

    public static class ViewHolder extends RecyclerView.ViewHolder {
        final ImageView flagView;
        final TextView nameView, capitalView;
        ViewHolder(View view){
            super(view);
            flagView = view.findViewById(R.id.flag);
            nameView = view.findViewById(R.id.name);
            capitalView = view.findViewById(R.id.capital);
        }
```

```
    }
}
```

Адаптер, который используется в RecyclerView, должен наследоваться от абстрактного класса **RecyclerView.Adapter**. Этот класс определяет три метода:

- **onCreateViewHolder**: возвращает объект ViewHolder, который будет хранить данные по одному объекту State.
- **onBindViewHolder**: выполняет привязку объекта ViewHolder к объекту State по определенной позиции.
- **getItemCount**: возвращает количество объектов в списке

Для хранения данных в классе адаптера определен статический класс ViewHolder, который использует определенные в **list_item.xml** элементы управления.

Теперь определим в файле **activity_main.xml** элемент RecyclerView:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:padding="16dp">
    <androidx.recyclerview.widget.RecyclerView
        android:id="@+id/list"
        android:layout_width="0dp"
```

```
        android:layout_height="0dp"

        app:layoutManager="androidx.recyclerview.widget.LinearLayoutManager"

        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintBottom_toBottomOf="parent"/>
</androidx.constraintlayout.widget.ConstraintLayout>
```

Следует учитывать, что RecyclerView расположен в пакете **androidx.recyclerview.widget** и является частью тулкита **Android Jetpack**, поэтому при использовании виджета указывается полное его название с учетом пакета (в принципе как и для ConstraintLayout):

```
<androidx.recyclerview.widget.RecyclerView ....
```

Для RecyclerView следует установить атрибут app:layoutManager, который указает на тип менеджера компоновки. Менеджер компоновки представляет объект, который представлен классом LayoutManager. По умолчанию библиотека RecyclerView предоставляет три реализации данного менеджера:

- **LinearLayoutManager:** упорядочивает элементы в виде списка с одной колонкой

- **GridLayoutManager**: упорядочивает элементы в виде грида со столлбцами и строками. Грид может упорядочивать элементы по горизонтали (горизонтальный грид) или по вертикали (вертикальный грид)

- **StaggeredGridLayoutManager**: аналогичен GridLayoutManager, однако не требует установки для каждого элемента в строке имели одну и ту же высоту (для вертикального грида) и одну и ту же ширну (для горизонтального грида)

В данном случае с помощью значения **androidx.recyclerview.widget.LinearLayoutManager** указываем, что элементы будут располагаться в виде простого списка. Обратите внимание, что класс LinearLayoutManager также расположен в библиотеке RecyclerView и поэтому при указании значения указывается полное название класса с именем его пакета.

И в конце изменим класс **MainActivity**:

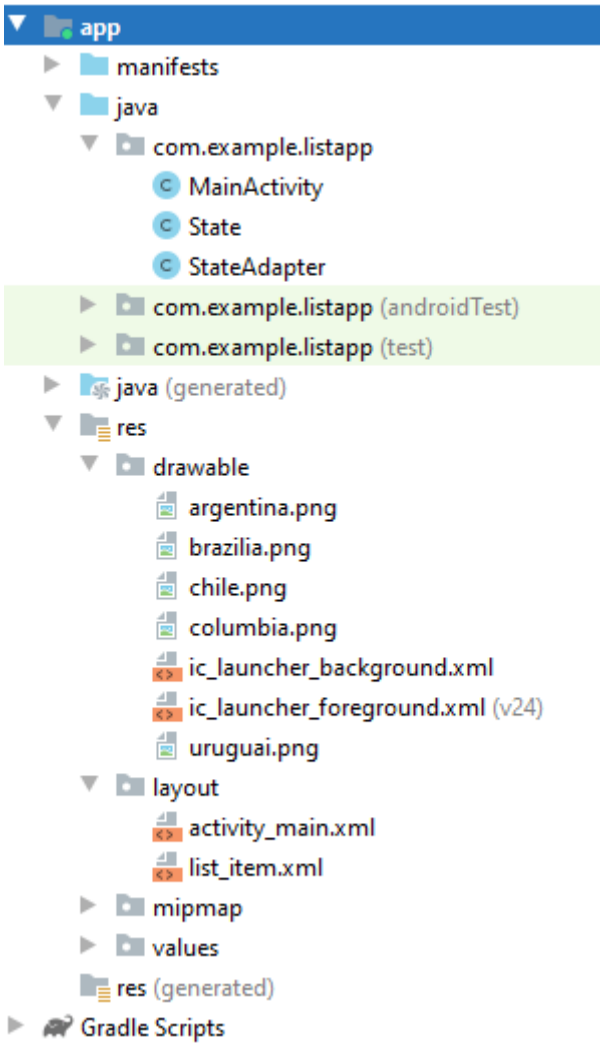
```
package com.example.listapp;


import androidx.appcompat.app.AppCompatActivity;
import androidx.recyclerview.widget.RecyclerView;
import android.os.Bundle;
import java.util.ArrayList;


public class MainActivity extends AppCompatActivity {


    ArrayList<State> states = new ArrayList<State>();
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        // начальная инициализация списка
        setInitialData();
        RecyclerView recyclerView = findViewById(R.id.list);
        // создаем адаптер
        StateAdapter adapter = new StateAdapter(this, states);
        // устанавливаем для списка адаптер
        recyclerView.setAdapter(adapter);
```

```
    }
    private void setInitialData(){

        states.add(new State ("Бразилия", "Бразилиа", R.drawable.brazilia));
        states.add(new State ("Аргентина", "Буэнос-Айрес", R.drawable.argentina));
        states.add(new State ("Колумбия", "Богота", R.drawable.columbia));
        states.add(new State ("Уругвай", "Монтевидео", R.drawable.uruguai));
        states.add(new State ("Чили", "Сантьяго", R.drawable.chile));
    }
}
```

