**ЛЕКЦІЯ 11. ЕЛЕМЕНТИ УПРАВЛІННЯ**

**6.1 Елементи управління TextView, EditView, Button, Checkbox, RadioButton.**

**Створення обробників подій та прив’язка їх до елементів управління**

**6.1.1 TextView**

Для простого перегляду тексту на екрані призначений елемент *TextView*. Він відображає текст без можливості редагування. Деякі його основні атрибути:

● **android:text**: встановлює текст елемента;

● **android:textSize**: встановлює висоту тексту, в якості одиниць виміру для вказівки висоти використовуються *sp*;

● **android:background**: задає фоновий колір елемента у вигляді кольору в шістнадцятковому вигляді або у вигляді колірного ресурсу;

● **android:textColor**: задає колір тексту;

● **android:textAllCaps**: при значенні *true* робить всі символи в тексті великими;

● **android:textDirection**: встановлює напрямок тексту. За замовчуванням використовується напрямок зліва направо, але за допомогою значення *rtl* можна встановити напрямок справа наліво;

● **android:textAlignment**: задає вирівнювання тексту. Може приймати наступні значення:

o center : вирівнювання по центру;

o textStart : на початку тексту відносно абзацу;

o textEnd : в кінці тексту;

o viewStart : по лівому краю (відносно екрана);

o viewEnd : по правому краю;

● **android:fontFamily**: встановлює тип шрифту. Може приймати наступні значення:

o monospace;

o serif;

o serif-monospace;

o sans-serif;

o sans-serif-condensed;

o sans-serif-smallcaps;

o sans-serif-light;

o casual;

o cursive.

Наприклад, визначимо три текстових поля:

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

<LinearLayout xmlns:android=["http://schemas.android.com/apk/res/android"](http://schemas.android.com/apk/res/android) android:id=["@+id](mailto:@+id)/activity\_main"

android:orientation="vertical" android:layout\_width="match\_parent" android:layout\_height="match\_parent">

<TextView android:layout\_height="wrap\_content" android:layout\_width="match\_parent" android:layout\_margin="10dp" android:text="Hello Android 7" android:fontFamily="sans-serif" android:textSize="26sp" android:background="#ffebee" android:textColor="#f44336" />

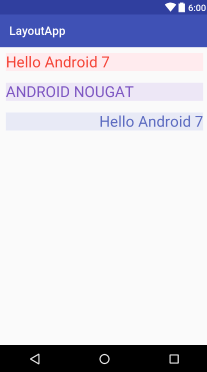
<TextView android:layout\_height="wrap\_content" android:layout\_width="match\_parent" android:layout\_margin="10dp" android:text="Android Nougat" android:textAllCaps="true" android:textSize="26sp" android:background="#ede7f6" android:textColor="#7e57c2" />

<TextView android:layout\_height="wrap\_content" android:layout\_width="match\_parent" android:layout\_margin="10dp" android:text="Hello Android 7" android:textAlignment="textEnd"

android:textSize="26sp" android:background="#e8eaf6" android:textColor="#5c6bc0" />

</LinearLayout>

На екрані вони відображуватимуться так:



Задання елемента в коді теж не є складним. Наприклад, створимо елемент і виведемо його на екран:

public class MainActivity extends AppCompatActivity {

[@Override](mailto:@Override)

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

LinearLayout linearLayout=new LinearLayout(this); TextView textView1=new TextView(this); textView1.setBackgroundColor(0xffe8eaf6); // колір фону textView1.setTextColor(0xff5c6bc0); // колір тексту textView1.setAllCaps(true); // всі букви великі textView1.setTextAlignment(TextView.TEXT\_ALIGNMENT\_CENTER); //

вирівнювання тексту по центру

textView1.setText("Android Nougat 7"); // текстове значення textView1.setTypeface(Typeface.create("casual", Typeface.NORMAL)); // шрифт textView1.setTextSize(26); // висота тексту

