МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
імені ВОЛОДИМИРА ДАЛЯ

КОНСПЕКТ ЛЕКЦІЙ

навчально-методичного комплексу
дистанційного курсу дисципліни
«Програмування для мобільних платформ»
Частина ІІ
(для студентів спеціальностей
F2 «Інженерія програмного забезпечення»,
F6 «Інформаційні технології та системи»)
(Електронне видання)

ЗАТВЕРДЖЕНО
на засіданні кафедри
інформаційних техногологій та програмування
Протокол № 10 від 09.06.2025 р.

КИЇВ 2025

УДК 330.88:338:65.01

Конспект лекцій навчально-методичного комплексу дистанційного курсу дисципліни «Програмування для мобільних платформ» Частина II (для студентів спеціальностей F2 «Інженерія програмного забезпечення», F6 «Інформаційні технології та системи») (Електронне видання) / Уклад.: Ратов Д. В. - Київ: вид-во СНУ ім. В. Даля, 2025.– 188с.

Укладено на основі ОПП підготовки бакалаврів спеціальностей F2 «Інженерія програмного забезпечення», F6 «Інформаційні технології та системи», робочої навчальної програми з дисципліни „Програмування для мобільних платформ”.

Укладач: Д. В. Ратов, доц., к.т.н.

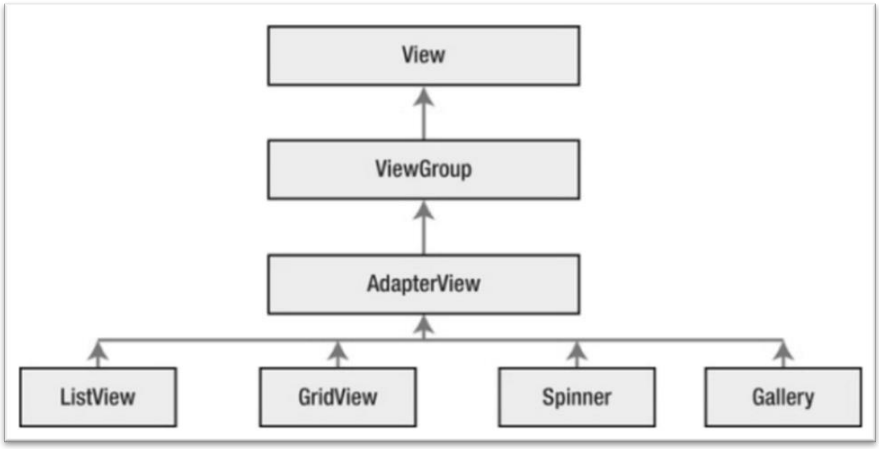
Рецензент О. І. Захожай, доц., д.т.н.

ЗМІСТ

1. АДАПТЕРИ ТА СПИСКИ

1.1. ListView та ArrayAdapter

Android представляє широку палітру елементів, які представляють списки. Усі вони є спадкоємцями класу android.widget.AdapterView. Це такі віджети, як ListView, GridView, Spinner. Вони можуть бути контейнерами для інших елементів управління



Під час роботи зі списками ми маємо справу з трьома компонентами. По-перше, це візуальний елемент або віджет, що на екрані представляє список (ListView, GridView) і який відображає дані. По-друге, це джерело даних - масив, об'єкт ArrayList, база даних і т.д., в якому знаходяться дані, що відображаються самі. І по-третє, це адаптер – спеціальний компонент, який пов'язує джерело даних із віджетом списку.

Одним із найпростіших та найпоширеніших елементів списку є віджет ListView. Розглянемо зв'язок елемента ListView з джерелом даних з допомогою однієї з таких адаптерів - класу ArrayAdapter.

Клас ArrayAdapter є найпростішим адаптером, який пов'язує масив даних з набором елементів TextView, з яких, наприклад, може складатися ListView. Тобто у разі джерелом даних виступає масив об'єктів. ArrayAdapter викликає у кожного об'єкта метод toString() для приведення до рядкового вигляду і отриманий рядок встановлює елемент TextView.

Подивимося на прикладі. Отже, розмітка програми може виглядати 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>
```

Тут також визначено елемент ListView, який виводитиме список об'єктів. Тепер перейдемо до коду activity та зв'яжемо ListView через ArrayAdapter з деякими даними:

```
        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 {

        // набір даних, які зв'яжемо зі списком
        String[] countries = { "Бразилія", "Аргентина", "Колумбія", "Чілі", "Уругвай"};

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

        // отримуємо елемент ListView
        ListView countriesList = findViewById(R.id.countriesList);

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

        // Встановлюємо для списку адаптер
        countriesList.setAdapter(adapter);
        }
        }
```

Тут спочатку отримуємо id елемент ListView і потім створюємо для нього адаптер.

Для створення адаптера використовувався наступний конструктор ArrayAdapter<String>(this,android.R.layout.simple_list_item_1, countries), де

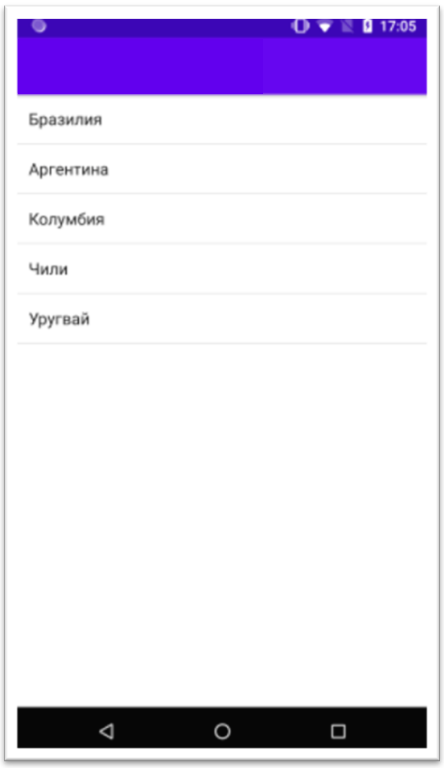
- **this**: поточний об'єкт activity
- **android.R.layout.simple_list_item_1**: файл розмітки списку, який представляє фреймворк за замовчуванням. Він знаходиться в папці Android SDK на шляху

platforms/[android-номер_версії]/data/res/layout. Якщо нас не задовольняє стандартна розмітка списку, ми можемо створити свою і потім у коді змінити id на id потрібної нам розмітки

- **countries**: масив даних. Тут необов'язково вказувати саме масив, це може бути перелік ArrayList<T>.

Насамкінець необхідно встановити для ListView адаптер за допомогою методу setAdapter().

В результаті ми отримаємо наступне відображення:



1.2. Ресурс string-array та ListView

У минулій темі було розглянуто, як виводити масив рядків за допомогою ArrayAdapter у ListView. При цьому масив рядків визначався програмно коду java. Однак подібний масив рядків набагато зручніше було б зберігати у файлі xml як ресурс.

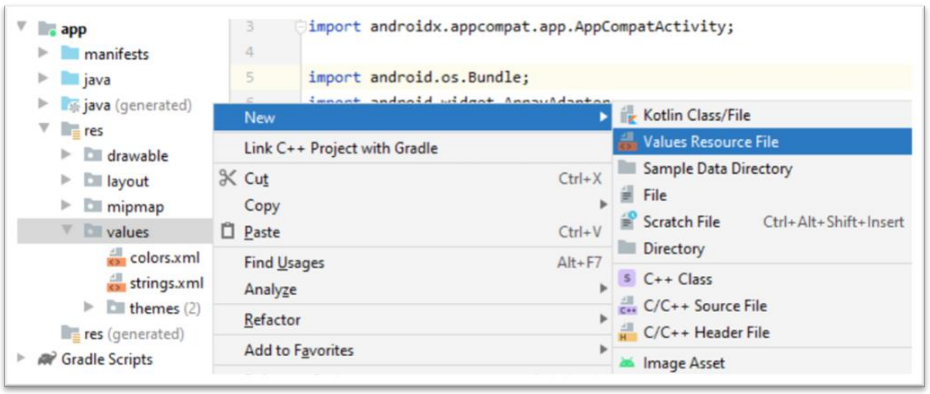
Ресурси масивів рядків є елементом типу string-array. Вони можуть бути розташовані в каталозі res/values в xml-файлі з довільним ім'ям.

Визначення масивів рядків мають наступний синтаксис:

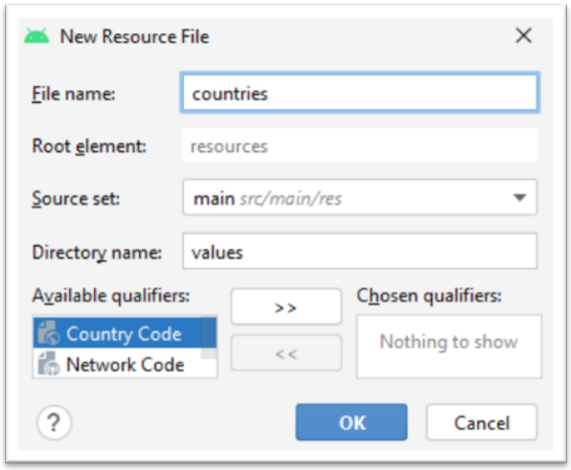
```
<?xml version="1.0" encoding="utf-8"?>
<resources>
<string-array name="ім'я_масиву_рядок">
<item>елемент</item>
</string-array>
</resources>
```

Масив рядків задається за допомогою елемента <string-array>, атрибут name якого може мати довільне значення, яким потім будуть посилатися на цей масив. Усі елементи масиву представляють набір значень <item>

Наприклад, додамо до папки res/values новий файл. Для цього натиснемо правою кнопкою миші на даний каталог і меню виберемо пункт New -> Value Resource file:



У вікні назвемо файл як countries:



Після додавання файлу в папку res/values змінимо його вміст так:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
<string-array name="countries">
<item>Бразилія</item>
<item>Аргентина</item>
<item>Колумбія</item>
<item>Чилі</item>
<item>Уругвай</item>
</string-array>
</resources>
```

У файлі розмітки activity_main.xml залишається визначення елемента 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>
```

Тепер зв'яжемо ресурс і ListView у коді 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);

// отримуємо елемент ListView
ListView countriesList = findViewById(R.id.countriesList);

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

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

// Встановлюємо для списку адаптер
countriesList.setAdapter(adapter);
}
}
```

Для отримання ресурсу в коді java застосовується вираз R.array. Назва_ресурсу.

Але нам необов'язково додавати список рядків до ListView програмно. Цей елемент має атрибут entries, який як значення може приймати ресурс string-array:

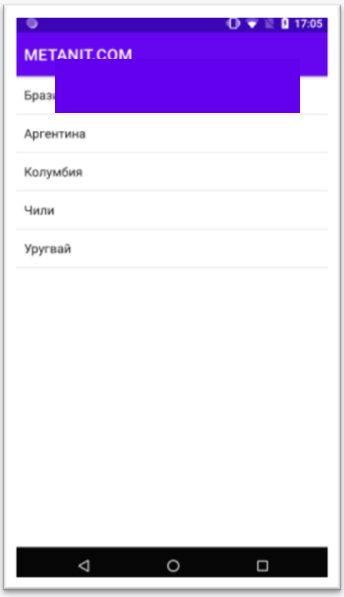
```
<?xml version="1.0" encoding="utf-8"?>
```

```
<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>
```

У цьому випадку код MainActivity ми можемо скоротити 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 буде той самий:



1.3. Вибір елемента у ListView

У попередніх темах було розглянуто, як можна завантажувати дані в ListView, пов'язувати його з джерелом даних. Але крім простого виведення списку елементів ListView дозволяє вибирати елемент та обробляти його вибір. Розглянемо як це зробити. Визначимо наступну розмітку у файлі 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>
```

Тепер зв'яжемо список ListView із джерелом даних та закріпимо за ним слухач натискання на елемент списку:

```
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 = { "Бразилія", "Аргентина", "Колумбія", "Чілі", "Уругвай"};
@Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_main);

// отримуємо елемент TextView
TextView selection = findViewById(R.id.selection);
// отримуємо елемент ListView
ListView countriesList = findViewById(R.id.countriesList);
// створюємо адаптер
ArrayAdapter<String> adapter=new ArrayAdapter(this,
android.R.layout.simple_list_item_1, countries);
// Встановлюємо для списку адаптер
countriesList.setAdapter(adapter);
// додаємо для списку слухач
countriesList.setOnItemClickListener(new AdapterView.OnItemClickListener(){
@Override
public void onItemClick(AdapterView<?> parent, View v, int position, long id)
{
// за позицією отримуємо обраний елемент
String selectedItem = countries[position];
// Встановлення тексту елемента TextView
selection.setText(selectedItem);
}
});
}
}
```

Отже, методsetAdapterпов'язує елемент ListView із певним адаптером. Далі для обробки вибору списку встановлюється слухач OnItemClickListener. Цей слухач має один

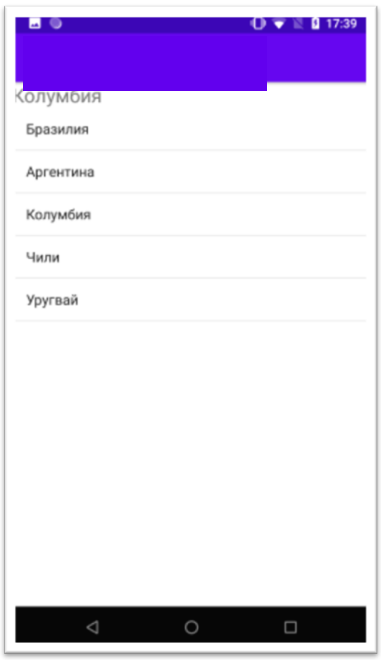
метод onItemClick, через параметри якого ми можемо отримати виділений елемент та супутні дані. Так, він приймає такі параметри:

- **parent**: натиснутий елемент AdapterView (в ролі якого у цьому випадку виступає наш елемент ListView)
- **view**: натиснутий віджет усередині AdapterView
- **position**: індекс натиснутого віджету всередині AdapterView
- **id**: ідентифікатор рядка натиснутого елемента

Використовуючи ці параметри, ми можемо у різний спосіб отримати виділений елемент.

Наприклад, в даному випадку отримуючи індекс натиснутого віджету, який відповідає індексу елемента в масиві рядків, ми можемо встановити відповідний елемент у масиві рядків і таким чином отримати його текст:

```
countriesList.setOnItemClickListener(new  AdapterView.OnItemClickListener(){
@Override
public void onItemClick(AdapterView<?>  parent, View v, int position, long id)
{
// за позицією отримуємо обраний елемент
String selectedItem = countries[position];
// Встановлення тексту елемента TextView
selection.setText(selectedItem);
}
});
```



Також ми можемо отримати виділений елемент з AdapterView, який передається як перший параметр -AdapterView<?> parent. Так, у даному випадку ми знаємо, що кожен елемент у AdapterView фактично представляє рядок або об'єкт String, тому в даному випадку можна отримати виділений елемент так:

```
countriesList.setOnItemClickListener(new  AdapterView.OnItemClickListener(){
@Override
public void onItemClick(AdapterView<?> parent, View v, int position, long id)
{
// отримуємо вибраний елемент
String selectedItem = (String)parent.getItemAtPosition(position);
// Встановлення тексту елемента TextView
selection.setText(selectedItem);
}
});
```

МетодgetItemAtPositionповертає виділений елемент за індексом. Це може бути актуальним, якщо ми використовуємо як джерело даних не масив рядків, створений у коді Java, а, наприклад, ресурс<string-array>, заданий у файлі XML.

По-третє, ми можемо використовувати виділений елемент, який передається як другий параметр -View v. Так, в даному випадку адаптер використовує як тип розмітки ресурс - android.R.layout.simple_list_item_1, а це означає, що виділений елемент представляє елемент TextView, в якому виводиться цей текст. Тому в даному випадку ми також могли б отримати виділений елемент так:

```
countriesList.setOnItemClickListener(new  AdapterView.OnItemClickListener(){
@Override
public void onItemClick(AdapterView<?> parent, View v, int position, long id)
{
// отримуємо вибраний елемент
TextView textView = (TextView) v;
String selectedItem = (String)textView.getText();
// Встановлення тексту елемента TextView
selection.setText(selectedItem);
// або так
// selection.setText(textView.getText());
}
});
```

1.4. Множинний вибір у списку

Іноді потрібно вибрати не один елемент як за замовчуванням, а кілька. Для цього, по-перше, у розмітці списку треба встановити атрибутandroid: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>
```

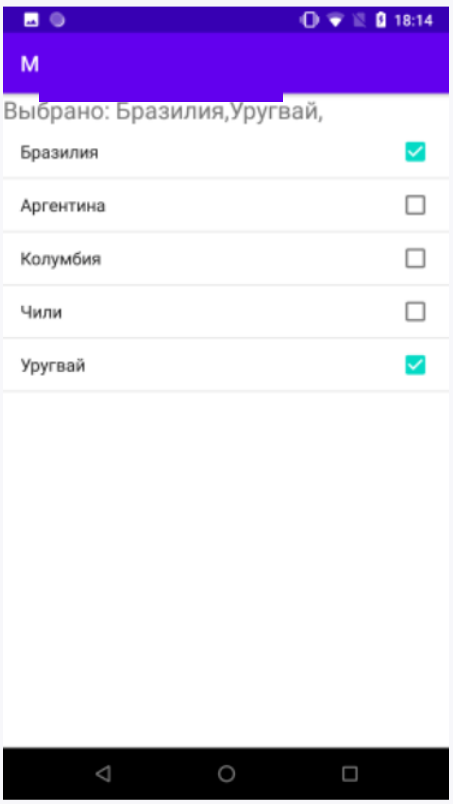
Тепер визначимо у коді MainActivity обробку вибору елементів списку:

```
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 = { "Бразилія", "Аргентина", "Колумбія", "Чілі", "Уругвай"};
@Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_main);

// отримуємо елемент TextView
TextView selection = findViewById(R.id.selection);
// отримуємо елемент ListView
ListView countriesList = findViewById(R.id.countriesList);
// створюємо адаптер
// створюємо адаптер
ArrayAdapter<String> adapter=new ArrayAdapter(this,
android.R.layout.simple_list_item_multiple_choice, countries);
```

```
// Встановлюємо для списку адаптер
countriesList.setAdapter(adapter);
// додаємо для списку слухач
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]+",";
}
// Встановлення тексту елемента TextView
selection.setText("Вибрано: " + SelectedItems);
}
});
}
}
```

Ресурсandroid.R.layout.simple_list_item_multiple_choiceпредставляє стандартну розмітку, яку надає фреймворк, для створення списку з множинним вибором.

А при виборі елементів ми отримуємо всі обрані позиції в об'єкт SparseBooleanArray, потім пробігаємось по всьому масиву, і якщо позиція елемента в масиві є в SparseBooleanArray, тобто вона позначена, то додаємо зазначений елемент у рядок.



1.5. Додавання та видалення в ArrayAdapter та ListView

Після прив'язки ListView до джерела даних через адаптер ми можемо працювати з даними - додавати, видаляти, змінювати лише через адаптер. ListView служить лише для відображення даних.

Для керування даними ми можемо використовувати методи адаптера або безпосередньо джерела даних. Наприклад, клас ArrayAdapter надає такі методи управління даними:

- **add(T object)**: додає елемент object в кінець масиву
- **void addAll(T... items)**: додає всі елементи items в кінець масиву
- **void addAll(Collection<? extends T> collection)**: додає колекцію елементів collection в кінець масиву.
- **clear()**: видаляє всі елементи зі списку
- **void insert(T object, int index)**: додає елемент object у масив за індексом index
- **void remove(T object)**: видаляє елемент object із масиву.

Однак після застосування вищевказаних методів зміни торкнуться лише масиву, що є джерелом даних. Щоб синхронізувати зміни з елементомListView, Треба викликати у адаптера методnotifyDataSetChanged().

Наприклад, визначимо у файлі activity_main.xml такі елементи:

```
<?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 виведення списку призначений ListView із можливістю множинного вибору елементів. Для додавання та видалення визначено дві кнопки. Для введення нового об'єкта до списку призначено поле EditText.

Тепер змінимо клас 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>();
```

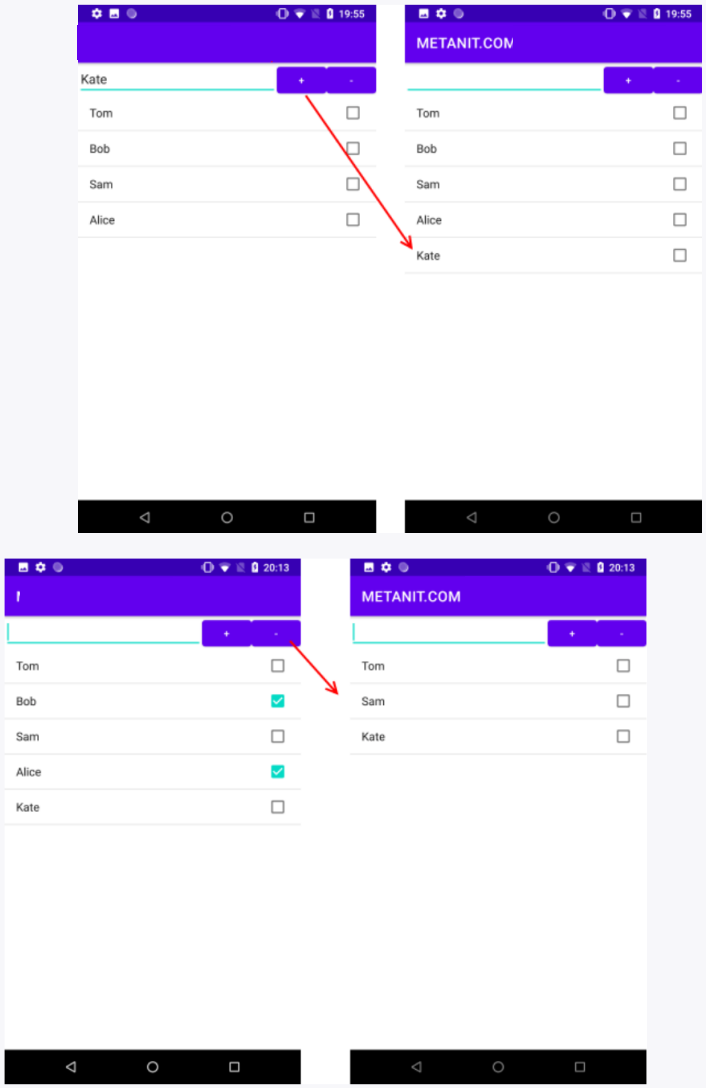
```
ArrayList<String> selectedUsers = new ArrayList<String>();
ArrayAdapter<String> adapter;
ListView usersList;
@Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_main);

// Додаємо початкові елементи
Collections.addAll(users, "Tom", "Bob", "Sam", "Alice");
// отримуємо елемент ListView
usersList = findViewById(R.id.usersList);
// створюємо адаптер
adapter=new ArrayAdapter(this,
android.R.layout.simple_list_item_multiple_choice, users);
// Встановлюємо для списку адаптер
usersList.setAdapter(adapter);

// обробка установки та зняття позначки у списку
usersList.setOnItemClickListener(new AdapterView.OnItemClickListener(){
@Override
public void onItemClick(AdapterView<?> parent, View v, int position, long
 id)
{
// отримуємо натиснутий елемент
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){
// отримуємо та видаляємо виділені елементи
for(int i=0; i< selectedUsers.size();i++){
adapter.remove(selectedUsers.get(i));
}
```

```
        // знімаємо всі раніше встановлені позначки
        usersList.clearChoices();
        // Очищаємо масив вибраних об'єктів
        selectedUsers.clear();

        adapter.notifyDataSetChanged();
        }
        }
```

З додаванням все відносно просто: отримуємо введений рядок та додаємо до списку за допомогою методу adapter.add(). Щоб оновити ListView після додавання, викликається метод adapter.notifyDataSetChanged().

А для видалення створюється додатковий список selectedUsers, який міститиме виділені елементи. Для отримання виділених елементів та додавання їх до списку використовується слухач AdapterView.OnItemClickListener, методonItemClick()якого викликається під час встановлення або зняття позначки з елемента, тобто при будь-якому натисканні на елемент.

По натисканні на кнопку видалення пробігаємося по списку виділених елементів і викликаємо для кожного метод adapter.remove().

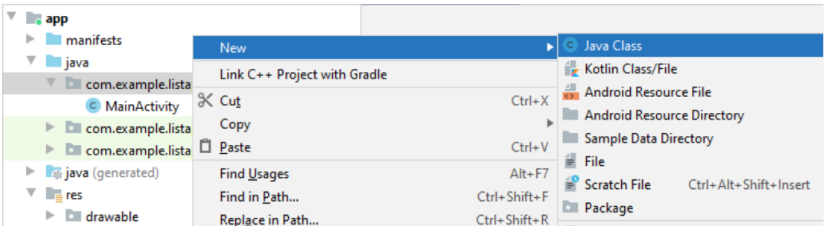


1.6. Розширення списків та створення адаптера

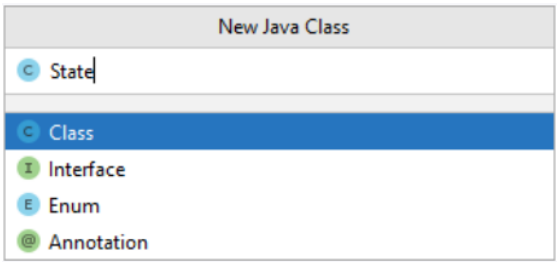
Традиційні списки ListView, що використовують стандартні адаптери ArrayAdapter, чудово працюють із масивами рядків. Однак частіше ми стикатимемося з складнішими за структурою списками, де один елемент представляє не один рядок, а кілька рядків, картинок та інших компонентів.

Для створення складного списку нам треба перевизначити один із використовуваних адаптерів. Оскільки, зазвичай, використовується ArrayAdapter, саме його ми й перевизначимо.

Але спочатку визначимо модель, дані якої відображатимуться у списку. Для цього додамо до того ж каталогу, де знаходиться клас MainActivity, новий клас. Для цього натиснемо на даний каталог правою кнопкою миші та в меню виберемо New -> Java Class:



У вікні вкажемо для класу, що додається, ім'я State



Після додавання змінимо клас State таким чином:

```
package com.example.listapp;
public class State {
private String name; // Назва
private String capital; // столиця
private int flagResource; // ресурс прапора
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;
}
}
```

Цей клас зберігає два рядкові поля - назву держави та її столицю, а також числове поле, яке вказуватиме на ресурс зображення з папки drawable, яке відображатиме прапор держави.Далі додамо до папки res/layout новий файл list_item.xml, який представлятиме розмітку одного елемента у списку:

```
<?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="Назва"
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="Столиця"
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>
```

Кожен елемент матиме зображення у вигляді ImageView та два компоненти TextView для відображення назви та столиці держави.

Після цього додамо в каталог, де знаходяться класи MainActivity і State, новий клас, який назвемо 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 ПозиціяInflater 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;
}
}
```

Вся взаємодія зі списком тут йтиме через клас StateAdapter. У конструкторі StateAdapter нам треба передати до конструктора базового класу три параметри:

- контекст, у якому використовується клас. У його ролі як правило виступає клас Activity
- ресурс розмітки інтерфейсу, який використовуватиметься для створення одного елемента в ListView
- набір об'єктів, які будуть виводитись у ListView

У конструкторі StateAdapter ми отримуємо ресурс розмітки та набір об'єкта та зберігаємо їх в окремі змінні. Крім того, для створення об'єкта View за отриманим ресурсом розмітки буде потрібно об'єкт LayoutInflater, який також зберігається в змінну.

У методіgetView()встановлюється відображення списку. Цей метод приймає три параметри:

- position: передає позицію елемента всередині адаптера, для якого створюється уявлення
- convertView: старе уявлення елемента, яке за наявності використовується ListView з метою оптимізації
- parent: батьківський компонент для представлення елемента

В даному випадку за допомогою об'єкта LayoutInflater створюємо об'єкт View для кожного окремого елемента у списку:

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

Зі створеного об'єкта View отримуємо елементи ImageView та TextView за 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);
```

Це елементи, визначені у файлі list_item.xml. Тут ми їх отримуємо.

Далі використовуючи параметр position, отримуємо об'єкт State, для якого створюється розмітка:

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

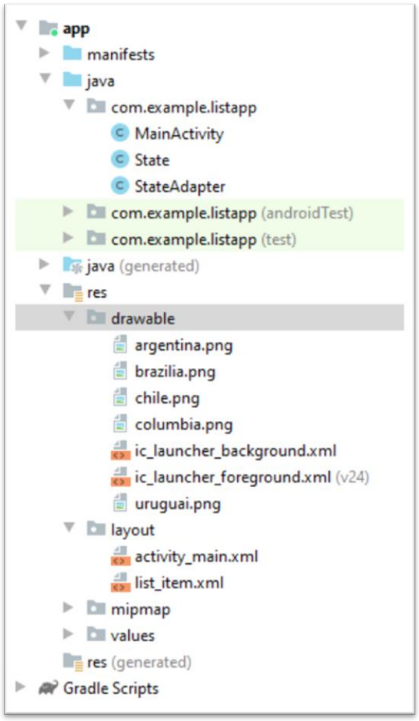
Потім отримані елементи ImageView та TextView наповнюємо з отриманого за позицією об'єкта State:

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

І наприкінці створений для відображення об'єкта State елемент View повертається з методу:

```
return view;
```

Для використання зображень додамо до папки res/drawable кілька зображень, у моєму випадку це п'ять зображень прапорів країн. У результаті проект виглядатиме так:



У файлі activity_main.xml визначимо ListView, в який будуть завантажуватись дані:

```
<?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 у файлі MainActivity з'єднаємо StateAdapter з ListView:

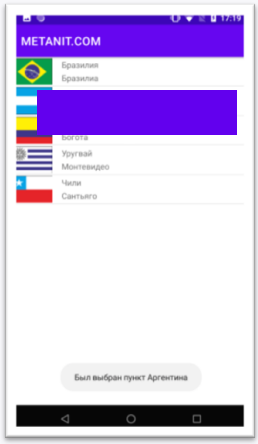
```
package com.example.listapp;

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);
        // Початкова ініціалізація списку
        setInitialData();
        // отримуємо елемент ListView
        countriesList = findViewById(R.id.countriesList);
        // створюємо адаптер
        StateAdapter stateAdapter = New StateAdapter(this, R.layout.list_item, states);
        // Встановлюємо адаптер
        countriesList.setAdapter(stateAdapter);
        // слухач вибору списку
        AdapterView.OnItemClickListener itemListener=new
AdapterView.OnItemClickListener() {
        @Override
        public void onItemClick(AdapterView<?> parent, View v, int position, long id) {
        // отримуємо вибраний пункт
        State selectedState = (State)parent.getItemAtPosition(position);
        Toast.makeText(getApplicationContext(), "Було обрано пункт " +
selectedState.getName(),
        Toast.LENGTH_SHORT).show();
        }
        };
        countriesList.setOnItemClickListener(itemListener);
        }
        private void setInitialData(){
        states.add(new State ("Бразилія", "Бразилія", R.drawable.brazilia));
        states.add(new State ("Аргентина", "Буенос-Айрес", R.drawable.argentina));
        states.add(new State ("Колумбія", "Богота", R.drawable.columbia));
        states.add(new State ("Уругвай", "Монтевідео", R.drawable.uruguai));
        states.add(new State ("Чилі", "Сантьяго", R.drawable.chile));
        }
        }
```

Як джерело даних тут виступає клас ArrayList, який отримує дані у методі setInitialData. Кожному об'єкту State, що додається, у списку передається назва держави, його столиця і ресурс зображення з папки res/drawable, який представляє прапор держави.

При створенні адаптера йому передається раніше створений ресурс розмітки list_item.xml та список states:

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



1.7. **Оптимізація адаптера та View Holder**

Минулої теми було створено кастомний адаптер, який дозволяв працювати зі складними списками об'єктів:

```
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 ПозиціяInflater 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;
}
}
```

Але цей адаптер має один дуже великий мінус - при прокручуванні в ListView, якщо в списку дуже багато об'єктів, то для кожного елемента, коли він потрапить у зону видимості, повторно викликатиметься метод getView, в якому буде створюватися новий об'єкт View. Відповідно буде збільшуватися споживання пам'яті та знижуватись продуктивність. Тому оптимізуємо код методу 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;
}
```

ПараметрconvertViewвказує на елемент View, який використовується для об'єкта у списку за позицією position. Якщо раніше вже створювався View для цього об'єкта, параметр convertView вже містить деяке значення, яке ми можемо використовувати.

У цьому випадку ми повторно використовуватимемо вже створені об'єкти і збільшимо продуктивність, однак цей код можна ще більше оптимізувати. Справа в тому, що одержання елементів з id теж щодо витратна операція. Тому далі оптимізуємо код StateAdapter, змінивши його так:

```
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 ПозиціяInflater 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 зберігання посилань на елементи ImageView і TextView визначено внутрішній приватний клас ViewHolder, який у конструкторі отримує об'єкт View, що містить ImageView і TextView.

У методіgetViewякщо convertView дорівнює null (тобто якщо раніше для об'єкта не створено розмітку) створюємо об'єкт ViewHolder, який зберігаємо в тег у convertView:

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

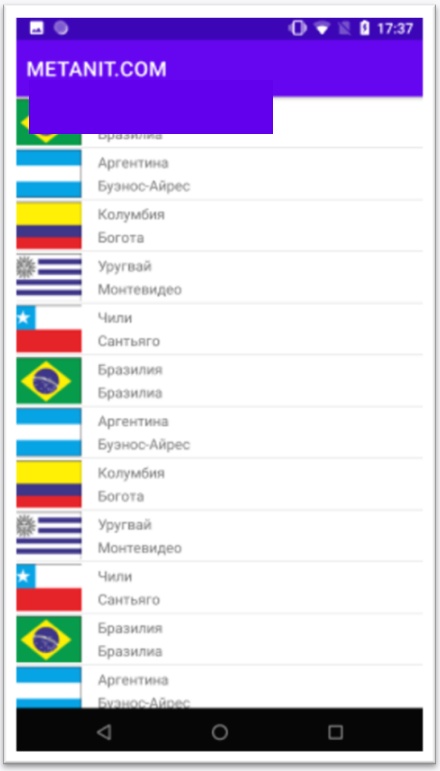
Якщо розмітка для об'єкта в ListView вже раніше була створена, то назад отримуємо ViewHolder з тега:

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

Потім також для ImageView і TextView в ViewHolder встановлюються значення об'єкта State:

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

І тепер ListView особливо при великих списках працюватиме плавніше і продуктивніше, ніж у минулій темі:



1.8. Складний список із кнопками

Раніше було розглянуто кастомні адаптери, які дозволяють виводити у списки складні дані. Тепер підемо далі та розглянемо, як ми можемо додати до списків інші елементи, наприклад, кнопки, та обробляти їхні події.

Для цього спочатку визначимо наступний клас:

```
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;
}
}
```

Цей клас зберігає назву, кількість продукту, а також одиницю виміру. І об'єкти цього класи виводитимемо до списку.

Для цього до папки res/layout додамо новий 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>
```

Тут визначено два текстові поля для виведення назви та кількості продукту та дві кнопки для додавання та видалення однієї одиниці продукту.

Тепер додамо клас адаптера, який назвемо 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 ПозиціяInflater 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 кожної кнопки тут визначений обробник натискання, в якому ми зменшуємо, або збільшуємо кількість продукту на одиницю і потім встановлюємо текст у відповідному текстовому полі.

Далі у файлі activity_main.xml визначимо елемент ListView:

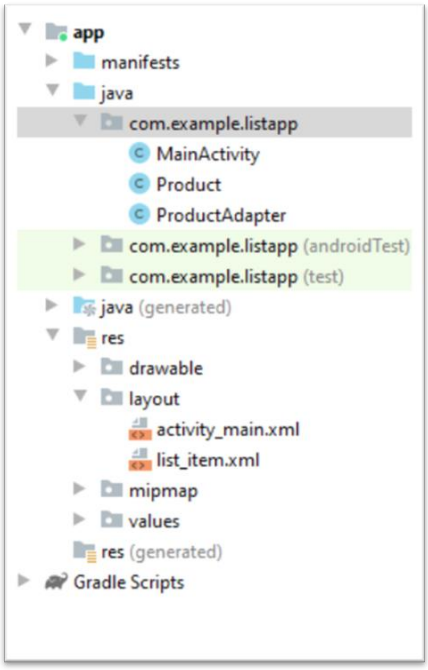
```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:app="http://schemas.android.com/apk/res-auto"
android:layout_width="match_parent"
android:layout_height="match_parent">
<ListView
android:id="@+id/productList"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

І змінимо клас 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("Картопля", "кг."));
products.add(new Product("Чай", "шт."));
products.add(new Product("Яйця", "шт."));
products.add(new Product("Молоко", "Л."));
products.add(new Product("Макарони", "кг."));
}
ListView productList = findViewById(R.id.productList);
ProductAdapter adapter = новий ProductAdapter(this, R.layout.list_item, products);
productList.setAdapter(adapter);
}
}
```

В результаті вийде наступний проект (рис. 1)



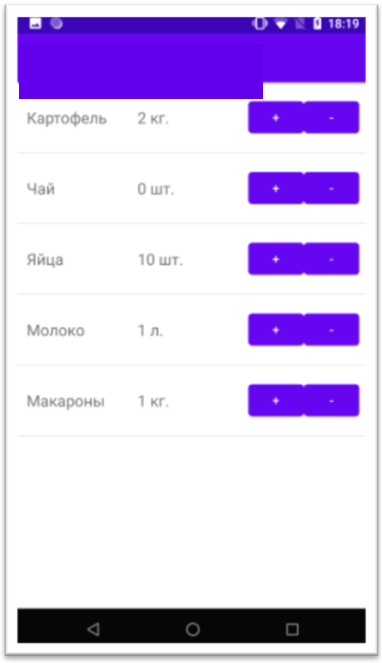


Рис.1 Рис.2

І після запуску програми ми зможемо керувати кількістю продуктів через кнопки (Мал.2)

1.9. ListActivity

Для спрощення доступу до елементів списку використається клас ListActivity. ListActivity є клас, успадкований від Activity і розроблений спеціально для роботи зі списками.

Отже, подивимося з прикладу. По-перше, визначимо у файлі розмітки activity_main.xml елемент 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="@android:id/list"
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>
```

Зверніть увагу на оголошення ідентифікатора ListView:android:id="@android:id/list". Подібне оголошення обов'язково, щоб ListActivity розпізнала список та могла б ним керувати.

Крім ListView у файлі розмітки інтерфейсу також можуть бути інші елементи. Але в цьому випадку обмежимося лише елементом ListView.

Далі змінимо код класу MainActivity:

```
package com.example.listapp;

import android.app.ListActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.Toast;
```

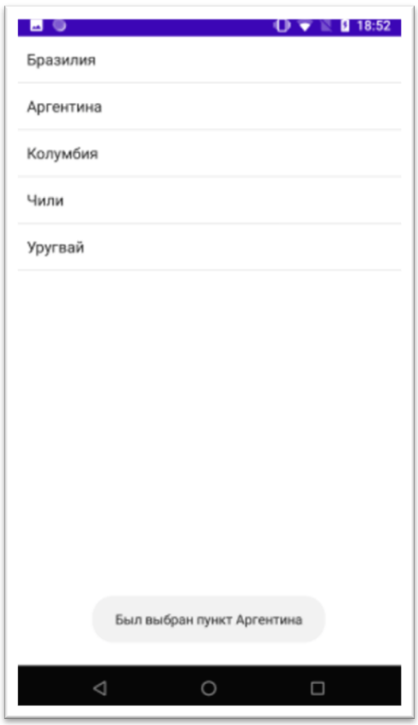
```
public class MainActivity extends ListActivity {

String[] countries = { "Бразилія", "Аргентина", "Колумбія", "Чілі", "Уругвай"};

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

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

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

Toast.makeText(getApplicationContext(), "Було обрано пункт " +
parent.getItemAtPosition(position).toString(), Toast.LENGTH_SHORT).show();
}
};
getListView().setOnItemClickListener(itemListener);
}
}
```

Тепер клас MainActivity розширює базовий клас ListActivity.

Тут, як і у випадку з ListView, ми створюємо адаптер ArrayAdapter, тільки встановлюємо його через методsetListAdapter, який визначено у ListActivity.

Далі створюється об'єкт слухача OnItemClickListener, який оброблятиме вибір елементів списку. Його єдиний методonItemClickаналогічний тому, що розбирався в минулому розділі, за винятком, що тут ми виводимо повідомлення з текстом вибраного елемента.

Наприкінці ми використовуємо методgetListView(), який повертає об'єкт ListView І потім встановлюємо для нього вищезазначений слухач.



1.10. Випадаючий перелік Spinner

Spinnerявляє собою список, що випадає. Визначимо у файлі розмітки activity_main.xml елемент 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>
```

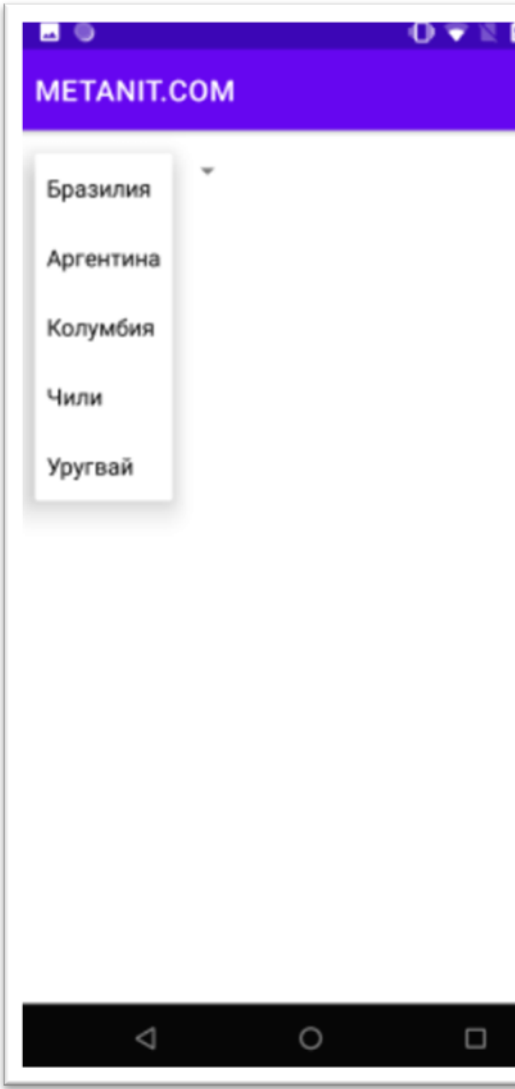
Як джерело даних, як і для ListView, для Spinner може бути простий список або масив, створений програмно, або ресурсstring-array. Взаємодія з джерелом даних також йтиме через адаптер. В даному випадку визначимо джерело програмно у вигляді масиву в коді MainActivity:

```
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 = { "Бразилія", "Аргентина", "Колумбія", "Чілі", "Уругвай"};
        @Override
        protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Spinner spinner = findViewById(R.id.spinner);
        // Створюємо адаптер ArrayAdapter за допомогою масиву рядків та стандартної розмітки елемента spinner
        ArrayAdapter<String> adapter=new ArrayAdapter(this, android.R.layout.simple_spinner_item, countries);
        // Визначаємо розмітку для використання під час вибору елемента
        adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
        // Застосовуємо адаптер до елемента spinner
        spinner.setAdapter(adapter);
        }
        }
```

Використовуваний при створенні ArrayAdapter ресурс android.R.layout.simple_spinner_item надається платформою і є стандартною розміткою для створення списку, що випадає.