С помощью метода setInitialData() устанавливается набор начальных данных. В данном случае имеется в виду, что в папке res/drawables находится ряд ресурсов изображений для объектов State.

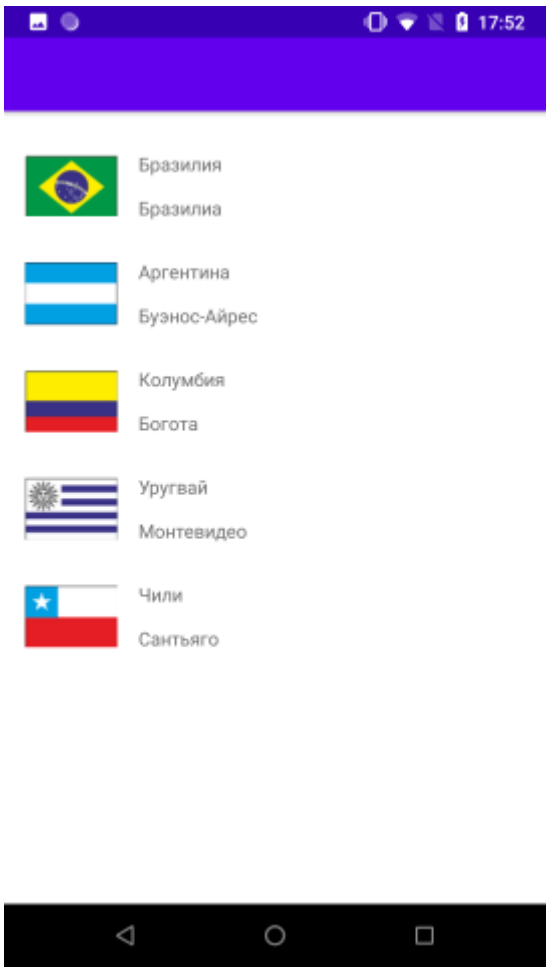
Как и в случае с выводом списка через ListView здесь сначала получаем элемент RecyclerView, создаем адаптер и устанавливаем адаптер для RecyclerView.

Весь проект в итоге будет выглядеть следующим образом:



Проект для RecyclerView в Android и Java

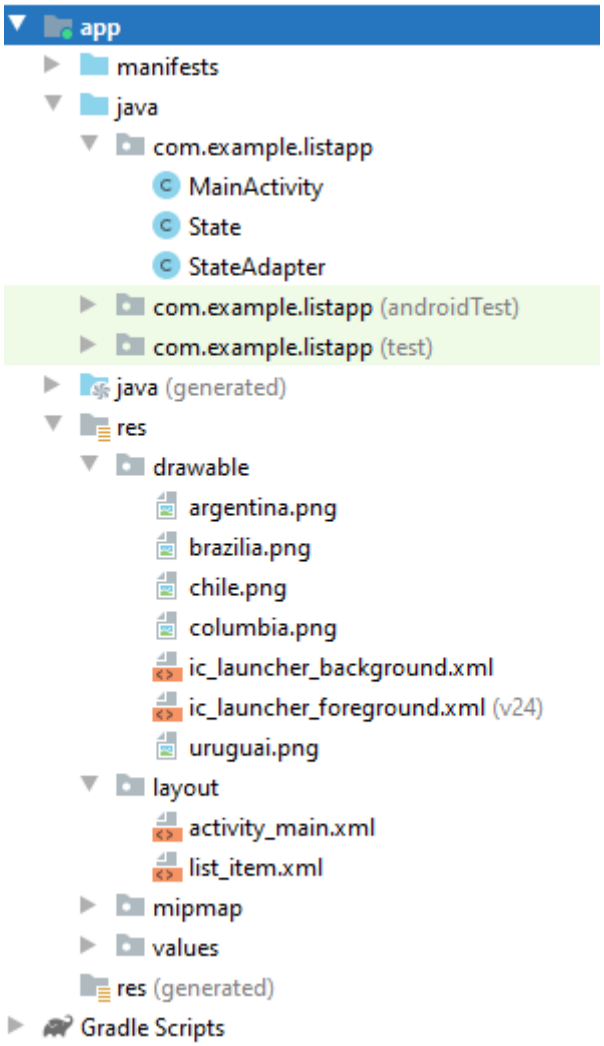
В результате RecyclerView выведет набор объектов:



Список RecyclerView в Android и Java

Обработка выбора элемента в RecyclerView

При работе с виджетом RecyclerView неизбежно встанет вопрос, а как обработать выбор элемента в RecyclerView. И тут надо заметить, что в отличие от других типов виджетов для работы со списками RecyclerView по умолчанию не предоставляет какого-то специального метода, с помощью которого можно было бы определить слушатель нажатия на элемент в списке. Поэтому всю инфраструктуру необходимо определять самому разработчику. Но, к счастью, в реальности все не так сложно. Для примера возьмем проект из прошлого материала темы:



Проект для RecyclerView в Android и Java

Итак, для представления данных в проекте есть класс **State**, который представляет государство:

```
package com.example.listapp;


public class State {

    private String name; // название
    private String capital;  // столица
    private int flagResource; // ресурс флага

    public State(String name, String capital, int flag){
```

```
    this.name=name;
    this.capital=capital;
    this.flagResource=flag;
}

public String getName() {
    return this.name;
}

public void setName(String name) {
    this.name = name;
}

public String getCapital() {
    return this.capital;
}

public void setCapital(String capital) {
    this.capital = capital;
}

public int getFlagResource() {
    return this.flagResource;
}

public void setFlagResource(int flagResource) {
    this.flagResource = flagResource;
}
```

```
}
```

Класс State содержит поля для хранения названия и столицы страны, а также ссылку на ресурс изображения флага страны. В данном случае предполагается, что в папке res/drawable будут располагаться файлы изображений флагов для используемых стран.

В папке **res/layout** для вывода одного объекта State в списке определен следующий файл **list_item.xml**:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="wrap_content">
    <ImageView
        android:id="@+id/flag"
        android:layout_width="70dp"
        android:layout_height="50dp"
        android:layout_marginTop="16dp"
        android:layout_marginBottom="16dp"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toLeftOf="@+id/name"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintBottom_toBottomOf="parent" />

    <TextView
        android:id="@+id/name"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
```

```
        android:layout_marginLeft="16dp"
        android:text="Название"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintLeft_toRightOf="@+id/flag"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintBottom_toTopOf="@+id/capital" />

    <TextView
        android:id="@+id/capital"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginLeft="16dp"
        android:text="Столица"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintLeft_toRightOf="@+id/flag"
        app:layout_constraintTop_toBottomOf="@+id/name"
        app:layout_constraintBottom_toBottomOf="parent" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