LinearLayout.LayoutParams layoutParams = new LinearLayout.LayoutParams

(ViewGroup.LayoutParams.MATCH\_PARENT, ViewGroup.LayoutParams.WRAP\_CONTENT);

layoutParams.setMargins(20,20,20,20); // зовнішні відступи textView1.setLayoutParams(layoutParams); // розміри linearLayout.addView(textView1);

setContentView(linearLayout);

}

}

Іноді необхідно вивести на екран посилання або телефон, після натискання на

які відбувалася б певна дія. Для цього в *TextView* визначено атрибут **android:autoLink** :

<TextView android:id=["@+id/display\_message" android:text="Посетите сайт https://metanit.com" android:textSize="21sp" android:layout\_width="wrap\_content" android:layout\_height="wrap\_content" android:autoLink="web|email" />](mailto:@+id)



Властивість **android:autoLink** може приймати декілька значень:

o none: відключає всі посилання;

o web: включає всі веб-посилання;

o email: включає посилання на електронні адреси; o phone: включає посилання на номери телефонів; o map: включає посилання на карту;

o all: включає всі перераховані вище посилання.

Тобто при налаштуванні android: *autoLink="web"* якщо в тексті є адреса *url*, то ця адреса буде виділятися, а при натисканні на нього буде здійснено перехід до веб- браузера, який відкриє сторінку за цією адресою. За допомогою прямої риски ми можемо об’єднувати умови, як в даному випадку: *android:autoLink="web|email*".

**6.1.2 EditText**

Елемент EditText є підкласом класу *TextView*. Він також представляє текстове поле, але вже з можливістю введення і редагування тексту. Таким чином, в EditText ми можемо використовувати всі ті ж можливості, що і в *TextView*.

З тих атрибутів, що не розглядалися в підрозділі про *TextView*, слід зазначити атрибут *android:hint*. Він дозволяє задати текст, який буде відображатися в якості підказки, якщо елемент *EditText* порожній. Крім того, ми можемо використовувати атрибут *android:inputType*, який дозволяє задати клавіатуру для введення. Зокрема, серед його значень можна виділити наступні:

● **text:** звичайна клавіатура для введення однорядкового тексту;

● **textMultiLine:** багаторядкове текстове поле;

● **textEmailAddress:** звичайна клавіатура, на якій присутній символ @,

орієнтована на введення *email*;

● **textUri:** звичайна клавіатура, на якій присутній символ /, орієнтована на введення інтернет-адрес;

● **textPassword:** клавіатура для введення пароля;

● **textCapWords:** при введенні перший введений символ слова являє велику літеру, інші – малі;

● **number:** числова клавіатура;

● **phone:** клавіатура в стилі звичайного телефону;

● **date:** клавіатура для введення дати;

● **time:** клавіатура для введення часу;

● **datetime:** клавіатура для введення дати і часу.

Використовуємо EditText на практиці:

<LinearLayout xmlns:andr[oid="http://schemas.android.com/apk/res/androi](http://schemas.android.com/apk/res/android)d" android:id=["@+id](mailto:@+id)/activity\_main" android:layout\_width="match\_parent" android:layout\_height="match\_parent" android:orientation="vertical"

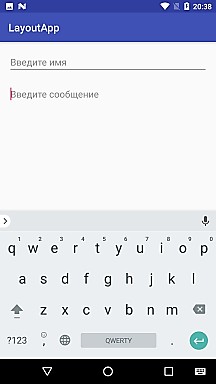
android:padding="16dp">

<EditText android:layout\_width="match\_parent" android:layout\_height="wrap\_content" android:hint="Введите имя" />

<EditText android:layout\_marginTop="16dp" android:layout\_width="match\_parent" android:layout\_height="match\_parent" android:hint="Введіть повідомлення" android:inputType="textMultiLine" android:gravity="top" />

</LinearLayout>

Перше поле тут звичайне однорядкове, а друге – багаторядкове. Щоб в другому полі текст вирівнювався по верху, додатково встановлюється атрибут *android:gravity="top"*. В результаті отримаємо:



Однією з можливостей елемента *EditText* також є можливість обробити введені символи під час введення користувача. Для цього визначимо в файлі *activity\_main.xml* наступну розмітку:

<LinearLayout xmlns:andr[oid="http://schemas.android.com/apk/res/androi](http://schemas.android.com/apk/res/android)d" android:id=["@+id](mailto:@+id)/activity\_main" android:layout\_width="match\_parent" android:layout\_height="match\_parent" android:orientation="vertical"

android:padding="16dp">

<TextView android:id=["@+id](mailto:@+id)/textView" android:textSize="34sp" android:layout\_width="match\_parent" android:layout\_height="wrap\_content" />

<EditText android:layout\_width="match\_parent" android:layout\_height="wrap\_content" android:hint="Введіть ім’я" android:id=["@+id](mailto:@+id)/editText" />

</LinearLayout>

Передбачається, що введені в *EditText* символи тут же будуть відображатися в елементі *TextView*. Для цього також змінимо код *MainActivity*:

public class MainActivity extends AppCompatActivity {

[@Override](mailto:@Override)

protected void onCreate(Bundle savedInstanceState) { super.onCreate(savedInstanceState); setContentView(R.layout.activity\_main);

EditText editText = (EditText) findViewById(R.id.editText);

editText.addTextChangedListener(new TextWatcher() {

public void afterTextChanged(Editable s) {}

public void beforeTextChanged(CharSequence s, int start, int count, int after) {

}

public void onTextChanged(CharSequence s, int start, int before, int count) {

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

textView.setText(s);

}

});

}

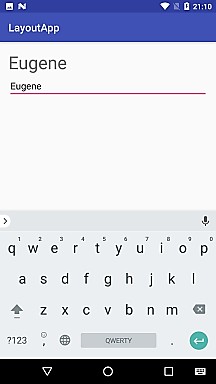
}

За допомогою методу *addTextChangedListener*() тут до елементу *EditText*

додається слухач введення тексту – об’єкт *TextWatcher*. Для його використання нам треба реалізувати три методу, але в реальності нам вистачить реалізації методу *onTextChanged*, який викликається при зміні тексту. Введений текст передається в цей метод в якості параметра *CharSequence*. У самому методі просто передаємо цей текст в елемент *TextView*.

В результаті при введенні в EditText все символи також будуть відображатися в

*TextView*:



**6.1.3 Button**

Одним з часто використовуваних елементів є кнопки, які представлені класом *android.widget.Button*. Ключовою особливістю кнопок є можливість взаємодії з користувачем через натискання. Деякі ключові атрибути, які можна задати у кнопок:

● **text:** задає текст на кнопці;

● **textColor:** задає колір тексту на кнопці;

● **background:** задає фоновий колір кнопки;

● **textAllCaps:** при значенні *true* встановлює текст в верхньому регістрі. За замовчуванням якраз і застосовується значення true;

● **onClick:** задає обробник натиснення кнопки. Змінимо код в *activity\_main.xml* наступним чином:

<LinearLayout xmlns:andr[oid="http://schemas.android.com/apk/res/androi](http://schemas.android.com/apk/res/android)d" android:id=["@+id](mailto:@+id)/activity\_main" android:layout\_width="match\_parent" android:layout\_height="match\_parent" android:orientation="vertical"

android:padding="16dp">

<TextView android:id=["@+id](mailto:@+id)/textView" android:textSize="34sp"

android:layout\_width="match\_parent" android:layout\_height="wrap\_content" />

<EditText android:layout\_width="match\_parent" android:layout\_height="wrap\_content" android:hint="Введіть ім’я" android:id=["@+id](mailto:@+id)/editText" />

<Button android:layout\_width="wrap\_content" android:layout\_height="wrap\_content" android:text="Click" android:onClick="sendMessage" />

</LinearLayout>

За допомогою атрибута **android:onClick** можна задати метод в коді Java, який буде обробляти натискання кнопки. Так, в наведеному вище прикладі це метод *sendMessage*. Тепер перейдемо до коду *MainActivity* і пропишемо в ньому такий метод: public class MainActivity extends AppCompatActivity {

[@Override](mailto:@Override)