За допомогою методуadapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item)встановлюються додаткові візуальні можливості списку. А ресурс android.R.layout.simple_spinner_dropdown_item, що передається в метод, використовується для візуалізації випадаючого списку і також надається платформою.



1.11.Обробка вибору елемента

Використовуючи слухач OnItemSelectedListener, зокрема його методonItemSelected(), ми можемо обробляти вибір елемента зі списку. Спочатку додамо в розмітку інтерфейсу текстове поле, яке виводитиме вибраний елемент:

```
<?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>
```

І змінимо код MainActivity, визначивши для елемента Spinner слухач 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 = { "Бразилія", "Аргентина", "Колумбія", "Чілі", "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);
// Створюємо адаптер ArrayAdapter за допомогою масиву рядків та стандартної
```

```
розмітки елемента spinner
        ArrayAdapter<String>  adapter=new  ArrayAdapter(this,
android.R.layout.simple_spinner_item,  countries);
        // Визначаємо розмітку для використання під час вибору елемента
        adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item
);
        // Застосовуємо адаптер до елемента spinner
        spinner.setAdapter(adapter);

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

        // Отримуємо обраний об'єкт
        String item = (String)parent.getItemAtPosition(position);
        selection.setText(item);
        }
        @Override
        public void onNothingSelected(AdapterView<?>  parent) {
        }
        };
        spinner.setOnItemSelectedListener(itemSelectedListener);
        }
        }
```

МетодonItemSelectedслухача OnItemSelectedListener отримує чотири параметри:

- parent: об'єкт Spinner, в якому сталася подія вибору елемента
- view: об'єкт View всередині Spinnera, який представляє обраний елемент
- position: індекс вибраного елемента в адаптері
- id: ідентифікатор рядка елемента, який був обраний

Отримавши позицію вибраного елемента, ми можемо знайти його у списку:

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

Для встановлення слухача OnItemSelectedListener у класі Spinner застосовується метод setOnItemSelectedListener.

Віджет автодоповнення AutoCompleteTextView

AutoCompleteTextViewпредставляє елемент, створений на основі класуEditTextта володіє можливістю автодоповнення

По-перше, оголосимо в ресурсі розмітці цей елемент:

```
<?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="Введіть місто"
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 дозволяє задати напис, який відображається внизу списку, а властивість android:completionThreshold встановлює, скільки символів треба ввести, щоб почати працювати автодоповнення. Тобто в даному випадку після введення одного символу повинен з'явитися список з підстановками.

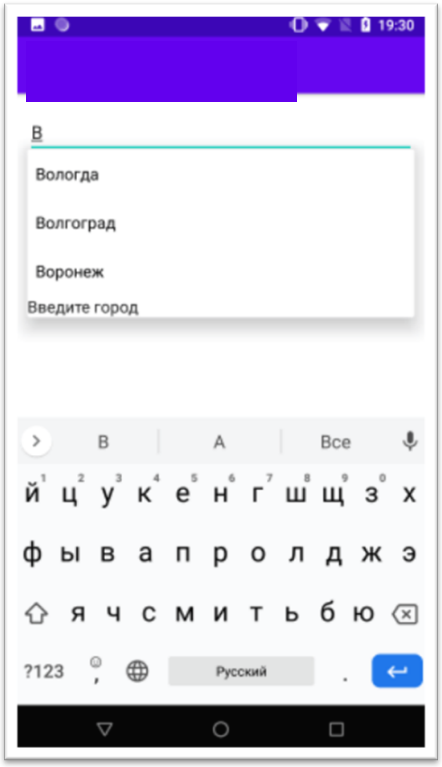
Як і у випадку з елементами ListView та Spinner, AutoCompleteTextView підключається до джерела даних через адаптер. Джерелом даних знову ж таки може бути масив або список об'єктів, або ресурс string-array.

Тепер підключимо до віджету масив рядків у класі 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", "Вороніж"};
@Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_main);
// Отримуємо посилання на елемент AutoCompleteTextView у розмітці
AutoCompleteTextView autoCompleteTextView = findViewById(R.id.autocomplete);
// Створюємо адаптер для автозаповнення елемента AutoCompleteTextView
ArrayAdapter<String> adapter=new ArrayAdapter (this, R.layout.support_simple_spinner_dropdown_item, cities);
autoCompleteTextView.setAdapter(adapter);
}
```

```
}
```

Після введення в текстове поле однієї літери з'явиться список з варіантами доповнення, де можна вибрати кращий:



1.12. **MultiAutoCompleteTextView**

Цей віджет доповнює функціональність елемента AutoCompleteTextView. MultiAutoCompleteTextView дозволяє використовувати автодоповнення не тільки для одного рядка, але й для окремих слів. Наприклад, якщо вводиться слово і після нього ставиться кома, то автозаповнення все одно буде працювати для слів, що вводяться після коми або іншого роздільника.

MultiAutoCompleteTextView має таку форму оголошення, як і AutoCompleteTextView:

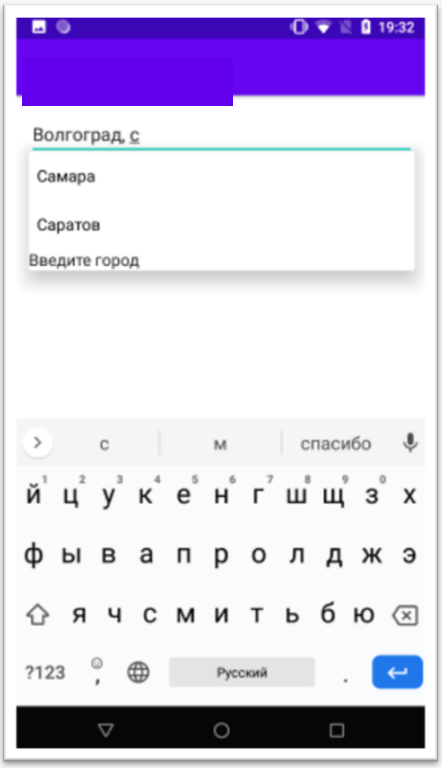
```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:app="http://schemas.android.com/apk/res-auto"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:padding="16dp">
<MultiAutoCompleteTextView
android:id="@+id/autocomplete"
android:layout_width="0dp"
android:layout_height="wrap_content"
android:completionHint="Введіть місто"
android:completionThreshold="1"
app:layout_constraintLeft_toLeftOf="parent"
app:layout_constraintRight_toRightOf="parent"
```

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

Щоб увімкнути MultiAutoCompleteTextView у коді, потрібно встановити токен роздільника:

```
        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",
"Вороніж"};
        @Override
        protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        // Отримуємо посилання на елемент AutoCompleteTextView у розмітці
        MultiAutoCompleteTextView autoCompleteTextView =
findViewById(R.id.autocomplete);
        // Створюємо адаптер для автозаповнення елемента MultiAutoCompleteTextView
        ArrayAdapter<String> adapter=new ArrayAdapter(this,
R.layout.support_simple_spinner_dropdown_item, cities);
        autoCompleteTextView.setAdapter(adapter);
        // Установка коми як роздільник
        autoCompleteTextView.setTokenizer(new
MultiAutoCompleteTextView.CommaTokenizer());
        }
        }
```

Тут як роздільник використовується вбудований роздільник на основі коми CommaTokenizer(). За потреби ми можемо створити свої роздільники.



1.13.**GridView**

Елемент GridView представляє відображення у вигляді таблиці – набору рядків та стовпців.

Основні атрибути GridView:

- **android:columnWidth**: встановлює фіксовану ширину стовпців.
- **android:gravity**: встановлює вирівнювання вмісту всередині кожного осередку
- **android:horizontalSpacing**: встановлює горизональні відступи між стовпцями
- **android:numColumns**: встановлює кількість стовпців
- **android:stretchMode**: встановлює, як стовпці будуть розтягуватися та займати простір контейнера Може приймати такі значення:
    - columnWidth: кожен стовпець рівномірно розтягується по всій ширині. Еквівалентно до значення2
    - none: стовпці не розтягуються Еквівалентно до значення0
    - spacingWidth: між стовпцями утворюються відступи Еквівалентно до значення1
    - spacingWidthUniform: між стовпцями утворюються рівномірні відступи Еквівалентно до значення3
- **android:verticalSpacing**: встановлює вертикальні відступи між рядками.

Основні методи класу GridView:

- **int getColumnWidth()**: повертає ширину стовпців
- **int getHorizontalSpacing()**: повертає розмір горизонтального відступу
- **int getNumColumns()**: повертає кількість стовпців
- **int getStretchMode()**: повертає режим розтягування простору всередині гриду
- **int getVerticalSpacing()**: повертає розмір вертикального відступу
- **void setAdapter(ListAdapter adapter)**: встановлює адаптер для підключення до джерела даних
- **void setColumnWidth(int columnWidth)**: встановлює ширину стовпців.
- **void setHorizontalSpacing(int horizontalSpacing)**: встановлює розмір горизонтального відступу
- **void setNumColumns(int numColumns)**: встановлює кількість стовпців
- **void setStretchMode(int stretchMode)**: встановлює режим розтягування простору всередині гриду.
- **void setVerticalSpacing(int verticalSpacing)**: встановлює розмір вертикального відступу
- **void setSelection(int position)**: встановлює поточний виділений елемент

Визначимо GridView у 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 даному випадку вказуємо, що грид матиме 2 стовпці, які розтягуються рівномірно по всій ширині гриду, а між осередками будуть відступи по горизонталі та вертикалі 16 dp.

Тепер, як і у випадку з ListView, треба встановити зв'язок із 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 = { "Бразилія", "Аргентина", "Чілі", "Колумбія", "Уругвай"};
@Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_main);
// отримуємо елемент GridView
GridView countriesList = findViewById(R.id.gridview);
// створюємо 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(),"Ви вибрали"
+ parent.getItemAtPosition(position).toString(),
Toast.LENGTH_SHORT).show();
}
};
countriesList.setOnItemClickListener(itemListener);
}
}
```

Для обробки натискання GridView застосовується слухач AdapterView.OnItemClickListener.

1.14.**RecyclerView**

Елемент RecyclerView призначений для оптимізації роботи зі списками та багато в чому дозволяє підвищити продуктивність у порівнянні зі стандартним ListView.

Для представлення даних додамо в проект у ту саму папку, де розташований клас MainActivity, новий клас Java, який назвемо State:

```
package com.example.listapp;
public class State {
private String name; // Назва
private String capital; // столиця
private int flagResource; // ресурс прапора
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;
}
}
```

Клас State містить поля для зберігання назви та столиці країни, а також посилання на ресурс зображення прапора країни. У цьому випадку передбачається, що в папці res/drawable будуть розміщуватися файли зображень прапорів для країн, що використовуються.

Допустимо, ми хочемо вивести список об'єктів State за допомогою RecyclerView. Для цього додамо до папки res/layout новий 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="Назва"
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="Столиця"
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>
```

Цей файл визначає розмітку для виведення одного об'єкта State.

Як і у випадку з ListView, для виведення складних об'єктів у RecyclerView необхідно визначити свій адаптер. Тому додамо в ту саму папку, де розташований клас MainActivity і State, новий клас Java, який назвемо 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 ПозиціяInflater 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, який використовується в RecyclerView, повинен успадковуватись від абстрактного класу RecyclerView.Adapter. Цей клас визначає три методи:

- **onCreateViewHolder**: повертає об'єкт ViewHolder, який зберігатиме дані по одному об'єкту State.
- **onBindViewHolder**: виконує прив'язку об'єкта ViewHolder до об'єкта State за певною позицією.
- **getItemCount**: повертає кількість об'єктів у списку

Для зберігання даних у класі адаптера визначено статичний клас ViewHolder, який використовує елементи керування, визначені в list_item.xml.

Тепер визначимо у файлі activity_main.xml елемент 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>
```

Слід враховувати, що RecyclerView розташований у пакеті androidx.recyclerview.widget і є частиною тулкіта Android Jetpack, тому при використанні віджету вказується повна назва з урахуванням пакета (в принципі як і для ConstraintLayout):

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

Для RecyclerView слід встановити атрибут app:layoutManager, який вказує тип менеджера компоновки. Менеджер компонування представляє об'єкт, представлений класом LayoutManager. За замовчуванням бібліотека RecyclerView надає три реалізації цього менеджера:

- **LinearLayoutManager**: упорядковує елементи у вигляді списку з однією колонкою
- **GridLayoutManager**: упорядковує елементи у вигляді грида зі стовпцями та рядками Грід може впорядковувати елементи по горизонталі (горизонтальний грід) або по вертикалі (вертикальний грід)
- **StaggeredGridLayoutManager**: аналогічний GridLayoutManager, однак не вимагає установки для кожного елемента в рядку мали ту саму висоту (для вертикального грида) і ту саму широку (для горизонтального грида)

В даному випадку за допомогою значення androidx.recyclerview.widget.LinearLayoutManager вказуємо, що елементи будуть розміщені у вигляді простого списку. Зверніть увагу, що клас LinearLayoutManager також розташований у бібліотеці RecyclerView і тому при вказівці значення вказується повна назва класу з ім'ям пакета.

І насамкінець змінимо клас 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);
// Початкова ініціалізація списку
setInitialData();
RecyclerView recyclerView = findViewById(R.id.list);
// створюємо адаптер
StateAdapter adapter = новий StateAdapter(this, states);
// Встановлюємо для списку адаптер
recyclerView.setAdapter(adapter);
}
private void setInitialData(){
states.add(new State ("Бразилія", "Бразилія", R.drawable.brazilia));
states.add(new State ("Аргентина", "Буенос-Айрес", R.drawable.argentina));
states.add(new State ("Колумбія", "Богота", R.drawable.columbia));
states.add(new State ("Уругвай", "Монтевідео", R.drawable.uruguai));
states.add(new State ("Чилі", "Сантьяго", R.drawable.chile));
}
}
```

За допомогою методуsetInitialData()встановлюється набір початкових даних. В даному випадку мається на увазі, що в папці res/drawables міститься ряд ресурсів зображень для об'єктів State.

Як і у випадку з виведенням списку через ListView, тут спочатку отримуємо елемент RecyclerView, створюємо адаптер і встановлюємо адаптер для RecyclerView.

Весь проект буде виглядати так (рис.1).

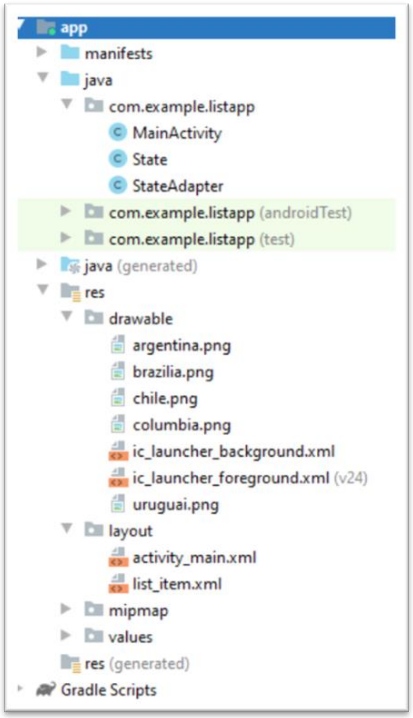


Рис.1

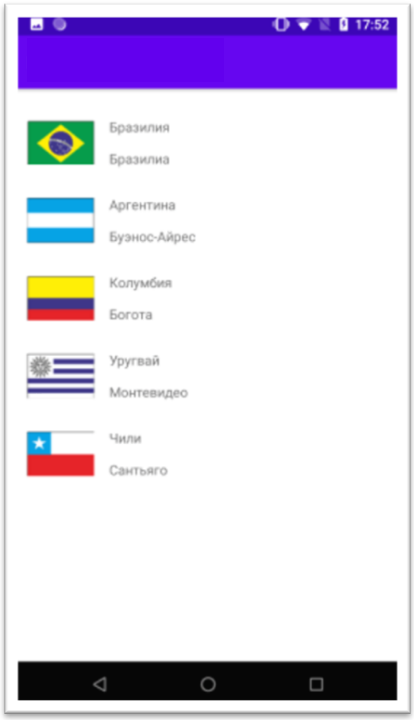
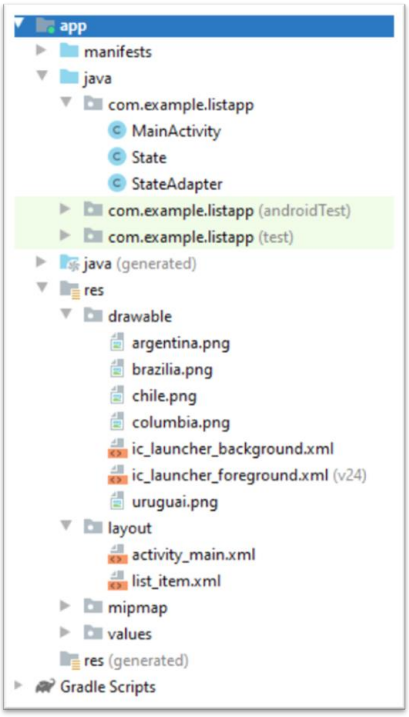


Рис.2

В результаті RecyclerView виведе набір об'єктів (рис.2)

1.15.Обробка вибору елемента у RecyclerView

При роботі з віджетом RecyclerView неминуче постає питання, а як обробити вибір елемента RecyclerView. І тут слід зазначити, що на відміну від інших типів віджетів для роботи зі списками RecyclerView за замовчуванням не надає якогось спеціального методу, за допомогою якого можна визначити слухач натискання на елемент у списку. Тому всю інфраструктуру необхідно визначати самому розробнику. Але на щастя насправді все не так складно. Наприклад візьмемо проект з минулої теми:



Отже, для представлення даних у проекті є клас State, який представляє державу:

```
package com.example.listapp;

public class State {

private String name; // Назва
private String capital; // столиця
private int flagResource; // ресурс прапора

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;
}
}
```

Клас State містить поля для зберігання назви та столиці країни, а також посилання на ресурс зображення прапора країни. У цьому випадку передбачається, що в папці res/drawable будуть розміщуватися файли зображень прапорів для країн, що використовуються.

У папці res/layout для виведення одного об'єкта State у списку визначено наступний 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="Назва"
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="Столиця"
        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>
```

Тепер перейдемо до класу StateAdapter і в такий спосіб визначимо його 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 ПозиціяInflater 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);
}
}
}
```

Тут я зупинюся на тих моментах, які були додані в порівнянні з кодом попередньої статті.

Насамперед нам треба визначити інтерфейс слухача події натискання. Для цього в класі StateAdapter визначено інтерфейс:

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

Інтерфейс визначає один методonStateClick(), який, як передбачається, буде викликатись при виборі об'єкта State і який отримуватиме обраний об'єкт State та його позицію у списку.

Наступний момент - визначення в класі адаптера змінної для зберігання об'єкта цього інтерфейсу та отримання для неї значення в конструкторі:

```
private final OnStateClickListener  onClickListener;

StateAdapter(Context  context,  List<State>  states,  OnStateClickListener
```

```
onClickListener) {

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

Таким чином, поза кодом адаптера ми можемо встановити будь-який об'єкт слухача та передати його до адаптера.

І третій момент - виклик методу слухача при натисканні на елемент у списку методу 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());

        // обробка натискання
        holder.itemView.setOnClickListener(new  View.OnClickListener(){
        @Override
        public void onClick(View v)
        {
        // Викликаємо метод слухача, передаючи йому дані
        onClickListener.onStateClick(state,  position);
        }
        });
        }
```

Клас ViewHolder має поле itemView, яке представляє інтерфейс для одного об'єкта у списку та фактично об'єкт View. А цей об'єкт має методsetOnClickListener(), через який можна підключити стандартний слухач натискання OnClickListener і в його методіonClick()викликати метод нашого інтерфейсу, передавши йому необхідні дані - вибраний об'єкт State та його позицію у списку.

Може виникнути питання, а чому б тут і не обробити натискання на елемент? Навіщо створювати додатковий інтерфейс, його змінну і викликати його метод? Звичайно, ми можемо спробувати тут обробити натискання, але це не є хорошою або поширеною практикою, оскільки, можливо, ми захочемо визначити обробку натискання в класі MainActivity виходячи з того, коду, який там визначений (або з якогось іншого місця ззовні) .

У файлі activity_main.xml залишається той самий візуальний інтерфейс:

```
        <?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>
```

І насамкінець змінимо клас 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);
        // Початкова ініціалізація списку
        setInitialData();
        RecyclerView recyclerView = findViewById(R.id.list);
        // визначаємо слухача натискання елемента у списку
        StateAdapter.OnStateClickListener stateClickListener = new
StateAdapter.OnStateClickListener() {
        @Override
        public void onStateClick(State state, int position) {

        Toast.makeText(getApplicationContext(), "Було обрано пункт " + state.getName(),
        Toast.LENGTH_SHORT).show();
        }
        };
        // створюємо адаптер
        StateAdapter adapter = новий StateAdapter(this, states, stateClickListener);
        // Встановлюємо для списку адаптер
        recyclerView.setAdapter(adapter);
        }
        private void setInitialData(){

        states.add(new State ("Бразилія", "Бразилія", R.drawable.brazilia));
```

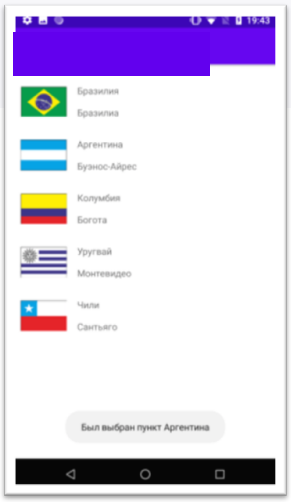
```
        states.add(new State ("Аргентина", "Буенос-Айрес", R.drawable.argentina));
        states.add(new State ("Колумбія", "Богота", R.drawable.columbia));
        states.add(new State ("Уругвай", "Монтевідео", R.drawable.uruguai));
        states.add(new State ("Чилі", "Сантьяго", R.drawable.chile));
        }
        }
```

При створенні адаптера йому передається певний у класі MainActivity слухач:

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

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

Тут просто виводиться спливаюче повідомлення

про вибраний елемент списку.



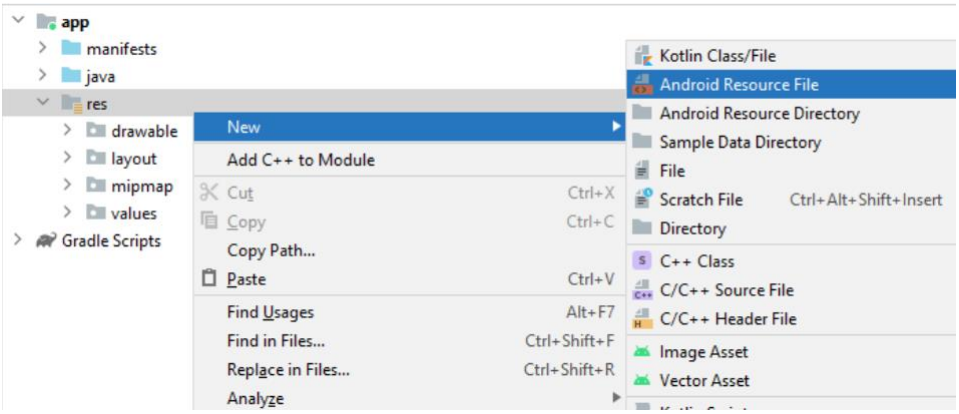
2. МЕНЮ

2.1. Створення меню

Меню в додатках представляє клас android.view.Menu, і кожна діяльність асоціюється з об'єктом цього типу. Об'єкт android.view.Menu може включати різну кількість елементів, а ті можуть зберігати поделементи.

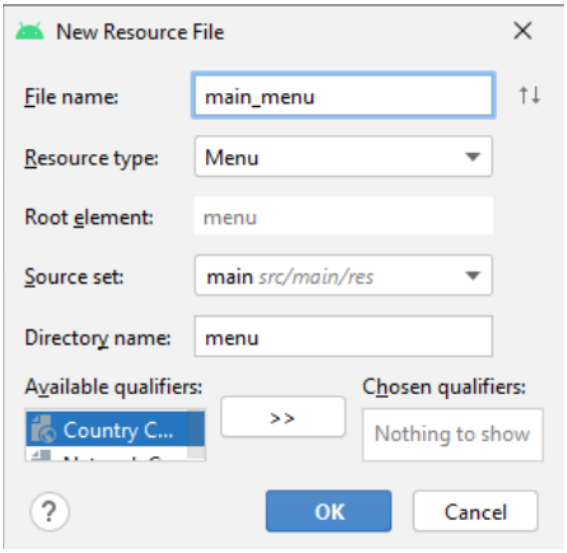
2.2. Визначення меню в xml

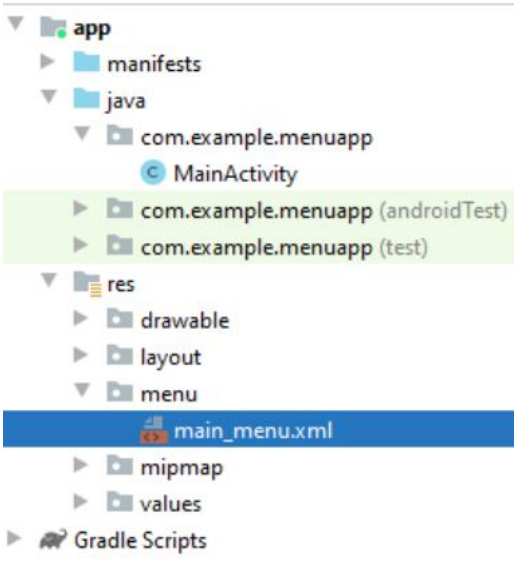
Меню, як і файли інтерфейсу або зображень, також є ресурсом. Однак за умови створення нового проекту з Empty Activity за замовчуванням немає жодних ресурсів меню, тому за необхідності їх потрібно додавати вручну. Так, для визначення ресурсів меню в проекті натиснемо правою кнопкою миші в проекті на каталог res і далі в списку виберемо пункт New -> Android Resource File:



Далі у вікні вкажемо для імені файлу назву main_menu, а для поля Resource Type (тип ресурсу) виберемо Menu:

Після цього в каталозі res буде створено підкаталог menu, в якому буде файл main_menu.xml.





За промовчанням цей файл визначає один порожній елемент menu:

```
<?xmlversion="1.0" encoding="utf-8"?>
<menuxmlns:android="http://schemas.android.com/apk/res/android">

</menu>
```

Змінимо вміст файлу, визначивши кілька пунктів:

```
<?xmlversion="1.0" encoding="utf-8"?>
<menuxmlns:android="http://schemas.android.com/apk/res/android">
<item
android:id="@+id/action_settings"
android:orderInCategory="1"
android:title="Налаштування"/>
<item
android:id="@+id/save_settings"
android:orderInCategory="3"
android:title="Зберегти"/>
<item
android:id="@+id/open_settings"
android:orderInCategory="2"
android:title="Відкрити"/>
</menu>
```

Тег <menu> є кореневим вузлом файлу та визначає меню, що складається з одного або декількох елементів <item> та <group>.

Елемент <item> представляє об'єкт MenuItem, який є одним із елементів меню. Цей елемент може містити внутрішній елемент <menu>, за допомогою якого створюється підменю.

Елемент <item> включає наступні атрибути, які визначають його зовнішній вигляд та поведінку:

- **android:id**: унікальний id елемент меню, який дозволяє його пізнати при виборі користувачем і знайти через пошук ресурсу по id
- **android:icon**: посилання на ресурс drawable, який визначає зображення для елемента (android:icon="@drawable/ic_help")
- **android:title**: посилання ресурс рядка, що містить заголовок елемента. За промовчанням значення "Settings"
- **android:orderInCategory**: порядок проходження елемента в меню

2.3. Наповнення меню елементами

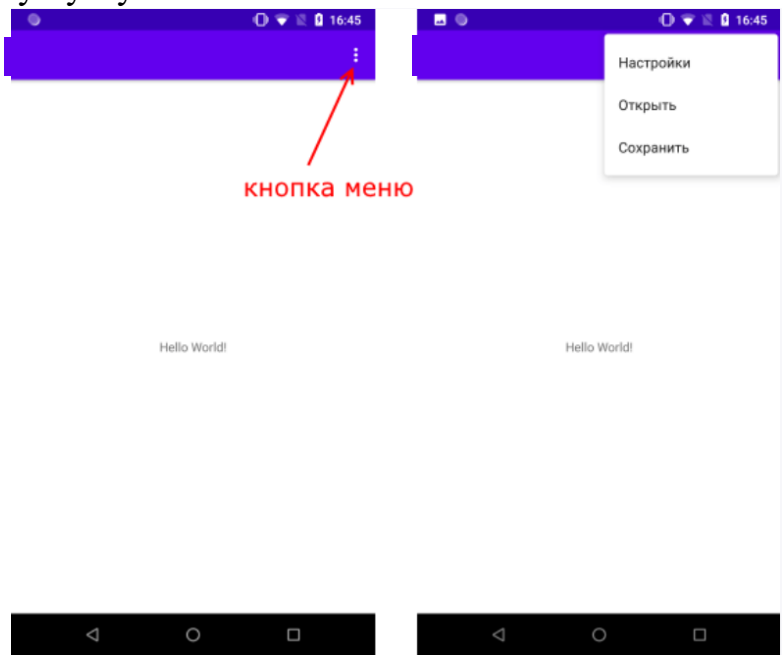
Ми визначили меню з трьома елементами, але визначення елементів у файлі ще не створює меню. Це лише декларативний опис. Щоб вивести його на екран, ми повинні використовувати його в класі Activity. Для цього необхідно перевизначити метод onCreateOptionsMenu. Отже, перейдемо до класу MainActivity і змінимо його так:

```
packagecom.example.menuapp;
```

```
importandroidx.appcompat.app.AppCompatActivity;
importandroid.os.Bundle;
importandroid.view.Menu;
publicclass MainActivity extends AppCompatActivity {
@Override
protectedvoid onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_main);
}
@Override
publicboolean onCreateOptionsMenu(Menu menu) {
getMenuInflater().inflate(R.menu.main_menu, menu);
returntrue;
}
}
```

Метод getMenuInflater повертає об'єкт MenuInflater, який викликається метод inflate(). Цей метод як перший параметр приймає ресурс, що представляє наш декларативний опис меню xml, і наповнює їм об'єкт menu, переданий як другий параметр.

Запустимо програму за замовчуванням і натисніть кнопку меню у правому верхньому кутку:



2.4. Обробка натискань у меню

Якщо ми натиснемо на будь-який пункт меню, то нічого не станеться. Щоб прив'язати до меню дії, нам треба перевизначити у класі діяльність наОпціїІмSelected.

Для виведення вибраного елемента меню у файлі activity_main.xml визначимо текстове поле з id=header:

```
<?xmlversion="1.0" encoding="utf-8"?>
```

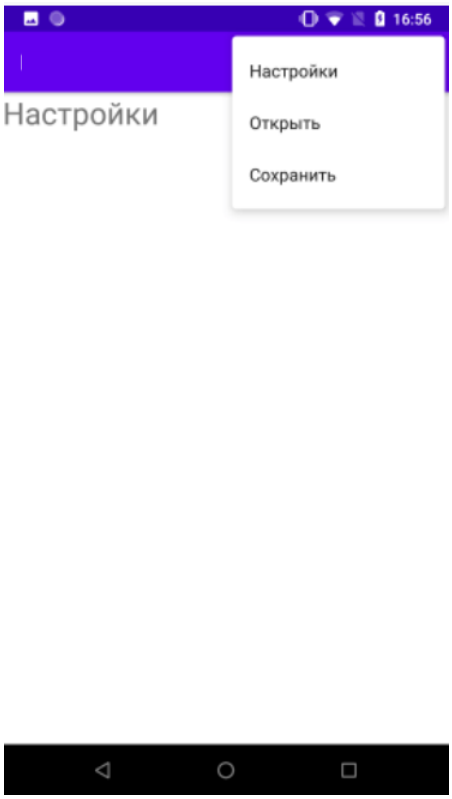
```
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:app="http://schemas.android.com/apk/res-auto"
android:layout_width="match_parent"
android:layout_height="match_parent">
<TextView
android:id="@+id/selectedMenuItem"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Hello World!"
android:textSize="28sp"
app:layout_constraintLeft_toLeftOf="parent"
app:layout_constraintTop_toTopOf="parent"/>
</androidx.constraintlayout.widget.ConstraintLayout>
```

І змінимо клас MainActivity:

```
packagecom.example.menuapp;
importandroidx.appcompat.app.AppCompatActivity;
importandroid.os.Bundle;
importandroid.view.Menu;
importandroid.view.MenuItem;
importandroid.widget.TextView;
publicclass MainActivity extends AppCompatActivity {
@Override
protectedvoid onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_main);
}
@Override
publicboolean onCreateOptionsMenu(Menu menu) {
getMenuInflater().inflate(R.menu.main_menu, menu);
returntrue;
}
@Override
publicboolean onOptionsItemSelected(MenuItem item) {
intid = item.getItemId();
TextView headerView = findViewById(R.id.selectedMenuItem);
switch(id){
caseR.id.action_settings :
headerView.setText("Налаштування");
returntrue;
caseR.id.open_settings:
headerView.setText("Відкрити");
returntrue;
caseR.id.save_settings:
headerView.setText("Зберегти");
```

```
returntrue;
}
//headerView.setText(item.getTitle());
returnsuper.onOptionsItemSelected(item);
}
}
```

Щоб зрозуміти, який пункт меню обрано, спочатку отримуємо його ідентифікатор int id = item.getItemId(). Потім пробігаємося в конструкції switch.



Варто зазначити, що в даному випадку, якщо наше завдання полягало, щоб просто виводити текст вибраного пункту меню, то замість конструкції switch просто могли написати так:

```
headerView.setText(item.getTitle());
```

2.5. Програмне створення меню

Крім визначення елементів меню xml, можна також створити меню програмним способом. Для додавання нових пунктів меню використається метод add() класу Menu.

Наприклад, змінимо код MainActivity:

```
packagecom.example.menuapp;

importandroidx.appcompat.app.AppCompatActivity;
importandroid.os.Bundle;
importandroid.view.Menu;
importandroid.view.MenuItem;
importandroid.widget.TextView;
```

```
publicclass MainActivity extends AppCompatActivity {
@Override
protectedvoid onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_main);
}
@Override
publicboolean onCreateOptionsMenu(Menu menu) {
super.onCreateOptionsMenu(menu);
menu.add("Налаштування");
menu.add("Відкрити");
menu.add("Зберегти");
returntrue;
}
@Override
publicboolean onOptionsItemSelected(MenuItem item) {
String title = item.getTitle().toString();
TextView headerView = findViewById(R.id.selectedMenuItem);
headerView.setText(title);
returnsuper.onOptionsItemSelected(item);
}
}
```

Використана версія методу add() приймає заголовок пункту меню.

2.6. Групи в меню та підменю

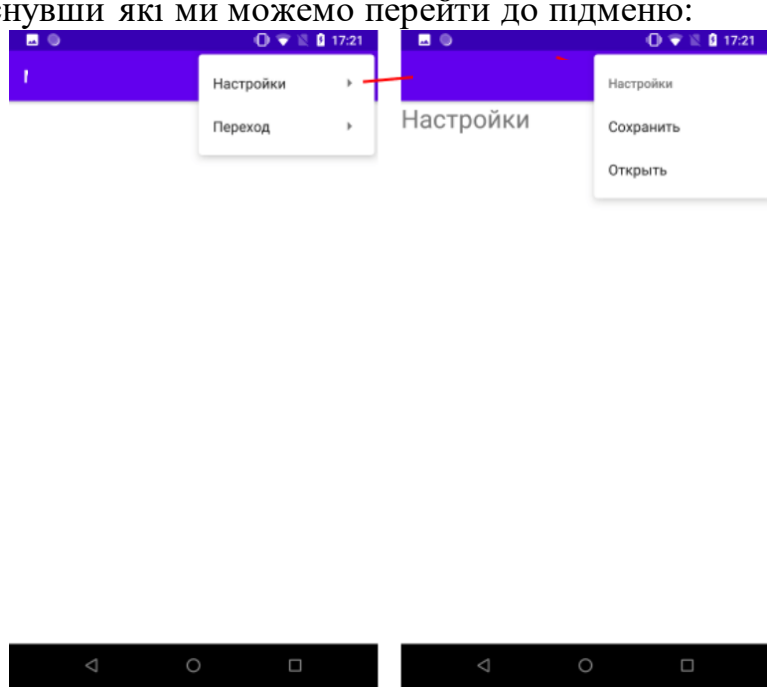
2.6.1. Створення підменю

Для створення підменю у файлі розмітки меню визначимо внутрішній елемент menu:

```
<?xmlversion="1.0" encoding="utf-8"?>
<menuxmlns:android="http://schemas.android.com/apk/res/android">
<item
android:id="@+id/action_settings"
android:title="Налаштування">
<menu>
<itemandroid:id="@+id/save_settings"
android:title="Зберегти"/>
<itemandroid:id="@+id/open_settings"
android:title="Відкрити"/>
</menu>
</item>
<item
android:id="@+id/action_move"
android:title="Перехід">
```

```
<menu>
<itemandroid:id="@+id/forward"
android:title="Вперед"/>
<itemandroid:id="@+id/back"
android:title="Назад"/>
</menu>
</item>
</menu>
```

Після натискання меню відобразяться елементи верхнього рівня, натиснувши які ми можемо перейти до підменю:



2.6.2. Групи у меню

Використання елемента group дозволяє оформити елементи меню до групи:

```
<?xmlversion="1.0" encoding="utf-8"?>
<menuxmlns:android="http://schemas.android.com/apk/res/android">

<groupandroid:checkableBehavior="single">
<item
android:id="@+id/action_settings"
android:title="Налаштування"
android:checked="true"/>
<itemandroid:id="@+id/save_settings"
android:title="Зберегти"/>
<itemandroid:id="@+id/open_settings"
android:title="Відкрити"/>
</group>
</menu>
```

У групі ми можемо встановити атрибут android:checkableBehavior. Цей атрибут може приймати такі значення: single (у кожного елемента створюється радіокнопка), all (кожний елемент створюється прапорець) і none.

В даному випадку для кожного елемента створюватиметься радіокнопка (візуально кружок). І для першого елемента встановлюється зазначена кнопка (android:checked="true").