Теперь перейдем к классу StateAdapter и следующим образом определим его код:

```
package com.example.listapp;

import android.content.Context;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ImageView;
```

```
import android.widget.TextView;
import androidx.recyclerview.widget.RecyclerView;

import java.util.List;

public class StateAdapter  extends RecyclerView.Adapter<StateAdapter.ViewHolder>{

    interface OnStateClickListener{
        void onStateClick(State state, int position);
    }

    private final OnStateClickListener onClickListener;

    private final LayoutInflater inflater;
    private final List<State> states;

    StateAdapter(Context context, List<State> states, OnStateClickListener onClickListener) {
        this.onClickListener = onClickListener;
        this.states = states;
        this.inflater = LayoutInflater.from(context);
    }
    @Override
    public StateAdapter.ViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {

        View view = inflater.inflate(R.layout.list_item, parent, false);
        return new ViewHolder(view);
    }
```

```
@Override
public void onBindViewHolder(StateAdapter.ViewHolder holder, int position) {
    State state = states.get(position);
    holder.flagView.setImageResource(state.getFlagResource());
    holder.nameView.setText(state.getName());
    holder.capitalView.setText(state.getCapital());

    holder.itemView.setOnClickListener(new View.OnClickListener(){
        @Override
        public void onClick(View v)
        {
            onClickListener.onStateClick(state, position);
        }
    });
}

@Override
public int getItemCount() {
    return states.size();
}

public static class ViewHolder extends RecyclerView.ViewHolder {
    final ImageView flagView;
    final TextView nameView, capitalView;
    ViewHolder(View view){
        super(view);
        flagView = view.findViewById(R.id.flag);
        nameView = view.findViewById(R.id.name);
```

```
            capitalView = view.findViewById(R.id.capital);
        }
    }
}
```

Здесь я остановлюсь на тех моментах, которые были добавлены по сравнению с кодом из прошлого материала.

Прежде всего нам надо определить интерфейс слушателя события нажатия. Для этого в классе StateAdapter определен интерфейс:

```
interface OnStateClickListener{
    void onStateClick(State state, int position);
}
```

Интерфейс определяет один метод onStateClick(), который, как предполагается, будет вызываться при выборе объекта State и который будет получать выбранный объект State и его позицию в списке.

Следующий момент - определение в классе адаптера переменной для хранения объекта этого интерфейса и получение для нее значения в конструкторе:

```
private final OnStateClickListener onClickListener;

StateAdapter(Context context, List<State> states, OnStateClickListener
onClickListener) {

    this.onClickListener = onClickListener;
    // .....................
}
```

Таким образом, вне кода адаптера мы можем установить любой объект слушателя и передать его в адаптер.

И третий момент - вызов метода слушателя при нажатии на элемент в списке в методе **onBindViewHolder**:

```
public void onBindViewHolder(StateAdapter.ViewHolder holder, int position) {
    State state = states.get(position);
    holder.flagView.setImageResource(state.getFlagResource());
    holder.nameView.setText(state.getName());
    holder.capitalView.setText(state.getCapital());

    // обработка нажатия
    holder.itemView.setOnClickListener(new View.OnClickListener(){
        @Override
        public void onClick(View v)
        {
            // вызываем метод слушателя, передавая ему данные
            onClickListener.onStateClick(state, position);
        }
    });
}
```

Класс **ViewHolder** имеет поле **itemView**, которое представляет интерфейс для одного объекта в списке и фактически объект **View**. А у этого объекта есть метод setOnClickListener(), через который можно подлючить стандартный слушатель нажатия **OnClickListener** и в его методе onClick() вызвать метод нашего интерфейса, передав ему необходимые данные - выбранный объект State и его позицию в списке.

Может возникнуть вопрос, а почему бы сразу тут и не обработать нажатие на элемент? К чему создавать дополнительный интерфейс, его переменную и

вызывать его метод? Конечно, мы можем попыться прямо тут обработать нажатия, но это не является хорошей или распространенной практикой, поскольку, возможно, мы захотим определить обработку нажатия в классе MainActivity исходя из того, кода, который там определен (или из какого-то другого места извне).

В файле activity_main.xml остается тот же самый визуальный интерфейс:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:padding="16dp">
    <androidx.recyclerview.widget.RecyclerView
        android:id="@+id/list"
        android:layout_width="0dp"
        android:layout_height="0dp"

        app:layoutManager="androidx.recyclerview.widget.LinearLayoutManager"

        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintBottom_toBottomOf="parent"/>
</androidx.constraintlayout.widget.ConstraintLayout>
```

И в конце изменим класс MainActivity:

```
package com.example.listapp;
```

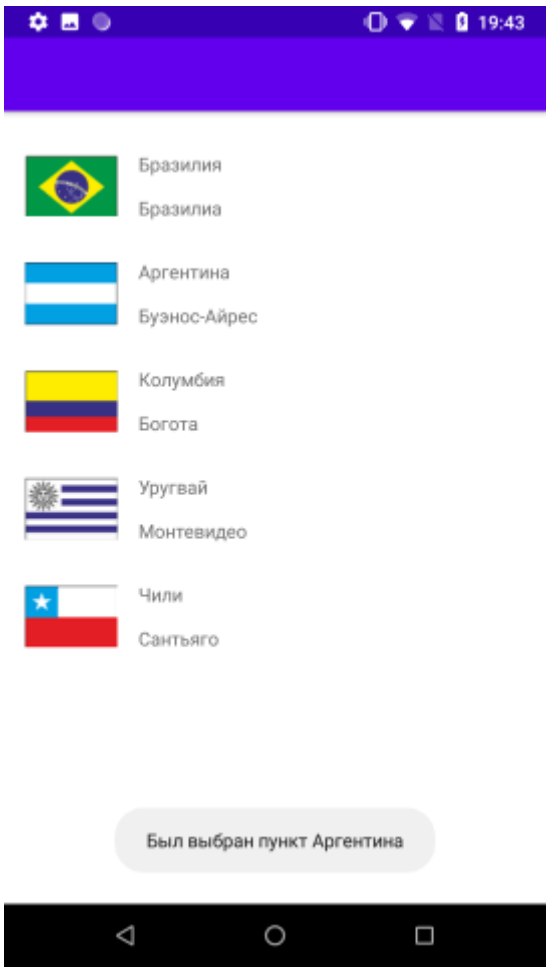
```
import androidx.appcompat.app.AppCompatActivity;
import androidx.recyclerview.widget.RecyclerView;
import android.os.Bundle;
import android.widget.Toast;
import java.util.ArrayList;

public class MainActivity extends AppCompatActivity {

    ArrayList<State> states = new ArrayList<State>();
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        // начальная инициализация списка
        setInitialData();
        RecyclerView recyclerView = findViewById(R.id.list);
        // определяем слушателя нажатия элемента в списке
        StateAdapter.OnStateClickListener stateClickListener = new
StateAdapter.OnStateClickListener() {
            @Override
            public void onStateClick(State state, int position) {

                Toast.makeText(getApplicationContext(), "Был выбран пункт " +
state.getName(),
                        Toast.LENGTH_SHORT).show();
            }
        };
        // создаем адаптер
        StateAdapter adapter = new StateAdapter(this, states, stateClickListener);
```

```
        // устанавливаем для списка адаптер
        recyclerView.setAdapter(adapter);
    }
    private void setInitialData(){

        states.add(new State ("Бразилия", "Бразилиа", R.drawable.brazilia));
        states.add(new State ("Аргентина", "Буэнос-Айрес", R.drawable.argentina));
        states.add(new State ("Колумбия", "Богота", R.drawable.columbia));
        states.add(new State ("Уругвай", "Монтевидео", R.drawable.uruguai));
        states.add(new State ("Чили", "Сантьяго", R.drawable.chile));
    }
}
```

При создании адаптера ему передается определенный в классе MainActivity слушатель:

```
StateAdapter.OnStateClickListener stateClickListener = new
StateAdapter.OnStateClickListener() {
    @Override
    public void onStateClick(State state, int position) {

        Toast.makeText(getApplicationContext(), "Был выбран пункт " +
state.getName(),
                Toast.LENGTH_SHORT).show();
    }
};
```

Здесь просто выводится всплывающее сообщение о выбранном элементе списка.



OnClickListener в ViewHolder и RecyclerView в Android и Java

Задание

1. Изучить применение адаптеров и списков. Реализовать пример связи

элемента **ListView** с источником данных с набором элементов **TextView** с помощью одного из таких адаптеров - класса **ArrayAdapter.**

2. Реализовать пример с использованием ресурса string-array и ListView.
3. Реализовать выбор элемента в ListView.
4. Реализовать множественный выбор в списке.
5. Реализовать добавление и удаление в ArrayAdapter и ListView
6. Реализовать расширение списков и создание адаптера.
7. Реализовать оптимизацию адаптера и применение View Holder
8. Реализовать сложный список с кнопками
9. Реализовать выпадающий список Spinner
10. Реализовать обработку выбора элемента используя слушатель OnItemSelectedListener
11. Реализовать виджет автодополнения AutoCompleteTextView
12. Реализовать виджет MultiAutoCompleteTextView.
13. Реализовать пример с элементом GridView
14. Реализовать пример с элементом RecyclerView
15. Реализовать обработку выбора элемента в RecyclerView