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

}

// Обробка натискання на кнопку

public void sendMessage(View view) {

TextView textView=(TextView) findViewById(R.id.textView); EditText editText=(EditText) findViewById(R.id.editText);

textView.setText("Ласкаво просимо, "+editText.getText());

}

}

При створенні методу для обробки натискання слід враховувати наступні

моменти:

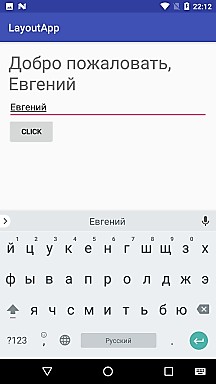
● метод повинен оголошуватися з модифікатором *public*;

● метод повинен повертати значення *void*;

● як параметр метод повинен приймати об’єкт *View*, який представляє натиснуту кнопку.

В даному випадку після натискання на кнопку в *TextView* виводиться текст з

*EditText*:



Аналогічний приклад повністю в коді MainActivity:

public class MainActivity extends AppCompatActivity { EditText editText;

TextView textView;

[@Override](mailto:@Override)

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

LinearLayout linearLayout=new LinearLayout(this); linearLayout.setOrientation(LinearLayout.VERTICAL); textView=new TextView(this); textView.setLayoutParams(new LinearLayout.LayoutParams(

LinearLayout.LayoutParams.MATCH\_PARENT, LinearLayout.LayoutParams.WRAP\_CONTENT

));

linearLayout.addView(textView);

editText=new EditText(this); editText.setHint("Введіть ім’я"); editText.setLayoutParams(new LinearLayout.LayoutParams(

LinearLayout.LayoutParams.MATCH\_PARENT, LinearLayout.LayoutParams.WRAP\_CONTENT

));

linearLayout.addView(editText);

Button button=new Button(this);

button.setText("CLICK");

button.setLayoutParams(new LinearLayout.LayoutParams( LinearLayout.LayoutParams.WRAP\_CONTENT,

LinearLayout.LayoutParams.WRAP\_CONTENT

));

linearLayout.addView(button);

button.setOnClickListener(new View.OnClickListener() {

public void onClick(View v) {

textView.setText("Ласкаво просимо, "+editText.getText());

}

});

setContentView(linearLayout);

}

}

При програмному створенні кнопки ми можемо визначити у неї слухач

натискання *View.OnClickListener* і за допомогою його методу *onClick* також обробити натискання:

button.setOnClickListener(new View.OnClickListener() {

public void onClick(View v) {

// Обробка натискання

}

});

**6.1.4 Checkbox**

Елементами *Checkbox* є прапорці, які можуть перебувати в відміченому і невіміченому стані. Прапорці дозволяють виробляти множинний вибір з кількох значень. Визначимо в файлі розмітки *activity\_main.xml* кілька прапорців:

<LinearLayout xmlns:andr[oid="http://schemas.android.com/apk/res/androi](http://schemas.android.com/apk/res/android)d" android:id=["@+id](mailto:@+id)/activity\_main" android:layout\_width="match\_parent" android:layout\_height="match\_parent" android:orientation="vertical"

android:padding="16dp">

<TextView android:id=["@+id](mailto:@+id)/selection" android:layout\_width="match\_parent" android:layout\_height="wrap\_content" android:textSize="26sp" />

<CheckBox android:id=["@+id](mailto:@+id)/java" android:layout\_width="wrap\_content" android:layout\_height="wrap\_content" android:text="Java" android:textSize="26sp" android:onClick="onCheckboxClicked"/>

<CheckBox android:id=["@+id](mailto:@+id)/javascript" android:layout\_width="wrap\_content" android:layout\_height="wrap\_content" android:text="JavaScript" android:textSize="26sp" android:onClick="onCheckboxClicked"/>

</LinearLayout>

Атрибут **android:onClick**, як і в випадку з простими кнопками, дозволяє задати обробник натиснення на прапорець. Визначимо обробник натиснення в коді *MainActivity*:

public class MainActivity extends AppCompatActivity {

[@Override](mailto:@Override)