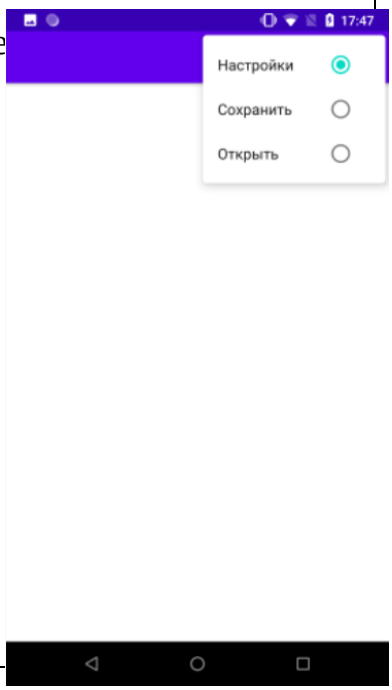
Нехай буде визначено текстове поле у файлі розмітки інтерфейсу activity_main.xml:

```
<?xmlversion="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/selectedMenuItem"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:textSize="28sp"
app:layout_constraintLeft_toLeftOf="parent"
app:layout_constraintTop_toTopOf="parent"/>
</androidx.constraintlayout.widget.ConstraintLayout>
```

А в класі MainActivity визначимо виділення радіокнопки у вибраного пункту меню:

```
packagecom.example.menuapp;
importandroidx.appcompat.app.AppCompatActivity;
importandroid.os.Bundle;
importandroid.view.Menu;
importandroid.view.MenuItem;
importandroid.widget.TextView;
publicclass MainActivity extends AppCompatActivity {
@Override
protectedvoid onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_main);
}
@Override
publicboolean onCreateOptionsMenu(Menu menu) {

getMenuInflater().inflate(R.menu.main_menu, menu);
returntrue;
}
@Override
publicboolean onOptionsItemSelected(MenuItem item) {
```

```
intid = item.getItemId();
TextView headerView = findViewById(R.id.selecte
switch(id){
caseR.id.action_settings :
headerView.setText("Налаштування");
returntrue;
caseR.id.open_settings:
headerView.setText("Відкрити");
returntrue;
caseR.id.save_settings:
headerView.setText("Зберегти");
returntrue;
}
returnsuper.onOptionsItemSelected(item);
}
}
```

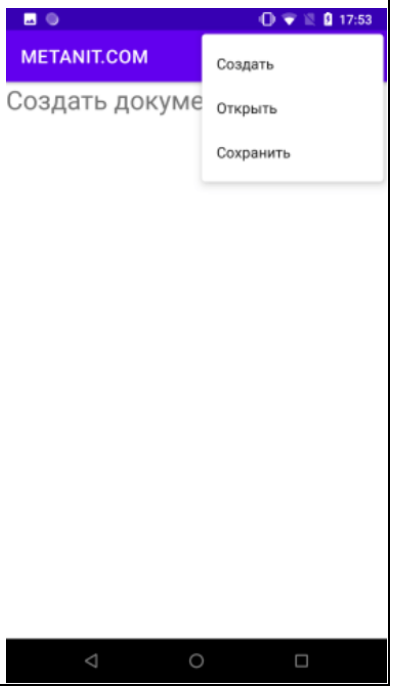


2.6.3. Програмне створення груп у меню та підменю

Також групи та підменю можна створювати програмним способом. Так, змінимо код MainActivity:

```
packagecom.example.menuapp;
importandroidx.appcompat.app.AppCompatActivity;
importandroid.os.Bundle;
importandroid.view.Menu;
importandroid.view.MenuItem;
importandroid.widget.TextView;
publicclass MainActivity extends AppCompatActivity {
@Override
protectedvoid onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_main);
}
@Override
publicboolean onCreateOptionsMenu(Menu menu) {
super.onCreateOptionsMenu(menu);
menu.add(0// Група
,1// id
,0//порядок
, "Створити"); // Заголовок
menu.add(0,2,1,"Відкрити");
menu.add(0,3,2,"Зберегти");
returntrue;
}
@Override
```

```
publicboolean onOptionsItemSelected(MenuItem item) {
intid = item.getItemId();
TextView headerView = findViewById(R.id.selectedMenuItem);
switch(id){
case1:
headerView.setText("Створити документ");
returntrue;
case2:
headerView.setText("Відкрити документ");
returntrue;
case3:
headerView.setText("Зберегти документ");
returntrue;
}
returnsuper.onOptionsItemSelected(item);
}
}
```



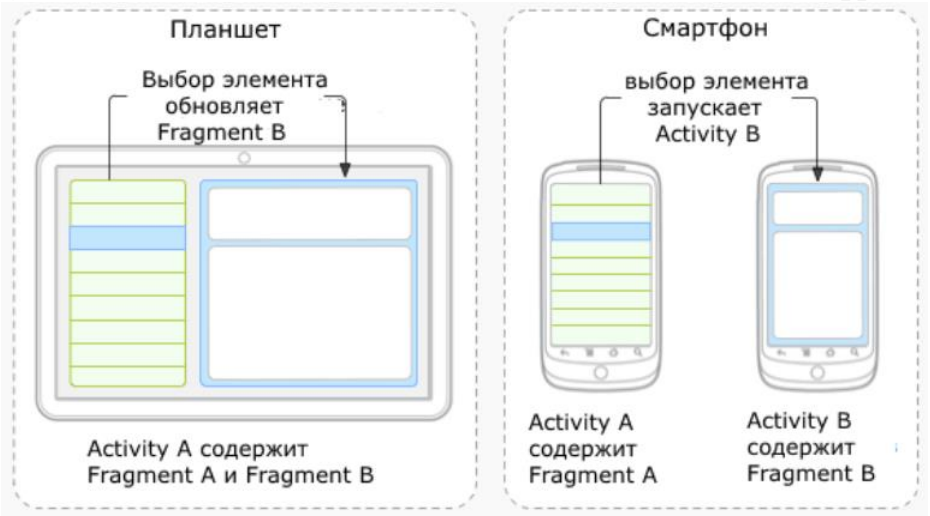
Використана версія методу add() додає пункт в меню, приймаючи наступні параметри: номер групи, id, порядок елемента в меню і заголовок елемента.

3. ФРАГМЕНТИ

3.1. Введення у фрагменти

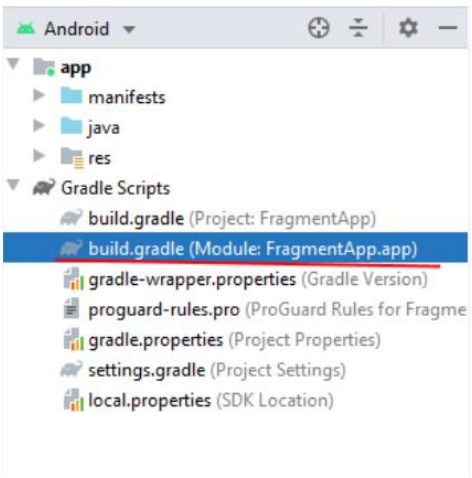
Організація програми на основі декількох дій не завжди може бути оптимальною. Світ ОС Android дуже фрагментований і складається з багатьох пристроїв. І якщо для мобільних апаратів з невеликими екранами взаємодія між різними activity виглядає досить непогано, то на великих екранах – планшетах, телевізорах вікна activity виглядали б не дуже через великий розмір екрану. Саме тому і виникла концепція фрагментів.

Фрагмент представляє шматочок візуального інтерфейсу, який може використовуватися повторно і багаторазово. У фрагмента може бути власний файл layout, фрагменти мають свій власний життєвий цикл. Фрагмент існує в контексті діяльності і має свій життєвий цикл, поза діяльністю відокремлено він існувати не може. Кожна діяльність може мати кілька фрагментів.



Для початку роботи з фрагментами створимо новий проект із порожньою MainActivity. І спочатку створимо перший фрагмент. Але відразу слід зазначити, що не вся функціональність фрагментів за замовчуванням може бути доступна в проекті, оскільки розміщується в окремій бібліотеці – AndroidX Fragment library. І спочатку необхідно підключити до проекту бібліотеку у файлі build.gradle.

Знайдемо у ньому секцію dependencies, яка виглядає за умовчанням приблизно так:



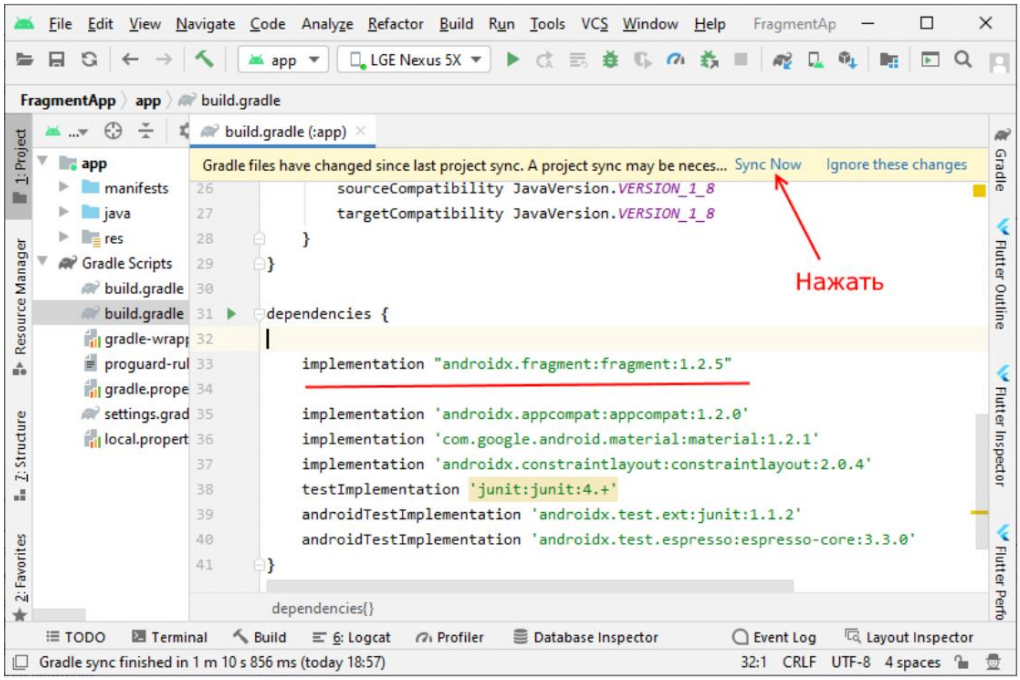
```
dependencies {
implementation 'androidx.appcompat:appcompat:1.3.1'
implementation 'com.google.android.material:material:1.4.0'
implementation 'androidx.constraintlayout:constraintlayout:2.1.0'
testImplementation 'junit:junit:4.+'
androidTestImplementation 'androidx.test.ext:junit:1.1.3'
androidTestImplementation 'androidx.test.espresso:espresso-core:3.4.0'
}
```

На її початок додамо рядок

```
implementation "androidx.fragment:fragment:1.3.6"
```

Тобто в моєму випадку вийде

```
dependencies {
implementation "androidx.fragment:fragment:1.3.6"
implementation 'androidx.appcompat:appcompat:1.3.1'
implementation 'com.google.android.material:material:1.4.0'
implementation 'androidx.constraintlayout:constraintlayout:2.1.0'
testImplementation 'junit:junit:4.+'
androidTestImplementation 'androidx.test.ext:junit:1.1.3'
androidTestImplementation 'androidx.test.espresso:espresso-core:3.4.0'
}
```



І потім натиснемо на посилання Sync Now.

Фактично, фрагмент - це звичайний клас Java, який успадковується від класу Fragment. Однак, як і клас Activity, фрагмент може використовувати xml-файли layout для визначення графічного інтерфейсу. І таким чином, ми можемо додати окремо клас Java, який представляє фрагмент, і файл XML для зберігання в ньому розмітки інтерфейсу, який використовуватиме фрагмент.

Отже, додамо в папку res/layout новий файл fragment_content.xml і визначимо наступний код:

```
<?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">
<Button
android:id="@+id/updateButton"
android:layout_width="0dp"
android:layout_height="wrap_content"
android:text="Оновити"
app:layout_constraintBottom_toTopOf="@+id/dateTextView"
app:layout_constraintLeft_toLeftOf="parent"
app:layout_constraintRight_toRightOf="parent"
app:layout_constraintTop_toTopOf="parent" />
<TextView
android:id="@+id/dateTextView"
android:layout_width="0dp"
android:layout_height="wrap_content"
android:text="Hello  from Fragment"
android:textSize="28sp"
app:layout_constraintLeft_toLeftOf="parent"
app:layout_constraintRight_toRightOf="parent"
app:layout_constraintTop_toBottomOf="@+id/updateButton" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

Фрагменти містять самі елементи управління, як і activity. Зокрема, тут визначено кнопку та текстове поле, які будуть складати інтерфейс фрагмента.

Тепер створимо сам клас фрагмента. Для цього додамо в одну папку з MainActivity новий клас. Для цього натиснемо на папку правою кнопкою миші та виберемо в меню New -> Java Class. Назвемо новий клас ContentFragment і визначимо у нього такий зміст:

```
package com.example.fragmentapp;
import androidx.fragment.app.Fragment;
public class ContentFragment extends Fragment {
public ContentFragment(){
super(R.layout.fragment_content);
}
}
```

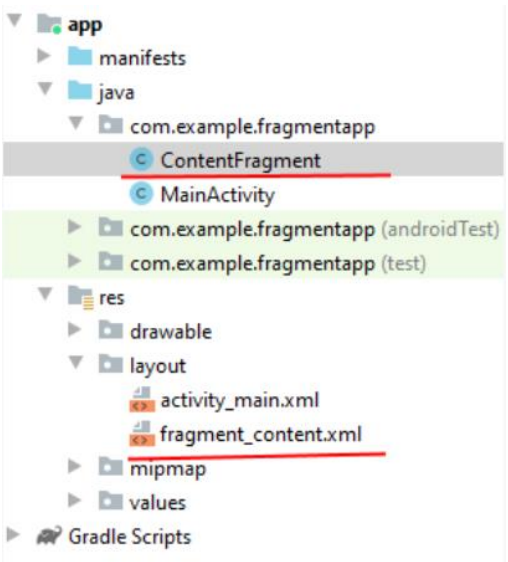
Клас фрагмента повинен успадковуватися від класу Fragment.

Щоб вказати, що фрагмент використовувати певний xml-файл layout, ідентифікатор ресурсу layout передається у виклик конструктора батьківського класу (тобто класу Fragment).

Весь проект виглядатиме так:

3.2. Додавання фрагмента до Activity

Для використання фрагмента додамо його до MainActivity. Для цього змінимо файл activity_main.xml, який визначає інтерфейс для MainActivity:




```
<?xml version="1.0" encoding="utf-8"?>
<androidx.fragment.app.FragmentContainerView
xmlns:android="http://schemas.android.com/apk/res/android"
android:id="@+id/fragment_container_view"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:name="com.example.fragmentapp.ContentFragment" />
```

Для додавання елемента застосовується елемент FragmentContainerView. По суті, FragmentContainerView представляє об'єкт View, який розширює клас FrameLayout і призначений спеціально для роботи з фрагментами. Власне, крім фрагментів, він більше нічого утримувати не може.

Його атрибут android:name вказує на ім'я класу фрагмента, який буде використовуватись. У моєму випадку повне ім'я класу фрагмента з облікового запису com.example.fragmentapp.ContentFragment.

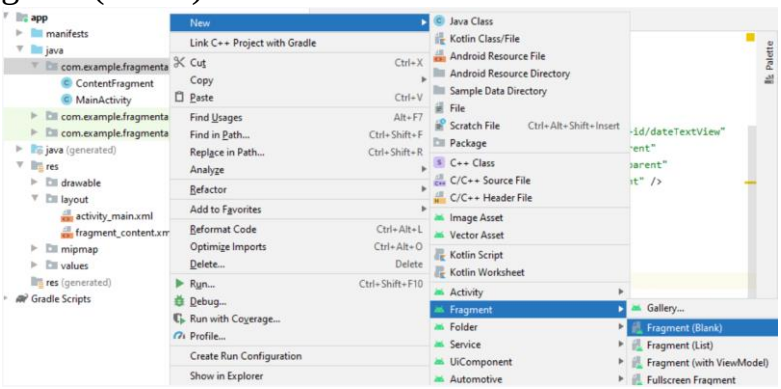
Код класу MainActivity залишається тим самим, що і при створенні проекту:

```
package com.example.fragmentapp;
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);
}}
```

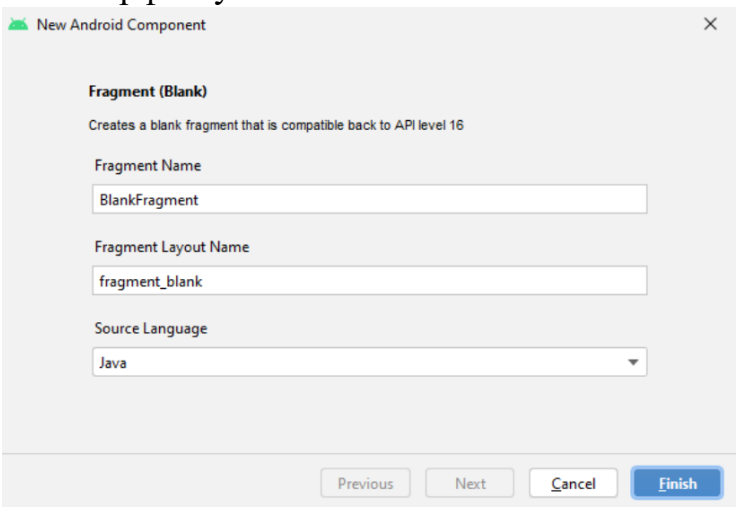
Якщо ми запустимо додаток, то ми побачимо фактично той самий інтерфейс, який ми могли б зробити і через activity, тільки в даному випадку інтерфейс буде визначено у фрагменті:

Варто відзначити, що Android Studio є готовим шаблоном для додавання фрагмента. Власне, скористаємося цим способом.

Для цього натиснемо на папку, де знаходиться клас MainActivity, правою кнопкою миші і в меню виберемо New -> Fragment -> Fragment(Blank):



Даний шаблон запропонувати вказати клас фрагмента та назву файлу пов'язаного з ним класу розмітки інтерфейсу.

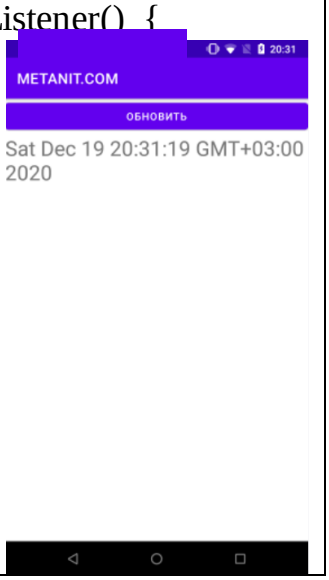


3.3. Додавання логіки до фрагмента

Фрагмент визначає кнопку. Тепер додамо до цієї кнопки певну дію. Для цього змінимо клас ContentFragment:

```
package com.example.fragmentapp;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;
import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.fragment.app.Fragment;
```

```
import java.util.Date;
public class ContentFragment extends Fragment {
public ContentFragment(){
super(R.layout.fragment_content);
}
@Override
public void onViewCreated(@NonNull View view, @Nullable Bundle savedInstanceState) {
super.onViewCreated(view, savedInstanceState);
Button updateButton = view.findViewById(R.id.updateButton);
TextView updateBox = view.findViewById(R.id.dateTextView);
updateButton.setOnClickListener(new View.OnClickListener() {
@Override
public void onClick(View v) {
String curDate = new Date().toString();
updateBox.setText(curDate);
}
});
}
}
```



Тут перевизначено метод onViewCreated класу Fragment, який викликається після створення об'єкта View для візуального інтерфейсу, який представляє цей фрагмент. Створений об'єкт View передається як перший параметр. І далі ми можемо отримати конкретні елементи управління в рамках цього об'єкта View, зокрема TextView і Button, і виконати деякі дії з ними. В даному випадку в обробнику натискання кнопки у текстовому полі відображається поточна дата.

3.4. Додавання фрагмента до коду

Крім визначення фрагмента в xaml-файлі інтерфейсу, ми можемо додати його динамічно в activity.

Для цього змінимо файл activity_main.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.fragment.app.FragmentContainerView
xmlns:android="http://schemas.android.com/apk/res/android"
android:id="@+id/fragment_container_view"
android:layout_width="match_parent"
android:layout_height="match_parent" />
```

І також змінимо клас MainActivity:

```
package com.example.fragmentapp;
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);
if (savedInstanceState == null) {
getSupportFragmentManager().beginTransaction()
.add(R.id.fragment_container_view, ContentFragment.class, null)
. commit ();
}
}
}
```

МетодgetSupportFragmentManager()повертає об'єктFragmentManager, який керує фрагментами.

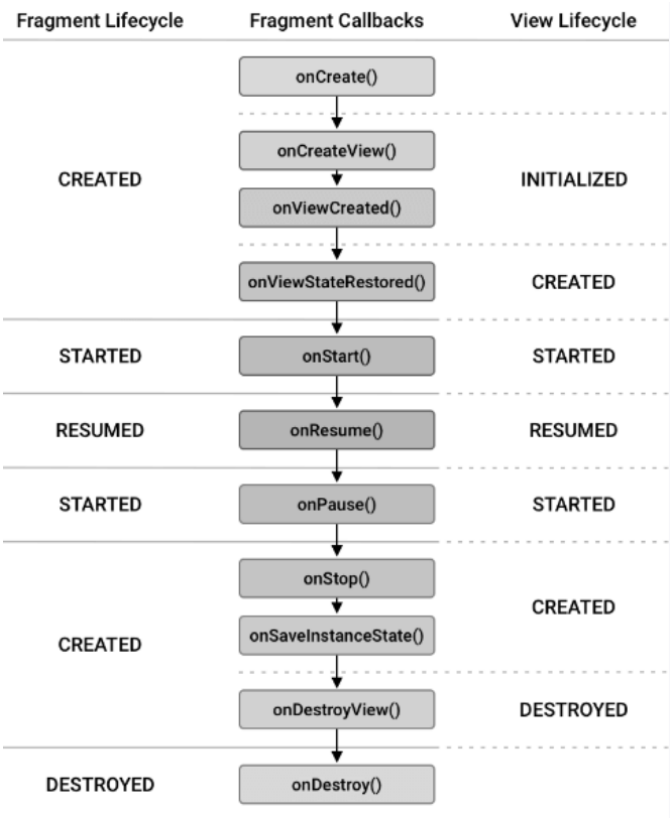
Об'єкт FragmentManager за допомогою методуbeginTransaction()створює об'єкт FragmentTransaction.

FragmentTransaction виконує два методи: add() та commit(). Метод add() додає фрагмент:add(R.id.fragment_container_view, new ContentFragment())-першим аргументом передається ресурс розмітки, до якого треба додати фрагмент (це визначений в activity_main.xml елементandroidx.fragment.app.FragmentContainerView). І методcommit()підтверджує та завершує операцію додавання.

Підсумковий результат такого додавання фрагмента буде тим самим, що і при явному визначенні фрагмента через елементFragmentContainerViewy розмітці інтерфейсу.

Життєвий цикл фрагментів

Кожен клас фрагмента успадковується від базового класу Fragment та має свій життєвий цикл, що складається з низки етапів:



Кожен етап життєвого циклу описується однією з констант перерахування Lifecycle.State:

- **INITIALIZED**
- **CREATED**

- **STARTED**
- **RESUMED**
- **DESTROYED**

Варто зазначити, що представлення фрагмента (його візуальний інтерфейс або View) має окремий цикл життя.

- При створенні фрагмент знаходиться у стані INITIALIZED. Щоб фрагмент пройшов решту етапів життєвого циклу, фрагмент необхідно передати в об'єкт FragmentManager, який далі визначає стан фрагмента і переводить фрагмент з одного стану в інший.
- Коли фрагмент додається до FragmentManager і прикріплюється до певного класу Activity, у фрагмента викликається метод onAttach(). Даний метод викликається до решти методів життєвого циклу. Після цього фрагмент переходить у стан CREATED
- **onCreate()**: відбувається створення фрагмента У цьому методі ми можемо отримати раніше збережений стан фрагмента через параметр методу Bundle savedInstanceState. (Якщо фрагмент створюється вперше, цей об'єкт має значення null) Цей метод викликається після виклику відповідного методу onCreate() у activity.

```
publicvoid onCreate(Bundle savedInstanceState)
```

- **onCreateView()**: фрагмент створює уявлення (View або візуальний інтерфейс). У цьому методі ми можемо встановити, який ізуальний інтерфейс використовуватиме фрагмент. При виконанні цього методу представлення фрагмента перетворюється на стан INITIALIZED. А сам фрагмент все ще перебуває в стані CREATED

```
publicView onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState)
```

- Перший параметр - об'єкт LayoutInflater дозволяє отримати вміст ресурсу layout і передати його у фрагмент.
- Другий параметр - об'єкт ViewGroup представляє контейнер, у якій завантажуватиметься фрагмент.
- Третій параметр – об'єкт Bundle представляє стан фрагмента. (Якщо фрагмент завантажується вперше, то дорівнює null)
- На виході метод повертає створене за допомогою LayoutInflater уявлення у вигляді об'єкта View - власне уявлення фрагмента
- **onViewCreated()**: викликається після створення уявлення фрагмента

```
publicvoid onViewCreated (View view, Bundle savedInstanceState)
```

- Перший параметр - об'єкт View - представлення фрагмента, яке було створено за допомогою методу onCreateView.
- Другий параметр – об'єкт Bundle представляє стан фрагмента. (Якщо фрагмент завантажується вперше, то дорівнює null)
- **onViewStateRestored()**: отримує стан уявлення фрагмента Після виконання цього методу представлення фрагмента перетворюється на стан CREATED

```
publicvoid onViewStateRestored (Bundle savedInstanceState)
```

- **onStart()**: викликається, коли фрагмент стає видимим і разом із поданням переходить у стан STARTED

```
publicvoid onStart ()
```

- **onResume()**: викликається, коли фрагмент стає активним, і користувач може взаємодіяти з ним. При цьому фрагмент та його уявлення переходять у стан RESUMED

```
publicvoid onResume()
```

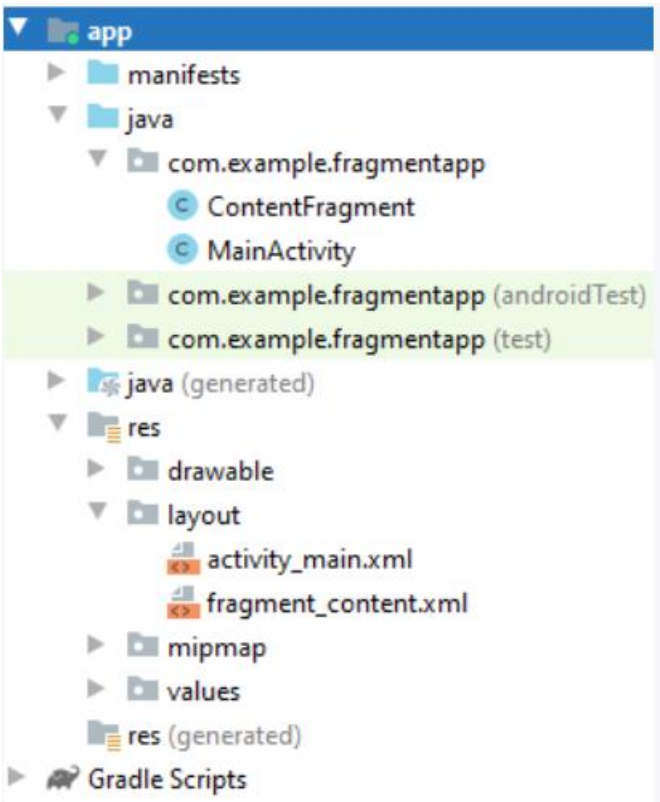
- **onPause()**: фрагмент продовжує залишатися видимим, але вже не активним і разом з поданням переходить у стан STARTED

```
publicvoid onPause ()
```

- **onStop()**: фрагмент більше не є видимим і разом з поданням переходить у стан CREATED

```
publicvoid onStop()
```

- На цьому етапі життєвого циклу ми можемо зберегти стан фрагмента за допомогою методу onSaveInstanceState(). Однак, варто враховувати, що виклик цього методу залежить від версії API. До API 28 onSaveInstanceState() викликається до onStop(), а починаючи API 28 після onStop().
- **onDestroyView()**: знищується уявлення фрагмента Подання переходить у стан DESTROYED
- **onDestroy()**: остаточне знищення фрагмента - він також переходить у стан DESTROYED
- Метод onDetach() викликається, коли фрагмент видаляється з FragmentManager і відкріплюється від класу Activity. Цей метод викликається після решти методів життєвого циклу.



У коді класу фрагмента ми можемо перевизначити всі або частину цих методів. Наприклад, нехай у нас буде визначено наступний проект:

У каталозі res/layout визначено файл layout для фрагмента - fragment_content.xml з наступною розміткою:

```
<?xmlversion="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">
```

```
<Button
android:id="@+id/updateButton"
android:layout_width="0dp"
android:layout_height="wrap_content"
android:text="Оновити"
app:layout_constraintBottom_toTopOf="@+id/dateTextView"
app:layout_constraintLeft_toLeftOf="parent"
app:layout_constraintRight_toRightOf="parent"
app:layout_constraintTop_toTopOf="parent"/>
<TextView
android:id="@+id/dateTextView"
android:layout_width="0dp"
android:layout_height="wrap_content"
android:textSize="26sp"
app:layout_constraintLeft_toLeftOf="parent"
app:layout_constraintRight_toRightOf="parent"
app:layout_constraintTop_toBottomOf="@+id/updateButton"/>

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

Клас фрагмента використовує даний файл для встановлення уявлення, а також визначає методи управління життєвим циклом:

```
packagecom.example.fragmentapp;
importandroid.content.Context;
importandroid.os.Bundle;
importandroid.view.LayoutInflater;
importandroid.view.View;
importandroid.view.ViewGroup;
importandroid.widget.Button;
importandroid.widget.TextView;
importandroidx.annotation.NonNull;
importandroidx.annotation.Nullable;
importandroidx.fragment.app.Fragment;
importandroid.util.Log;
importjava.util.Date;
publicclass ContentFragment extends Fragment {
privatefinal static String TAG = "ContentFragment";
publicContentFragment(){
Log.d(TAG, "Constructor");
}
@Override
publicvoid onAttach(@NonNull Context context) {
super.onAttach(context);
Log.d(TAG, "onAttach");
}
@Override
```

```
        publicvoid onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        Log.d(TAG, "onCreate");
        }
        @Override
        publicView onCreateView(LayoutInflater inflater, ViewGroup container,
        Bundle savedInstanceState) {
        Log.d(TAG, "onCreateView");
        returninflater.inflate(R.layout.fragment_content, container, false);
        }
        @Override
        publicvoid onViewCreated(@NonNull View view, @Nullable Bundle
savedInstanceState) {
        super.onViewCreated(view, savedInstanceState);
        Button updateButton = view.findViewById(R.id.updateButton);
        TextView updateBox = view.findViewById(R.id.dateTextView);
        updateButton.setOnClickListener(newView.OnClickListener() {
        @Override
        publicvoid onClick(View v) {
        String curDate = newDate().toString();
        updateBox.setText(curDate);
        }
        });
        Log.d(TAG, "onViewCreated");
        }
        @Override
        publicvoid onViewStateRestored(@Nullable Bundle savedInstanceState) {
        super.onViewStateRestored(savedInstanceState);
        Log.d(TAG, "onViewStateRestored");
        }
        @Override
        publicvoid onStart() {
        super.onStart();
        Log.d(TAG, "onStart");
        }
        @Override
        publicvoid onResume() {
        super.onResume();
        Log.d(TAG, "onResume");
        }
        @Override
        publicvoid onPause() {
        super.onPause();
        Log.d(TAG, "onPause");
        }
```

```
@Override
publicvoid onStop() {
super.onStop();
Log.d(TAG, "onStop");
}
@Override
publicvoid onDestroyView() {
super.onDestroyView();
Log.d(TAG, "onDestroyView");
}
@Override
publicvoid onDestroy() {
super.onDestroy();
Log.d(TAG, "onDestroy");
}
@Override
publicvoid onDetach() {
super.onDetach();
Log.d(TAG, "onDetach");
}
}
```

На відміну від минулої статті, де розглядалося створення фрагмента, тут фрагмент встановлює уявлення у методі onCreateView. Для цього метод inflate() об'єкта LayoutInflater передається ідентифікатор ресурсу layout і контейнер - об'єкт ViewGroup, в який буде завантажуватися фрагмент. Через війну спосіб inflate() повертає створене уявлення.

```
returninflater.inflate(R.layout.fragment_content,  container, false);
```

При виконанні методу onViewCreated() уявлення вже створено і воно передається як перший параметр - об'єкт View, через який за допомогою ідентифікаторів ми можемо отримати візуальні елементи - TextView та Button, які визначені у поданні.

Для інших методів життєвого циклу встановлено просте логування за допомогою методу Log.d().

Нехай у файлі activity_main.xml відбувається додавання фрагмента:

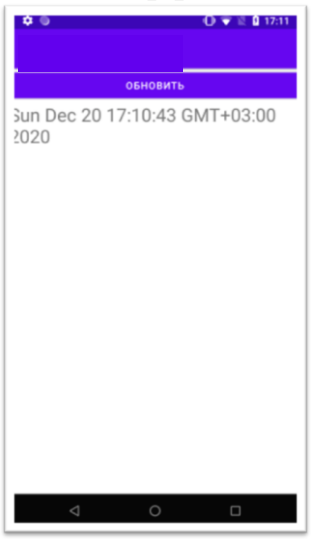
```
<?xmlversion="1.0" encoding="utf-8"?>
<androidx.fragment.app.FragmentContainerView
xmlns:android="http://schemas.android.com/apk/res/android"
android:id="@+id/fragment_container_view"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:name="com.example.fragmentapp.ContentFragment"/>
```

І клас**MainActivity**:

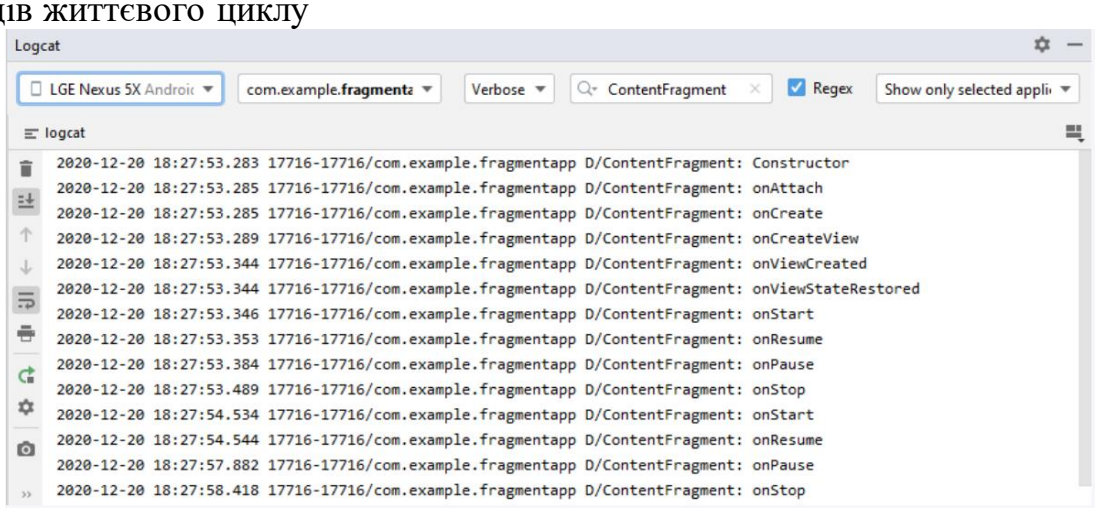
```
packagecom.example.fragmentapp;
importandroidx.appcompat.app.AppCompatActivity;
importandroid.os.Bundle;
```

```
publicclass MainActivity extends AppCompatActivity {
@Override
protectedvoid onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_main);
}
}
```

Якщо ми запустимо проект, то на екрані пристрою побачимо візуальний інтерфейс, визначений для фрагмента.



А у вікніLogcatв Android Studio можна буде спостерігати логування методів життєвого циклу



3.5. Взаємодія між фрагментами

Одна дія може використовувати кілька фрагментів, наприклад, з одного боку список, а з іншого - детальний опис вибраного елемента списку. У такій конфігурації діяльність використовує два фрагменти, які повинні взаємодіяти між собою. Розглянемо базові принципи взаємодії фрагментів у додатку.

Створимо новий проект із порожньою MainActivity. Далі створимо розмітку layout для фрагментів. Нехай у нас у додатку буде два фрагменти. Додамо до папки res/layout новий xml-файл fragment_list.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">
<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>
```

Тут визначено елемент ListView виведення списку об'єктів.

Також додамо для іншого фрагмента файл розмітки fragment_detail.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">
<TextView
android:id="@+id/detailsText"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_gravity="center"

android:text="Не вибрано"
android:textSize="22sp"
app:layout_constraintLeft_toLeftOf="parent"
app:layout_constraintTop_toTopOf="parent"
/>
</androidx.constraintlayout.widget.ConstraintLayout>
```

Обидва фрагменти будуть гранично простими: один міститиме список, а другий - текстове поле. Логіка програми буде така: при виборі елемента у списку в одному фрагменті вибраний елемент повинен відобразитись у текстовому полі, яке знаходиться у другому фрагменті.

Потім додамо в проект в одну папку з MainActivity власне класи фрагментів. Додамо новий клас ListFragment з таким вмістом:

```
package com.example.fragmentapp;
import android.content.Context;
```

```
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.ListView;
import androidx.fragment.app.Fragment;
public class ListFragment extends Fragment {
interface OnFragmentSendDataListener {
void onSendData(String data);
}
private OnFragmentSendDataListener fragmentSendDataListener;
String[] countries = { "Бразилія", "Аргентина", "Колумбія", "Чілі",
"Уругвай"};
@Override
public void onAttach(Context context) {
super.onAttach(context);
try {
fragmentSendDataListener=(OnFragmentSendDataListener) context;
} catch (ClassCastException e) {
throw new ClassCastException(context.toString()
+ "має реалізовувати інтерфейс OnFragmentInteractionListener");
}
}
@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container,
Bundle savedInstanceState) {
View view = inflater.inflate(R.layout.fragment_list, container, false);
// отримуємо елемент ListView
ListView countriesList = view.findViewById(R.id.countriesList);
// створюємо адаптер
ArrayAdapter<String> adapter=new ArrayAdapter(getContext(),
android.R.layout.simple_list_item_1, countries);
// Встановлюємо для списку адаптер
countriesList.setAdapter(adapter);
// додаємо для списку слухач
countriesList.setOnItemClickListener(new
AdapterView.OnItemClickListener(){
@Override
public void onItemClick(AdapterView<?> parent, View v, int position,
long id)
{
// отримуємо вибраний елемент
String selectedItem = (String)parent.getItemAtPosition(position);
```

```
// Надсилаємо дані Activity
fragmentSendDataListener.onSendData(selectedItem);
}
});
return view;
}
}
```

Фрагменти що неспроможні безпосередньо взаємодіяти між собою. Для цього треба звертатися до контексту, як якого виступає клас Activity. Для звернення до діяльності, як правило, створюється вкладений інтерфейс. У цьому випадку він називається OnFragmentSendDataListener з одним методом.