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

}

public void onCheckboxClicked(View view) {

CheckBox language=(CheckBox) view; // отримуємо прапорець

boolean checked=language.isChecked(); // перевіряємо, чи він відмічений

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

// Перевіряємо, який саме прапорець відмічено

switch(view.getId()) {

case R.id.java:

if (checked){

selection.setText("Java");

}

break;

case R.id.javascript:

if (checked)

selection.setText("JavaScript");

break;

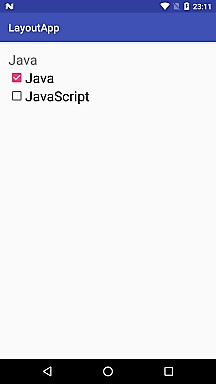
}

}

}

В якості параметра в обробник натискання *onCheckboxClicked* передається

натиснутий прапорець. За допомогою методу *isChecked*() можна дізнатися, чи виділений прапорець – в цьому випадку метод повертає *true*. За допомогою конструкції *switch ... case* можна отримати *id* натиснутого прапорця і виконати відповідні дії.



Якщо нам просто треба взяти текст з обраного прапорця, то не обов’язково в даному випадку використовувати конструкцію *switch*, бо ми можемо скоротити весь код наступним чином:

public void onCheckboxClicked(View view) { CheckBox language=(CheckBox) view;

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

if(language.isChecked())

selection.setText(language.getText());

}

Але в даному випадку у нас є проблема: в текстовому полі відображається тільки

один виділений елемент. Змінимо код *MainActivity*, щоб показати обидва виділених елементи:

public class MainActivity extends AppCompatActivity {

[@Override](mailto:@Override)

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

}

public void onCheckboxClicked(View view) {

// отримуємо прапорці

CheckBox java=(CheckBox) findViewById(R.id.java);

CheckBox javascript=(CheckBox) findViewById(R.id.javascript); String selectedItems="";

if(java.isChecked())

selectedItems+=java.getText() + ", ";

if(javascript.isChecked())

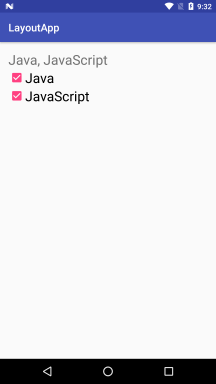
selectedItems+=javascript.getText();

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

selection.setText(selectedItems);

}

}



За допомогою слухача **OnCheckedChangeListener** можна відстежувати зміни прапорця. Цей слухач спрацьовує, коли ми встановлюємо або прибираємо позначку на прапорці. Наприклад, визначимо наступний checkbox:

<LinearLayout xmlns:andr[oid="http://schemas.android.com/apk/res/androi](http://schemas.android.com/apk/res/android)d" android:id=["@+id](mailto:@+id)/activity\_main" android:layout\_width="match\_parent" android:layout\_height="match\_parent" android:orientation="vertical"

android:padding="16dp">

<CheckBox android:id=["@+id](mailto:@+id)/enabled" android:layout\_width="wrap\_content" android:layout\_height="wrap\_content" android:checked="true" android:text="Ввімкнути" android:textSize="26sp" />

</LinearLayout>

Програмний код:

public class MainActivity extends AppCompatActivity {

[@Override](mailto:@Override)

protected void onCreate(Bundle savedInstanceState) { super.onCreate(savedInstanceState); setContentView(R.layout.activity\_main);

CheckBox enableBox = (CheckBox) findViewById(R.id.enabled);

enableBox.setOnCheckedChangeListener(new

CompoundButton.OnCheckedChangeListener() {

public void onCheckedChanged(CompoundButton buttonView, boolean isChecked) {

if (isChecked) {

Toast.makeText(getApplicationContext(), "Ввімкнено", Toast.LENGTH\_SHORT).show();

} else {

Toast.makeText(getApplicationContext(), "Вимкнено", Toast.LENGTH\_SHORT).show();

}

}

});

}

}

Слухач **OnCheckedChangeListener** визначено в базовому класі *CompoundButton*

і він визначає один метод – *onCheckedChanged*. Перший параметр цього методу *buttonView* – сам змінений прапорець *CheckBox*. А другий параметр *isChecked* вказує, чи відзначений прапорець. При зміні стану прапорця, буде виводитися відповідне повідомлення:



**6.1.5 RadioButton**

Подібну до прапорців функціональність надають перемикачі, які представлені класом *RadioButton*. Але на відміну від прапорців одноразово в групі перемикачів ми можемо вибрати тільки один перемикач. Щоб створити список перемикачів для вибору, спочатку треба створити об’єкт *RadioGroup* , який буде включати в себе всі перемикачі:

<LinearLayout xmlns:android=["http://schemas.android.com/apk/res/android"](http://schemas.android.com/apk/res/android) xmlns:t[ools="http://schemas.android.com/tools"](http://schemas.android.com/tools) android:layout\_width="match\_parent" android:layout\_height="match\_parent"

android:orientation="vertical" >

<TextView android:id=["@+id](mailto:@+id)/selection" android:layout\_width="match\_parent" android:layout\_height="wrap\_content" android:textSize="26sp"

/>

<RadioGroup xmlns:andr[oid="http://schemas.android.com/apk/res/android"](http://schemas.android.com/apk/res/android) android:id=["@+id](mailto:@+id)/radios"

android:layout\_width="match\_parent" android:layout\_height="wrap\_content" android:orientation="vertical" >

<RadioButton android:[id="@+id/java"](mailto:@+id)

android:layout\_width="wrap\_content" android:layout\_height="wrap\_content" android:text="Java" android:onClick="onRadioButtonClicked"/>

<RadioButton android:[id="@+id/javascript" android:layout\_width="wrap\_content" android:layout\_height="wrap\_content" android:text="JavaScript" android:onClick="onRadioButtonClicked"/>](mailto:@+id)

</RadioGroup>

</LinearLayout>

Оскільки клас *RadioGroup* є похідним від *LinearLayout*, то ми також можемо поставити вертикальну або горизонтальну орієнтацію списку, при тому включивши в нього не тільки власне перемикачі, а й інші об’єкти, наприклад, кнопку або *TextView*.

У класі *MainActivity* визначимо обробку вибору перемикачів:

public class MainActivity extends AppCompatActivity {

[@Override](mailto:@Override)

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

}

public void onRadioButtonClicked(View view) {

// якщо перемикач ввімкнено

boolean checked = ((RadioButton) view).isChecked();

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

// Отримуємо ввімкнений перемикач

switch(view.getId()) {

case R.id.java:

if (checked){

selection.setText("Выбран Java");

}

break;

case R.id.javascript:

if (checked){

selection.setText("Выбран JavaScript");

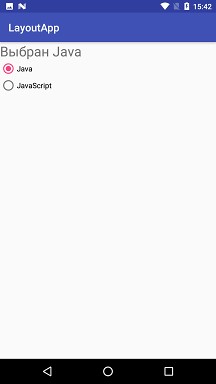
}

break;

}

}

}



Окрім обробки натискання на кожен окремий перемикач, ми можемо в цілому визначити для всього *RadioGroup* з його перемикачами слухач *OnCheckedChangeListener* і обробляти в ньому натискання. Для цього приберемо з розмітки у перемикачів атрибути **android:onClick**, а у елемента *RadioGroup* визначимо *id*:

<LinearLayout xmlns:android=["http://schemas.android.com/apk/res/android"](http://schemas.android.com/apk/res/android) xmlns:t[ools="http://schemas.android.com/tools"](http://schemas.android.com/tools) android:layout\_width="match\_parent" android:layout\_height="match\_parent"

android:orientation="vertical" >

<TextView android:id=["@+id](mailto:@+id)/selection" android:layout\_width="match\_parent" android:layout\_height="wrap\_content"

android:textSize="26sp"

/>

<RadioGroup xmlns:andr[oid="http://schemas.android.com/apk/res/android"](http://schemas.android.com/apk/res/android) android:id=["@+id](mailto:@+id)/radios"

android:layout\_width="match\_parent" android:layout\_height="wrap\_content" android:orientation="vertical">

<RadioButton android:[id="@+id/java" android:layout\_width="wrap\_content" android:layout\_height="wrap\_content" android:text="Java"/>](mailto:@+id)

<RadioButton android:[id="@+id/javascript" android:layout\_width="wrap\_content" android:layout\_height="wrap\_content" android:text="JavaScript"/>](mailto:@+id)

</RadioGroup>

</LinearLayout>

Далі в коді *MainActivity* призначимо об’єкту RadioGroup слухача:

public class MainActivity extends AppCompatActivity {

[@Override](mailto:@Override)

protected void onCreate(Bundle savedInstanceState) { super.onCreate(savedInstanceState); setContentView(R.layout.activity\_main);

RadioGroup radGrp=(RadioGroup)findViewById(R.id.radios);

// обробка зміни стану премикача

radGrp.setOnCheckedChangeListener(new RadioGroup.OnCheckedChangeListener() {

[@Override](mailto:@Override)

public void onCheckedChanged(RadioGroup arg0, int id) { TextView selection=(TextView) findViewById(R.id.selection); switch(id) {

case R.id.java: selection.setText("Выбран Java"); break;

case R.id.javascript:

}

}});

}

}

selection.setText("Выбран JavaScript");

break;

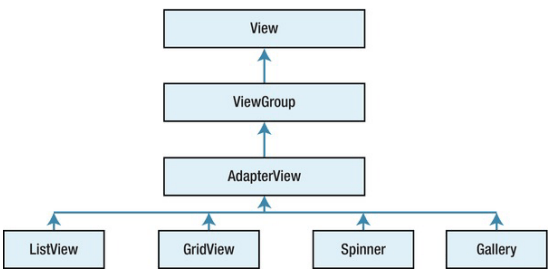
default:

break;

Слухач *RadioGroup.OnCheckedChangeListener* визначає метод *onCheckedChanged*(), в який передається об’єкт *RadioGroup* і *id* виділеного перемикача. Далі також ми можемо перевірити *id* і виконати певну обробку.

**6.2 Адаптери та списки**

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



При роботі зі списками ми маємо справу з трьома компонентами. По-перше, це самі елементи списків (*ListView*, *GridView*), які відображають дані. По-друге, це джерело даних – масив, об’єкт *ArrayList*, база даних і т. д., в якому знаходяться самі відображаються дані. І по-третє, це адаптери – спеціальні компоненти, які пов’язують джерело даних з елементом списку.

Розглянемо зв’язок елемента *ListView* з джерелом даних за допомогою одного з таких адаптерів – класу *ArrayAdapter*.

**6.2.1 ArrayAdapter**

Клас ArrayAdapter є простим адаптером, який пов’язує масив даних з набором елементів *TextView*, з яких, наприклад, може складатися *ListView*. Тобто в даному випадку джерелом даних виступає масив об’єктів. *ArrayAdapter* викликає у кожного об’єкта метод *toString*()і отриманий рядок встановлює в елемент *TextView*. Подивимося на прикладі. Отже, розмітка програми може виглядати так:

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

<RelativeLayout xmlns:andr[oid="http://schemas.android.com/apk/res/android"](http://schemas.android.com/apk/res/android) android:id=["@+id](mailto:@+id)/activity\_main"

android:layout\_width="match\_parent" android:layout\_height="match\_parent">

<ListView android:id=["@+id](mailto:@+id)/countriesList" android:layout\_width="match\_parent" android:layout\_height="match\_parent">

</ListView>

</RelativeLayout>

Тут також визначено елемент *ListView*, який буде виводити список об’єктів. Тепер перейдемо до коду activity і пов’яжемо *ListView* через *ArrayAdapter* з деякими даними:

public class MainActivity extends AppCompatActivity {

// набsр даних, які ми пов’яжемо із списком

String[] countries={"Бразилия", "Аргентина", "Колумбия", "Чили", "Уругвай"};

[@Override](mailto:@Override)

protected void onCreate(Bundle savedInstanceState) { super.onCreate(savedInstanceState); setContentView(R.layout.activity\_main);