```
interface OnFragmentSendDataListener {
void onSendData(String data);
}
private OnFragmentSendDataListener fragmentSendDataListener;
```

Але щоб взаємодіяти з іншим фрагментом через діяльність, нам треба прикріпити поточний фрагмент до діяльності. Для цього у класі фрагмента визначено методonAttach(Context context). У ньому відбувається встановлення об'єктаOnFragmentSendDataListener:

```
fragmentSendDataListener=(OnFragmentSendDataListener) context;
```

При обробці натискання на елемент у списку ми можемо надіслати Activity дані про вибраний об'єкт:

```
String selectedItem = (String)parent.getItemAtPosition(position);
fragmentSendDataListener.onSendData(selectedItem);
```

Таким чином, при виборі об'єкта у списку MainActivity отримає вибраний об'єкт.

Тепер визначимо клас другого фрагмента. Назвемо його DetailFragment:

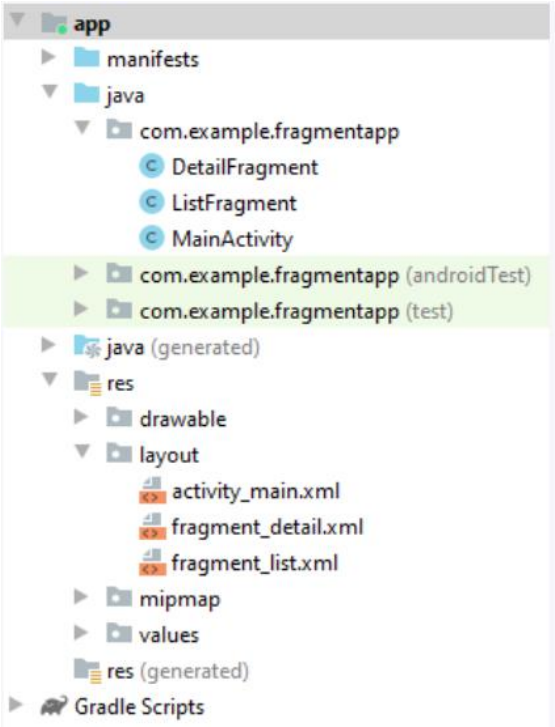
```
package com.example.fragmentapp;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.TextView;
import androidx.fragment.app.Fragment;
public class DetailFragment extends Fragment {
@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container,
Bundle savedInstanceState) {
return inflater.inflate(R.layout.fragment_detail, container, false);
}
// Поновлення текстового поля
public void setSelectedItem(String selectedItem) {
```

```
TextView view = getView().findViewById(R.id.detailsText);
view.setText(selectedItem);
}
}
```

Завдання цього фрагмента – висновок деякої інформації. Так як він не повинен передавати жодну інформацію іншому фрагменту, тут ми модем обмежитися лише перевизначенням методуonCreateView(), який як візуальний інтерфейс встановлює розмітку з файлу fragment_detail.xml

Але щоб імітувати взаємодію між двома фрагментами, тут також визначено методsetSelectedItem(), яка оновлює текст на текстовому полі.

У результаті вийде така структура:



Тепер змінимо файл розмітки 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" >
<androidx.fragment.app.FragmentContainerView
android:id="@+id/listFragment"
android:layout_width="0dp"
android:layout_height="0dp"
android:name="com.example.fragmentapp.ListFragment"
app:layout_constraintLeft_toLeftOf="parent"
app:layout_constraintTop_toTopOf="parent"
app:layout_constraintBottom_toTopOf="@+id/detailFragment"
app:layout_constraintRight_toRightOf="parent"/>
<androidx.fragment.app.FragmentContainerView
```

```
        android:id="@+id/detailFragment"
        android:layout_width="0dp"
        android:layout_height="0dp"
        android:name="com.example.fragmentapp.DetailFragment"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintTop_toBottomOf="@id/listFragment"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintRight_toRightOf="parent"/>

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

За допомогою двох елементів FragmentContainerView в MainActivity додаються два вище певні фрагменти.

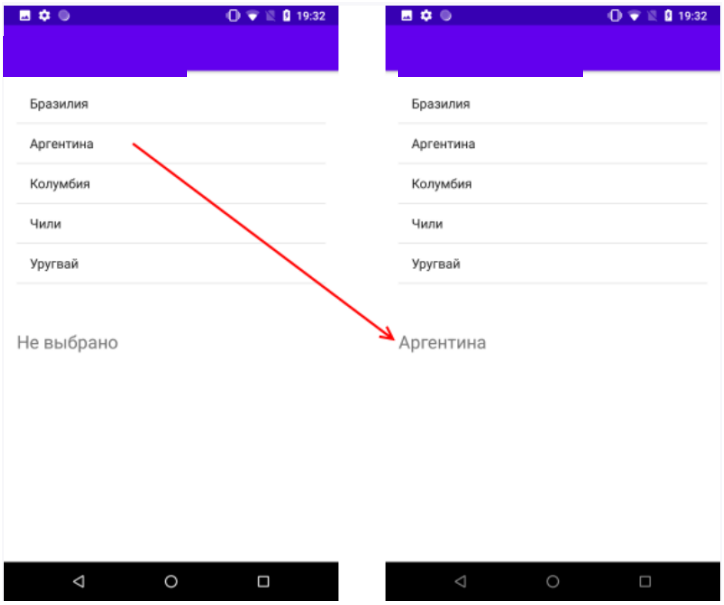
І насамкінець змінимо код MainActivity:

```
        package com.example.fragmentapp;
        import androidx.appcompat.app.AppCompatActivity;
        import android.os.Bundle;
        public class MainActivity extends AppCompatActivity implements
ListFragment.OnFragmentSendDataListener {
        @Override
        protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        }
        @Override
        public void onSendData(String selectedItem) {
        DetailFragment fragment = (DetailFragment)
getSupportFragmentManager()
        .findFragmentById(R.id.detailFragment);
        if (fragment! = null)
        fragment.setSelectedItem(selectedItem);
        }
        }
```

Для взаємодії фрагмента ListFragment з іншим фрагментом через MainActivity треба, щоб ця activity реалізовувала інтерфейс OnFragmentSendDataListener. Для цього реалізуємо методonSendData(), який отримує фрагмент DetailFragment і викликає у нього методsetSelectedItem()

В результаті вийде, що при виборі у списку у фрагменті ListFragment спрацьовуватиме слухач списку і зокрема його методonItemClick(AdapterView<?> parent, View v, int position, long id), який викличе методfragmentSendDataListener.onSendData(selectedItem);. fragmentSendDataListener встановлюється як MainActivity, тому буде викликаний метод setSelectedItem у фрагмента DetailFragment. Таким чином відбудеться взаємодія між двома фрагментами.

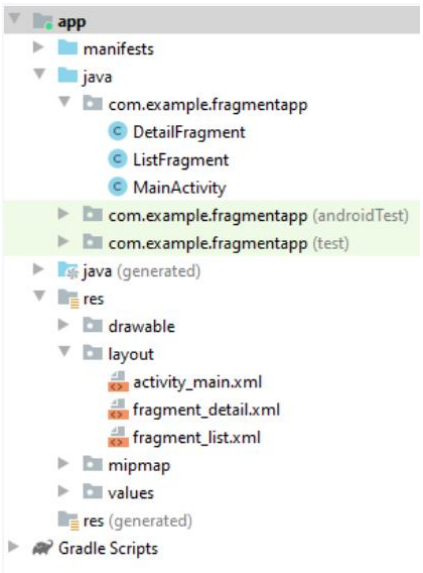
Якщо ми запустимо проект, то на екран буде виведено обидва фрагменти, які зможуть взаємодіяти між собою.



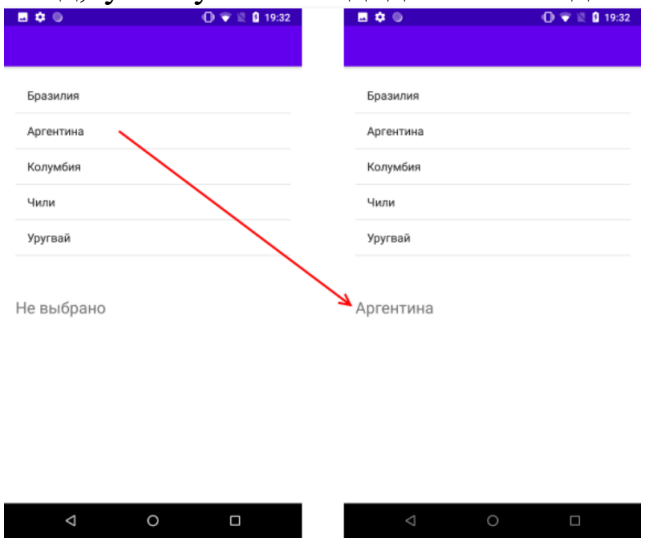
Однак поки фрагменти однаково виводяться в одній діяльності як в альбомній, так і в портретній орієнтації незалежно від пристрою. Тому оптимізуємо програму.

3.6. Фрагменти в альбомному та портретному режимі

У минулій темі було розроблено програму, яка виводить обидва фрагменти на екран. Продовжимо роботу із цим проектом. Усього було створено два фрагменти: ListFragment для відображення списку та DetailFragment для відображення вибраного елемента у списку. І MainActivity виводила обидва фрагменти на екран:



Але відображення двох і більше фрагментів за портретної орієнтації не дуже оптимально. Наприклад, у минулій темі додаток виглядав так:

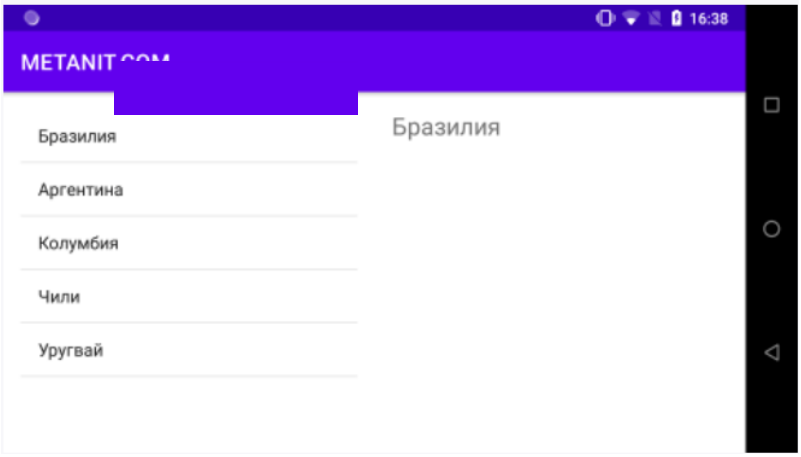


Але якщо список великий, другий фрагмент, який відображає вибраний елемент, відповідно йде вниз. При альбомній орієнтації вийде розташування

ще неоптимальніше. Тому спочатку змінимо файл activity_main.xml, щоб зручніше розташовувати фрагменти в альбомній орієнтації:

```
<?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" >
<androidx.fragment.app.FragmentContainerView
android:id="@+id/listFragment"
android:layout_width="0dp"
android:layout_height="0dp"
android:name="com.example.fragmentapp.ListFragment"
app:layout_constraintLeft_toLeftOf="parent"
app:layout_constraintTop_toTopOf="parent"
app:layout_constraintRight_toLeftOf="@id/detailFragment"
app:layout_constraintBottom_toBottomOf="parent" />
<androidx.fragment.app.FragmentContainerView
android:id="@+id/detailFragment"
android:layout_width="0dp"
android:layout_height="0dp"
android:name="com.example.fragmentapp.DetailFragment"
app:layout_constraintLeft_toRightOf="@id/listFragment"
app:layout_constraintTop_toTopOf="parent"
app:layout_constraintBottom_toBottomOf="parent"
app:layout_constraintRight_toRightOf="parent"/>
</androidx.constraintlayout.widget.ConstraintLayout>
```

Для зручнішого розташування при альбомної орієнтації, зазвичай, рішення досить просте - два фрагменти розташовуються горизонтально в ряд.

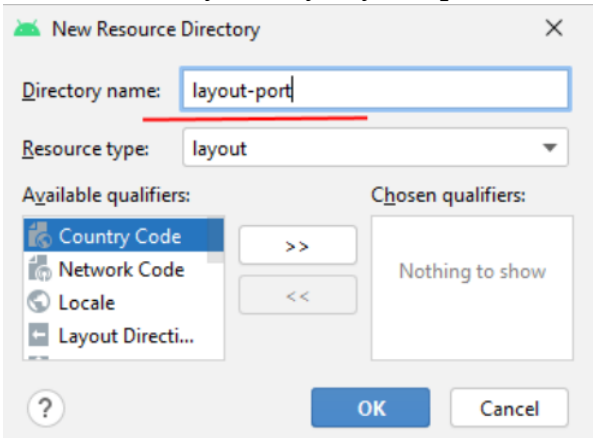


Тепер розглянемо як найбільш успішно розмістити фрагменти при портретної орієнтації. Нерідко у разі рішення наступне - одномоментно екран відображає лише один фрагмент.

Отже, створимо в проекті папки res, де зберігаються всі ресурси, підкаталог layout-port, який зберігатиме файли інтерфейсу для портретної

орієнтації. Для цього перейдемо до повного виду проекту. Натисніть на папку res правою кнопкою миші і в контекстному меню виберемо New -> Android Resource Directory:

Назвемо нову папку layout-port:

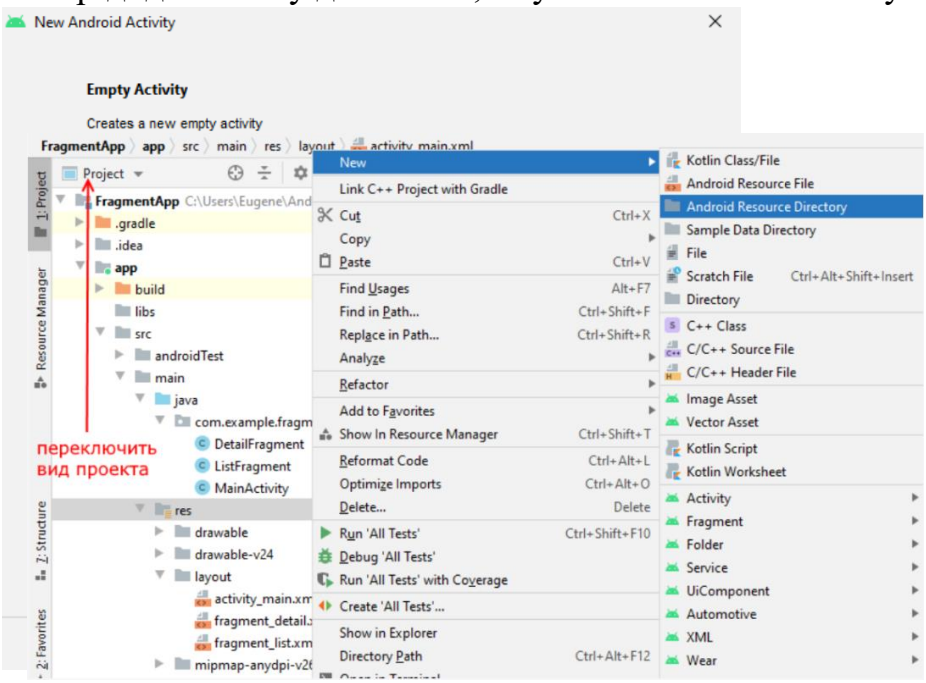


Далі додамо в res/layout-port новий файл activity_main.xml і визначимо наступний код:

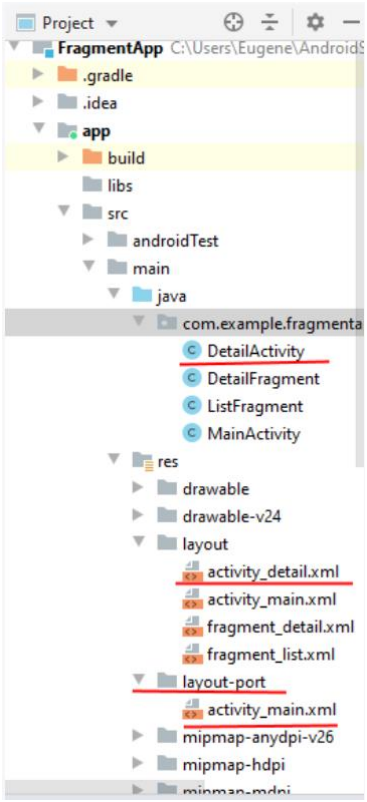
```
<?xml version="1.0" encoding="utf-8"?>
<androidx.fragment.app.FragmentContainerView
xmlns:android="http://schemas.android.com/apk/res/android"
android:id="@+id/listFragment"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:name="com.example.fragmentapp.ListFragment" />
```

Цей файл використовуватиметься для портретної орієнтації MainActivity. Таким чином, MainActivity у портретному режимі відображатиме лише один список.

Тепер додамо нову діяльність, яку назвемо DetailActivity:



У результаті проект виглядатиме так:



У папці res/layout у файлі activity_detail.xml визначимо для DetailActivity наступний інтерфейс:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.fragment.app.FragmentContainerView
xmlns:android="http://schemas.android.com/apk/res/android"
android:id="@+id/detailFragment"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:name="com.example.fragmentapp.DetailFragment" />
```

Таким чином, інтерфейс DetailActivity буде визначатися фрагментом DetailFragment, що завантажується.

Далі у коді DetailActivity визначимо наступний код:

```
package com.example.fragmentapp;
import androidx.appcompat.app.AppCompatActivity;
import android.content.res.Configuration;
import android.os.Bundle;
public class DetailActivity extends AppCompatActivity {
public static final String SELECTED_ITEM = "SELECTED_ITEM";
String selectedItem = "Не вибрано";
@Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
if (getResources().getConfiguration().orientation ==
Configuration.ORIENTATION_LANDSCAPE) {
finish();
return;
```

```
        }
        setContentView(R.layout.activity_detail);
        Bundle extras = getIntent().getExtras();
        if (extras! = null)
        selectedItem = extras.getString(SELECTED_ITEM);
        }
        @Override
        protected void onResume() {
        super.onResume();
        DetailFragment fragment = (DetailFragment)
getSupportFragmentManager()
        .findFragmentById(R.id.detailFragment);
        if (fragment! = null)
        fragment.setSelectedItem(selectedItem);
        }
        }
```

Тут насамперед перевіряємо конфігурацію. Так як ця діяльність призначена тільки для портретного режиму, то при альбомній орієнтації здійснюємо вихід:

```
        if (getResources().getConfiguration().orientation ==
Configuration.ORIENTATION_LANDSCAPE) {
        finish();
        return;
        }
```

Якщо пристрій знаходиться в портретному режимі, отримуємо передані дані за ключом "SELECTED_ITEM":

```
        Bundle extras = getIntent().getExtras();
        if (extras! = null)
        selectedItem = extras.getString(SELECTED_ITEM);
```

Передбачається, що за ключом "SELECTED_ITEM" передаватиметься вибраний елемент списку з MainActivity, коли він перебуватиме в портретній орієнтації.

І дуже важливий момент – нам треба передати це значення у текстове поле, визначене у фрагменті. Проте треба враховувати особливості життєвого циклу уявлення фрагмента. У цьому випадку перевизначається метод onResume(), тому що при викликі цього методу DetailActivity вже буде видно на екрані, і користувач зможе з нею взаємодіяти. А це також означає, що в цій точці вже буде активним і фрагмент та його уявлення. Наприклад, у методіonCreate()представлення фрагмента ще повністю створено, тому ми можемо у ньому отримати віджети, визначені у фрагменті. Зате можемо все це зробити у методіonResume().

```
        protected void onResume() {
        super.onResume();
        DetailFragment fragment = (DetailFragment)
getSupportFragmentManager()
```

```
.findFragmentById(R.id.detailFragment);
if (fragment! = null)
fragment.setSelectedItem(selectedItem);
}
```

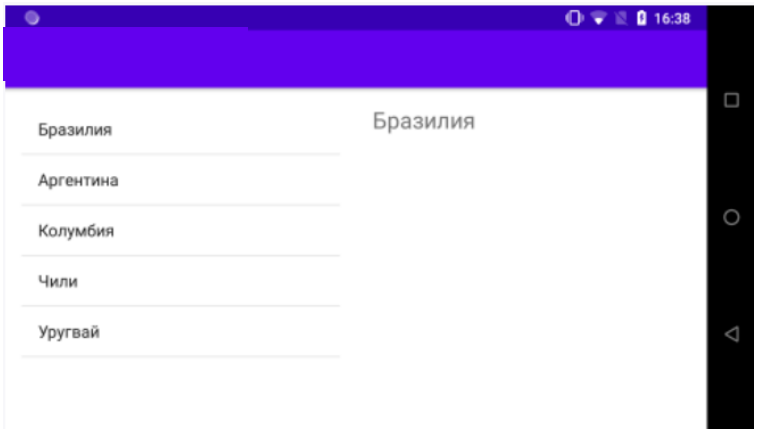
І в даному випадку ми отримуємо через методgetSupportFragmentManager()фрагмент DetailFragment і викликає його метод setSelectedItem(). Як аргумент цього методу передається рядкове значення, передане через Intent.

І також змінимо головну діяльність - MainActivity:

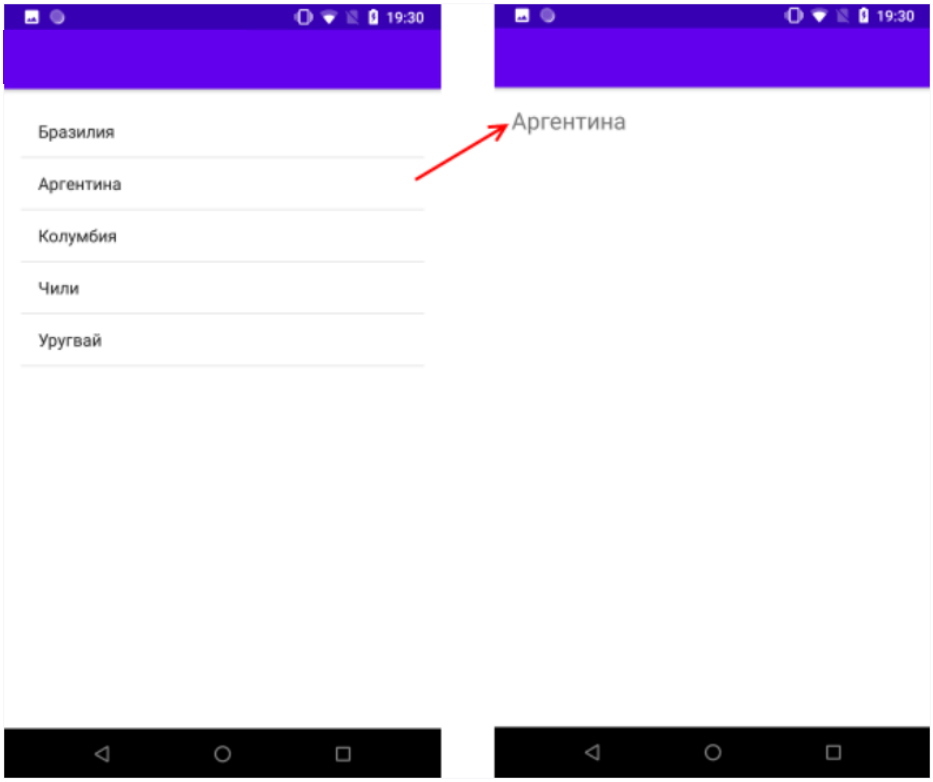
```
package com.example.fragmentapp;
import androidx.appcompat.app.AppCompatActivity;
import android.content.Intent;
import android.os.Bundle;
public class MainActivity extends AppCompatActivity implements
ListFragment.OnFragmentSendDataListener {
@Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_main);
}
@Override
public void onSendData(String selectedItem) {
DetailFragment fragment = (DetailFragment)
getSupportFragmentManager()
.findFragmentById(R.id.detailFragment);
if (fragment! = null && fragment.isVisible())
fragment.setSelectedItem(selectedItem);
else {
Intent intent = new Intent(getApplicationContext(),
DetailActivity.class);
intent.putExtra(DetailActivity.SELECTED_ITEM, selectedItem);
startActivity(intent);
}
}
}
```

За допомогою методуfragment.isVisible()ми можемо дізнатися, чи активний певний фрагмент у розмітці інтерфейсу. Якщо фрагмента DetailFragment в даний момент не бачимо, то використовується портретний режим, тому запускається DetailActivity. Інакше йде робота з фрагментом усередині MainActivity, яка в альбомному режимі відображає відразу два фрагменти – ListFragment та DetailFragment.

Запустимо програму і перейдемо в альбомний режим:



А при портретній орієнтації екран виглядатиме інакше:



4. НАЛАШТУВАННЯ ТА СТАН ПРОГРАМИ

4.1. Збереження стану програми

В одній з попередніх тем був розглянутий життєвий цикл Activity у додатку на Android, де після створення Activity викликався метод onRestoreInstanceState, який відновлював її стан, а перед завершенням роботи викликався метод onSaveInstanceState, який зберігав стан Actiity. Обидва ці методи як параметр приймають об'єкт Bundle, який якраз і зберігає стан activity:

```
protectedvoid onRestoreInstanceState(Bundle  saveInstanceState);
protectedvoid onSaveInstanceState(Bundle  saveInstanceState);
```

У якій ситуації може бути доречним використання подібних методів? Банальна ситуація - переворот екрану та перехід від портретної орієнтації до альбомної та навпаки. Якщо, наприклад, графічний інтерфейс містить текстове поле виведення TextView, і ми програмно змінюємо його текст, після зміни орієнтації екрана його може зникнути. Або якщо у нас глобальні змінні, то при зміні орієнтації екрана їх значення можуть бути скинуті до значень за замовчуванням.

Щоб точніше зрозуміти проблему, з якою ми можемо мати справу, розглянемо приклад. Змінимо файл activity_main таким чином:

```
<?xmlversion="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/nameBox"
android:layout_width="0dp"
android:layout_height="wrap_content"
android:hint="Введіть ім'я"
app:layout_constraintBottom_toTopOf="@id/saveButton"
app:layout_constraintLeft_toLeftOf="parent"
app:layout_constraintRight_toRightOf="parent"
app:layout_constraintTop_toTopOf="parent"/>
<Button
android:id="@+id/saveButton"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Зберегти"
android:onClick="saveName"
app:layout_constraintBottom_toTopOf="@id/nameView"
app:layout_constraintLeft_toLeftOf="parent"
app:layout_constraintTop_toBottomOf="@id/nameBox"/>
```

```
<TextView
android:id="@+id/nameView"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:textSize="18sp"
app:layout_constraintBottom_toTopOf="@id/getButton"
app:layout_constraintLeft_toLeftOf="parent"
app:layout_constraintTop_toBottomOf="@id/saveButton"/>
<Button
android:id="@+id/getButton"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Отримати ім'я"
android:onClick="getName"
app:layout_constraintLeft_toLeftOf="parent"
app:layout_constraintTop_toBottomOf="@id/nameView"/>
</androidx.constraintlayout.widget.ConstraintLayout>
```

Тут визначено поле EditText, яке вводимо ім'я. І також визначено кнопку для його збереження.

Далі для виведення збереженого імені призначено поле TextView, а отримання збереженого імені - друга кнопка.

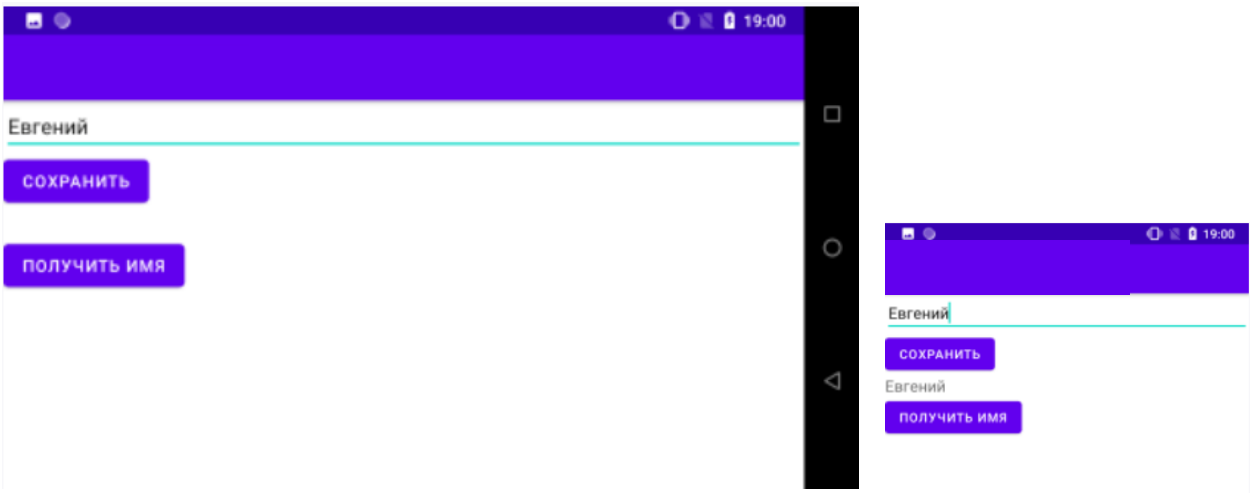
Тепер змінимо клас MainActivity:

```
packagecom.example.settingsapp;
importandroidx.appcompat.app.AppCompatActivity;
importandroid.os.Bundle;
importandroid.view.View;
importandroid.widget.EditText;
importandroid.widget.TextView;
publicclass MainActivity extends AppCompatActivity {
String name="undefined";
@Override
protectedvoid onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_main);
}
publicvoid saveName(View view) {
// Отримуємо введене ім'я
EditText nameBox = findViewById(R.id.nameBox);
name = nameBox.getText().toString();
}
publicvoid getName(View view) {

// отримуємо збережене ім'я
TextView nameView = findViewById(R.id.nameView);
nameView.setText(name);
}
}
```

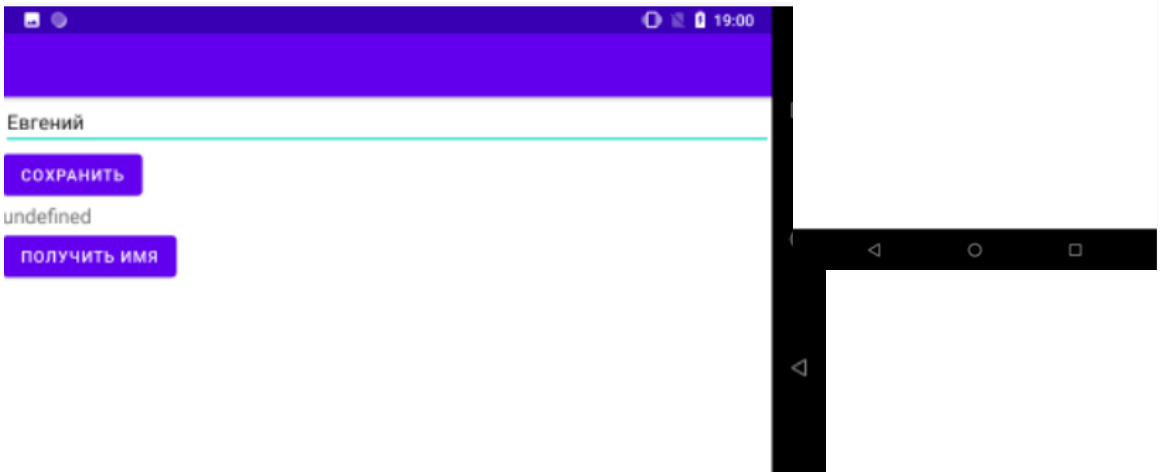
Для збереження імені у програмі визначено змінну name. При натисканні на першу кнопку зберігаємо текст із EditText у змінну name, а при натисканні на другу кнопку – назад отримуємо текст із змінної name у полі TextView.

Запустимо програму введемо якесь ім'я, збережемо і отримаємо його в TextView:

Але якщо ми перейдемо до альбомного режиму, TextView виявиться порожнім, незважаючи на те, що в нього начебто вже отримали потрібне значення:



І навіть якщо ми спробуємо наново отримати значення зі змінної name, ми побачимо, що вона обнулилася:



Щоб уникнути подібних ситуацій якраз і слід зберігати та відновлювати стан діяльності. Для цього змінимо код MainActivity:

```
packagecom.example.settingsapp;
importandroidx.appcompat.app.AppCompatActivity;
importandroid.os.Bundle;
importandroid.view.View;
importandroid.widget.EditText;
importandroid.widget.TextView;
publicclass MainActivity extends AppCompatActivity {
```

```
String name="undefined";
finalstatic String nameVariableKey = "NAME_VARIABLE";
TextView nameView;
@Override
protectedvoid onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_main);
nameView = findViewById(R.id.nameView);
}
// Збереження стану
@Override
protectedvoid onSaveInstanceState(Bundle outState) {
outState.putString(nameVariableKey, name);
super.onSaveInstanceState(outState);
}
// отримання раніше збереженого стану
@Override
protectedvoid onRestoreInstanceState(Bundle savedInstanceState) {
super.onRestoreInstanceState(savedInstanceState);
name = savedInstanceState.getString(nameVariableKey);
nameView.setText(name);
}
publicvoid saveName(View view) {
// Отримуємо введене ім'я
EditText nameBox = findViewById(R.id.nameBox);
// Зберігаємо його в змінну name
name = nameBox.getText().toString();
}
publicvoid getName(View view) {
// виводимо збережене ім'я
nameView.setText(name);
}
}
```

У методі onSaveInstanceState() зберігаємо стан. Для цього викликаємо у параметра Bundle метод putString(key, value), перший параметр якого - ключ, а другий - значення даних, що зберігаються. У разі ми зберігаємо рядок, тому викликаємо метод putString(). Для збереження об'єктів інших типів даних ми можемо викликати відповідний метод:

- **put()**: універсальний метод, який додає значення Object. Відповідно поле одержання дане значення необхідно перетворити до потрібного типу
- **putString()**: додає об'єкт типу String
- **putInt()**: додає значення типу int
- **putByte()**: додає значення типу byte
- **putChar()**: додає значення типу char
- **putShort()**: додає значення типу short

- **putLong()**: додає значення типу long
- **putFloat()**: додає значення типу float
- **putDouble()**: додає значення типу double
- **putBoolean()**: додає значення типу boolean
- **putCharArray()**: додає масив об'єктів char
- **putIntArray()**: додає масив об'єктів int
- **putFloatArray()**: додає масив об'єктів float
- **putSerializable()**: додає об'єкт інтерфейсу Serializable
- **putParcelable()**: додає об'єкт Parcelable

Кожен такий метод також як перший параметр приймає ключа, а як другий - значення.

У методі onRestoreInstanceState відбувається зворотний процес - за допомогою методу getString(key) по ключу отримуємо зі збереженого стану рядок по ключу. Відповідно, для отримання даних інших типів ми можемо використовувати аналогічні методи:

- **get()**: Універсальний метод, який повертає значення типу Object. Відповідно поле одержання дане значення необхідно перетворити до потрібного типу
- **getString()**: повертає об'єкт типу String
- **getInt()**: повертає значення типу int
- **getByte()**: повертає значення типу byte
- **getChar()**: повертає значення типу char
- **getShort()**: повертає значення типу short
- **getLong()**: повертає значення типу long
- **getFloat()**: повертає значення типу float
- **getDouble()**: повертає значення типу double
- **getBoolean()**: повертає значення типу boolean
- **getCharArray()**: повертає масив об'єктів char
- **getIntArray()**: повертає масив об'єктів int
- **getFloatArray()**: повертає масив об'єктів float
- **getSerializable()**: повертає об'єкт інтерфейсу Serializable
- **getParcelable()**: повертає об'єкт Parcelable

Наприклад розглянемо збереження-одержання складніших даних. Наприклад, об'єкти певного класу. Нехай у нас є клас User:

```
packagecom.example.settingsapp;
importjava.io.Serializable;
publicclass User implements Serializable {
privateString name;
privateint age;
publicUser(String name, int age) {
this.name = name;
this.age = age;
}
publicString getName() {
returnname;
}
publicvoid setName(String name) {
this.name = name;
}
publicint getAge() {
returnage;
}
publicvoid setAge(int age) {
this.age = age;
}
}
```

Клас User реалізує інтерфейс Serializable, тому ми можемо зберегти його об'єкти за допомогою методу putSerializable(), а отримати за допомогою методу getSerializable().

Нехай у нас буде наступний інтерфейс activity_main.xml:

```
<?xmlversion="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/nameBox"
android:layout_width="0dp"
android:layout_height="wrap_content"
android:hint="Введіть ім'я"
app:layout_constraintBottom_toTopOf="@id/yearBox"
app:layout_constraintLeft_toLeftOf="parent"
app:layout_constraintRight_toRightOf="parent"
app:layout_constraintTop_toTopOf="parent"/>
<EditText
android:id="@+id/yearBox"
android:layout_width="0dp"
android:layout_height="wrap_content"
android:hint="Введіть вік"
```

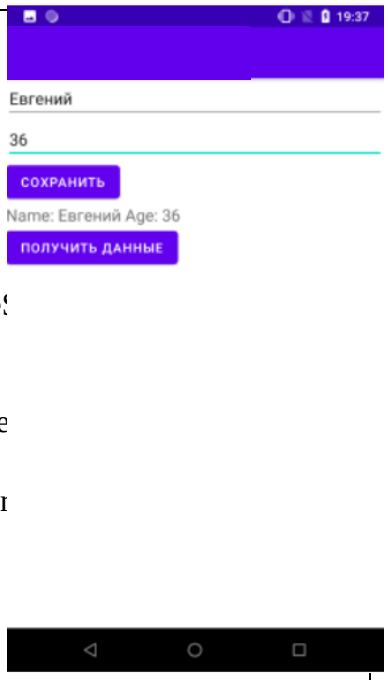
```
android:inputType="numberDecimal"
app:layout_constraintBottom_toTopOf="@id/saveButton"
app:layout_constraintLeft_toLeftOf="parent"
app:layout_constraintRight_toRightOf="parent"
app:layout_constraintTop_toTopOf="parent"/>
<Button
android:id="@+id/saveButton"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Зберегти"
android:onClick="saveData"
app:layout_constraintBottom_toTopOf="@id/dataView"
app:layout_constraintLeft_toLeftOf="parent"
app:layout_constraintTop_toBottomOf="@id/yearBox"/>
<TextView
android:id="@+id/dataView"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:textSize="18sp"
app:layout_constraintBottom_toTopOf="@id/getButton"
app:layout_constraintLeft_toLeftOf="parent"
app:layout_constraintTop_toBottomOf="@id/saveButton"/>
<Button
android:id="@+id/getButton"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Отримати дані"
android:onClick="getData"
app:layout_constraintLeft_toLeftOf="parent"
app:layout_constraintTop_toBottomOf="@id/dataView"/>
</androidx.constraintlayout.widget.ConstraintLayout>
```

Тут визначено два поля введення для імені та віку відповідно.

У класі MainActivity пропишемо логіку збереження та отримання даних:

```
packagecom.example.settingsapp;
importandroidx.appcompat.app.AppCompatActivity;
importandroid.os.Bundle;
importandroid.view.View;
importandroid.widget.EditText;
importandroid.widget.TextView;
publicclass MainActivity extends AppCompatActivity {
User user=newUser("undefined", 0);
finalstatic String userVariableKey = "USER_VARIABLE";
@Override
protectedvoid onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_main);
}
// Збереження стану
```

```
@Override
protectedvoid onSaveInstanceState(Bundle  outState) {
outState.putSerializable(userVariableKey,  user);
super.onSaveInstanceState(outState);
}
// отримання раніше збереженого стану
@Override
protectedvoid onRestoreInstanceState(Bundle  savedInstance
super.onRestoreInstanceState(savedInstanceState);
// отримуємо об'єкт User у змінну
user = (User)savedInstanceState.getSerializable(userVariable
TextView  dataView = findViewById(R.id.dataView);
dataView.setText("Name:"+  user.getName()  + "Age:" + user
}
publicvoid saveData(View view) {

// Отримуємо введені дані
EditText nameBox  = findViewById(R.id.nameBox);
EditText yearBox = findViewById(R.id.yearBox);
String name  = nameBox.getText().toString();
intage = 0; // Значення за замовчуванням, якщо користувач ввів некоректні дані
try{
age = Integer.parseInt(yearBox.getText().toString());
}
catch(NumberFormatException  ex){}
user=newUser(name,  age);
}
publicvoid getData  (View view) {
// отримуємо збережені дані
TextView  dataView = findViewById(R.id.dataView);
dataView.setText("Name:"+  user.getName()  + "Age:" + user.getAge());
```



Тут також зберігаємо дані змінну User, яка попередньо ініціалізована деякими даними за умовчанням. А при натисканні на кнопку отримання отримаємо дані зі змінної і передаємо їх для виведення в текстове поле.

4.2. Створення та отримання настройок SharedPreferences

Нерідко програмі потрібно зберігати невеликі шматочки даних для подальшого використання, наприклад, дані про користувача, налаштування конфігурації тощо. Для цього в Android існує концепція Preferences або налаштування. Налаштування є групою пар ключ-значення, які використовуються програмою.

Як значення можуть виступати дані таких типів: Boolean, Float, Integer, Long, String, набір рядків.

Налаштування спільними для всіх activity в додатку, але також можуть бути і налаштування безпосередньо для окремих activity

Налаштування зберігаються у файлах xml у незашифрованому вигляді у локальному сховищі. Вони невидимі, тому для простого користувача недоступні.

При роботі з налаштуваннями слід враховувати наступні моменти. Оскільки вони зберігаються у незашифрованому вигляді, то не рекомендується зберігати в них чутливі дані типу пароля чи номерів кредитних карток. Крім того, вони представляють дані, асоційовані з програмою, і через панель управління програмою в Налаштуваннях ОС користувач може видалити ці дані.

4.3. Загальні налаштування

Для роботи з налаштуваннями, що розділяються в класі Activity (точніше в його базовому класі Context) є метод getSharedPreferences():

```
importandroid.content.SharedPreferences;
//.....
SharedPreferences settings = getSharedPreferences("PreferencesName", MODE_PRIVATE);
```

Перший параметр методу вказує назву налаштувань. У цьому випадку назва - "PreferencesName". Якщо настройок з такою назвою немає, вони створюються при виклікі даного методу. Другий параметр вказує на режим доступу. В даному випадку режим описаний константою MODE_PRIVATE

Клас android.content.SharedPreferences надає ряд методів для керування налаштуваннями:

- contains(String key): повертає true, якщо в налаштуваннях збережено значення з ключем key
- getAll(): повертає всі збережені в установках значення
- getBoolean (String key, boolean defValue): повертає значення типу Boolean, яке має ключ key. Якщо елемент з таким ключем не виявиться, то повертається значення defValue, яке передається другим параметром
- getFloat(String key, float defValue): повертає значення типу float із ключем key. Якщо елемент з таким ключем не виявиться, то повертається значення defValue

- getInt(String key, int defValue): повертає значення типу int із ключем key
- getLong(String key, long defValue): повертає значення типу long із ключем key
- getString(String key, String defValue): повертає строкове значення з ключем key
- getStringSet(String key, Set<String> defValues): повертає масив рядків із ключем key
- edit(): повертає об'єкт SharedPreferences.Editor, який використовується для редагування налаштувань

Для керування налаштуваннями використовується об'єкт класу SharedPreferences.Editor, метод edit(), що повертається. Він визначає такі методи:

- clear(): видаляє всі настройки
- remove(String key): видаляє значення з ключем key.
- putBoolean(String key, boolean value): додає до налаштувань значення типу boolean з ключем key
- putFloat(String key, float value): додає в налаштування значення типу float з ключем key
- putInt(String key, int value): додає в налаштування значення int з ключем key
- putLong(String key, long value): додає до налаштувань значення типу long з ключем key
- putString(String key, String value): додає в налаштування рядок з ключем key
- putStringSet(String key, Set<String> values): додає в налаштування строковий масив
- commit(): підтверджує всі зміни в установках
- apply(): також, як і метод commit(), підтверджує всі зміни в налаштуваннях, проте змінений об'єкт SharedPreferences спочатку зберігається в тимчасовій пам'яті, і лише потім в результаті асинхронної операції записується на мобільний пристрій

Розглянемо приклад збереження та отримання налаштувань у програмі. Визначимо у файлі activity_main.xml наступний інтерфейс користувача:

```
<?xmlversion="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/nameBox"
android:layout_width="0dp"
android:layout_height="wrap_content"
android:hint="Введіть ім'я"
app:layout_constraintBottom_toTopOf="@id/saveButton"
app:layout_constraintLeft_toLeftOf="parent"
```

```
app:layout_constraintRight_toRightOf="parent"
app:layout_constraintTop_toTopOf="parent"/>
<Button
android:id="@+id/saveButton"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Зберегти"
android:onClick="saveName"
app:layout_constraintBottom_toTopOf="@id/nameView"
app:layout_constraintLeft_toLeftOf="parent"
app:layout_constraintTop_toBottomOf="@id/nameBox"/>
<TextView
android:id="@+id/nameView"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:textSize="18sp"
app:layout_constraintBottom_toTopOf="@id/getButton"
app:layout_constraintLeft_toLeftOf="parent"
app:layout_constraintTop_toBottomOf="@id/saveButton"/>
<Button
android:id="@+id/getButton"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Отримати ім'я"
android:onClick="getName"
app:layout_constraintLeft_toLeftOf="parent"
app:layout_constraintTop_toBottomOf="@id/nameView"/>
</androidx.constraintlayout.widget.ConstraintLayout>
```

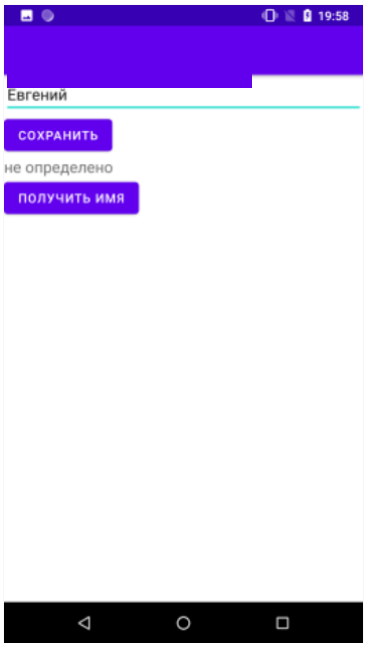
На екрані будуть дві кнопки - для збереження та для виведення раніше збереженого значення, а також поле для введення та текстове поля для виведення збереженої установки.

Визначимо методи обробників кнопок у класі MainActivity:

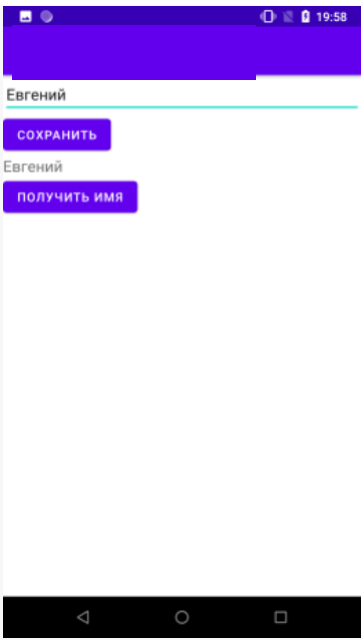
```
packagecom.example.settingsapp;
importandroidx.appcompat.app.AppCompatActivity;
importandroid.content.SharedPreferences;
importandroid.os.Bundle;
importandroid.view.View;
importandroid.widget.EditText;
importandroid.widget.TextView;
publicclass MainActivity extends AppCompatActivity {
privatestatic final String PREFS_FILE="Account";
privatestatic final String PREF_NAME = "Name";
SharedPreferences settings;
@Override
protectedvoid onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_main);
settings = getSharedPreferences(PREFS_FILE, MODE_PRIVATE);
}
```

```
publicvoid saveName(View view) {
// Отримуємо введене ім'я
EditText nameBox = findViewById(R.id.nameBox);
String name = nameBox.getText().toString();
// Зберігаємо його в налаштуваннях
SharedPreferences.Editor prefEditor = settings.edit();
prefEditor.putString(PREF_NAME, name);
prefEditor.apply();
}
publicvoid getName(View view) {
// отримуємо збережене ім'я
TextView nameView = findViewById(R.id.nameView);
String name = settings.getString(PREF_NAME,"не визначено");
nameView.setText(name);
}
}
```

За відсутності налаштувань при спробі їх отримати, програма виведе значення за промовчанням:



Тепер збережемо та виведемо заново збережене значення:



Нерідко виникає завдання автоматично зберігати дані, що вводяться при виході користувача з activity. Для цього ми можемо перевизначити метод onPause:

```
packagecom.example.settingsapp;
importandroidx.appcompat.app.AppCompatActivity;
importandroid.content.SharedPreferences;
importandroid.os.Bundle;
importandroid.view.View;
importandroid.widget.EditText;
publicclass MainActivity extends AppCompatActivity {
privatestatic final String PREFS_FILE="Account";
privatestatic final String PREF_NAME = "Name";
EditText nameBox;
SharedPreferences settings;
SharedPreferences.Editor prefEditor;
@Override
protectedvoid onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_main);
nameBox = findViewById(R.id.nameBox);
settings = getSharedPreferences(PREFS_FILE, MODE_PRIVATE);
// отримуємо налаштування
String name = settings.getString(PREF_NAME,"");
nameBox.setText(name);
}
@Override
protectedvoid onPause(){
super.onPause();
String name = nameBox.getText().toString();
// зберігаємо у налаштуваннях
prefEditor = settings.edit();
prefEditor.putString(PREF_NAME, name);
prefEditor.apply();
```

```
}
publicvoid saveName(View view) {
}
publicvoid getName(View view) {
}
}
```

4.4. Приватні налаштування

Крім загальних настройок, кожна activity може використовувати приватні, до яких доступ з інших activity буде неможливий. Для отримання налаштувань рівня activity використовується метод getPreferences(MODE_PRIVATE):

```
importandroid.content.SharedPreferences;
//.....
SharedPreferences settings = getPreferences(MODE_PRIVATE);
```

Тобто, на відміну від загальних налаштувань, тут не використовується назва групи налаштувань як перший параметр, як у методі getSharedPreferences(). Однак вся інша робота з додавання, отримання та зміни налаштувань буде аналогічна до роботи із загальними налаштуваннями.

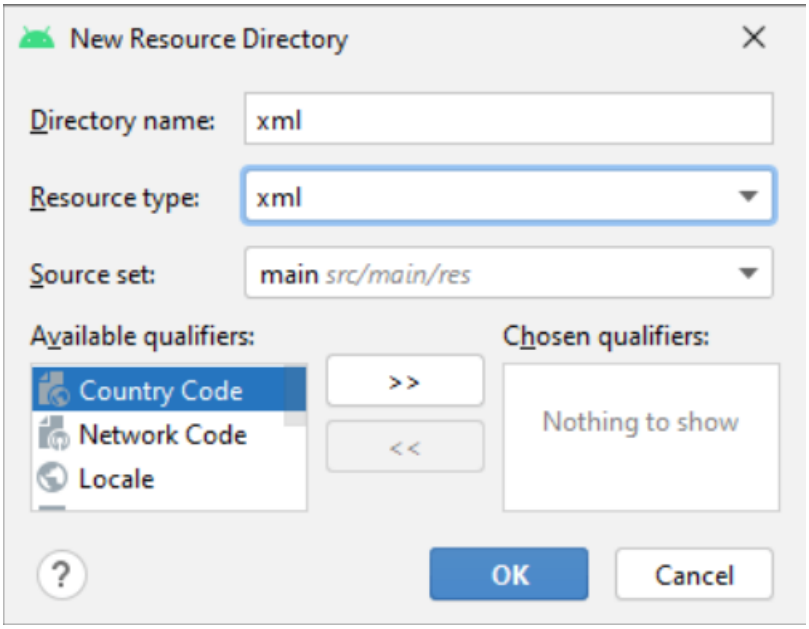
4.5. PreferenceFragmentCompat

Для спрощення роботи з групою налаштувань Android надає спеціальний тип фрагмента – PreferenceFragmentCompat. Розглянемо, як її використовувати.

Створимо новий проект і спочатку визначимо у файлі build.gradle необхідні залежності для роботи з PreferenceFragmentCompat:

```
implementation "androidx.fragment:fragment:1.3.6"
implementation "androidx.preference:preference:1.1.1"
```

Для визначення налаштувань додамо до папки res підпапку xml.



Потім до папки res/xml додамо новий файл, який назвемо settings.xml. І змінимо його так:

```
<?xmlversion="1.0" encoding="utf-8"?>
<PreferenceScreenxmlns:android=
"http://schemas.android.com/apk/res/android">
<EditTextPreference
android:key="login"
android:summary="Введіть логін"
android:title="Логін"/>
<CheckBoxPreference
android:key="enabled"
android:summary="Відображати логін"
android:title="Відображати"/>
</PreferenceScreen>
```

Тут у кореневому елементі PreferenceScreen встановлюються елементи EditTextPreference та CheckBoxPreference. Через кожен із цих елементів ми можемо взаємодіяти з певним налаштуванням.

Взагалі, у цьому випадку ми можемо використовувати ряд різних типів налаштувань:

- EditTextPreference: використовується елемент EditText для введення текстового значення
- CheckBoxPreference: використовується елемент CheckBox для встановлення логічних значень true або false
- SwitchPreference: використовується елемент Switch для встановлення логічних значень true або false ("on" та "off")
- RingtonePreference: використовує діалогове вікно для встановлення рінгтону зі списку рінгтонів для встановлення логічних значень true або false
- ListPreference: використовує список для вибору одного з визначених значень
- MultiSelectListPreference: також використовує список для вибору значень, але дозволяє вибрати кілька елементів

Для кожного елемента налаштування необхідно визначити щонайменше три атрибути:

- android:key: встановлює ключ налаштування у SharedPreferences
- android:title: назва налаштування для користувача
- android:summary: короткий опис цієї настройки для користувача

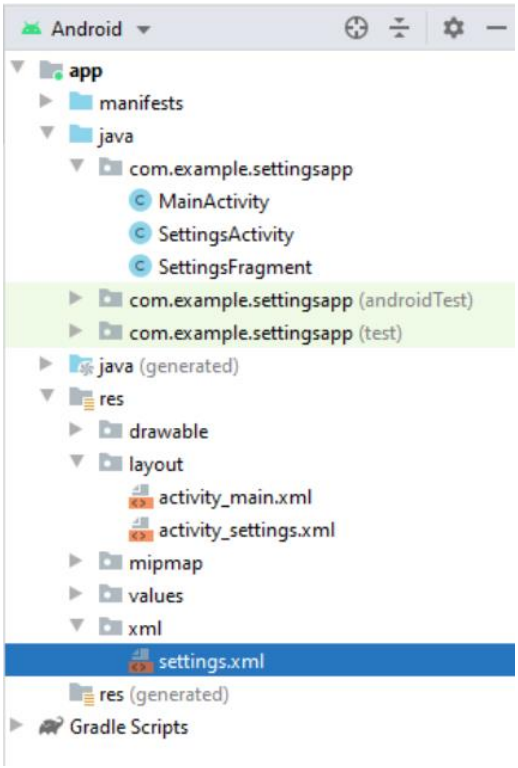
Далі додамо новий клас Java, який назвемо SettingsFragment:

```
packagecom.example.settingsapp;
importandroid.os.Bundle;
importandroidx.preference.PreferenceFragmentCompat;
publicclass SettingsFragment extends PreferenceFragmentCompat {
@Override
publicvoid onCreatePreferences(Bundle savedInstanceState, String rootKey) {
addPreferencesFromResource(R.xml.settings);
```

```
    }
    }
```

Фрагмент SettingsFragment успадковується від класу PreferenceFragmentCompat. У його методі onCreatePreferences викликається метод addPreferencesFromResource(), який передається id ресурсу xml з налаштуваннями (у разі раніше певний ресурс R.xml.settings).

І тепер додамо до проекту спеціальну activity для встановлення налаштувань. Назвемо її SettingsActivity. У результаті проект виглядатиме так:



У файлі layout для SettingsActivity - activity_settings.xml пропишемо наступний інтерфейс:

```
    <?xmlversion="1.0" encoding="utf-8"?>
    <androidx.fragment.app.FragmentContainerView
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/settings_container"
    android:layout_width="match_parent"
    android:layout_height="match_parent"/>
```

Тут визначено FragmentContainerView з id = settings_container - саме той елемент, в який завантажуватиметься фрагмент SettingsFragment.

У коді SettingsActivity визначимо завантаження фрагмента:

```
    packagecom.example.settingsapp;
    importandroidx.appcompat.app.AppCompatActivity;
    importandroid.os.Bundle;
    publicclass SettingsActivity extends AppCompatActivity {
    @Override
    protectedvoid onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_settings);
    getSupportFragmentManager()
```

```
.beginTransaction()
.replace(R.id.settings_container, newSettingsFragment())
. commit ();
}
}
```

SettingsActivity як розмітка інтерфейсу буде використовувати ресурс R.layout.activity_settings.

При запуску SettingsActivity завантажуватиме фрагмент SettingsFragment в елемент з id settings_container.

Далі перейдемо до головної діяльності - MainActivity. У файлі activity_main.xml визначимо текстове поле та кнопку:

```
<?xmlversion="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/settingsText"
android:layout_width="0dp"
android:layout_height="wrap_content"
android:textSize="18sp"
app:layout_constraintBottom_toTopOf="@id/settingsButton"
app:layout_constraintLeft_toLeftOf="parent"
app:layout_constraintRight_toRightOf="parent"
app:layout_constraintTop_toTopOf="parent"/>
<Button
android:id="@+id/settingsButton"
android:layout_width="0dp"
android:layout_height="wrap_content"
android:text="Налашгування"
android:onClick="setPrefs"
app:layout_constraintTop_toBottomOf="@id/settingsText"
app:layout_constraintLeft_toLeftOf="parent"
app:layout_constraintRight_toRightOf="parent"/>

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

І змінимо клас MainActivity:

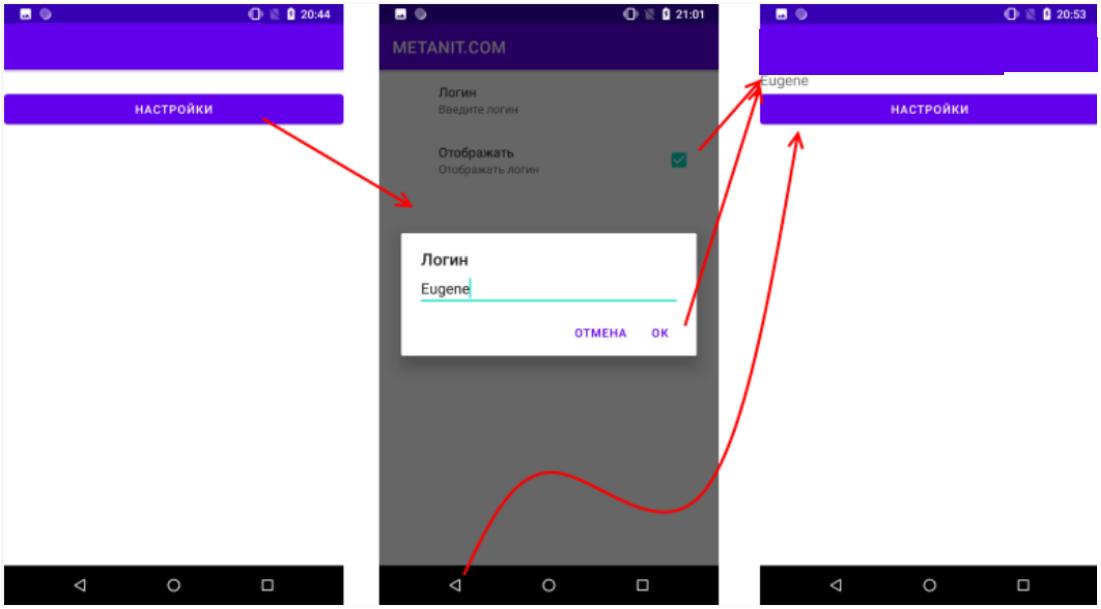
```
packagecom.example.settingsapp;
importandroidx.appcompat.app.AppCompatActivity;
importandroidx.preference.PreferenceManager;
importandroid.content.Intent;
importandroid.content.SharedPreferences;
importandroid.os.Bundle;
importandroid.view.View;
importandroid.widget.TextView;
```

```
publicclass MainActivity extends AppCompatActivity {
TextView settingsText;
booleanenabled;
String login;
@Override
protectedvoid onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_main);

settingsText = findViewById(R.id.settingsText);
}
@Override
publicvoid onResume() {
super.onResume();
SharedPreferences prefs=PreferenceManager.getDefaultSharedPreferences(this);
enabled = prefs.getBoolean("enabled", false);
login = prefs.getString("login", "не встановлено");
settingsText.setText(login);
if(enabled)
settingsText.setVisibility(View.VISIBLE);
else
settingsText.setVisibility(View.INVISIBLE);
}
publicvoid setPrefs(View view){
Intent intent = newIntent(this, SettingsActivity.class);
startActivity(intent);
}
}
```

У методі onResume() отримуємо всі налаштування. Якщо налаштування enabled одно true, відображаємо текстове поле з логіном.

У методі setPrefs(), який спрацьовує при натисканні на кнопку, виконується перехід до SettingsActivity.

При першому запуску настройок не буде, і логін не відображатиметься. Перейдемо на сторінку налаштувань і встановимо там логін та включимо його відображення, а потім повернемося на головну діяльність:



При цьому вручну нам нічого не треба зберігати, всі налаштування автоматично зберігаються функціоналом PreferenceFragmentCompat.

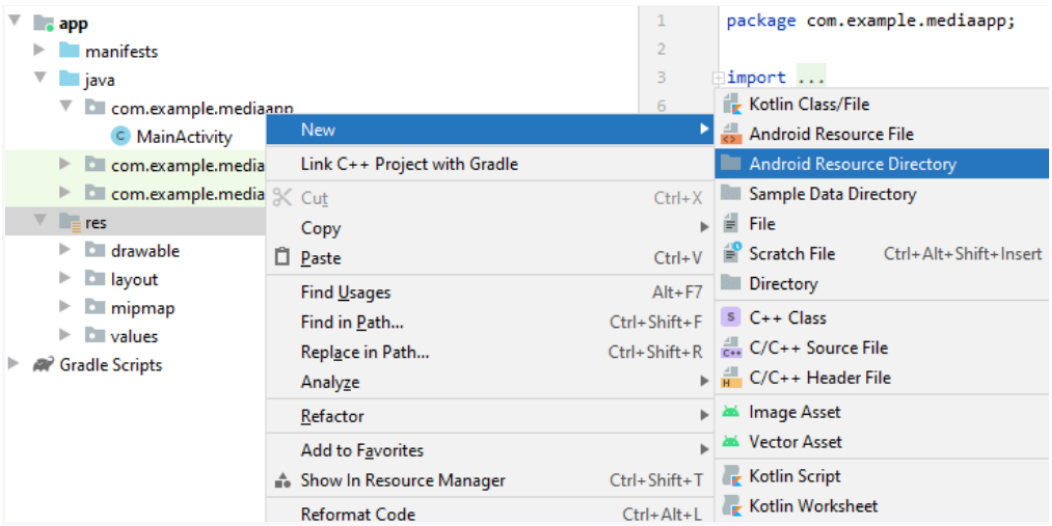
5. РОБОТА З МУЛЬТИМЕДІА

5.1. Робота з відео

Для роботи з відеоматеріалами у стандартному наборі віджетів Android визначено клас VideoView, що дозволяє відтворювати відео.

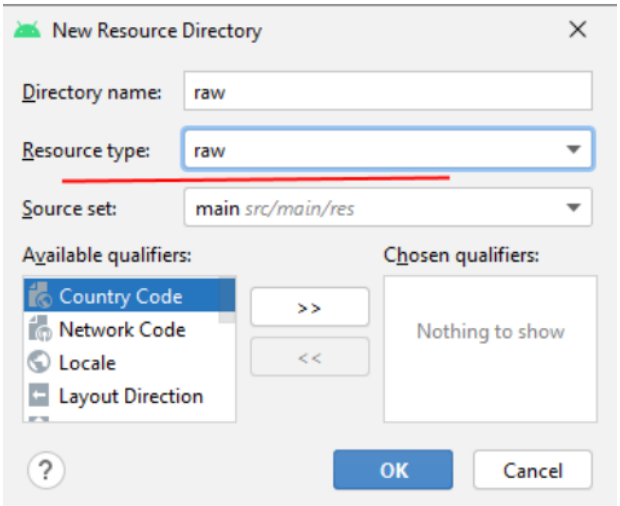
Які типи відеофайлів можна використовувати? Android підтримує більшість поширених типів відеофайлів, зокрема 3GPP (.3gp), WebM (.webm), Matroska (.mkv), MPEG-4 (.mp4).

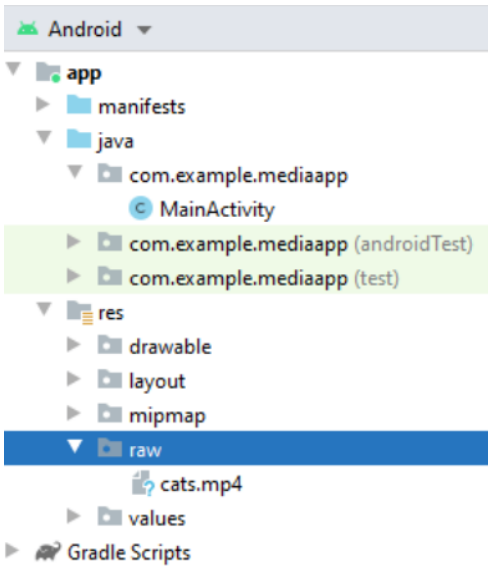
VideoView може працювати як з роликами, розміщеними на мобільному пристрої, так і відеоматеріалами з мережі. У цьому випадку використовуємо відеоролик, розміщений локально. Для цього додамо в проект якийсь відеоролик. Зазвичай відеоматеріали поміщають у проекті папку res/raw. За замовчуванням проект не містить такої папки, тому додамо до каталогу res підпапку raw. Для цього натиснемо на папку res правою кнопкою миші і в меню виберемо New -> Android Resource Directory:



Потім у вікні як тип папки вкажемо raw (що також буде використовуватися як назва папки):

Після додавання папки raw скопіюємо в неї якийсь відеофайл:





Тепер визначимо функціонал його відтворення. Для цього у файлі activity_main.xml вкажемо наступний код:

```
<?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">
<Button
android:id="@+id/playButton"
android:layout_width="0dp"
android:layout_height="wrap_content"
android:text="Play"
android:onClick="play"
app:layout_constraintBottom_toTopOf="@id/videoPlayer"
app:layout_constraintLeft_toLeftOf="parent"
app:layout_constraintRight_toLeftOf="@id/pauseButton"
app:layout_constraintTop_toTopOf="parent"  />
<Button
android:id="@+id/pauseButton"
android:layout_width="0dp"
android:layout_height="wrap_content"
android:text="Pause"
android:onClick="pause"
app:layout_constraintBottom_toTopOf="@id/videoPlayer"
app:layout_constraintLeft_toRightOf="@id/playButton"
app:layout_constraintRight_toLeftOf="@id/stopButton"
app:layout_constraintTop_toTopOf="parent"/>
<Button
android:id="@+id/stopButton"
android:layout_width="0dp"
android:layout_height="wrap_content"
android:text="Stop"
```

```
        android:onClick="stop"
        app:layout_constraintBottom_toTopOf="@id/videoPlayer"
        app:layout_constraintLeft_toRightOf="@id/pauseButton"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
        <VideoView android:id="@+id/videoPlayer"
        android:layout_height="0dp"
        android:layout_width="0dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toBottomOf="@id/playButton"/>

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

Для управління відтворенням відео тут визначено три кнопки: для запуску відео, для паузи та його зупинки.

Також змінимо код MainActivity:

```
        package com.example.mediaapp;
        import androidx.appcompat.app.AppCompatActivity;
        import android.net.Uri;
        import android.os.Bundle;
        import android.view.View;
        import android.widget.VideoView;
        public class MainActivity extends AppCompatActivity {
        VideoView VideoPlayer;
        @Override
        protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        videoPlayer = findViewById(R.id.videoPlayer);
        Uri myVideoUri= Uri.parse(" android.resource://" + getPackageName() + "/" +
R.raw.cats);
        videoPlayer.setVideoURI(myVideoUri);
        }
        public void play(View view){
        videoPlayer.start();
        }
        public void pause(View view) {
        videoPlayer.pause();
        }
        public void stop(View view){
        videoPlayer.stopPlayback();
        videoPlayer.resume();
        }
        }
```

По-перше, щоб керувати потоком відтворення, нам треба отримати об'єкт VideoView:videoPlayer = findViewById(R.id.videoPlayer);

Щоб вказати джерело відтворення, потрібний об'єкт Uri. В даному випадку за допомогою виразуUri myVideoUri= Uri.parse("android.resource://" + getPackageName() + "/" + R.raw.cats);отримуємо адресу відео усередині пакета програми.

Рядок URI має ряд частин: спочатку йдеUri-схема(http://або як тутandroid.resource://), потім назва пакета, одержуване через метод getPackageName(), і далі безпосередньо назва ресурсу відео з папки res/raw, яке збігається з назвою файлу.

Потім цей Uri встановлюється у VideoPlayer:videoPlayer.setVideoURI(myVideoUri);

Щоб керувати відеопотоком, обробники натискання кнопок викликають відповідну дію:

```
public void play(View view){
videoPlayer.start();
}
public void pause(View view) {
videoPlayer.pause();
}
public void stop(View view){
videoPlayer.stopPlayback();
videoPlayer.resume();
}
```

МетодvideoPlayer.start()починає або продовжує відтворення.

МетодvideoPlayer.pause()зупиняє відео.

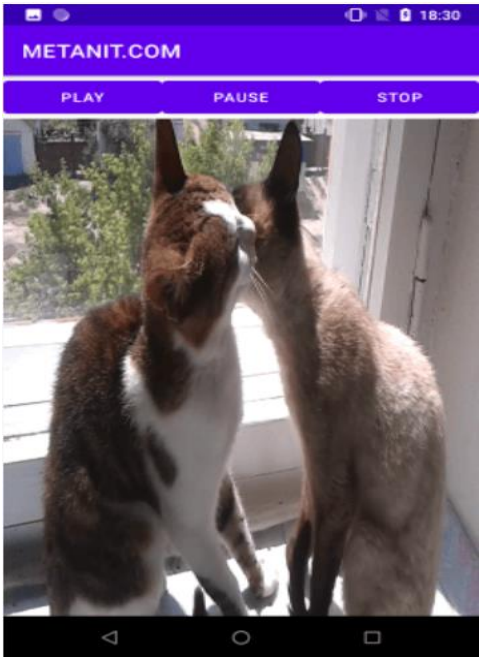
МетодvideoPlayer.stopPlayback()повністю зупиняє відео.

МетодvideoPlayer.resume()дозволяє знову розпочати відтворення відео з початку після його повної зупинки.

При запуску програми ми зможемо за допомогою відтворенням:

5.2. **MediaController**

За допомогою класу MediaController ми можемо додати до VideoView додаткові елементи керування. Для цього змінимо код MainActivity:



```
package com.example.mediaapp;
import androidx.appcompat.app.AppCompatActivity;
import android.net.Uri;
import android.os.Bundle;
import android.view.View;
import android.widget.VideoView;
import android.widget.MediaController;
public class MainActivity extends AppCompatActivity {
VideoView VideoPlayer;
@Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_main);
videoPlayer = findViewById(R.id.videoPlayer);
Uri myVideoUri= Uri.parse(" android.resource://" + getPackageName() + "/" +
R.raw.cats);
videoPlayer.setVideoURI(myVideoUri);
MediaController mediaController = New MediaController(this);
videoPlayer.setMediaController(mediaController);
mediaController.setMediaPlayer(videoPlayer);
}
public void play(View view){
videoPlayer.start();
}
public void pause(View view) {
videoPlayer.pause();
}
public void stop(View view){
videoPlayer.stopPlayback();
videoPlayer.resume();
}
}
```

І якщо ми запустимо програми, то при дотику VideoView внизу з'являться інструменти для керування відео. У принципі тепер і кнопки, які ми створили раніше, не потрібні:

5.3. **Відтворення файлу з Інтернету**

VideoView підтримує відтворення файлу з Інтернету. Але щоб це стало можливо, необхідно у файлі AndroidManifest.xml встановити дозвіл android.permission.INTERNET, тому що ми отримуємо дані з інтернету:

```
<uses-permission android:name="android.permission.INTERNET" />
```

Далі змінимо клас MainActivity:

```
package com.example.mediaapp;
import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.VideoView;
public class MainActivity extends AppCompatActivity {
VideoView VideoPlayer;
@Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_main);
videoPlayer = findViewById(R.id.videoPlayer);
videoPlayer.setVideoPath("http://techslides.com/demos/sample- videos/small.mp4");
}
public void play(View view){
videoPlayer.start();
}
public void pause(View view) {
videoPlayer.pause();
}
public void stop(View view){
videoPlayer.stopPlayback();
videoPlayer.resume();
```

```
    }
}
```

Тут нам треба в методvideoPlayer.setVideoPath()передати інтернет-адресу файлу, що відтворюється.

Відтворення аудіо

Для відтворення музики та інших аудіоматеріалів Android надає клас MediaPlayer.

Щоб відтворювати аудіо, MediaPlayer повинен знати, який ресурс (файл) потрібно виробляти. Встановити потрібний ресурс для відтворення можна трьома способами:

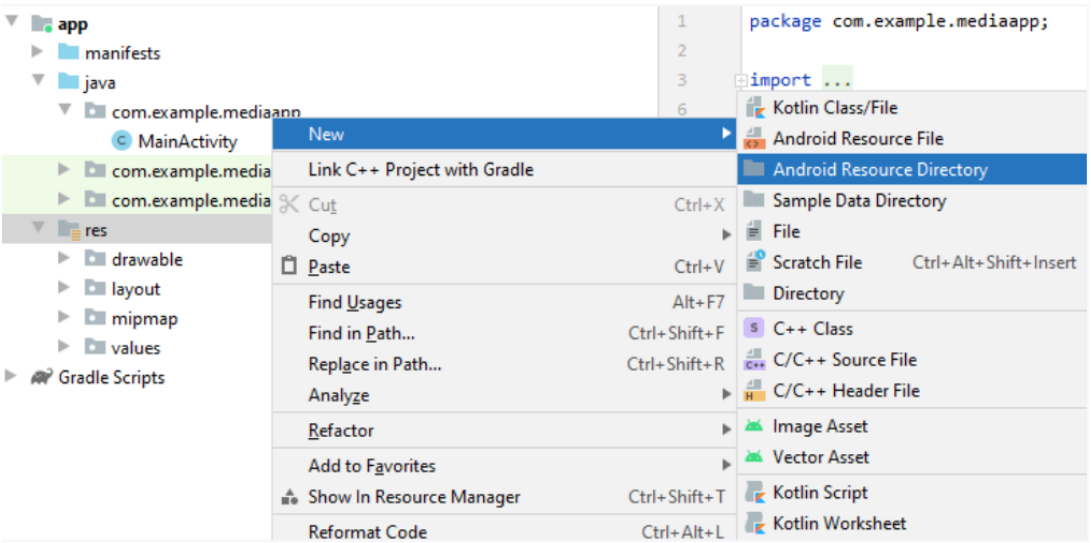
- у методcreate()об'єкта MediaPlayer передається id ресурсу, що представляє аудіо-файл
- у методcreate()об'єкта MediaPlayer передається об'єкт Uri, що представляє аудіо-файл
- у методsetDataSource()об'єкта MediaPlayer передається повний шлях до аудіофайлу

Після встановлення ресурсу викликається методprepare()абоprepareAsync()(Асинхронний варіант prepare()). Цей метод готує аудіофайл до відтворення, витягуючи з нього перші секунди. Якщо ми відтворюємо файл з мережі, краще використовувати prepareAsync().

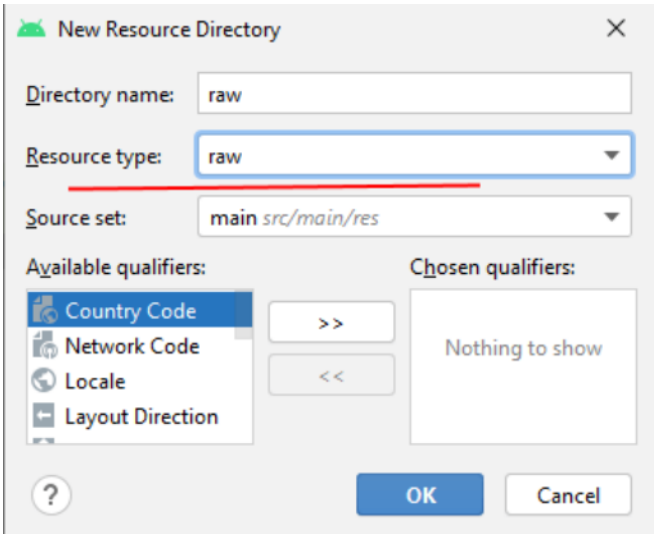
Для керування відтворенням у класі MediaPlayer визначено такі методи:

- start(): запускає аудіо
- pause(): призупиняє відтворення
- stop(): повністю зупиняє відтворення

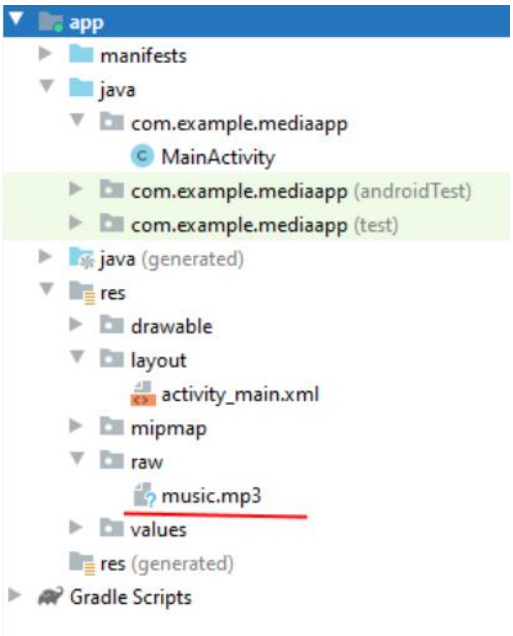
Отже, створимо новий проект. Як і у випадку з відео, аудіофайл повинен знаходитись у папці res/raw, тому додамо в проект в Android Studio таку папку. Для цього натиснемо на папку res правою кнопкою миші і в меню виберемо New -> Android Resource Directory:



Потім у вікні як тип папки вкажемо raw (що також буде використовуватися як назва папки):



І скопіюємо в неї якийсь аудіо-файл.



Для керування аудіопотоком визначимо у файлі activity_main.xml три кнопки:

```
<?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">
<Button
android:id="@+id/playButton"
android:layout_width="0dp"
android:layout_height="wrap_content"
android:text="Play"
android:onClick="play"
app:layout_constraintLeft_toLeftOf="parent"
```

```
app:layout_constraintRight_toLeftOf="@id/pauseButton"
app:layout_constraintTop_toTopOf="parent" />
<Button
android:id="@+id/pauseButton"
android:layout_width="0dp"
android:layout_height="wrap_content"
android:text="Pause"
android:onClick="pause"
app:layout_constraintLeft_toRightOf="@id/playButton"
app:layout_constraintRight_toLeftOf="@id/stopButton"
app:layout_constraintTop_toTopOf="parent"/>
<Button
android:id="@+id/stopButton"
android:layout_width="0dp"
android:layout_height="wrap_content"
android:text="Stop"
android:onClick="stop"
app:layout_constraintLeft_toRightOf="@id/pauseButton"
app:layout_constraintRight_toRightOf="parent"
app:layout_constraintTop_toTopOf="parent" />

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

І змінимо код класу MainActivity:

```
package com.example.mediaapp;
import androidx.appcompat.app.AppCompatActivity;
import android.media.MediaPlayer;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.Toast;
public class MainActivity extends AppCompatActivity {

MediaPlayer mPlayer;
button playbutton, pausebutton, stopbutton;
@Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_main);
mPlayer = MediaPlayer.create (this, R.raw.music);
mPlayer.setOnCompletionListener(new MediaPlayer.OnCompletionListener() {
@Override
public void onCompletion(MediaPlayer mp) {
stopPlay();
}
});
playButton = findViewById(R.id.playButton);
pauseButton = findViewById(R.id.pauseButton);
stopButton = findViewById(R.id.stopButton);
```

```
pauseButton.setEnabled(false);
stopButton.setEnabled(false);
}
private void stopPlay(){
mPlayer.stop();
pauseButton.setEnabled(false);
stopButton.setEnabled(false);
try {
mPlayer.prepare();
mPlayer.seekTo(0);
playButton.setEnabled(true);
}
catch (Throwable t) {
Toast.makeText(this, t.getMessage(), Toast.LENGTH_SHOR
}
}
public void play(View view){
mPlayer.start();
playButton.setEnabled(false);
pauseButton.setEnabled(true);
stopButton.setEnabled(true);
}
public void pause(View view) {
mPlayer.pause();
playButton.setEnabled(true);
pauseButton.setEnabled(false);
stopButton.setEnabled(true);
}
public void stop(View view){
stopPlay();
}
@Override
public void onDestroy() {
super.onDestroy();
if (mPlayer.isPlaying()) {
stopPlay();
}
}
}
```

Обробник кожної кнопки крім виклику певного методу MediaPlayer також перемикає доступність кнопок.

І якщо запуск та призупинення відтворення особливих складнощів не викликає, то при обробці повної зупинки відтворення ми можемо зіткнутися з низкою труднощів. Зокрема, коли ми виходимо з програми - повністю закриваємо його через диспетчер додатків або натискаємо кнопку Назад, то у нас для поточної Activity викликається метод onDestroy, activity знищується, але MediaPlayer продовжує працювати. Якщо ми повернемося до програми, то activity буде створено заново, але за допомогою кнопок ми

не зможемо керувати відтворенням. Тому в даному випадку перевизначаємо метод onDestroy, в якому завершуємо відтворення.

Для коректного завершення також визначено обробника природного завершення відтворенняНаCompletionListener, Дія якого буде аналогічна натисканню на кнопку "Стоп".

Додамо до відтворення індикатор гучності. Для цього у файлі activity_main.xml визначимо SeekBar:

```
<?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">
<Button
android:id="@+id/playButton"
android:layout_width="0dp"
android:layout_height="wrap_content"
android:text="Play"
android:onClick="play"
app:layout_constraintLeft_toLeftOf="parent"
app:layout_constraintRight_toLeftOf="@id/pauseButton"
app:layout_constraintTop_toTopOf="parent"
app:layout_constraintBottom_toTopOf="@id/volumeControl" />
<Button
android:id="@+id/pauseButton"
android:layout_width="0dp"
android:layout_height="wrap_content"
android:text="Pause"
android:onClick="pause"
app:layout_constraintLeft_toRightOf="@id/playButton"
app:layout_constraintRight_toLeftOf="@id/stopButton"
app:layout_constraintTop_toTopOf="parent"/>
<Button
android:id="@+id/stopButton"
android:layout_width="0dp"
android:layout_height="wrap_content"
android:text="Stop"
android:onClick="stop"
app:layout_constraintLeft_toRightOf="@id/pauseButton"
app:layout_constraintRight_toRightOf="parent"
app:layout_constraintTop_toTopOf="parent" />
<SeekBar
android:id="@+id/volumeControl"
android:layout_width="0dp"
android:layout_height="wrap_content"
android:layout_marginTop="32dp"
app:layout_constraintTop_toBottomOf="@id/playButton"
```

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

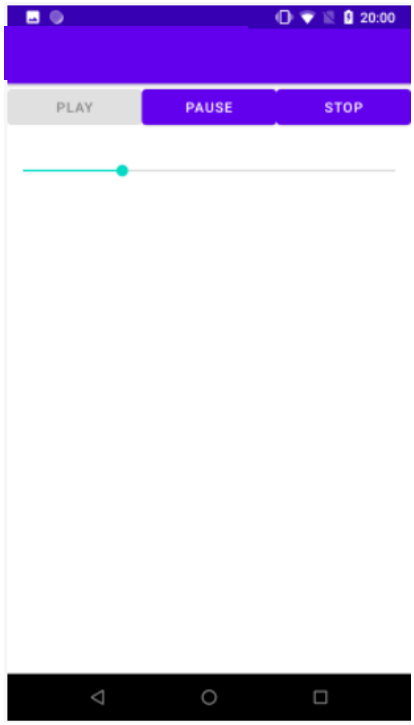
І далі змінимо код класу MainActivity:

```
package com.example.mediaapp;
import androidx.appcompat.app.AppCompatActivity;
import android.content.Context;
import android.media.AudioManager;
import android.media.MediaPlayer;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.SeekBar;
import android.widget.Toast;
public class MainActivity extends AppCompatActivity {
MediaPlayer mPlayer;
button playbutton, pausebutton, stopbutton;
SeekBar volumeControl;
AudioManager audioManager;
@Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_main);

mPlayer=MediaPlayer.create(this, R.raw.music);
mPlayer.setOnCompletionListener(new
 MediaPlayer.OnCompletionListener() {
@Override
public void onCompletion(MediaPlayer mp) {
stopPlay();
}
});
playButton = findViewById(R.id.playButton);
pauseButton = findViewById(R.id.pauseButton);
stopButton = findViewById(R.id.stopButton);

audioManager = (AudioManager)
getSystemService(Context.AUDIO_SERVICE);
int maxVolume =
audioManager.getStreamMaxVolume(AudioManager.STREAM_MUSIC);
int curValue =
audioManager.getStreamVolume(AudioManager.STREAM_MUSIC);
volumeControl = findViewById(R.id.volumeControl);
volumeControl.setMax(maxVolume);
volumeControl.setProgress(curValue);
volumeControl.setOnSeekBarChangeListener(new
 SeekBar.OnSeekBarChangeListener() {
@Override
```

```
public void onProgressChanged(SeekBar  seekBar, int
 progress, boolean fromUser) {
audioManager.setStreamVolume(AudioManager.STREAM_MUSIC,
progress, 0);
}
@Override
public void onStartTrackingTouch(SeekBar  seekBar) {
}
@Override
public void onStopTrackingTouch(SeekBar  seekBar) {
}
});
pauseButton.setEnabled(false);
stopButton.setEnabled(false);
}
private void stopPlay(){
mPlayer.stop();
pauseButton.setEnabled(false);
stopButton.setEnabled(false);
try {
mPlayer.prepare();
mPlayer.seekTo(0);
playButton.setEnabled(true);
}
catch (Throwable t) {
Toast.makeText(this,  t.getMessage(),
Toast.LENGTH_SHORT).show();
}
}
public void play(View view){
mPlayer.start();
playButton.setEnabled(false);
pauseButton.setEnabled(true);
stopButton.setEnabled(true);
}
public void pause(View view) {

mPlayer.pause();
playButton.setEnabled(true);
pauseButton.setEnabled(false);
stopButton.setEnabled(true);
}
public void stop(View view){
stopPlay();
}
@Override
public void onDestroy() {
super.onDestroy();
if (mPlayer.isPlaying()) {
```

```
        stopPlay();
        }
        }
        }
```

Для керування гучністю звуку застосовується клас AudioManager. А в за допомогою викликуaudioManager.setStreamVolume(AudioManager.STREAM_MUSIC, progress, 0);як другий параметр можна передати потрібне значення гучності.



6.

6. Робота із мережею. WebView

6.1. WebView

WebView представляє найпростіший елемент для рендерингу html-коду, що базується на движку WebKit. Завдяки цьому ми можемо використовувати WebView як примітивний веб-браузер, переглядаючи через нього контент із Інтернету. Використання движка WebKit гарантує, що відображення контенту буде відбуватися приблизно також, як і в інших браузерах, побудованих на цьому движку - Google Chrome і Safari.

Деякі основні методи класу WebView:

- boolean canGoBack(): повертає true, якщо перед поточною веб-сторінкою в історії навігації WebView ще є сторінки
- boolean canGoForward(): повертає true, якщо після поточної веб-сторінки в історії навігації WebView ще є сторінки
- clearCache(boolean includeDiskFiles): очищує кеш WebView
- clearFormData(): очищає даний автозаповнення полів форм.
- clearHistory(): очищає історію навігації
- String getUrl(): повертає адресу поточної веб-сторінки
- void goBack(): переходить до попередньої веб-сторінки в історії навігації
- void goForward(): переходить до наступної веб-сторінки в історії навігації
- void loadData(String data, String mimeType, String encoding): завантажує у веб-браузері дані у вигляді html-коду, використовуючи вказаний mime-тип та кодування
- void loadDataWithBaseURL (String baseUrl, String data, String mimeType, String encoding, String historyUrl): також завантажує у веб-браузері дані у вигляді html-коду, використовуючи вказаний mime-тип та кодування, як і метод loadData(). Однак крім того, як перший параметр приймає валідну адресу, з якою асоціюється завантажені дані.

Навіщо цей метод потрібен, якщо є loadData()? Вміст, завантажуваний методом loadData(), як значення для window.origin буде мати значення null, і таким чином, джерело вмісту, що завантажується, не зможе пройти перевірку на достовірність. Метод loadDataWithBaseURL() з валідними адресами (протокол може бути HTTP і HTTPS) дозволяє встановити джерело вмісту.

- void loadUrl(String url): завантажує веб-сторінку за певною адресою
- void postUrl(String url, byte[] postData): надсилає дані за допомогою запиту типу "POST" за певною адресою
- void zoomBy(float zoomFactor): змінює масштаб на визначений коефіцієнт
- boolean zoomIn(): збільшує масштаб
- boolean zoomOut(): зменшує масштаб

Працювати із WebView дуже просто. Визначимо цей елемент у розмітці layout:

```
<?xmlversion="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">
<WebView
android:id="@+id/webBrowser"
android:layout_width="0dp"
android:layout_height="0dp"
app:layout_constraintLeft_toLeftOf="parent"
app:layout_constraintRight_toRightOf="parent"
```

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

Для отримання доступу до інтернету з програми, необхідно вказати у файлі маніфесту AndroidManifest.xml відповідний дозвіл:

```
<uses-permissionandroid:name="android.permission.INTERNET"/>
```

Щоб завантажити певну сторінку WebView, через метод loadUrl() треба встановити її адресу:

```
packagecom.example.viewapp;
importandroidx.appcompat.app.AppCompatActivity;
importandroid.os.Bundle;
importandroid.webkit.WebView;

publicclass MainActivity extends AppCompatActivity {
@Override
protectedvoid onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_main);
WebView browser=findViewById(R.id.webBrowser);
browser.loadUrl("https://timetable.lond.lg.ua/");
}
}
```

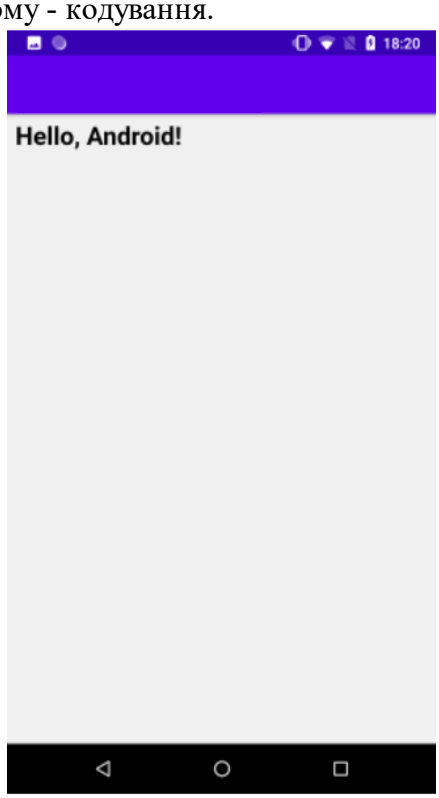
Замість визначення елемента в layout ми можемо створити WebView у коді Activity:

```
WebView browser=newWebView(this);
setContentView(browser);
browser.loadUrl("https://timetable.lond.lg.ua/");
```

Окрім завантаження конкретної сторінки з інтернету за допомогою методу loadData():

```
WebView browser = findViewById (R.id.webBrowser);
browser.loadData("<html><body><h2>Hello, Android!</h2></body></html>", "text/html", "UTF-8");
```

Першим параметром метод приймає рядок коду html, у другому - тип вмісту, а третьому - кодування.



6.2. JavaScript

За замовчуванням у WebView вимкнено javascript, щоб його включити треба застосувати метод setJavaScriptEnabled(true) об'єкта WebSettings:

```
importandroid.webkit.WebSettings;
//.....
WebView browser = findViewById(R.id.webBrowser);
WebSettings webSettings = browser.getSettings();
webSettings.setJavaScriptEnabled(true);
```

Завантаження даних та клас HttpURLConnection

На сьогоднішній день якщо не всі, більшість Android-пристроїв мають доступ до мережі інтернет. А велика кількість мобільних додатків так чи інакше взаємодіють із середовищем інтернету: завантажують файли, авторизуються та отримують інформацію із зовнішніх веб-сервісів тощо. Розглянемо, як ми можемо використовувати у своєму додатку доступ до Інтернету.

Серед стандартних елементів нам доступний віджет WebView, який може завантажувати контент із певної URL-адреси. Але цим можливості роботи з мережею Android не обмежуються. Для отримання даних із певного інтернет-ресурсу ми можемо використовувати класи HttpUrlConnection (для протоколу HTTP) та HttpsUrlConnection (для протоколу HTTPS) із стандартної бібліотеки Java.

Отже, створимо новий проект із порожньою MainActivity. Насамперед для роботи з мережею нам треба встановити у файлі маніфесту AndroidManifest.xml відповідний дозвіл:

```
<uses-permission android:name="android.permission.INTERNET"/>
```

У файлі activity_main.xml, який представляє розмітку MainActivity, визначимо наступний код:

```
<?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" >
<Button
android:id="@+id/downloadBtn"
android:layout_width="0dp"
android:layout_height="wrap_content"
android:text="Завантаження"
app:layout_constraintLeft_toLeftOf="parent"
app:layout_constraintRight_toRightOf="parent"
app:layout_constraintTop_toTopOf="parent" />
<WebView
android:id="@+id/webView"
android:layout_width="0dp"
android:layout_height="0dp"
app:layout_constraintLeft_toLeftOf="parent"
app:layout_constraintRight_toRightOf="parent"
app:layout_constraintTop_toBottomOf="@id/downloadBtn"
app:layout_constraintBottom_toTopOf="@id/scrollView" />
<ScrollView
android:id="@+id/scrollView"
android:layout_width="0dp"
android:layout_height="0dp"
app:layout_constraintLeft_toLeftOf="parent"
app:layout_constraintRight_toRightOf="parent"
app:layout_constraintTop_toBottomOf="@id/webView"
app:layout_constraintBottom_toBottomOf="parent">

<TextView android:id="@+id/content"
android:layout_width="match_parent"
android:layout_height="wrap_content" />
</ScrollView>
</androidx.constraintlayout.widget.ConstraintLayout>
```

Тут визначена кнопка для завантаження даних, а самі дані для прикладу завантажуються одночасно у вигляді рядка в текстове поле та елемент WebView. Оскільки даних може бути дуже багато, то текстове поле поміщене елемент ScrollView.

Оскільки завантаження даних може зайняти деякий час, то звернення до інтернет-ресурсу визначимо в окремому потоці і для цього змінимо код MainActivity таким чином:

```
package com.example.httpapp;
import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.webkit.WebView;
import android.widget.Button;
import android.widget.TextView;
import android.widget.Toast;
import java.io.BufferedReader;
```

```
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.net.URL;
import javax.net.ssl.HttpsURLConnection;
public class MainActivity extends AppCompatActivity {
@Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_main);
TextView contentView = findViewById(R.id.content);
WebView webView = findViewById(R.id.webView);
webView.getSettings().setJavaScriptEnabled(true);
Button btnFetch = findViewById(R.id.downloadBtn);
btnFetch.setOnClickListener(new View.OnClickListener() {
@Override
public void onClick(View v) {
contentView.setText("Завантаження...");
new Thread(new Runnable() {
public void run() {
try{
String content =
getContent("https://stackoverflow.com/");
webView.post(new Runnable() {
public void run() {
webView.loadDataWithBaseURL(
"https://stackoverflow.com/",
content, "text/html", "UTF-8",
"https://stackoverflow.com/");
Toast.makeText(getApplicationContext(),
"Дані завантажені", Toast.LENGTH_SHORT).show();
}
});
contentView.post(new Runnable() {
public void run() {
contentView.setText(content);
}
});
}
catch (IOException ex) {
contentView.post(new Runnable() {
public void run() {
contentView.setText("Помилка: " +
ex.getMessage());
Toast.makeText(getApplicationContext(), "Помилка",
Toast.LENGTH_SHORT).show();
}
});
}
}
}).start();
}
```

```
});
}
private String getContent(String path) throws IOException {
BufferedReader reader=null;
InputStream stream = null;
HttpsURLConnection connection = null;
try {
URL url=new URL(path);
connection =(HttpsURLConnection)url.openConnection();
connection.setRequestMethod("GET");
connection.setReadTimeout(10000);
connection.connect();
stream = connection.getInputStream();
reader = new BufferedReader(new InputStreamReader(stream));
StringBuilder buf=new StringBuilder();
String line;
while ((line=reader.readLine()) != null) {
buf.append(line).append("\n");
}
return(buf.toString());
}
finally {
if (reader! = null) {
reader.close();
}
if (stream! = null) {
stream.close();
}
if (connection! = null) {
connection.disconnect();
}
}
}
}
```

Безпосередньо для самого завантаження визначено методgetContent(), який завантажуватиме веб-сторінку за допомогою класу HttpsURLConnection і повертатиме код завантаженої сторінки у вигляді рядка.

Спочатку створюється елементHttpsURLConnection:

```
URL url=new URL(path);
connection =(HttpsURLConnection)url.openConnection();
connection.setRequestMethod("GET"); // встановлення методу отримання даних - GET
connection.setReadTimeout(10000); // встановлення таймауту перед виконанням - 10 000 мілісекунд
connection.connect(); // підключаємось до ресурсу
```

Після підключення відбувається зчитування із вхідного потоку:

```
stream = connection.getInputStream();
reader = new BufferedReader(new InputStreamReader(stream));
```

Використовуючи вхідний потік, ми можемо рахувати його в рядок.

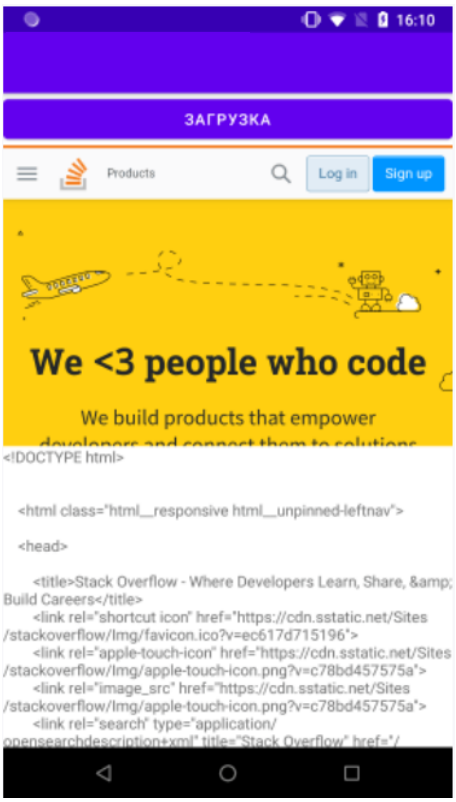
Цей методgetContent()потім буде викликатися в обробнику натискання кнопки:

```
Button btnFetch = (Button)findViewById(R.id.downloadBtn);
btnFetch.setOnClickListener(new View.OnClickListener() {
```

```
@Override
public void onClick(View v) {
contentView.setText("Завантаження...");
new Thread(new Runnable() {
public void run() {
try{
String content = getContent("https://stackoverflow.com/");
```

Оскільки завантаження може зайняти тривалий час, то методgetContent()в окремому потоці за допомогою об'єктів Thread та Runnable. Наприклад, у цьому випадку звернення йде до ресурсу "https://stackoverflow.com/".

Запустимо програму та натиснемо на кнопку. І за наявності інтернету програма завантажить головну сторінку з "https://stackoverflow.com/" та відобразить її у WebView та TextView:



Звичайно, цей спосіб навряд чи підходить для перегляду інтернет-сторінок, проте таким чином ми можемо отримувати будь-які дані (не інтернет-сторінки) від різних веб-сервісів, наприклад, у форматі xml або json (наприклад, різні курси валют, показники погоди), використовуючи спеціальні api, а потім після обробки показувати їх користувачеві.

7. Робота з файловою системою

7.1. Читання та збереження файлів

Робота з налаштуваннями рівня activity та програми дозволяє зберегти невеликі дані окремих типів (string, int), але для роботи з великими масивами даних, такими як графічні файли, файли мультимедіа тощо, нам доведеться звертатися до файлової системи.

ОС Android побудовано на основі Linux. Цей факт знаходить своє відображення у роботі з файлами. Так, у шляхах до файлів як розмежувач в Linux використовує слєш "/", а не зворотний слєш "\" (як у Windows). А всі назви файлів і каталогів є реєстрозалежними, тобто "data" це не те саме, що і "Data".

Програма Android зберігає свої дані в каталозі /data/data/<назва_пакета>/ і, як правило, щодо цього каталогу буде йти робота.

Для роботи з файлами абстрактний клас android.content.Context визначає низку методів:

- boolean deleteFile (String name): видаляє певний файл
- String[] fileList (): отримує всі файли, що містяться в підкаталозі /files у каталозі програми
- File getCacheDir(): отримує посилання на підкаталог cache у каталозі програми
- File getDir(String dirName, int mode): отримує посилання на підкаталог у каталозі програми, якщо такого підкаталогу немає, він створюється
- File getExternalCacheDir(): отримує посилання на папку /cache зовнішньої файлової системи пристрою
- File getExternalFilesDir(String type): отримує посилання на каталог /files зовнішньої файлової системи пристрою
- File getFileStreamPath(String filename): повертає абсолютний шлях до файлу у файловій системі
- FileInputStream openFileInput(String filename): відкриває файл для читання
- FileOutputStream openFileOutput (String name, int mode): відкриває файл для запису

Всі файли, які створюються та редагуються в додатку, зазвичай зберігаються в підкаталозі /files в каталозі програми.

Для безпосереднього читання та запису файлів застосовуються також стандартні класи Java із пакета java.io.

Отже, застосуємо функціонал читання-запису файлів у програмі. Нехай у нас буде наступна примітивна розмітка layout:

```
<?xmlversion="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/editor"
android:layout_width="0dp"
android:layout_height="0dp"
android:textSize="18sp"
android:gravity="start"
app:layout_constraintLeft_toLeftOf="parent"
app:layout_constraintRight_toRightOf="parent"
```

```
app:layout_constraintBottom_toTopOf="@id/save_text"
app:layout_constraintTop_toTopOf="parent"/>
<Button
android:id="@+id/save_text"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:onClick="saveText"
android:text="Зберегти"
app:layout_constraintLeft_toLeftOf="parent"
app:layout_constraintRight_toRightOf="parent"
app:layout_constraintBottom_toTopOf="@id/text"
app:layout_constraintTop_toBottomOf="@id/editor"/>
<TextView
android:id="@+id/text"
android:layout_width="0dp"
android:layout_height="0dp"
android:gravity="start"
android:textSize="18sp"
app:layout_constraintLeft_toLeftOf="parent"
app:layout_constraintRight_toRightOf="parent"
app:layout_constraintBottom_toTopOf="@+id/open_text"
app:layout_constraintTop_toBottomOf="@+id/save_text"/>
<Button
android:id="@+id/open_text"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:onClick="openText"
android:text="Відкрити"
app:layout_constraintLeft_toLeftOf="parent"
app:layout_constraintRight_toRightOf="parent"
app:layout_constraintBottom_toBottomOf="parent"
app:layout_constraintTop_toBottomOf="@+id/text"/>
</androidx.constraintlayout.widget.ConstraintLayout>
```

Поле EditText призначене для введення тексту, а TextView – для виведення раніше збереженого тексту. Для збереження та відновлення тексту додано дві кнопки.

Тепер у коді Activity пропишемо обробники кнопок із збереженням та читанням файлу:

```
packagecom.example.filesapp;
importandroidx.appcompat.app.AppCompatActivity;
importandroid.os.Bundle;
importandroid.view.View;
importandroid.widget.EditText;
importandroid.widget.TextView;
importandroid.widget.Toast;
importjava.io.FileInputStream;
importjava.io.FileOutputStream;
importjava.io.IOException;
publicclass MainActivity extends AppCompatActivity {
privatefinal static String FILE_NAME = "content.txt";
@Override
protectedvoid onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
```

```
setContentView(R.layout.activity_main);
}
// Збереження файлу
publicvoid saveText(View view) {
FileOutputStream fos = null;
try{
EditText textBox = findViewById(R.id.editor);
String text = textBox.getText().toString();

fos = openFileOutput(FILE_NAME, MODE_PRIVATE);
fos.write(text.getBytes());
Toast.makeText(this, "Файл збережений", Toast.LENGTH_SHORT).show();
}
catch(IOException ex) {
Toast.makeText(this, ex.getMessage(), Toast.LENGTH_SHORT).show();
}
finally{
try{
if(fos!=null)
fos.close();
}
catch(IOException ex) {
Toast.makeText(this, ex.getMessage(), Toast.LENGTH_SHORT).show();
}
}
}
// Відкриття файлу
publicvoid openText(View view) {

FileInputStream fin = null;
TextView textView = findViewById(R.id.text);
try{
fin = openFileInput(FILE_NAME);
byte[] bytes = newbyte[fin.available()];
fin.read(bytes);
String text = newString (bytes);
textView.setText(text);
}
catch(IOException ex) {
Toast.makeText(this, ex.getMessage(), Toast.LENGTH_SHORT).show();
}
finally{
try{
if(fin!=null)
fin.close();
}
catch(IOException ex) {

Toast.makeText(this, ex.getMessage(), Toast.LENGTH_SHORT).show();
}
}
}
```

```
}
```

При натисканні на кнопку збереження буде створюватись потік виводу FileOutputStream fos = openFileOutput(FILE_NAME, MODE_PRIVATE)

В даному випадку введений текст зберігатиметься у файлі "content.txt". При цьому використовуватиметься режим MODE_PRIVATE

Система дозволяє створювати файли з двома різними режимами:

- MODE_PRIVATE: файли можуть бути доступні лише власникові програми (за замовчуванням)
- MODE_APPEND: дані можуть бути додані до кінця файлу

Тому якщо файл "content.txt" вже існує, то він буде перезаписаний. Якщо нам треба було дописати файл, тоді треба було б використовувати режим MODE_APPEND:

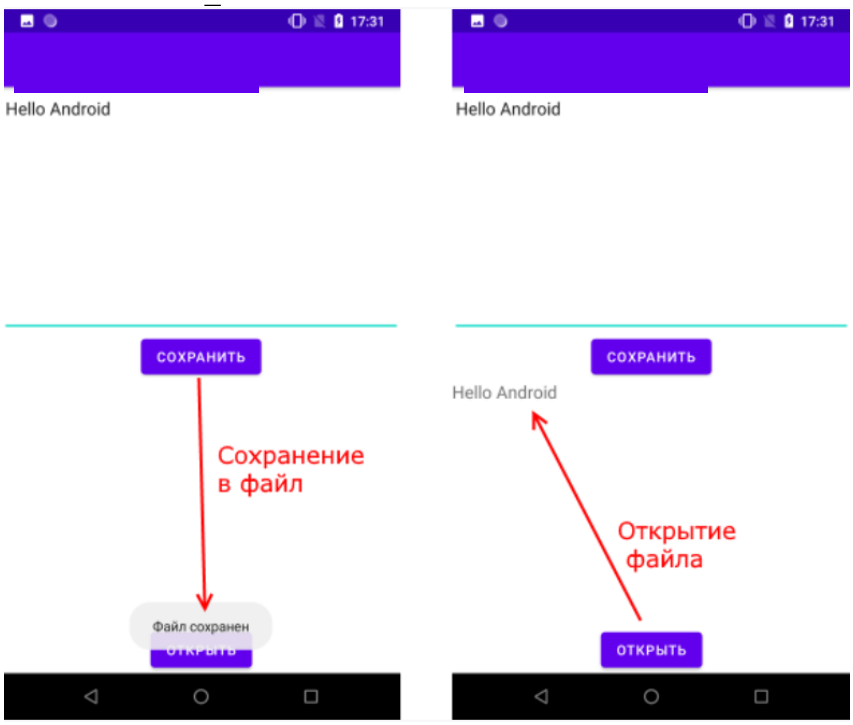
```
FileOutputStream fos = openFileOutput(FILE_NAME, MODE_APPEND);
```

Для читання файлу застосовується потік введення FileInputStream:

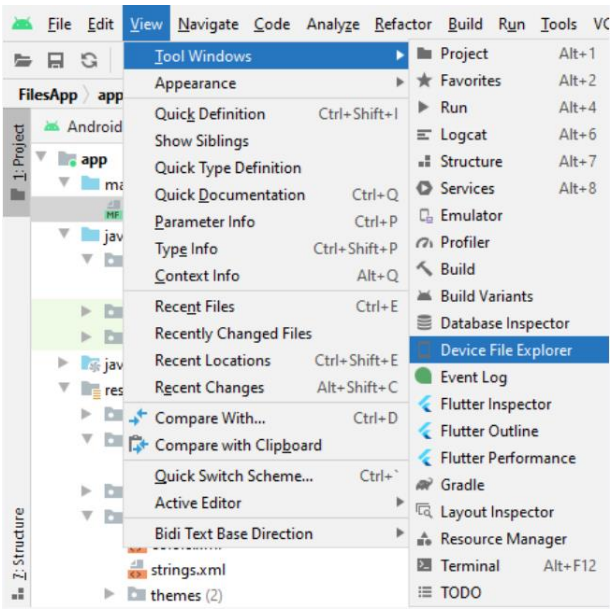
```
FileInputStream fin = openFileInput(FILE_NAME);
```

Докладніше про використання потоків вводу-виводу можна прочитати в посібнику Java:https://metanit.com/java/tutorial/6.3.php

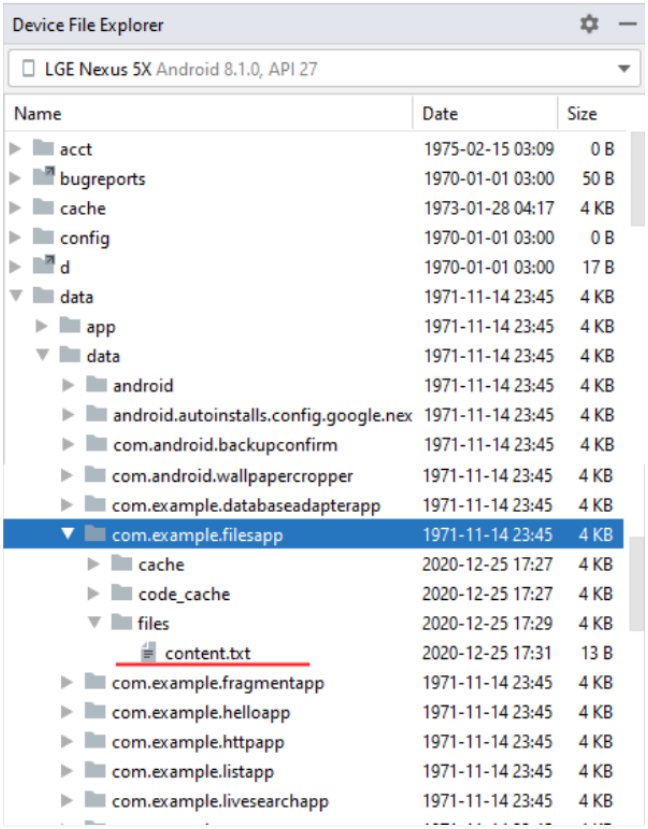
У результаті після натискання кнопки збереження весь текст буде збережено у файлі /data/data/назва_пакета/files/content.txt



Де фізично знаходиться файл? Щоб побачити його на підключеному пристрої, перейдемо в Android Stud в меню до пункту View -> Tool Windows -> Device File Explorer



Після цього відкриється вікно Device File Explorer для перегляду файлової системи пристрою. І в папці data/data/[назва_пакета_програми]/files ми зможемо знайти збережений файл.



7.2. Розміщення файлів у зовнішньому сховищі

Минулої теми ми розглянули збереження та читання файлів з каталогу програми. За промовчанням такі файли доступні лише для самої програми. Проте ми можемо поміщати та працювати з файлами із зовнішнього сховища програми. Це також дозволить іншим програмам відкривати дані файли та за необхідності змінювати.

Весь механізм роботи з файлами буде таким самим, як і при роботі зі сховищем програми. Ключовою відмінністю тут буде отримання та використання шляхів до зовнішнього сховища через метод getExternalFilesDir() класу Context.

Отже, нехай у файлі activity_main.xml буде така сама розмітка інтерфейсу:

```
<?xmlversion="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/editor"
android:layout_width="0dp"
android:layout_height="0dp"
android:textSize="18sp"
android:gravity="start"
app:layout_constraintLeft_toLeftOf="parent"
app:layout_constraintRight_toRightOf="parent"
app:layout_constraintBottom_toTopOf="@id/save_text"
app:layout_constraintTop_toTopOf="parent"/>
<Button
android:id="@+id/save_text"
android:layout_width="wrap_content"
android:layout_height="wrap_content"

android:onClick="saveText"
android:text="Зберегти"
app:layout_constraintLeft_toLeftOf="parent"
app:layout_constraintRight_toRightOf="parent"
app:layout_constraintBottom_toTopOf="@id/text"
app:layout_constraintTop_toBottomOf="@id/editor"/>

<TextView
android:id="@+id/text"
android:layout_width="0dp"
android:layout_height="0dp"
android:gravity="start"
android:textSize="18sp"

app:layout_constraintLeft_toLeftOf="parent"
app:layout_constraintRight_toRightOf="parent"
app:layout_constraintBottom_toTopOf="@+id/open_text"
app:layout_constraintTop_toBottomOf="@+id/save_text"/>
<Button
android:id="@+id/open_text"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:onClick="openText"
android:text="Відкрити"
app:layout_constraintLeft_toLeftOf="parent"
app:layout_constraintRight_toRightOf="parent"
app:layout_constraintBottom_toBottomOf="parent"
app:layout_constraintTop_toBottomOf="@+id/text"/>

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

А код класу MainActivity буде виглядати так:

```
packagecom.example.filesapp;

importandroidx.appcompat.app.AppCompatActivity;
importandroid.os.Bundle;
importandroid.view.View;
importandroid.widget.EditText;
importandroid.widget.TextView;
importandroid.widget.Toast;

importjava.io.File;
importjava.io.FileInputStream;
importjava.io.FileOutputStream;
importjava.io.IOException;

publicclass MainActivity extends AppCompatActivity {

privatefinal static String FILE_NAME = "document.txt";
@Override
protectedvoid onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_main);
}
privateFile getExternalPath() {
returnnew File(getExternalFilesDir(null), FILE_NAME);
}
// Збереження файлу
publicvoid saveText(View view) {

try(FileOutputStream fos = newFileOutputStream(getExternalPath())) {
EditText textBox = findViewById(R.id.editor);
String text = textBox.getText().toString();
fos.write(text.getBytes());
Toast.makeText(this, "Файл збережений", Toast.LENGTH_SHORT).show();
}
catch(IOException ex) {

Toast.makeText(this, ex.getMessage(), Toast.LENGTH_SHORT).show();
}
}
// Відкриття файлу
publicvoid openText(View view) {

TextView textView = findViewById(R.id.text);
File file = getExternalPath();
// якщо файл не існує, вихід із методу
if(!file.exists()) return;
try(FileInputStream fin = newFileInputStream(file)) {
byte[] bytes = newbyte[fin.available()];
fin.read(bytes);
String text = newString (bytes);
textView.setText(text);
```

```
}
catch(IOException ex) {

Toast.makeText(this, ex.getMessage(), Toast.LENGTH_SHORT).show();
}
}
}
```

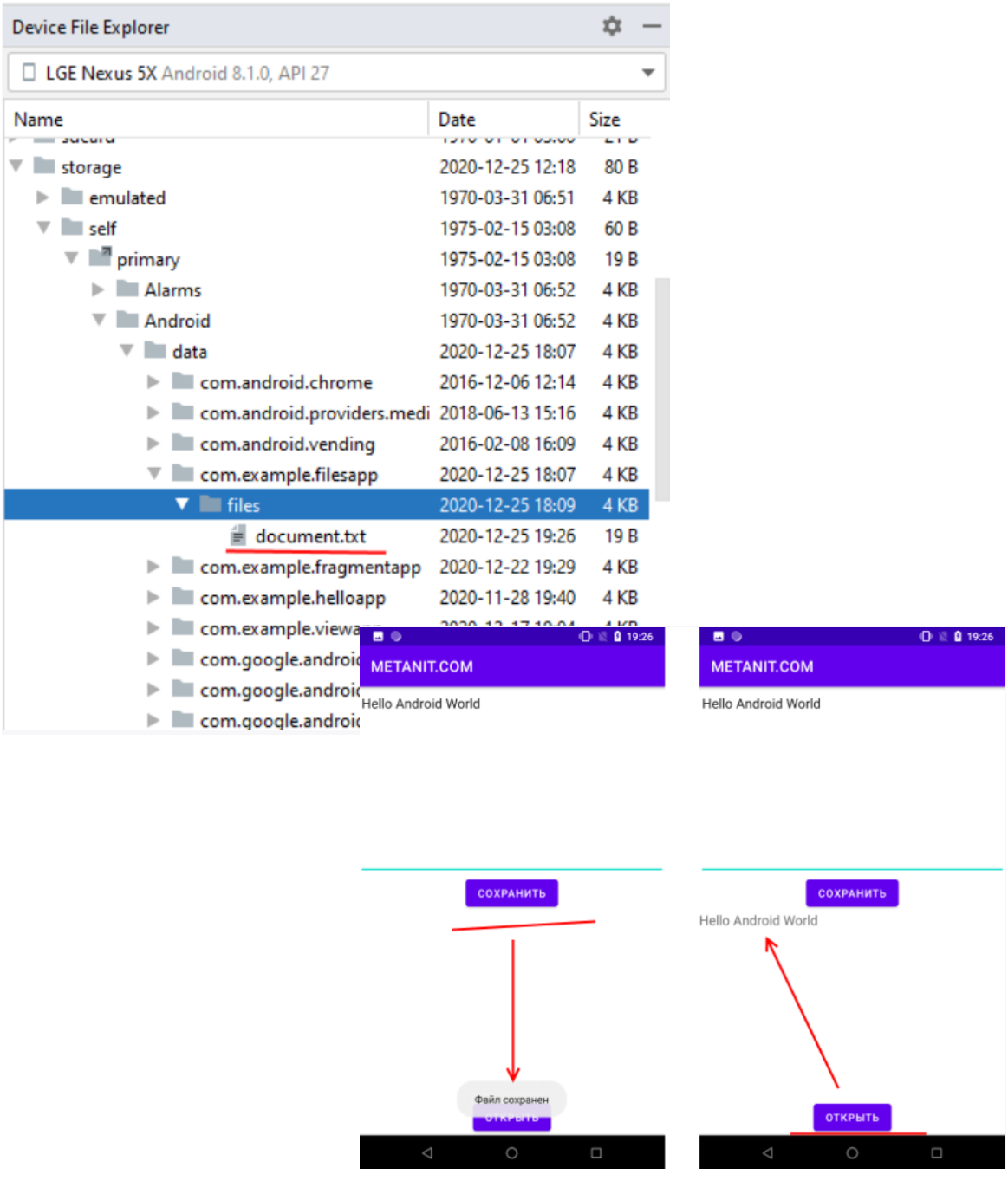
За допомогою виразу getExternalFilesDir(null) отримуємо доступ до папки програми у зовнішньому сховищі та встановлюємо об'єкт файлу:

```
privateFile getExternalPath() {
returnnew File(getExternalFilesDir(null), FILE_NAME);
}
```

Як параметр передається тип папки, але в даному випадку він нам не важливий, тому передається значення null

Всі інші дії запису/читання файлу будуть такими ж, як[redacted] темі у випадку роботи з локальною папкою[redacted]

І після операції запису на смартфоні через Device File Explorer ми зможемо побачити створений файл у папці storage/self/primary/Android/data/[назва_пакета]/files:



8. Робота з базами даних SQLite

8.1. Підключення до бази даних SQLite

В Android є вбудована підтримка однієї з найпоширеніших систем управління базами даних – SQLite. Для цього в пакеті android.database.sqlite визначено набір класів, які дозволяють працювати з базами даних SQLite. І кожна програма може створити свою базу даних.

Щоб використовувати SQLite в Android, потрібно створити базу даних за допомогою виразу на мові SQL. Після цього база даних зберігатиметься в каталозі програми по дорозі:

```
DATA/data/[Назва_програми]/databases/[Назва_файлу
_бази_даних]
```

ОС Android вже містить ряд вбудованих бад SQLite, які використовуються стандартними програмами - для списку контактів, для зберігання фотографій з камери, музичних альбомів і т.д.

Основну функціональність роботи з базами даних надає пакет android.database. Функціональність безпосередньо для роботи з SQLite знаходиться у пакеті android.database.sqlite.

База даних SQLite представлена класом android.database.sqlite.SQLiteDatabase. Він дозволяє виконувати запити до бд, виконувати з нею різні маніпуляції.

Клас android.database.sqlite.SQLiteCursor надає запит та дозволяє повертати набір рядків, які відповідають цьому запиту.

Клас android.database.sqlite.SQLiteQueryBuilder дозволяє створювати SQL запити.

Самі sql-вирази представлені класом android.database.sqlite.SQLiteStatement, які дозволяють за допомогою плейсхолдерів вставляти у вирази динамічні дані.

Клас android.database.sqlite.SQLiteOpenHelper дозволяє створити базу даних з усіма таблицями, якщо їх ще немає.

SQLite застосовує таку систему типів даних:

- INTEGER: представляє ціле число, аналог типу int в java
- REAL: представляє число з плаваючою точкою, аналог float та double у java
- TEXT: представляє набір символів, аналог String і char в java
- BLOB: представляє масив бінарних даних, наприклад, зображення, аналог типу int в java

Дані, що зберігаються, повинні представляти відповідні типи в java.

- Створення та відкриття бази даних

Для створення або відкриття нової бази даних із коду Activity в Android ми можемо викликати метод openOrCreateDatabase(). Цей метод може приймати три параметри:

- назва для бази даних
- числове значення, яке визначає режим роботи (як правило, у вигляді константи MODE_PRIVATE)
- необов'язковий параметр у вигляді об'єкта SQLiteDatabase.CursorFactory, який представляє фабрику створення курсору для роботи з бд

Наприклад, створення бази даних app.db:

```
SQLiteDatabase db = getBaseContext().openOrCreateDatabase("app.db",
MODE_PRIVATE, null);
```

Для виконання запиту до бази даних можна використовувати метод execSQL класу SQLiteDatabase. У цей спосіб передається SQL-вираз. Наприклад, створення бази даних таблиці users:

```
SQLiteDatabase db = getBaseContext().openOrCreateDatabase("app.db", MODE_PRIVATE, null);
db.execSQL("CREATE TABLE IF NOT EXISTS users (name TEXT, age INTEGER)");
```

Якщо нам треба не просто виконати вираз, але й отримати з бд будь-які дані, то використовується метод rawQuery(). Цей метод як параметр приймає SQL-вираз, а також набір значень для вираження sql. Наприклад, отримання всіх об'єктів із бази даних:

```
SQLiteDatabase db = getBaseContext().openOrCreateDatabase("app.db", MODE_PRIVATE, null);
db.execSQL("CREATE TABLE IF NOT EXISTS users (name TEXT, age INTEGER)");
Cursor query = db.rawQuery("SELECT * FROM users;", null);
if(query.moveToFirst()){
String name = query.getString(0);
intage = query.getInt(1);
}
```

Метод db.rawQuery() повертає об'єкт Cursor, за допомогою якого ми можемо отримати отримані дані.

Можлива ситуація, коли в базі даних не буде об'єктів і для цього методом query.moveToFirst() намагаємося переміститися до першого об'єкта, отриманого з бд. Якщо цей метод поверне значення false, це означає, що запит не отримав жодних даних з бд.

Тепер для роботи з базою даних зробимо найпростіший додаток. Для цього створимо новий проект.

У файлі activity_main.xml визначимо найпростіший графічний інтерфейс:

```
<?xmlversion="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">
<Button
android:id="@+id/button"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Click"
android:onClick="onClick"
app:layout_constraintBottom_toTopOf="@id/textView"
app:layout_constraintLeft_toLeftOf="parent"
app:layout_constraintTop_toTopOf="parent"
<TextView
android:id="@+id/textView"
android:layout_width="wrap_content"
```

```
android:layout_height="wrap_content"
android:textSize="22sp"
app:layout_constraintTop_toBottomOf="@id/button"
app:layout_constraintLeft_toLeftOf="parent"/>
</androidx.constraintlayout.widget.ConstraintLayout>
```

А в класі MainActivity визначимо взаємодію з базою даних:

```
packagecom.example.sqliteapp;
importandroidx.appcompat.app.AppCompatActivity;
importandroid.database.Cursor;
importandroid.database.sqlite.SQLiteDatabase;
importandroid.os.Bundle;
importandroid.view.View;
importandroid.widget.TextView;
publicclass MainActivity extends AppCompatActivity {
@Override
protectedvoid onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_main);
}
publicvoid onClick(View view) {
SQLiteDatabase db = getBaseContext().openOrCreateDatabase("app.db", MODE_PRIVATE, null);
db.execSQL("CREATE TABLE IF NOT EXISTS users (name TEXT, age INTEGER, UNIQUE(name))");
db.execSQL("INSERT OR IGNORE INTO users VALUES ('Tom Smith', 23), ('John Dow', 31);");
Cursor query = db.rawQuery("SELECT * FROM users;", null);
TextView textView = findViewById(R.id.textView);
textView.setText("");
while(query.moveToNext()){
String name = query.getString(0);
intage = query.getInt(1);
textView.append("Name:"+ name + "Age:" + age + "\n");
}
query.close();
db.close();
}
}
```

По натисканні на кнопку тут спочатку створюється в базі даних app.db нова таблиця users, а потім до неї додаються два об'єкти в базу даних за допомогою SQL-вираження INSERT.

Далі за допомогою виразу SELECT отримуємо всіх доданих користувачів із бази даних у вигляді курсору Cursor.

Викликом query.moveToNext() переміщуємось у циклі while послідовно по всіх об'єктах.

Для отримання даних із курсору застосовуються методи query.getString(0) та query.getInt(1). У дужках до методів передається номер стовпця, з якого ми отримуємо дані. Наприклад, вище ми додали спочатку ім'я користувача у вигляді рядка, а потім вік у вигляді числа. Значить, нульовим стовпцем йтиме рядкове значення, яке отримуємо за

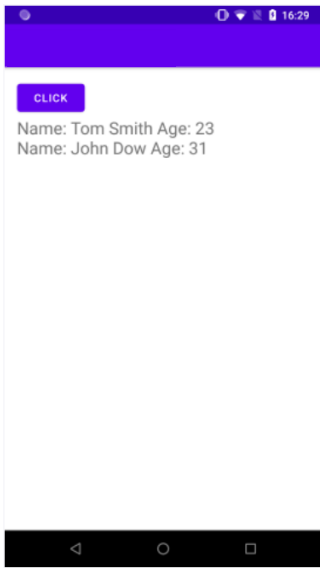
допомогою методу getString(), а наступним - першим стовпцем йде числове значення, для якого застосовується метод getInt().

Після завершення роботи з курсором та базою даних ми закриваємо всі пов'язані об'єкти:

```
query.close();
db.close();
```

Якщо ми не закриємо курсор, можемо зіткнутися з проблемою витоку пам'яті.

І якщо ми звернемося до програми, то після натискання на кнопку в текстове поле буде виведено додані дані:



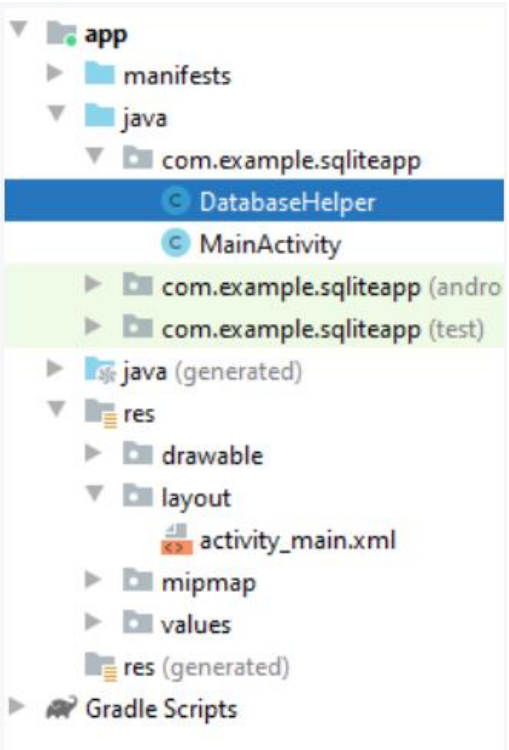
8.2. SQLiteOpenHelper і SimpleCursorAdapter, отримання даних з SQLite

У минулій темі було розглянуто, як підключатися до бази даних SQLite та виконувати запити. Тепер підемо далі та створимо повністю інтерфейс для роботи з базою даних.

Отже, створимо новий проект.

Для спрощення роботи з базами даних SQLite в Android часто застосовується клас SQLiteOpenHelper. Для використання необхідно створити клас-спадкоємець від SQLiteOpenHelper, перевизначивши як мінімум два його методи:

- onCreate(): викликається при спробі доступу до бази даних, але коли ця база даних ще не створена
- onUpgrade(): викликається, коли необхідно оновити схему бази даних. Тут можна перестворити раніше створену базу даних onCreate(), встановивши відповідні правила перетворення від старої бд до нової



Тому додамо в проект, у ту саму папку, де знаходиться клас MainActivity, новий клас DatabaseHelper:

```
packagecom.example.sqliteapp;
importandroid.database.sqlite.SQLiteOpenHelper;
importandroid.database.sqlite.SQLiteDatabase;
importandroid.content.Context;
publicclass DatabaseHelper extends SQLiteOpenHelper
{
privatestatic final String DATABASE_NAME =
"userstore.db"; // Назва бд
privatestatic final int SCHEMA = 1; // версія бази
даних
staticfinal String TABLE = "users"; // Назва таблиці в
бд
// назви стовпців
publicstatic final String COLUMN_ID = "_id";
publicstatic final String COLUMN_NAME="name";
publicstatic final String COLUMN_YEAR = "рок";
publicDatabaseHelper(Context context) {
super(context, DATABASE_NAME, null, SCHEMA);
}
@Override
publicvoid onCreate(SQLiteDatabase db) {

db.execSQL("CREATE TABLE users ("+ COLUMN_ID
+ " INTEGER PRIMARY KEY AUTOINCREMENT,"+
COLUMN_NAME
```

```
    + "TEXT,"+ COLUMN_YEAR + "INTEGER);");
    // Додавання початкових даних
    db.execSQL("INSERT INTO "+ TABLE +" ("+
COLUMN_NAME
    + ","+ COLUMN_YEAR + ") VALUES ('Том Сміт',
1981);");
    }
    @Override
    publicvoid onUpgrade(SQLiteDatabase db, int
oldVersion, int newVersion) {
    db.execSQL("DROP TABLE IF EXISTS"+TABLE);
    onCreate(db);
    }
    }
```

Якщо база даних відсутня або її версія (яка задається в змінній SCHEMA) вище за поточну, то спрацьовує метод onCreate().

Для виконання запитів до бази даних нам знадобиться об'єкт SQLiteDatabase, який представляє базу даних. Метод onCreate() отримує як параметр базу даних програми.

Для виконання запитів до SQLite використовується метод execSQL(). Він приймає sql-вираз CREATE TABLE, що створює таблицю. Тут також у разі потреби ми можемо виконати й інші запити, наприклад, додати будь-які початкові дані. Так, у цьому випадку за допомогою того ж методу та виразу sql INSERT додається один об'єкт у таблицю.

У методі onUpgrade() відбувається оновлення схеми БД. В даному випадку для прикладу використаний примітивний похід з видаленням попередньої бази даних за допомогою sql-вираження DROP та подальшим її створенням. Але насправді якщо вам необхідно зберегти дані, цей метод може включати складнішу логіку - додавання нових стовпців, видалення непотрібних, додавання додаткових даних і т.д.

Далі визначимо у файлі activity_main.xml наступну розмітку:

```
    <?xmlversion="1.0" encoding="utf-8"?>
    <androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/and
roid"
    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/header"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:textSize="18sp"
    app:layout_constraintBottom_toTopOf="@+id/list"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent"
    />
```

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

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

Тут визначено список ListView, для відображення отриманих даних, із заголовком, який виводитиме кількість отриманих об'єктів.

І змінимо код класу MainActivity наступним чином:

```
    packagecom.example.sqliteapp;

    importandroidx.appcompat.app.AppCompatActivity;
    importandroid.widget.SimpleCursorAdapter;
    importandroid.database.Cursor;
    importandroid.database.sqlite.SQLiteDatabase;
    importandroid.os.Bundle;
    importandroid.widget.ListView;
    importandroid.widget.TextView;
    publicclass MainActivity extends AppCompatActivity {
    ListView userList;
    TextView header;
    DatabaseHelper databaseHelper;
    SQLiteDatabase db;
    Cursor userCursor;
    SimpleCursorAdapter userAdapter;
    @Override
    protectedvoid onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    header = findViewById(R.id.header);
    userList = findViewById(R.id.list);
    databaseHelper                                    =
newDatabaseHelper(getApplicationContext());
    }
    @Override
    publicvoid onResume() {
    super.onResume();
    // відкриваємо підключення
    db = databaseHelper.getReadableDatabase();
    //отримуємо дані з бд як курсора
    userCursor    =    db.rawQuery("select    *    from    "+
```

```
DatabaseHelper.TABLE, null);
    // визначаємо, які стовпці з курсору виводитимуться
у ListView
    String[] headers = newString[]
{DatabaseHelper.COLUMN_NAME,
DatabaseHelper.COLUMN_YEAR};
    // Створюємо адаптер, передаємо в нього курсор
    userAdapter = newSimpleCursorAdapter(this,
android.R.layout.two_line_list_item,
    користувачаCursor, headers,
newint[]{android.R.id.text1, android.R.id.text2}, 0);
    header.setText("Знайдено елементів: "+
userCursor.getCount());
    userList.setAdapter(userAdapter);
    }
    @Override
    publicvoid onDestroy(){
    super.onDestroy();
    // Закриваємо підключення та курсор
    db.close();
    userCursor.close();
    }
    }
```

У методі onCreate() відбувається створення об'єкта SQLiteOpenHelper. Сама ініціалізація об'єктів для роботи з базою даних відбувається у методі onResume(), який спрацьовує після методу onCreate().

Щоб отримати об'єкт бази даних, необхідно використовувати метод getReadableDatabase() (отримання бази даних для читання) або getWritableDatabase(). Так як в даному випадку ми будемо тільки зчитувати дані з бд, то скористаємося першим методом:

```
    db = sqlHelper.getReadableDatabase();
```

- Отримання даних та Cursor

Android надає різні способи здійснення запитів до об'єкта SQLiteDatabase. У більшості випадків ми можемо застосовувати метод rawQuery(), який приймає два параметри: SQL-вираз SELECT і додатковий параметр, який визначає параметри запиту.

Після виконання запиту rawQuery() повертає об'єкт Cursor, який зберігає результат виконання SQL-запиту:

```
    userCursor = db.rawQuery("select * from "+ DatabaseHelper.TABLE, null);
```

Клас Cursor пропонує ряд методів для керування вибіркою, зокрема:

- getCount(): отримує кількість вилучених з бази даних об'єктів
- Методи moveToFirst() і moveToNext() дозволяють переходити до першого та наступного елементів вибірки. Метод isAfterLast() дозволяє перевірити, чи досягнуто кінець вибірки.
- Методи get*(columnIndex) (наприклад, getLong(), getString()) дозволяють за індексом стовпця звернутися до даного стовпця поточного рядка
- CursorAdapter

Додатково для управління курсором Android є клас CursorAdapter. Він дозволяє адаптувати отриманий за допомогою курсору набір для відображення у списокових

елементах на зразок ListView. Як правило, під час роботи з курсором використовується підклас CursorAdapter - SimpleCursorAdapter. Хоча можна використовувати інші адаптери, типу ArrayAdapter.

```
    userAdapter = newSimpleCursorAdapter(this,
android.R.layout.two_line_list_item,
    користувачаCursor, headers,
newint[]{android.R.id.text1, android.R.id.text2}, 0);
    userList.setAdapter(userAdapter);
```

Конструктор класу SimpleCursorAdapter приймає шість параметрів:

1. Першим параметром виступає контекст, з яким асоціюється адаптер, наприклад, поточна діяльність
2. Другий параметр - ресурс розмітки інтерфейсу, який використовуватиметься для відображення результатів вибірки
3. Третій параметр – курсор
4. Четвертий параметр - список стовпців із вибірки, які відображатимуться у розмітці інтерфейсу
5. П'ятий параметр - елементи всередині ресурсу розмітки, які відображатимуть значення стовпців із четвертого параметра
6. Шостий параметр – прапори, що задають поведінки адаптера

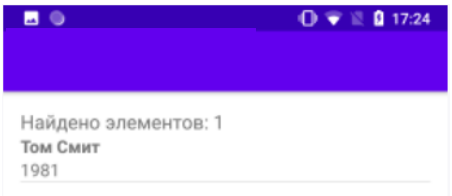
При використанні CursorAdapter та його підкласів слід враховувати, що вибірка курсора повинна включати цілий стовпець з назвою _id, який має бути унікальним для кожного елемента вибірки. Значення цього стовпця при натисканні на елемент списку потім передається метод обробки onListItemClick(), завдяки чому ми можемо по id ідентифікувати натиснутий елемент.

В даному випадку у нас перший стовпець якраз називається "_id".

Після завершення роботи курсор має бути закритий методом close()

І також треба враховувати, що якщо ми використовуємо курсор SimpleCursorAdapter, то ми не можемо використовувати метод close(), поки не завершимо використання SimpleCursorAdapter. Тому метод cursor краще викликати в методі onDestroy() фрагмента або activity.

І якщо ми запустимо програму, то побачимо список з одного доданого елемента:

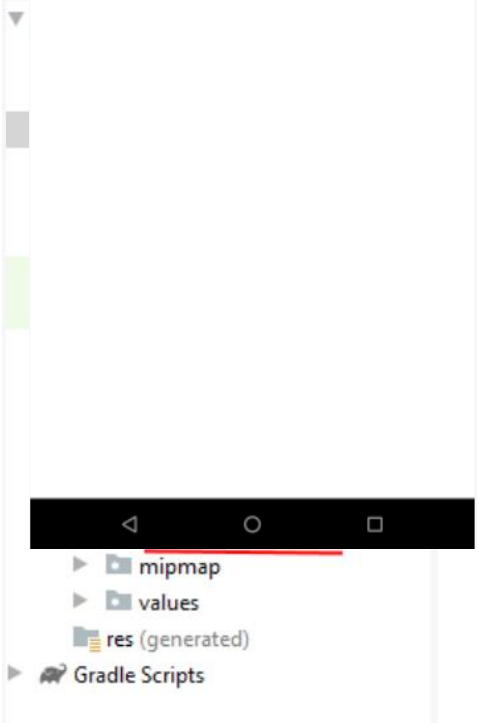


8.3. Додавання, видалення та оновлення даних у SQLite

Продовжимо роботу з проектом із минулої теми, де ми отримуємо дані. Тепер додамо до нього стандартну CRUD-логіку (створення, оновлення, видалення).

Щоб не нагромаджувати форму з головною діяльністю, всі інші дії по роботі з даними будуть відбуватися на іншому екрані. Додамо до проекту новий клас activity, який назвемо UserActivity:

У файлі activity_user.xml визначимо універсальну форму для



додавання/оновлення/видалення даних:

```
<?xmlversion="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/and
roid"
xmlns:app="http://schemas.android.com/apk/res-auto"
android:layout_width="match_parent"
android:layout_height="match_parent">
<EditText
android:id="@+id/name"
android:layout_width="0dp"
android:layout_height="wrap_content"
android:hint="Введіть ім'я"
app:layout_constraintBottom_toTopOf="@+id/year"
app:layout_constraintTop_toTopOf="parent"
app:layout_constraintLeft_toLeftOf="parent"
app:layout_constraintRight_toRightOf="parent"/>
<EditText
android:id="@+id/year"
android:layout_width="0dp"
android:layout_height="wrap_content"
android:hint="Введіть рік народження"
app:layout_constraintTop_toBottomOf="@+id/name"
app:layout_constraintBottom_toTopOf="@+id/saveButto
n"
app:layout_constraintLeft_toLeftOf="parent"
app:layout_constraintRight_toRightOf="parent"/>
<Button
android:id="@+id/saveButton"
android:layout_width="0dp"
android:layout_height="wrap_content"
android:text="Зберегти"
android:onClick="save"
app:layout_constraintHorizontal_weight="1"
app:layout_constraintTop_toBottomOf="@+id/year"
app:layout_constraintLeft_toLeftOf="parent"
app:layout_constraintRight_toLeftOf="@+id/deleteButt
on"
/>
<Button
android:id="@+id/deleteButton"
android:layout_width="0dp"
android:layout_height="wrap_content"
android:text="Видалити"
android:onClick="delete"
```

```
    app:layout_constraintHorizontal_weight="1"
    app:layout_constraintTop_toBottomOf="@+id/year"
    app:layout_constraintLeft_toRightOf="@+id/saveButton
"
    app:layout_constraintRight_toRightOf="parent"
    />
    </androidx.constraintlayout.widget.ConstraintLayout>
```

І також змінимо код UserActivity:

```
    packagecom.example.sqliteapp;

    importandroidx.appcompat.app.AppCompatActivity;

    importandroid.content.ContentValues;
    importandroid.content.Intent;
    importandroid.database.Cursor;
    importandroid.database.sqlite.SQLiteDatabase;
    importandroid.os.Bundle;
    importandroid.view.View;
    importandroid.widget.Button;
    importandroid.widget.EditText;

    publicclass UserActivity extends AppCompatActivity {

    EditText nameBox;
    EditText yearBox;
    Button del Button;
    Button saveButton;

    DatabaseHelper sqlHelper;
    SQLiteDatabase db;
    Cursor userCursor;
    longuserId=0;
    @Override
    protectedvoid onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_user);

    nameBox = findViewById(R.id.name);
    yearBox = findViewById(R.id.year);
    delButton = findViewById(R.id.deleteButton);
    saveButton = findViewById(R.id.saveButton);

    sqlHelper = newDatabaseHelper(this);
    db = sqlHelper.getWritableDatabase();
```

```
    Bundle extras = getIntent().getExtras();
    if(extras! = null) {
    userId = extras.getLong("id");
    }
    // якщо 0, то додавання
    if(userId > 0) {
    // отримуємо елемент з id з бд
    userCursor = db.rawQuery("select * from"+
DatabaseHelper.TABLE + "where" +
    DatabaseHelper.COLUMN_ID + "=?",
newString[]{String.valueOf(userId)});
    userCursor.moveToFirst();
    nameBox.setText(userCursor.getString(1));
    yearBox.setText(String.valueOf(userCursor.getInt(2)));
    userCursor.close();
    } else{
    // приховуємо кнопку видалення
    delButton.setVisibility(View.GONE);
    }
    }

    publicvoid save (View view) {
    ContentValues cv=newContentValues();
    cv.put(DatabaseHelper.COLUMN_NAME,
nameBox.getText().toString());
    cv.put(DatabaseHelper.COLUMN_YEAR,
Integer.parseInt(yearBox.getText().toString()));

    if(userId > 0) {
    db.update(DatabaseHelper.TABLE, cv,
DatabaseHelper.COLUMN_ID + "="+ userId, null);
    } else{
    db.insert(DatabaseHelper.TABLE, null, cv);
    }
    goHome();
    }
    publicvoid delete (View view) {
    db.delete(DatabaseHelper.TABLE, "_id = ?",
newString[]{String.valueOf(userId)});
    goHome();
    }
    privatevoid goHome(){
    // закриваємо підключення
    db.close();
    // Перехід до головної діяльності
```

```
    Intent intent = newIntent(this, MainActivity.class);
    intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP
| Intent.FLAG_ACTIVITY_SINGLE_TOP);
    startActivity(intent);
    }
    }
```

При оновленні або видаленні об'єкта зі списку з головної activity в UserActivity буде передаватися id об'єкта:

```
    longuserId=0;
    //.................
    Bundle extras = getIntent().getExtras();
    if(extras! = null) {
    userId = extras.getLong("id");
    }
```

Якщо MainActivity не було передано id, то встановлюємо його значення 0, отже, у нас буде додавання, а не редагування/видалення

Якщо id визначено, то отримуємо з нього з бази даних об'єкт для редагування/видалення:

```
    if(id < 0) {

    userCursor = db.rawQuery("select * from"+
DatabaseHelper.TABLE + "where" +
    DatabaseHelper.COLUMN_ID + "=?",
newString[]{String.valueOf(id)});
    userCursor.moveToFirst();
    nameBox.setText(userCursor.getString(1));
    yearBox.setText(String.valueOf(userCursor.getInt(2)));
    userCursor.close();
    }
```

Інакше просто приховуємо кнопку видалення.

Для виконання операцій із вставлення, оновлення та видалення даних SQLiteDatabase має методи insert(), update() та delete(). Ці методи викликаються в обробниках кнопок:

```
    db.delete(DatabaseHelper.TABLE, "_id = ?",
newString[]{String.valueOf(id)});
```

У метод delete() передається назва таблиці, а також стовпець, яким відбувається видалення, і його значення. Як критерій можна вибрати кілька стовпців, тому третім параметром йде масив. Знак питання ? позначає параметр, замість якого підставляється значення третього параметра.

8.4. ContentValues

Для додавання чи оновлення нам потрібно створити об'єкт ContentValues. Цей об'єкт представляє словник, який містить набір пар "ключ-значення". Для додавання цього

словника нового об'єкта застосовується метод put. Перший параметр методу – це ключ, а другий – значення, наприклад:

```
ContentValues cv=newContentValues();
cv.put("NAME", "Tom");
cv.put("YEAR", 30);
```

В якості значень метод put можна передавати рядки, цілі числа, числа з плаваючою точкою

У цьому випадку додаються введені в текстове поля значення:

```
ContentValues cv=newContentValues();
cv.put(DatabaseHelper.COLUMN_NAME,
nameBox.getText().toString());
cv.put(DatabaseHelper.COLUMN_YEAR,
Integer.parseInt(yearBox.getText().toString()));
```

При оновленні метод update() передається назва таблиці, об'єкт ContentValues і критерій, за яким відбувається оновлення (в даному випадку стовпець id):

```
db.update(DatabaseHelper.TABLE, cv,
DatabaseHelper.COLUMN_ID + "="+ userId, null);
```

Метод insert() приймає назву таблиці, об'єкт ContentValues з значеннями, що додаються. Другий параметр є необов'язковим: він передає стовпець, до якого треба додати значення NULL:

```
db.insert(DatabaseHelper.TABLE, null, cv);
```

Замість цих методів, як і в минулій темі, можна використовувати метод execSQL() з точною вказівкою sql-вираження. У той же час методи delete/insert/update мають перевагу - вони повертають id зміненого запису, за яким ми можемо дізнатися про успішність операції, або -1 у разі невдалої операції:

```
longresult = db.insert(DatabaseHelper.TABLE, null,
cv);
if(result>0){

// дії
}
```

Після кожної операції виконується метод goHome(), який повертає на головну діяльність.

Після цього нам потрібно виправити код MainActivity, щоб вона ініціювала виконання коду в UserActivity. Для цього змінимо код activity_main.xml:

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

<Button
android:id="@+id/addButton"
android:layout_width="0dp"
android:layout_height="wrap_content"
```

```
        android:textSize="18sp"
        android:text="Додати"
        android:onClick="add"
        app:layout_constraintBottom_toTopOf="@+id/list"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        />
        <ListView
        android:id="@+id/list"
        android:layout_width="0dp"
        android:layout_height="0dp"
        app:layout_constraintTop_toBottomOf="@+id/addButto
n"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"/>

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

У цьому випадку була додана кнопка виклику UserActivity.

Також змінимо код класу MainActivity:

```
packagecom.example.sqliteapp;
importandroidx.appcompat.app.AppCompatActivity;
importandroid.content.Intent;
importandroid.view.View;
importandroid.widget.AdapterView;
importandroid.widget.SimpleCursorAdapter;
importandroid.database.Cursor;
importandroid.database.sqlite.SQLiteDatabase;
importandroid.os.Bundle;
importandroid.widget.ListView;
publicclass MainActivity extends AppCompatActivity {
ListView userList;
DatabaseHelper databaseHelper;
SQLiteDatabase db;
Cursor userCursor;
SimpleCursorAdapter userAdapter;
@Override
protectedvoid onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_main);
userList = findViewById(R.id.list);
userList.setOnItemClickListener(newAdapterView.OnItemClickListe
ner() {
```

```
    @Override
    publicvoid onItemClick(AdapterView<?> parent, View view, int
position, long id) {
    Intent intent = newIntent(getApplicationContext(),
UserActivity.class);
    intent.putExtra("id", id);
    startActivity(intent);
    }
    });
    databaseHelper = newDatabaseHelper(getApplicationContext());
    }
    @Override
    publicvoid onResume() {
    super.onResume();
    // відкриваємо підключення
    db = databaseHelper.getReadableDatabase();
    //отримуємо дані з бд як курсора
    userCursor = db.rawQuery("select * from"+ DatabaseHelper.TABLE,
null);
    // визначаємо, які стовпці з курсору виводитимуться у ListView
    String[] headers = newString[]{DatabaseHelper.COLUMN_NAME,
DatabaseHelper.COLUMN_YEAR};
    // Створюємо адаптер, передаємо в нього курсор
    userAdapter = newSimpleCursorAdapter(this,
android.R.layout.two_line_list_item,
    користувачаCursor, headers, newint[]{android.R.id.text1,
android.R.id.text2}, 0);
    userList.setAdapter(userAdapter);
    }

    // за натисканням на кнопку запускаємо UserActivity для
додавання даних
    publicvoid add (View view) {
    Intent intent = newIntent(this, UserActivity.class);
    startActivity(intent);
    }

    @Override
    publicvoid onDestroy() {
    super.onDestroy();
    // Закриваємо підключення та курсор
    db.close();
    userCursor.close();
    }
    }
```

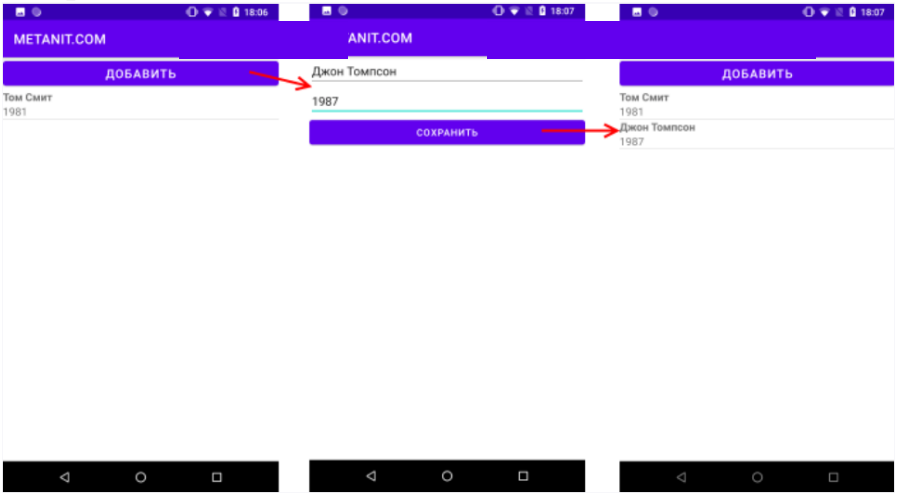
При натисканні на кнопку запускається UserActivity, при цьому не передається жодного id, тобто в UserActivity id дорівнюватиме нулю, значить буде йти додавання даних:

```
publicadd(View view){
Intent intent = newIntent(this, UserActivity.class);
startActivity(intent);
}
```

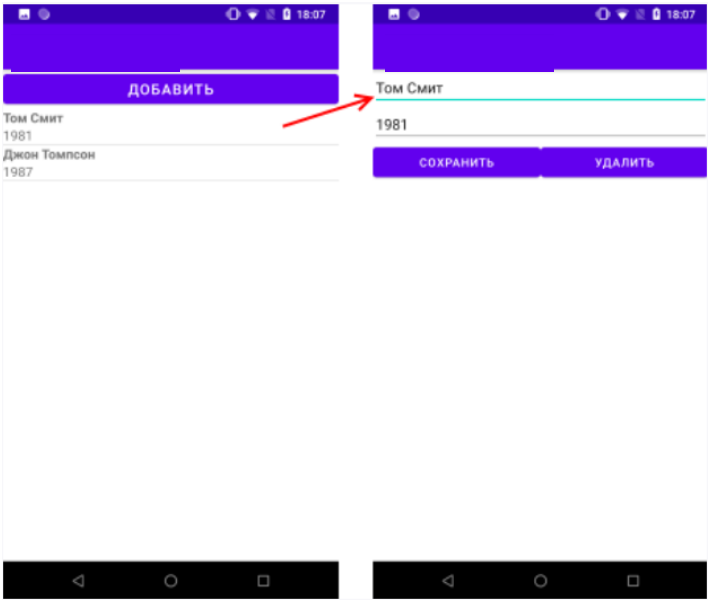
Іншу ситуацію представляє обробник натискання на елемент списку - при натисканні також запускатиметься UserActivity, але тепер передаватиметься id обраного запису:

```
publicvoid onItemClick(AdapterView<?> parent, View view, int position, long id) {
Intent intent = newIntent(getApplicationContext(), UserActivity.class);
intent.putExtra("id", id);
startActivity(intent);
}
```

Запустимо програму та натиснемо на кнопку, яка має перенаправляти на UserActivity:



При натисканні MainActivity на елемент списку цей елемент потрапить на UserActivity, де його можна буде видалити або підредагувати:

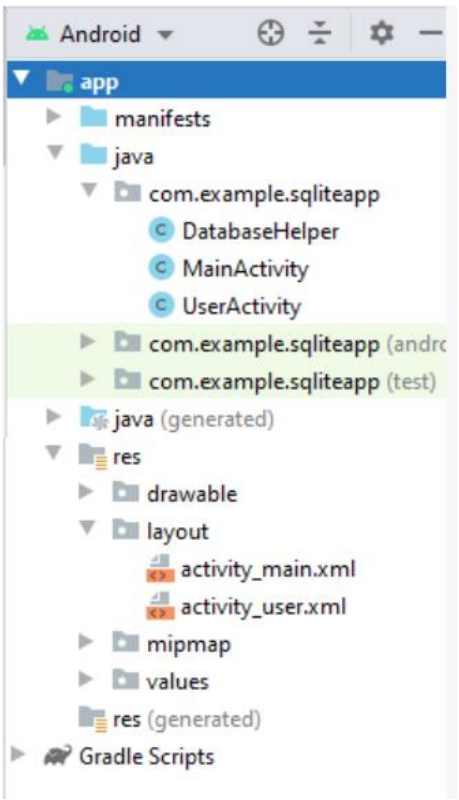


8.5. Використання існуючої бази даних SQLite

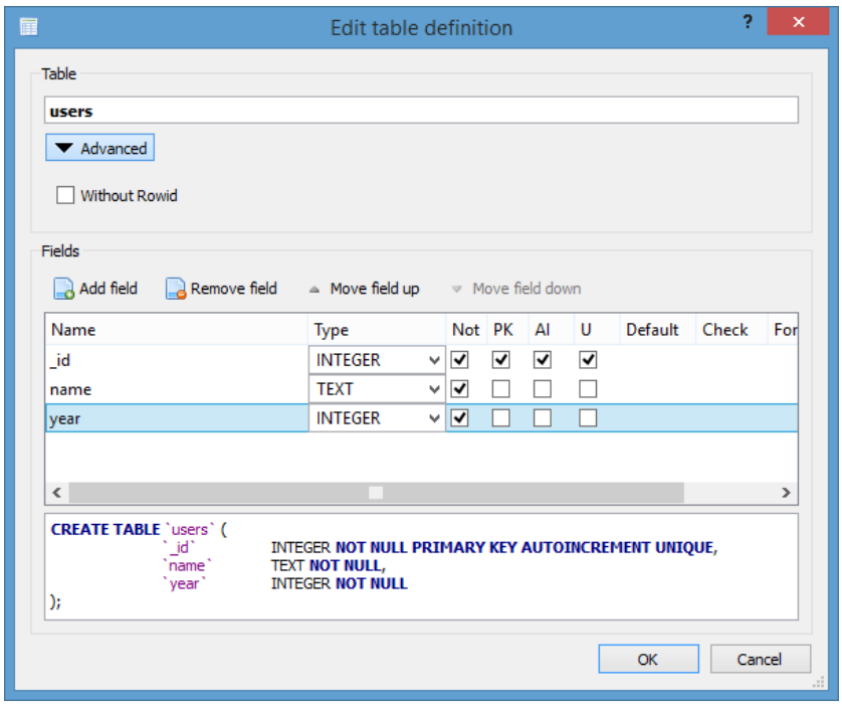
Крім створення нової бази даних, ми також можемо використовувати вже існуючу. Це може бути більш переважно, тому що в цьому випадку база даних програми вже міститиме всю необхідну інформацію.

Візьмемо проект, створений у попередній темі, де у нас була MainActivity, яка виводила список об'єктів, та UserActivity, яка дозволяла додавати, редагувати та видаляти об'єкти з БД

Для початку створимо базу даних SQLite. У цьому може допомогти такий інструмент як Sqlitebrowser. Він безкоштовний та доступний для різних операційних систем за адресоюhttps://sqlitebrowser.org/. Хоча можна використовувати інші способи для створення початкової БД.



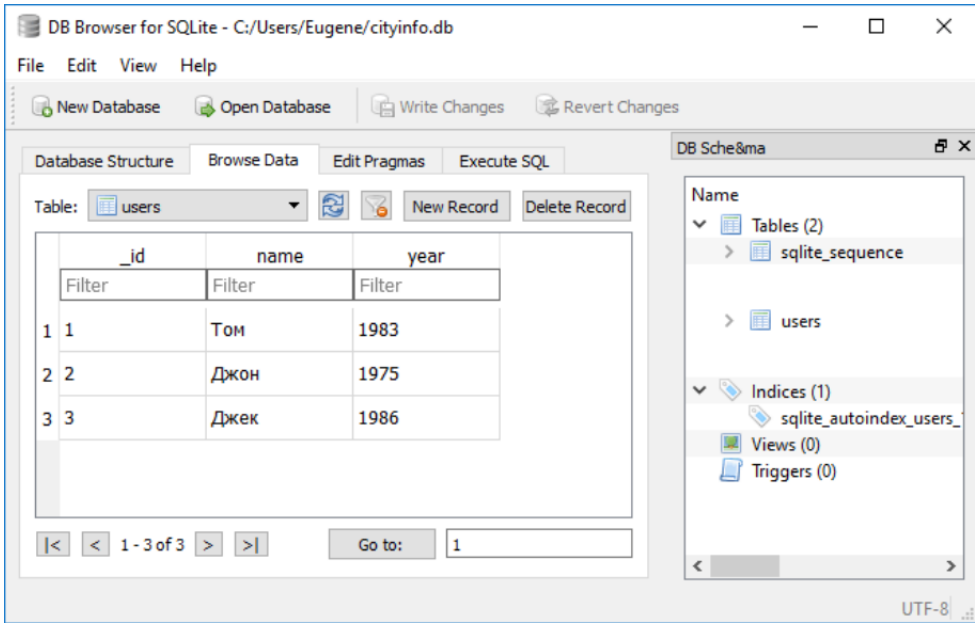
Sqlitebrowser представляє графічний інтерфейс для створення бази даних та визначення в ній всіх необхідних таблиць:



Як видно на скріншоті, я визначаю таблицю users із трьома полями: _id, name, age. Загальна команда на створення таблиці буде такою:

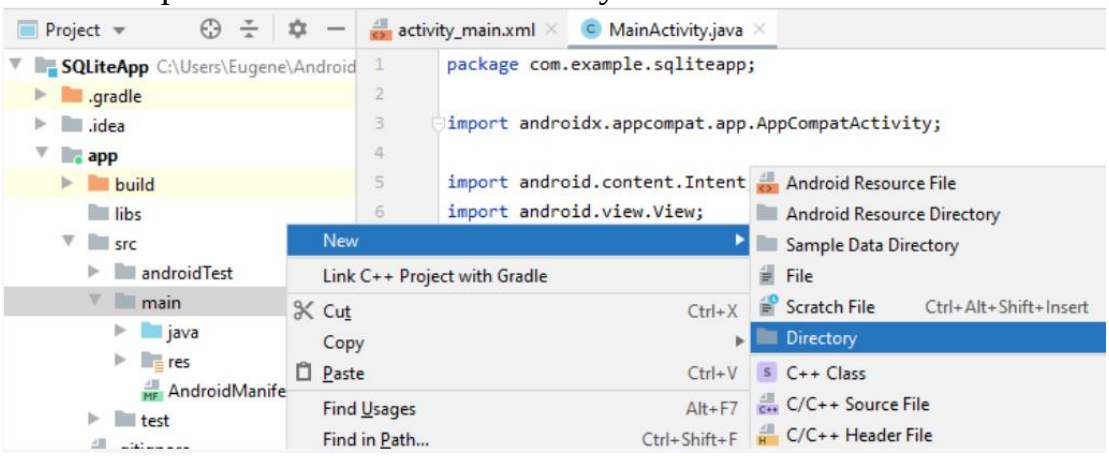
```
CREATETABLE `users` (
`_id` INTEGERNOT NULL PRIMARY KEY AUTOINCREMENT UNIQUE,
`name` TEXT NOTNULL,
`year` INTEGERNOT NULL
);
```

Там же у програмі додамо кілька елементів у створену таблицю:

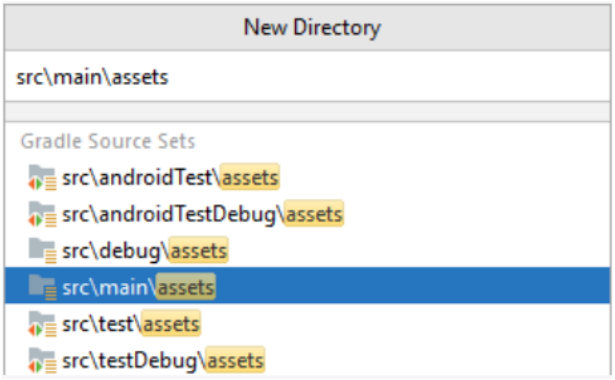


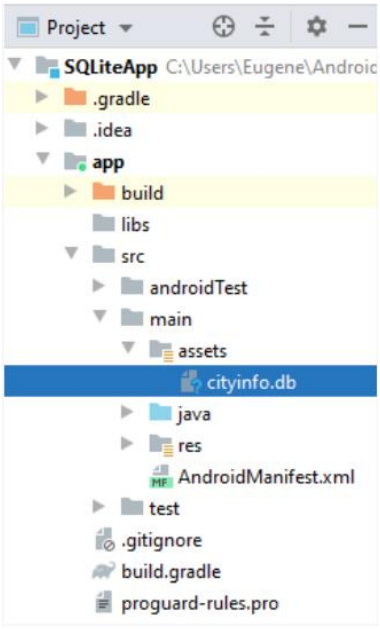
Після створення таблиці додамо в проект Android Studio папку assets, а в папку assets - щойно створену базу даних. Для цього перейдемо до повного визначення

проекту, натиснемо на папку main правою кнопкою миші і в меню виберемо New -> Directory:



Потім у вікні, що з'явилося, виберемо пункт src\main\assets і натиснемо на Enter для її додавання в проект::





І потім скопіюємо в неї нашу базу даних:

У нашому випадку база даних має назву "cityinfo.db".

Змінимо код DatabaseHelper наступним чином:

```
packagecom.example.sqliteapp;

importandroid.database.SQLException;
importandroid.database.sqlite.SQLiteOpenHelper;
importandroid.database.sqlite.SQLiteDatabase;
importandroid.content.Context;
importandroid.util.Log;

importjava.io.File;
importjava.io.FileOutputStream;
importjava.io.IOException;
importjava.io.InputStream;
importjava.io.OutputStream;
```

```
    classDatabaseHelper extends SQLiteOpenHelper {
    privatestatic String DB_PATH; // Повний шлях до
бази даних
    privatestatic String DB_NAME = "cityinfo.db";
    privatestatic final int SCHEMA = 1; // версія бази
даних
    staticfinal String TABLE = "users"; // Назва таблиці в
бд
    // назви стовпців
    staticfinal String COLUMN_ID = "_id";
    staticfinal String COLUMN_NAME="name";
    staticfinal String COLUMN_YEAR = "рок";
    privateContext myContext;

    DatabaseHelper(Context context) {
    super(context, DB_NAME, null, SCHEMA);
    this.myContext=context;
    DB_PATH       =context.getFilesDir().getPath()       +
DB_NAME;
    }

    @Override
    publicvoid onCreate(SQLiteDatabase db) { }
    @Override
    publicvoid    onUpgrade(SQLiteDatabase    db,    int
oldVersion, int newVersion) { }

    voidcreate_db(){

    File file = newFile(DB_PATH);
    if(!file.exists()) {
    //отримуємо локальну бд як потік
    try(InputStream                myInput                =
myContext.getAssets().open(DB_NAME);
    // Відкриваємо порожню бд
    OutputStream                myOutput                =
newFileOutputStream(DB_PATH)) {

    // побайтово копіюємо дані
    byte[] buffer = newbyte[1024];
    intlength;
    while((length = myInput.read(buffer)) > 0) {
    myOutput.write(buffer, 0, length);
    }
```

```
myOutput.flush();
}
catch(IOException ex) {
Log.d("DatabaseHelper", ex.getMessage());
}
}
}
publicSQLiteDatabase open()throws SQLException {

returnSQLiteDatabase.openDatabase(DB_PATH, null,
SQLiteDatabase.OPEN_READWRITE);
}
}
```

За замовчуванням база даних буде розміщуватися у зовнішньому сховищі, що виділяється для програми в папці /data/data/[назва_пакета]/databases/, і щоб отримати повний шлях до бази даних у конструкторі використовується вираз:

```
DB_PATH =context.getFilesDir().getPath() +
DB_NAME;
```

Метод onCreate() нам не потрібен, тому що нам не потрібне створення вбудованої бази даних. Зате тут визначено додатковий метод create_db(), метою якого є копіювання бази даних з папки assets в те місце, яке вказано в змінній DB_PATH.

Крім цього, тут також визначено метод відкриття бази даних open() за допомогою методу SQLiteDatabase.openDatabase()

Новий спосіб організації підключення змінить використання DatabaseHelper в activity. Так, оновимо клас MainActivity:

```
packagecom.example.sqliteapp;

importandroidx.appcompat.app.AppCompatActivity;

importandroid.content.Intent;
importandroid.view.View;
importandroid.widget.AdapterView;
importandroid.widget.SimpleCursorAdapter;
importandroid.database.Cursor;
importandroid.database.sqlite.SQLiteDatabase;
importandroid.os.Bundle;
importandroid.widget.ListView;
publicclass MainActivity extends AppCompatActivity {
ListView userList;
DatabaseHelper databaseHelper;
SQLiteDatabase db;
Cursor userCursor;
SimpleCursorAdapter userAdapter;
@Override
```

```
    protectedvoid onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    userList = findViewById(R.id.list);
    userList.setOnItemClickListener(newAdapterView.OnIte
mClickListener() {
    @Override
    publicvoid onItemClick(AdapterView<?> parent, View
view, int position, long id) {
    Intent intent = newIntent(getApplicationContext(),
UserActivity.class);
    intent.putExtra("id", id);
    startActivity(intent);
    }
    });
    databaseHelper =
newDatabaseHelper(getApplicationContext());
    // створюємо базу даних
    databaseHelper.create_db();
    }

    @Override
    publicvoid onResume() {
    super.onResume();
    // відкриваємо підключення
    db = databaseHelper.open();
    //отримуємо дані з бд як курсора
    userCursor = db.rawQuery("select * from"+
DatabaseHelper.TABLE, null);
    // визначаємо, які стовпці з курсору виводитимуться у
ListView
    String[] headers =
newString[]{DatabaseHelper.COLUMN_NAME,
DatabaseHelper.COLUMN_YEAR};
    // Створюємо адаптер, передаємо в нього курсор
    userAdapter = newSimpleCursorAdapter(this,
android.R.layout.two_line_list_item,
    користувачаCursor, headers,
    newint[]{android.R.id.text1, android.R.id.text2}, 0);
    userList.setAdapter(userAdapter);
    }

    // за натисканням на кнопку запускаємо UserActivity
для додавання даних
    publicvoid add (View view) {
```

```
Intent intent = newIntent(this, UserActivity.class);
startActivity(intent);
}

@Override
publicvoid onDestroy() {
super.onDestroy();
// Закриваємо підключення та курсор
db.close();
userCursor.close();
}
}
```

І також змінимо клас UserActivity:

```
packagecom.example.sqliteapp;

importandroidx.appcompat.app.AppCompatActivity;

importandroid.content.ContentValues;
importandroid.content.Intent;
importandroid.database.Cursor;
importandroid.database.sqlite.SQLiteDatabase;
importandroid.os.Bundle;
importandroid.view.View;
importandroid.widget.Button;
importandroid.widget.EditText;

publicclass UserActivity extends AppCompatActivity {

EditText nameBox;
EditText yearBox;
Button del Button;
Button saveButton;

DatabaseHelper sqlHelper;
SQLiteDatabase db;
Cursor userCursor;
longuserId=0;
@Override
protectedvoid onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_user);

nameBox = findViewById(R.id.name);
yearBox = findViewById(R.id.year);
delButton = findViewById(R.id.deleteButton);
```

```
    saveButton = findViewById(R.id.saveButton);

    sqlHelper = newDatabaseHelper(this);
    db = sqlHelper.open();

    Bundle extras = getIntent().getExtras();
    if(extras! = null) {
    userId = extras.getLong("id");
    }
    // якщо 0, то додавання
    if(userId > 0) {
    // отримуємо елемент з id з бд
    userCursor = db.rawQuery("select * from"+
DatabaseHelper.TABLE + "where" +
    DatabaseHelper.COLUMN_ID + "=?",
newString[]{String.valueOf(userId)});
    userCursor.moveToFirst();
    nameBox.setText(userCursor.getString(1));
    yearBox.setText(String.valueOf(userCursor.getInt(2)));
    userCursor.close();
    } else{
    // приховуємо кнопку видалення
    delButton.setVisibility(View.GONE);
    }
    }
    publicvoid save (View view) {
    ContentValues cv=newContentValues();
    cv.put(DatabaseHelper.COLUMN_NAME,
nameBox.getText().toString());
    cv.put(DatabaseHelper.COLUMN_YEAR,
Integer.parseInt(yearBox.getText().toString()));

    if(userId > 0) {
    db.update(DatabaseHelper.TABLE, cv,
DatabaseHelper.COLUMN_ID + "="+ userId, null);
    } else{
    db.insert(DatabaseHelper.TABLE, null, cv);
    }
    goHome();
    }
    publicvoid delete (View view) {
    db.delete(DatabaseHelper.TABLE, "_id = ?",
newString[]{String.valueOf(userId)});
    goHome();
    }
```

```
    privatevoid goHome(){
    // закриваємо підключення
    db.close();
    // Перехід до головної діяльності
    Intent intent = newIntent(this, MainActivity.class);
    intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP |
Intent.FLAG_ACTIVITY_SINGLE_TOP);
    startActivity(intent);
    }
    }
```

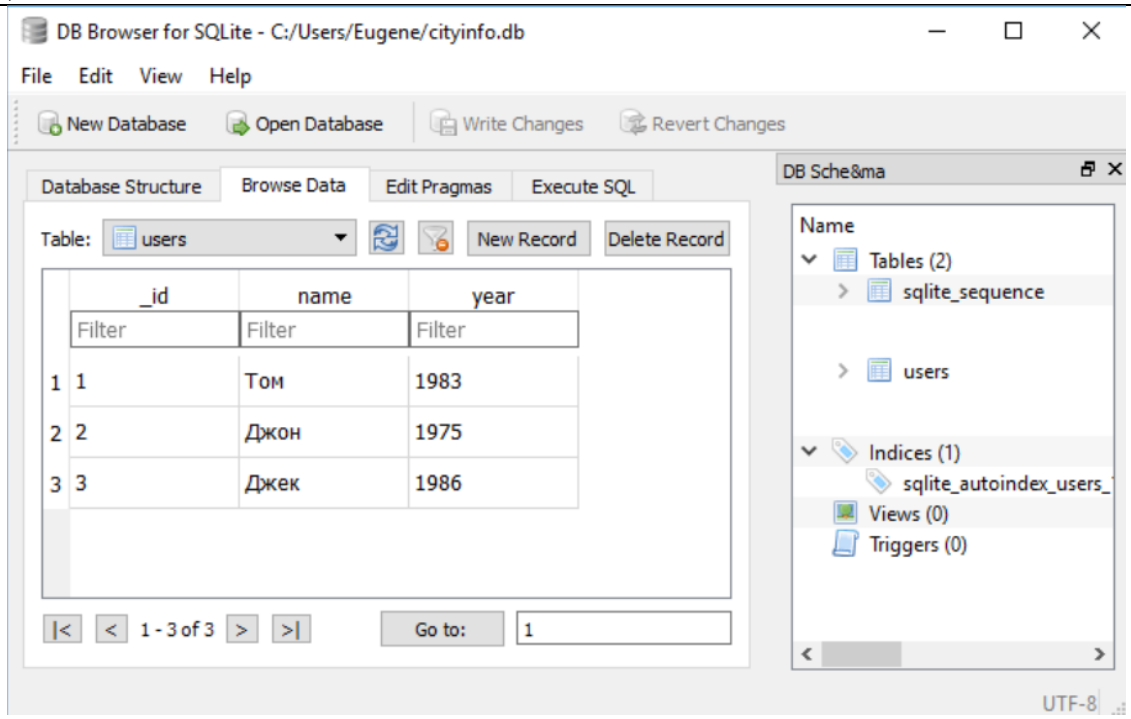
Вся решта роботи з даними буде тією ж, що й у минулих темах:

8.6. Динамічний пошук бази даних SQLite

Розглянемо, як ми можемо створити в додатку Android динамічний пошук по базі даних SQLite.

Отже, створимо новий проект із порожньою MainActivity. Для цього проекту візьмемо базу даних із минулої теми (або створимо нову). Ця база даних називається cityinfo і має одну таблицю users із трьома полями _id, name, age:

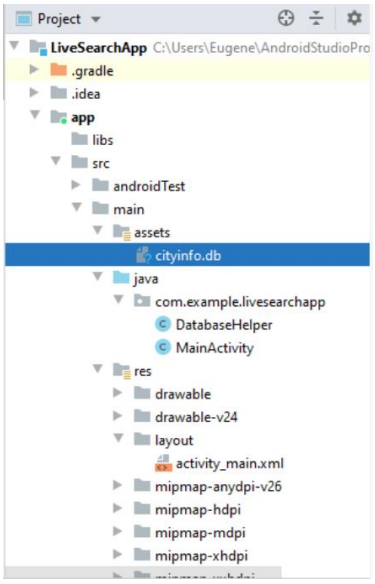
```
CREATETABLE `users` (
`_id` INTEGERNOT NULL PRIMARY KEY AUTOINCREMENT UNIQUE,
`name` TEXT NOTNULL,
`year` INTEGERNOT NULL
);
```



І також додамо в проект в Android Studio папку assets, а в папку assets - щойно створену базу даних:

У нашому випадку база даних має назву "cityinfo.db".

Як показано вище на скріншоті, додавши до проекту в одну папку з MainActivity новий клас DatabaseHelper:



```
packagecom.example.livesearchapp;

importandroid.database.SQLException;
importandroid.database.sqlite.SQLiteOpenHelper;
importandroid.database.sqlite.SQLiteDatabase;
importandroid.content.Context;
importandroid.util.Log;

importjava.io.File;
importjava.io.FileOutputStream;
importjava.io.IOException;
importjava.io.InputStream;
importjava.io.OutputStream;

classDatabaseHelper extends SQLiteOpenHelper {
privatestatic String DB_PATH; // Повний шлях до
бази даних
privatestatic String DB_NAME = "cityinfo.db";
privatestatic final int SCHEMA = 1; // версія бази
даних
staticfinal String TABLE = "users"; // Назва таблиці в
бд
// назви стовпців
staticfinal String COLUMN_ID = "_id";
staticfinal String COLUMN_NAME="name";
staticfinal String COLUMN_YEAR = "рок";
privateContext myContext;

DatabaseHelper(Context context) {
super(context, DB_NAME, null, SCHEMA);
this.myContext=context;
DB_PATH =context.getFilesDir().getPath() +
DB_NAME;
}

@Override
publicvoid onCreate(SQLiteDatabase db) { }
@Override
publicvoid onUpgrade(SQLiteDatabase db, int
oldVersion, int newVersion) { }

voidcreate_db(){

File file = newFile(DB_PATH);
if(!file.exists()) {
```

```
    //отримуємо локальну бд як потік
    try(InputStream myInput =
myContext.getAssets().open(DB_NAME);
    // Відкриваємо порожню бд
    OutputStream myOutput =
newFileOutputStream(DB_PATH)) {

    // побайтово копіюємо дані
    byte[] buffer = newbyte[1024];
    intlength;
    while((length = myInput.read(buffer)) > 0) {
    myOutput.write(buffer, 0, length);
    }
    myOutput.flush();
    }
    catch(IOException ex) {
    Log.d("DatabaseHelper", ex.getMessage());
    }
    }
    }
    publicSQLiteDatabase open()throws SQLException {

    returnSQLiteDatabase.openDatabase(DB_PATH, null,
SQLiteDatabase.OPEN_READWRITE);
    }
    }
```

Перейдемо до файлу activity_main.xml, який визначає візуальний інтерфейс, та змінимо його таким чином:

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

    <EditTextandroid:id="@+id/userFilter"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:hint="Пошук"
    app:layout_constraintBottom_toTopOf="@+id/userList"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintTop_toTopOf="parent"/>
```

```
    <ListView
    android:id="@+id/userList"
    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_toBottomOf="@+id/userFilter
"
    />

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

Отже, у нас буде елемент ListView для відображення списку та текстове поле для фільтрації.

Тепер змінимо код MainActivity:

```
    packagecom.example.livesearchapp;

    importandroidx.appcompat.app.AppCompatActivity;

    importandroid.database.Cursor;
    importandroid.database.SQLException;
    importandroid.database.sqlite.SQLiteDatabase;
    importandroid.os.Bundle;
    importandroid.text.Editable;
    importandroid.text.TextWatcher;
    importandroid.widget.EditText;
    importandroid.widget.FilterQueryProvider;
    importandroid.widget.ListView;
    importandroid.widget.SimpleCursorAdapter;

    publicclass MainActivity extends AppCompatActivity {

    DatabaseHelper sqlHelper;
    SQLiteDatabase db;
    Cursor userCursor;
    SimpleCursorAdapter userAdapter;
    ListView userList;
    EditText userFilter;

    @Override
    protectedvoid onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    userList = (ListView) findViewById (R.id.userList);
    userFilter = (EditText) findViewById (R.id.userFilter);
```

```
    sqlHelper                                                    =
newDatabaseHelper(getApplicationContext());
    // створюємо базу даних
    sqlHelper.create_db();
    }
    @Override
    publicvoid onResume() {
    super.onResume();
    try{
    db = sqlHelper.open();
    userCursor    =    db.rawQuery("select    *    from"+
DatabaseHelper.TABLE, null);
    String[]                    headers                          =
newString[]{DatabaseHelper.COLUMN_NAME,
DatabaseHelper.COLUMN_YEAR};
    userAdapter       =       newSimpleCursorAdapter(this,
android.R.layout.two_line_list_item,
    користувачаCursor,                                  headers,
newint[]{android.R.id.text1, android.R.id.text2}, 0);

    // якщо текстовому полі є текст, виконуємо
фільтрацію
    // Ця перевірка потрібна під час переходу від однієї
орієнтації екрану на іншу
    if(!userFilter.getText().toString().isEmpty())
    userAdapter.getFilter().filter(userFilter.getText().toStr
ing());

    // встановлення слухача зміни тексту
    userFilter.addTextChangedListener(newTextWatcher()
{

    publicvoid afterTextChanged(Editable s) { }

    publicvoid    beforeTextChanged(CharSequence    s,    int
start, int count, int after) { }
    // при зміні тексту виконуємо фільтрацію
    publicvoid onTextChanged(CharSequence s, int start,
int before, int count) {

    userAdapter.getFilter().filter(s.toString());
    }
    });
```

```
    // встановлюємо провайдер фільтрації
    userAdapter.setFilterQueryProvider(newFilterQueryPro
vider() {
    @Override
    publicCursor runQuery(CharSequence constraint) {

    if(constraint == null || constraint.length() == 0) {

    returndb.rawQuery("select * from " +
DatabaseHelper.TABLE, null);
    }
    else{
    returndb.rawQuery("select * from" +
DatabaseHelper.TABLE + "where" +
    DatabaseHelper.COLUMN_NAME + " like ?",
newString[]{"%" + constraint.toString() + "%"});
    }
    }
    });

    userList.setAdapter(userAdapter);
    }
    catch(SQLException ex){}
    }
    @Override
    publicvoid onDestroy(){
    super.onDestroy();
    // Закриваємо підключення та курсор
    db.close();
    userCursor.close();
    }
    }
```

Перш за все треба зазначити, що для фільтрації даних в адаптері, нам треба отримати фільтр адаптера, а цей фільтр виконати метод filter():

```
    userAdapter.getFilter().filter(s.toString());
```

Цей метод filter() передає ключ пошуку.

Для текстового поля ми можемо відстежувати зміни вмісту за допомогою слухача:

```
    userFilter.addTextChangedListener(newTextWatcher()
{

    publicvoid afterTextChanged(Editable s) {
    }

    publicvoid beforeTextChanged(CharSequence s, int
start, int count, int after) {
```

```
    }
    // при зміні тексту виконуємо фільтрацію
    publicvoid onTextChanged(CharSequence s, int start,
int before, int count) {

    userAdapter.getFilter().filter(s.toString());
    }
    });
```

У слухачі TextWatcher у методі onTextChanged якраз і викликається метод filter(), який передається введена користувачем у текстове поле послідовність символів.

Сам виклик методу filter() мало впливає. Нам нало ще визначити провайдер фільтрації адаптера, які і інкапсулюватиме реальну логіку фільтрації:

```
    userAdapter.setFilterQueryProvider(newFilterQueryProvi
der() {
    @Override
    publicCursor runQuery(CharSequence constraint) {

    if(constraint == null || constraint.length() == 0) {

    returndb.rawQuery("select * from " +
DatabaseHelper.TABLE, null);
    }
    else{
    returndb.rawQuery("select * from" +
DatabaseHelper.TABLE + "where" +
    DatabaseHelper.COLUMN_NAME + " like ?",
newString[]{"%" + constraint.toString() + "%"});
    }
    }
    });
```

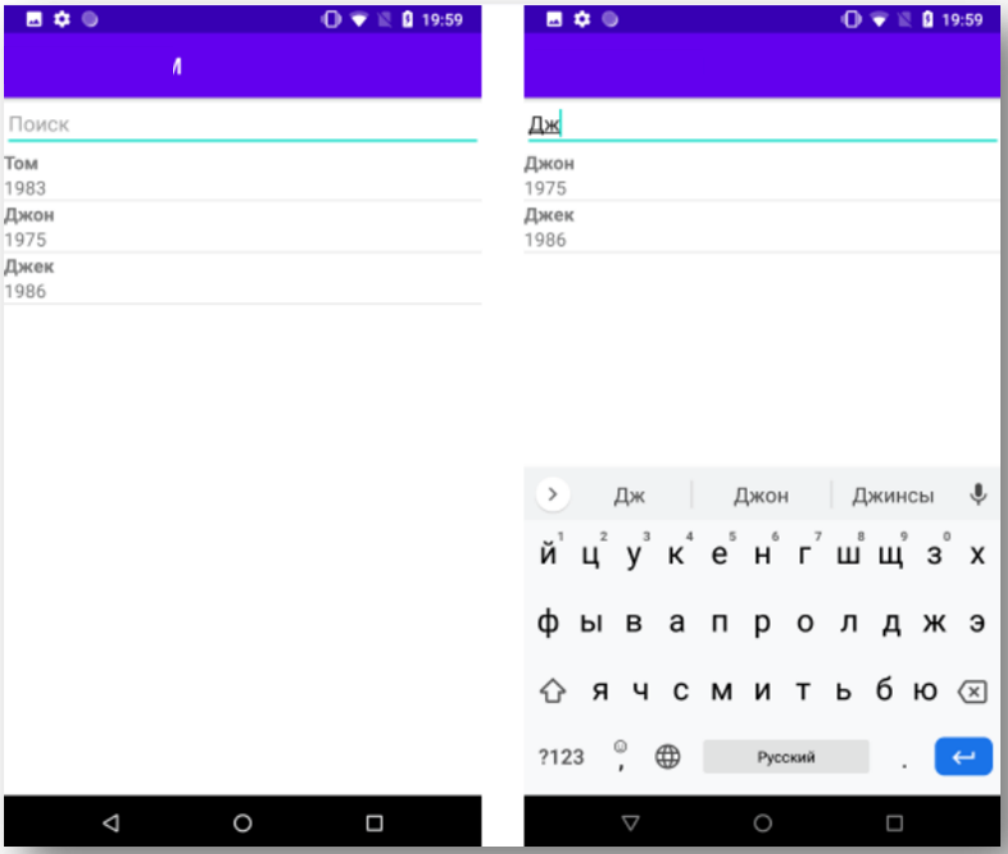
Сутність цього провайдера полягає у виконанні SQL-виражень до бд, а саме конструкцій "select from" та "select from where like". Дані найпростіші вирази виконують регістрозалежну фільтрацію. В результаті адаптар отримує фільтровані дані.

Слід також зазначити наступний код:

```
    if(!userFilter.getText().toString().isEmpty())
    userAdapter.getFilter().filter(userFilter.getText().toStri
ng());
```

Цей код нам потрібен при зміні орієнтації (наприклад, з портретної на альбомну). І якщо орієнтація пристрою змінена, але в текстовому полі все ж таки є деякі текст-фільтр, то виконується фільтрація. Інакше вона не виконувалася б.

І після запуску ми зможемо насолодитися фільтруванням даних:


19:59
Поиск
Том
1983
Джон
1975
Джек
1986
19:59
Дж
Джон
1975
Джек
1986
Дж
Джон
Джинсы
?123
Русский

8.7. Модель, репозиторій та робота з базою даних

У попередніх темах була розглянута взаємодія з базою даних через клас SimpleCursorAdapter. Але є й інші методи роботи з даними, коли ми абстрагуємося від структури таблиці і працюємо через модель, проте взаємодія з базою даних виробляється практично через реалізацію паттерна репозиторій.

Так, створимо новий проект з порожньою MainActivity і насамперед додамо до нього клас моделі, який назвемо User:

```
packagecom.example.databaseadapterapp;
publicclass User {
privatelong id;
privateString name;
privateint year;
User(longid, String name, int year) {
this.id = id;
this.name = name;
this.year = year;
}
publiclong getId() {
returnid;
}
publicString getName() {
returnname;
}
publicvoid setName(String name) {
this.name = name;
}
publicint getYear() {
returnyear;
}
publicvoid setYear(int year) {
this.year = year;
}
@Override
publicString toString() {
returnthis.name + ":" + this.year;
}
}
```

У цьому проекті ми будемо працювати фактично з тими самими даними, що й раніше з даними користувачів, які мають унікальний ідентифікатор, ім'я та рік народження. І модель User описує ці дані.

Для взаємодії з базою даних SQLite додамо новий клас DatabaseHelper:

```
packagecom.example.databaseadapterapp;

importandroid.database.sqlite.SQLiteOpenHelper;
```

```
importandroid.database.sqlite.SQLiteDatabase;
importandroid.content.Context;

publicclass DatabaseHelper extends SQLiteOpenHelper
{
privatestatic final String DATABASE_NAME =
"userstore.db"; // Назва бд
privatestatic final int SCHEMA = 1; // версія бази
даних
staticfinal String TABLE = "users"; // Назва таблиці в
бд
// назви стовпців
publicstatic final String COLUMN_ID = "_id";
publicstatic final String COLUMN_NAME="name";
publicstatic final String COLUMN_YEAR = "рок";

publicDatabaseHelper(Context context) {
super(context, DATABASE_NAME, null, SCHEMA);
}

@Override
publicvoid onCreate(SQLiteDatabase db) {

db.execSQL("CREATE TABLE"+ TABLE + "(" +
COLUMN_ID
+ " INTEGER PRIMARY KEY AUTOINCREMENT,"+
COLUMN_NAME
+ "TEXT,"+ COLUMN_YEAR + "INTEGER);");
// Додавання початкових даних
db.execSQL("INSERT INTO "+ TABLE +" ("+
COLUMN_NAME
+ ","+ COLUMN_YEAR + ") VALUES ('Том Сміт',
1981);");
}
@Override
publicvoid onUpgrade(SQLiteDatabase db, int
oldVersion, int newVersion) {
db.execSQL("DROP TABLE IF EXISTS"+TABLE);
onCreate(db);
}
}
```

Також для роботи з базою даних додамо до проекту клас DatabaseAdapter:

```
packagecom.example.databaseadapterapp;
```

```
importandroid.content.ContentValues;
importandroid.content.Context;
importandroid.database.Cursor;
importandroid.database.DatabaseUtils;
importandroid.database.sqlite.SQLiteDatabase;
importjava.util.ArrayList;
importjava.util.List;

publicclass DatabaseAdapter {

privateDatabaseHelper dbHelper;
privateSQLiteDatabase database;

publicDatabaseAdapter(Context context){
dbHelper =
newDatabaseHelper(context.getApplicationContext());
}

publicDatabaseAdapter open(){
database = dbHelper.getWritableDatabase();
returnthis;
}

publicvoid close(){
dbHelper.close();
}

privateCursor getAllEntries(){
String[] columns = newString[]
{DatabaseHelper.COLUMN_ID,
DatabaseHelper.COLUMN_NAME,
DatabaseHelper.COLUMN_YEAR};
returndatabase.query(DatabaseHelper.TABLE, columns,
null, null, null, null, null);
}

publicList<User> getUsers(){
ArrayList<User> users = newArrayList<>();
Cursor cursor = getAllEntries();
while(cursor.moveToNext()){
intid =
cursor.getInt(cursor.getColumnIndex(DatabaseHelper.COLU
MN_ID));
String name =
cursor.getString(cursor.getColumnIndex(DatabaseHelper.C
```

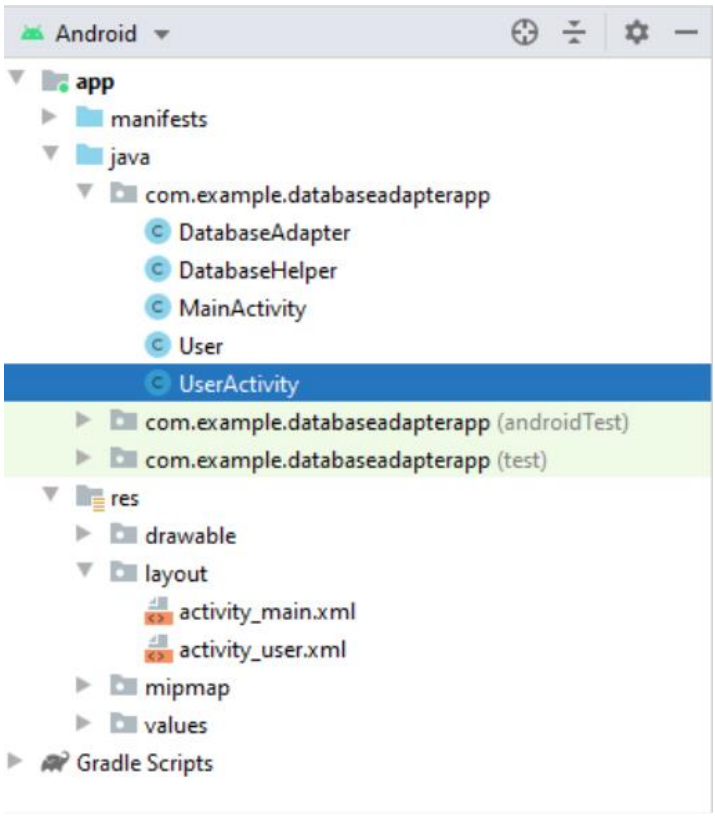
```
OLUMN_NAME));
    intyear =
cursor.getInt(cursor.getColumnIndex(DatabaseHelper.COLU
MN_YEAR));
    users.add(newUser(id, name, year));
    }
    cursor.close();
    returnusers;
    }

    publiclong getCount(){
    returnDatabaseUtils.queryNumEntries(database,
DatabaseHelper.TABLE);
    }

    publicUser getUser(long id){
    User user=null;
    String query = String.format("SELECT * FROM %s
WHERE %s=?",DatabaseHelper.TABLE,
DatabaseHelper.COLUMN_ID);
    Cursor cursor = database.rawQuery(query,
newString[]{ String.valueOf(id)});
    if(cursor.moveToFirst()){
    String name =
cursor.getString(cursor.getColumnIndex(DatabaseHelper.C
OLUMN_NAME));
    intyear =
cursor.getInt(cursor.getColumnIndex(DatabaseHelper.COLU
MN_YEAR));
    user=newUser(id, name, year);
    }
    cursor.close();
    returnuser;
    }

    publiclong insert(User user){

    ContentValues cv=newContentValues();
    cv.put(DatabaseHelper.COLUMN_NAME,
user.getName());
    cv.put(DatabaseHelper.COLUMN_YEAR,
user.getYear());

    returndatabase.insert(DatabaseHelper.TABLE, null,
cv);
```

```
    }

    publiclong delete(long userId){

    String whereClause = "_id = ?";
    String[]                whereArgs                =
newString[]{String.valueOf(userId)};
    returndatabase.delete(DatabaseHelper.TABLE,
whereClause, whereArgs);
    }

    publiclong update(User user){

    String whereClause = DatabaseHelper.COLUMN_ID +
"="+ user.getId();
    ContentValues cv=newContentValues();
    cv.put(DatabaseHelper.COLUMN_NAME,
user.getName());
    cv.put(DatabaseHelper.COLUMN_YEAR,
user.getYear());
    returndatabase.update(DatabaseHelper.TABLE,      cv,
whereClause, null);
    }
    }
```

Фактично цей клас виконує роль репозиторію даних. Щоб взаємодіяти з БД він визначає методи open() та close(), які відповідно відкривають та закривають підключення до бази даних.

Безпосередньо для роботи з даними в класі визначені методи insert() (додавання), delete() (видалення), update() (оновлення), getUsers() (отримання всіх користувачів з таблиці) та getUser() (отримання одного користувача за id).

Як інтерфейс користувача будемо відштовхуватися від того функціоналу, який використовувався в минулих темах. Так, додамо до проекту новий клас Activity - UserActivity. У результаті весь проект виглядатиме так:



У файлі activity_user.xml у папці res/layout визначимо для UserActivity найпростіший інтерфейс:

```
<?xmlversion="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/name"
android:layout_width="0dp"
android:layout_height="wrap_content"
android:hint="Введіть ім'я"
app:layout_constraintBottom_toTopOf="@+id/year"
app:layout_constraintTop_toTopOf="parent"
app:layout_constraintLeft_toLeftOf="parent"
app:layout_constraintRight_toRightOf="parent"/>
<EditText
android:id="@+id/year"
android:layout_width="0dp"
android:layout_height="wrap_content"
android:hint="Введіть рік народження"
app:layout_constraintTop_toBottomOf="@+id/name"
app:layout_constraintBottom_toTopOf="@+id/saveButto
n"
```

```
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent"/>
    <Button
    android:id="@+id/saveButton"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:text="Зберегти"
    android:onClick="save"
    app:layout_constraintHorizontal_weight="1"
    app:layout_constraintTop_toBottomOf="@+id/year"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toLeftOf="@+id/deleteButt
on"
    />
    <Button
    android:id="@+id/deleteButton"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:text="Видалити"
    android:onClick="delete"
    app:layout_constraintHorizontal_weight="1"
    app:layout_constraintTop_toBottomOf="@+id/year"
    app:layout_constraintLeft_toRightOf="@+id/saveButton
"
    app:layout_constraintRight_toRightOf="parent"
    />
    </androidx.constraintlayout.widget.ConstraintLayout>
```

У класі UserActivity визначимо логіку додавання/зміни/видалення користувача:

```
    packagecom.example.databaseadapterapp;

    importandroidx.appcompat.app.AppCompatActivity;

    importandroid.content.Intent;
    importandroid.os.Bundle;
    importandroid.view.View;
    importandroid.widget.Button;
    importandroid.widget.EditText;

    publicclass UserActivity extends AppCompatActivity {

    privateEditText nameBox;
    privateEditText yearBox;
    privateButton del Button;
    privateDatabaseAdapter adapter;
```

```
    privatelong userId=0;
    @Override
    protectedvoid onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_user);
    nameBox = findViewById(R.id.name);
    yearBox = findViewById(R.id.year);
    delButton = findViewById(R.id.deleteButton);
    adapter=newDatabaseAdapter(this);
    Bundle extras = getIntent().getExtras();
    if(extras! = null) {
    userId = extras.getLong("id");
    }
    // якщо 0, то додавання
    if(userId > 0) {
    // отримуємо елемент з id з бд
    adapter.open();
    User user = adapter.getUser(userId);
    nameBox.setText(user.getName());
    yearBox.setText(String.valueOf(user.getYear()));
    adapter.close();
    } else{
    // приховуємо кнопку видалення
    delButton.setVisibility(View.GONE);
    }
    }
    publicvoid save (View view) {
    String name = nameBox.getText().toString();
    intyear                                                    =
Integer.parseInt(yearBox.getText().toString());
    User user=newUser(userId, name, year);

    adapter.open();
    if(userId > 0) {
    adapter.update(user);
    } else{
    adapter.insert(user);
    }
    adapter.close();
    goHome();
    }
    publicvoid delete (View view) {
    adapter.open();
    adapter.delete(userId);
    adapter.close();
```

```
goHome();
}
privatevoid goHome(){
// Перехід до головної діяльності
Intent intent = newIntent(this, MainActivity.class);
intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP
| Intent.FLAG_ACTIVITY_SINGLE_TOP);
startActivity(intent);
}
}
```

Ця дія використовується для додавання/редагування/видалення одного об'єкта User. Якщо в UserActivity передається параметр id, то ми перебуваємо в режимі редагування користувача, тому звертаємося до методу getUser() класу DatabaseAdapter для отримання потрібного користувача.

Для додавання/зміни/видалення користувача натисканням на кнопку викликається відповідний метод класу DatabaseAdapter.

У файлі activity_main.xml у папці res/layout визначимо візуальний інтерфейс для MainActivity:

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

<Button
android:id="@+id/addButton"
android:layout_width="0dp"
android:layout_height="wrap_content"
android:text="Додати"
android:onClick="add"
app:layout_constraintBottom_toTopOf="@+id/list"
app:layout_constraintTop_toTopOf="parent"
app:layout_constraintLeft_toLeftOf="parent"
app:layout_constraintRight_toRightOf="parent"
/>
<ListView
android:id="@+id/list"
android:layout_width="0dp"
android:layout_height="0dp"
app:layout_constraintTop_toBottomOf="@+id/addButto
n"
app:layout_constraintBottom_toBottomOf="parent"
app:layout_constraintLeft_toLeftOf="parent"
app:layout_constraintRight_toRightOf="parent"/>
```

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

Тут є елемент ListView для виведення об'єктів з таблиці та кнопка переходу до UserActivity для додавання користувача.

І змінимо код MainActivity:

```
packagecom.example.databaseadapterapp;

importandroidx.appcompat.app.AppCompatActivity;

importandroid.content.Intent;
importandroid.os.Bundle;
importandroid.view.View;
importandroid.widget.AdapterView;
importandroid.widget.ArrayAdapter;
importandroid.widget.ListView;

importjava.util.List;

publicclass MainActivity extends AppCompatActivity {

privateListView userList;
ArrayAdapter<User> arrayAdapter;
@Override
protectedvoid onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_main);

userList = findViewById(R.id.list);

userList.setOnItemClickListener(newAdapterView.OnIte
mClickListener() {
@Override
publicvoid onItemClick(AdapterView<?> parent, View
view, int position, long id) {
User user =arrayAdapter.getItem(position);
if(user!=null) {
Intent intent = newIntent(getApplicationContext(),
UserActivity.class);
intent.putExtra("id", user.getId());
startActivity(intent);
}
}
});
}
```

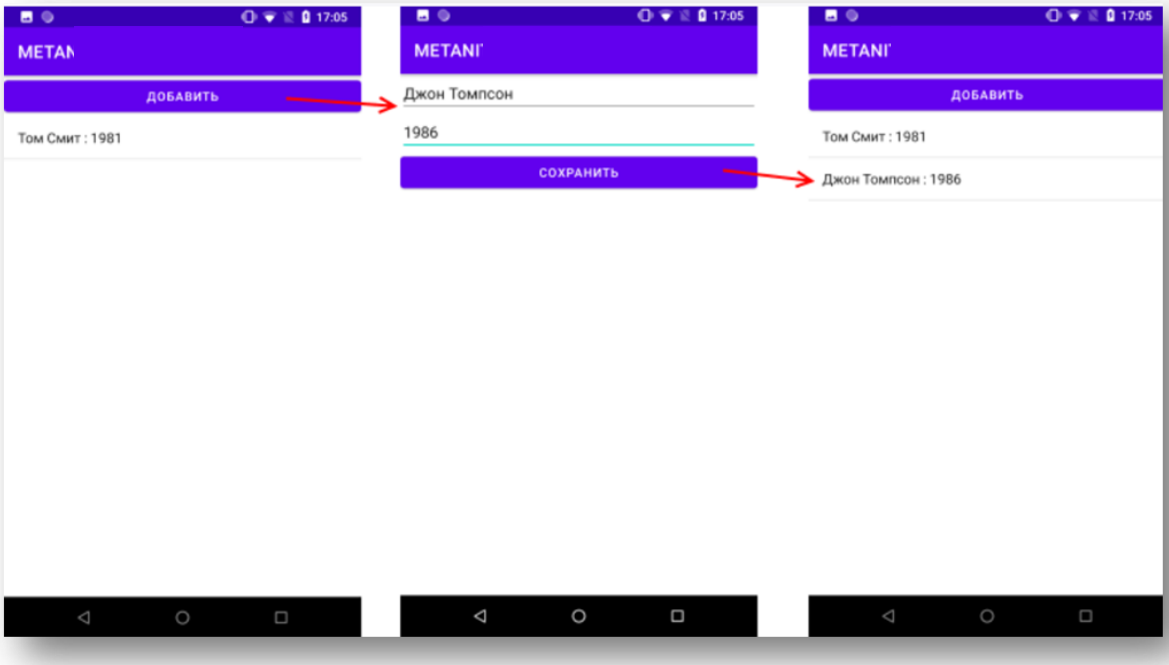
```
    @Override
    publicvoid onResume() {
    super.onResume();
    DatabaseAdapter adapter=newDatabaseAdapter(this);
    adapter.open();

    List<User> users = adapter.getUsers();

    arrayAdapter = новийArrayAdapter<>(this,
android.R.layout.simple_list_item_1, users);
    userList.setAdapter(arrayAdapter);
    adapter.close();
    }
    // за натисканням на кнопку запускаємо UserActivity
для додавання даних
    publicadd(View view){
    Intent intent = newIntent(this, UserActivity.class);
    startActivity(intent);
    }
    }
```

У перевизначеному методі onResume() через об'єкт DatabaseAdapter отримуємо всіх користувачів з бази даних і через ArrayAdapter виводимо їх у ListView.

При натисканні на елемент ListView запускаємо UserActivity, передаючи їй ID виділеного користувача.

При натисканні на кнопку просто викликаємо UserActivity.

При запуску MainActivity відобразить список користувачів з бази даних, а під час переходу до UserActivity ми зможемо підредагувати або додати користувачів:



Навчальне видання

КОНСПЕКТ ЛЕКЦІЙ
навчально-методичного комплексу
дистанційного курсу дисципліни
«Програмування для мобільних платформ»
Частина ІІ
(для студентів спеціальностей
F2 «Інженерія програмного забезпечення»,
F6 «Інформаційні технології та системи»)

Укладач:
Ратов Денис Валентинович

Редактор *Д.В.Ратов*
Техн. редактор *Д.В.Ратов*
Оригінал - макет *Д.В.Ратов*

Підписано до друку _____
Формат $60\times84\,{}^{1}\!/_{16}$. Папір типограф. Гарнитура *Times*.
Друк офсетний. Ум. друк. арк.___. Обл.-вид.арк._____.
Тираж ___ прим. Вид. №_____. Замовл. №_____. Ціна договірна.

Видавництво Східноукраїнського національного університету
імені Володимира Даля

Свідоцтво про реєстрацію: серія ДК № 1620 від 18.12.03 р.
Адреса університета: вул. Іоанна Павла ІІ, 17
м. Київ, 01042, Україна
e-mail: vidavnictvoSNU.ua@gmail.com.