// отримуємо ListView

ListView countriesList=(ListView) findViewById(R.id.countriesList);

// створюємо адаптер

ArrayAdapter<String> adapter=new ArrayAdapter(this,

android.R.layout.simple\_list\_item\_1, countries)

// задаємо адаптер для списку

countriesList.setAdapter(adapter);

}

}

Тут спочатку отримуємо по *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:

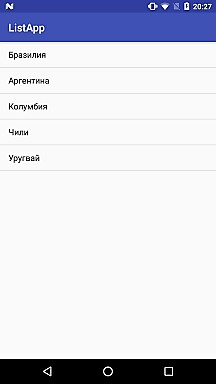
● *this* : поточний об’єкт *activity*;

● *android.R.layout.simple\_list\_item\_1* : файл розмітки списку, який фреймворк являє за замовчуванням. Він знаходиться в папці Android SDK *platforms/[android- номер\_версіі]/data/res/layout*. Якщо нас не задовольняє стандартна розмітка списку, ми можемо створити свою і потім в коді змінити *id* на *id* потрібної нам розмітки

● *countries* : масив даних. Тут необов’язково вказувати саме масив, це може бути список *ArrayList<T>*.

В кінці необхідно встановити для *ListView* адаптер за допомогою методу

*setAdapter()*. У підсумку ми отримаємо наступне відображення:



**6.2.2 Ресурс string-array і ListView**

У попередньому підпункті було розглянуто, як виводити масив рядків (стрічок) за допомогою *ArrayAdapter* в *ListView*. При цьому масив рядків визначався програмно в коді *java*. Однак подібний масив рядків набагато зручніше було б зберігати в файлі *xml* у вигляді ресурсу.

Ресурси типу масивів рядків представляють елемент типу *string*-array. Вони можуть знаходиться 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\_масивe\_стрічок">

<item>елемент</item>

</string-array>

</resources>

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

Наприклад, додамо в папку *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 на такий:

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

<resources>

<string-array name="countries">

<item>Бразилия</item>

<item>Аргентина</item>

<item>Колумбия</item>

<item>Чили</item>

<item>Уругвай</item>

</string-array>

</resources>

В файлі розмітки *activity\_main.xml* залишається визначення елемента *ListView*:

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

<RelativeLayout xmlns:andr[oid="http://schemas.android.com/apk/res/android"](http://schemas.android.com/apk/res/android) android:id=["@+id](mailto:@+id)/activity\_main"

android:layout\_width="match\_parent" android:layout\_height="match\_parent">

<ListView android:id=["@+id](mailto:@+id)/countriesList" android:layout\_width="match\_parent" android:layout\_height="match\_parent">

</ListView>

</RelativeLayout>

Тепер зв’яжемо ресурс і *ListView* в коді *MainActivity*:

public class MainActivity extends AppCompatActivity {

[@Override](mailto:@Override)

protected void onCreate(Bundle savedInstanceState) { super.onCreate(savedInstanceState); setContentView(R.layout.activity\_main);

// отримуємо ListView

ListView countriesList = (ListView) 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);

// задаємо адаптер для списку

countriesList.setAdapter(adapter);

}

}

Для отримання ресурсу в коді *java* застосовується вираз *R.array.назва\_ресурсу*.

Але нам необов’язково додавати список рядків в *ListView* програмно. У цього елемента є атрибут *entries*, який в якості значення може приймати ресурс *string-array*:

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

<RelativeLayout xmlns:android="[http://schemas.android.com/apk/res/android"](http://schemas.android.com/apk/res/android) android:id=["@+id](mailto:@+id)/activity\_main"

android:layout\_width="match\_parent" android:layout\_height="match\_parent">

<ListView android:entries=["@array](mailto:@array)/countries" android:id=["@+id](mailto:@+id)/countriesList" android:layout\_width="match\_parent" android:layout\_height="match\_parent">

</ListView>

</RelativeLayout>

В цьому випадку код *MainActivity* ми можемо скоротити до стандартного:

public class MainActivity extends AppCompatActivity {

[@Override](mailto:@Override)

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

}

}

А результат буде таким же самим:

