ОСНОВНІ ЕЛЕМЕНТИ УПРАВЛІННЯ

План:

- TextView
- EditText
- Button
- Додаток Калькулятор
- Вспливаючі вікна. Toast
- Snackbar
- Checkbox
- Програмне встановлення ширини і висоти
- OnCheckedChangeListener
- ToggleButton
- RadioButton
- OnCheckedChangeListener
- DatePicker і TimePicker
- Цифровий і аналоговий годинник
- Повзунок SeekBar

TextView

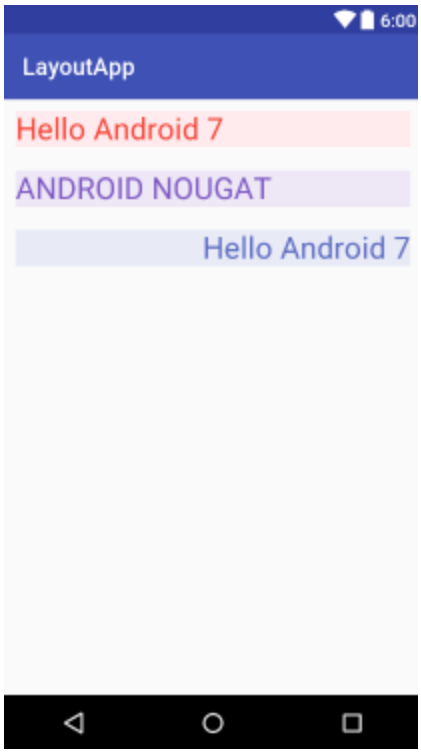
Для простого перегляду тексту на екрані призначений елемент **TextView** . Він просто відображає текст без можливості його редагування. Деякі його основні атрибути:

- **android: text** : встановлює текст елемента
- **android: textSize** : встановлює висоту тексту, в якості одиниць виміру для вказівки висоти використовуються **sp**
- **android: background** : задає фоновий колір елементу у вигляді кольору в шістнадцятковій формі запису 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:
  - center: Вирівнювання по центру
  - textStart: По лівому краю

  - textEnd: По правому краю
  - viewStart: По лівому краю
  - viewEnd: По правому краю
- **android: fontFamily** : встановлює тип шрифту. Може приймати наступні значення:
  - monospace
  - serif
  - serif-monospace
  - sans-serif
  - sans-serif-condensed
  - sans-serif-smallcaps
  - sans-serif-light
  - casual
  - cursive
  - cursive

Наприклад, визначимо три текстових поля:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+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>
```



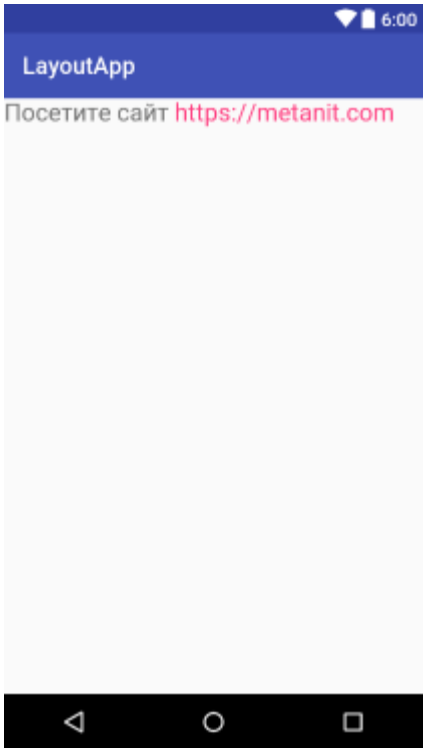
Установка елемента в коді теж не відрізняється складністю. Наприклад, створимо елемент і виведемо його на екран:

```
package com.example.ozm.helloworld;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.ViewGroup;
import android.widget.LinearLayout;
import android.widget.TextView;
import android.graphics.Typeface;

public class MainActivity extends AppCompatActivity {

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

        LinearLayout linearLayout = new LinearLayout(this);
        TextView textView1 = new TextView(this);
        // установка кольору фону
        textView1.setBackgroundColor(0xffe8eaf6);
        // установка кольору тексту
        textView1.setTextColor(0xff5c6bc0);
        // робимо всі букви великими
        textView1.setAllCaps(true);
        // встановлюємо вирівнювання тексту по центру
        textView1.setTextAlignment(TextView.TEXT_ALIGNMENT_CENTER);
        // встановлюєм текст
        textView1.setText("Android Nougat 7");
        // встановлюємо шрифт
        textView1.setTypeface(Typeface.create("casual", Typeface.NORMAL));
        // встановлюємо висоту тексту
        textView1.setTextSize(26);

        LinearLayout.LayoutParams layoutParams = new
LinearLayout.LayoutParams
```

```
                    (ViewGroup.LayoutParams.MATCH_PARENT,
ViewGroup.LayoutParams.WRAP_CONTENT);
                // встановлення зовнішніх відступів
                layoutParams.setMargins(20,20,20,20);
                // встановлюємо розміри
                textView1.setLayoutParams(layoutParams);
                linearLayout.addView(textView1);
                setContentView(linearLayout);
            }
        }
```

Іноді необхідно вивести на екран якесь посилання, або телефон, після натискання на які проводилося б певним чином впливати. Для цього в TextView визначено атрибут **android: autoLink** :

```
<TextView android:id="@+id/display_message"
    android:text="Відвідайте сайт https://metanit.com"
    android:textSize="21sp"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:autoLink="web|email" />
```



android: autoLink може приймати кілька значень:

- none: Відключає всі посилання
- web: Включає всі веб-посилання
- email: Включає посилання на електронні адреси
- phone: Включає посилання на номери телефонів
- map: Включає посилання на карту
- all: Включає всі перераховані вище посилання

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

EditText

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

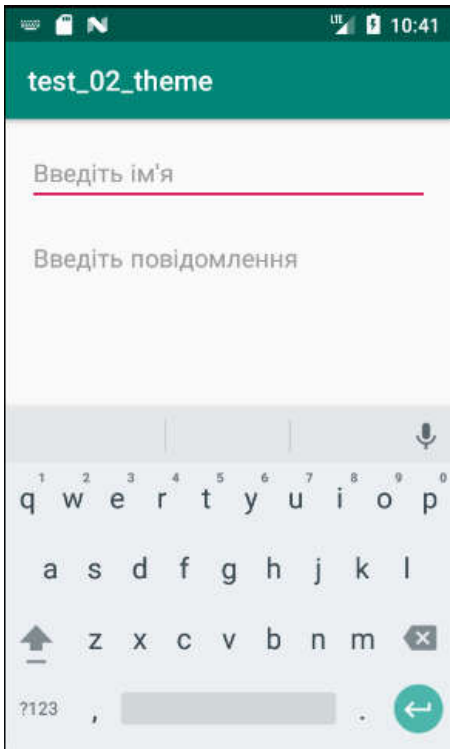
З тих атрибутів, що не розглядалися в темі про TextView, слід зазначити атрибут **android: hint** . Він дозволяє задати текст, який буде відображатися в якості підказки, якщо елемент EditText порожній. Крім того, ми можемо використовувати атрибут **android: inputType** , який дозволяє задати клавіатуру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

- text: Звичайна клавіатура для введення однострочного тексту
- textMultiLine: Многострочное 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 - рядкові
- 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:

```
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+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="Введіть ім'я" />
    <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 - многострочное. Щоб у другому полі текст вирівнювався по верху, додатково встановлюється атрибут `android:gravity="top"`.



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:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/activity_main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="16dp">

    <TextView
        android:id="@+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/editText" />

</LinearLayout>
```

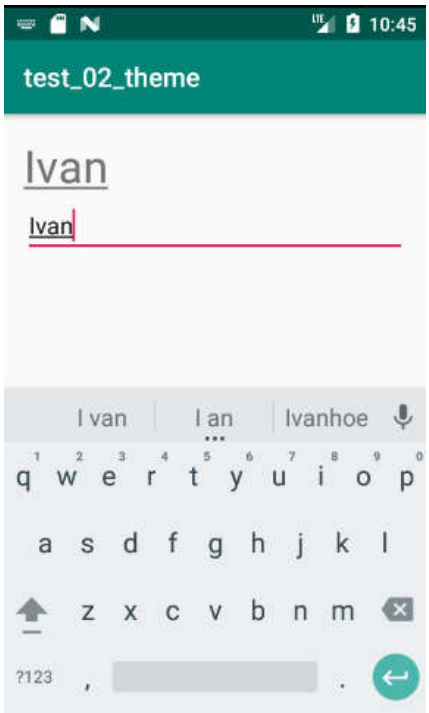
Передбачається, що введені в EditText символи тут же будуть відображатися в елементі TextView. І для цього також змінимо код MainActivity:

```
package com.example.i.test_02_theme;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.text.Editable;
import android.text.TextWatcher;
import android.widget.EditText;
import android.widget.TextView;
```

```
public class MainActivity extends AppCompatActivity {

    @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 реалності 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:



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 натискання кнопки

Отже, змінимо код в **activity_main.xml** наступним чином:

```
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/activity_main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="16dp">

    <TextView
        android:id="@+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/editText" />
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Click"
        android:onClick="sendMessage" />

</LinearLayout>
```

При допомогою атрибута android:onClickможна 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:

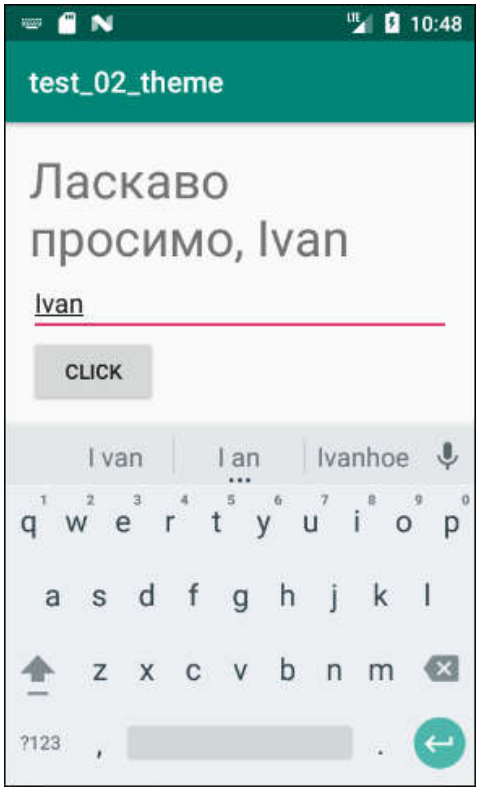
```
package com.example.i.test_02_theme;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.EditText;
import android.widget.TextView;

public class MainActivity extends AppCompatActivity {

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

```
        setContentView(R.layout.activity_main);
    }
    // Обробка натискання кнопки
    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 обробки натискання слід враховувати наступні моменти:

- Метод повинен оголошуватися з модифікатором public
- Повинен повертати значення void
- Як параметр приймати об'єкт View. Цей об'єкт View і являє собою натиснуту кнопку

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



Аналогічний приклад повністю в коді MainActivity:

```
package com.example.i.test_02_theme;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.LinearLayout;
import android.widget.TextView;

public class MainActivity extends AppCompatActivity {

    EditText editText;
    TextView textView;
    @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) {
                    // Обробка натискання
                    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) {
            // Обрабка натискання
            }
            });
```

Додаток Калькулятор

Знаючи деякі основи компонування і такі елементи як TextView, EditText і Button, вже можна скласти більш менш повноцінне додаток. В даному випадку ми зробимо простенький калькулятор.

Для цього створимо новий проект і визначимо в файлі **activity_main.xml** наступний інтерфейс:

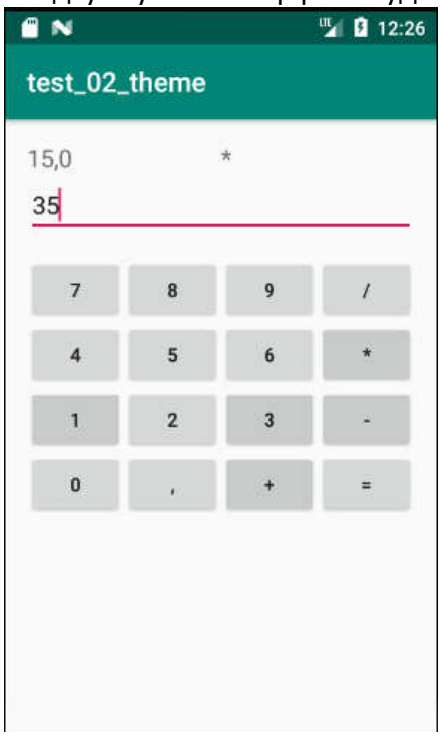
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/activity_main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
```

```
android:orientation="vertical"
android:padding="16dp">
<LinearLayout
    android:id="@+id/result"
    android:layout_width="match_parent"
    android:layout_height="wrap_content">

    <TextView
        android:id="@+id/resultField"
        android:layout_width="0dp"
        android:layout_weight="1"
        android:textSize="18sp"
        android:layout_height="wrap_content" />
    <TextView
        android:id="@+id/operationField"
        android:layout_width="0dp"
        android:layout_weight="1"
        android:textSize="18sp"
        android:layout_height="wrap_content"
        />
</LinearLayout>
<EditText
    android:id="@+id/numberField"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:inputType="phone" />
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="16dp">
    <Button
        android:layout_weight="1"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:text="7"
        android:onClick="onNumberClick"/>
    <Button
        android:layout_weight="1"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:text="8"
        android:onClick="onNumberClick"/>
    <Button
        android:layout_weight="1"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:text="9"
        android:onClick="onNumberClick"/>
    <Button
        android:layout_weight="1"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:text="/"
        android:onClick="onOperationClick"/>
</LinearLayout>
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content">
    <Button
        android:layout_weight="1"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:text="4"
        android:onClick="onNumberClick"/>
```

```
        <Button
            android:layout_weight="1"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:text="5"
            android:onClick="onNumberClick"/>
        <Button
            android:layout_weight="1"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:text="6"
            android:onClick="onNumberClick"/>
        <Button
            android:layout_weight="1"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:text="*"
            android:onClick="onOperationClick"/>
    </LinearLayout>
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content">
        <Button
            android:layout_weight="1"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:text="1"
            android:onClick="onNumberClick"/>
        <Button
            android:layout_weight="1"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:text="2"
            android:onClick="onNumberClick"/>
        <Button
            android:layout_weight="1"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:text="3"
            android:onClick="onNumberClick"/>
        <Button
            android:layout_weight="1"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:text="-"
            android:onClick="onOperationClick"/>
    </LinearLayout>
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content">
        <Button
            android:layout_weight="1"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:text="0"
            android:onClick="onNumberClick"/>
        <Button
            android:layout_weight="1"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:text="."
            android:onClick="onNumberClick"/>
        <Button
            android:layout_weight="1"
```

```
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:text="+"
            android:onClick="onOperationClick"/>
        <Button
            android:layout_weight="1"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:text="="
            android:onClick="onOperationClick"/>
    </LinearLayout>
</LinearLayout>
```

У підсумку весь інтерфейс буде виглядати наступним чином:



Кореневої контейнер компонування представляє елемент LinearLayout з вертикальною орієнтацією. Перший елементв ньому - горизонтальний елемент LinearLayout, в якому визначені два текстових поля TextView: одне для виведення результату обчислень і одне для виведення поточного знака операції.

Потім йде елемент EditText, призначений для введення чисел.

І далі розташовані чотири елементи LinearLayout з горизонтальними рядами кнопок. Щоб все кнопки займали рівне простір всередині контейнера, для них встановлені атрибути android:layout_weight="1"і android:layout_width="0dp".

Крім того, для числових кнопок в якості обробника натискання встановлено метод onNumberClick, а для кнопок зі знаками операцій атрибут onClick вказує на метод onOperationClick.

Тепер змінимо клас **MainActivity** :

```
package com.example.i.test_02_theme;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
```

```
public class MainActivity extends AppCompatActivity {

    TextView resultField; // текстовое поле для виводу результату
    EditText numberField;   // поле для вводу числа
    TextView operationField;    // текстове поле для виводу знаку операції
    Double operand = null;  // операнд операції
    String lastOperation = "="; // остання операція

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        // отримуємо всі поля по id з activity_main.xml
        resultField =(TextView) findViewById(R.id.resultField);
        numberField = (EditText) findViewById(R.id.numberField);
        operationField = (TextView) findViewById(R.id.operationField);
    }
    // збереження стану
    @Override
    protected void onSaveInstanceState(Bundle outState) {
        outState.putString("OPERATION", lastOperation);
        if(operand!=null)
            outState.putDouble("OPERAND", operand);
        super.onSaveInstanceState(outState);
    }
    // отримання раніше збереженого стану
    @Override
    protected void onRestoreInstanceState(Bundle savedInstanceState) {
        super.onRestoreInstanceState(savedInstanceState);
        lastOperation = savedInstanceState.getString("OPERATION");
        operand= savedInstanceState.getDouble("OPERAND");
        resultField.setText(operand.toString());
        operationField.setText(lastOperation);
    }
    // обробка натискання на числову кнопку
    public void onNumberClick(View view){

        Button button = (Button)view;
        numberField.append(button.getText());

        if(lastOperation.equals("=") && operand!=null){
            operand = null;
        }
    }
    // обробка натискання на кнопку операції
    public void onOperationClick(View view){

        Button button = (Button)view;
        String op = button.getText().toString();
        String number = numberField.getText().toString();
        // якщо введено що небудь
        if(number.length()>0){
            number = number.replace(',', '.');
            try{
                performOperation(Double.valueOf(number), op);
            }catch (NumberFormatException ex){
                numberField.setText("");
            }
        }
        lastOperation = op;
        operationField.setText(lastOperation);
    }
```

```
        private void performOperation(Double number, String operation){
    // якщо операнд раніше не був встановлений (при вводі самої першої операції)
            if(operand ==null){
                operand = number;
            }
            else{
                if(lastOperation.equals("=")){
                    lastOperation = operation;
                }
                switch(lastOperation){
                    case "=":
                        operand =number;
                        break;
                    case "/":
                        if(number==0){
                            operand =0.0;
                        }
                        else{
                            operand /=number;
                        }
                        break;
                    case "*":
                        operand *=number;
                        break;
                    case "+":
                        operand +=number;
                        break;
                    case "-":
                        operand -=number;
                        break;
                }
            }
            resultField.setText(operand.toString().replace('.', ','));
            numberField.setText("");
        }
    }
```

Розберемо цей код. Спочатку в методі onCreate()отримуємо все поля з activity_main.xml, текст яких буде змінюватися:

```
resultField =(TextView) findViewById(R.id.resultField);
numberField = (EditText) findViewById(R.id.numberField);
operationField = (TextView) findViewById(R.id.operationField);
```

Результат операції буде потрапляти в змінну operand, яка представляє тип Double, а знак операції - в змінну lastOperation:

```
Double operand = null;
String lastOperation = "=";
```

Так як при переході від портретної орієнтації до альбомної або навпаки ми можемо втратити все введені дані, то щоб їх не втратити, ми їх зберігаємо в методі onSaveInstanceState()і назад отримуємо в методі onRestoreInstanceState().

При натисканні на цифрові клавіші буде викликатися метод onNumberClick, в якому додаємо введену цифру або знак коми до тексту в поле numberField:

```
Button button = (Button)view;
        numberField.append(button.getText());
```

```
        if(lastOperation.equals("=") && operand!=null){
            operand = null;
        }
```

При цьому якщо остання операція являла собою отримання результату (знак "дорівнює"), то ми скидаємо змінну operand.

У методі onOperationClickвідбувається обробка натискання на кнопку зі знаком операції:

```
Button button = (Button)view;
String op = button.getText().toString();
String number = numberField.getText().toString();
// якщо введено що небудь
if(number.length()>0){
    number = number.replace(',', '.');
    try{
        performOperation(Double.valueOf(number), op);
    }catch (NumberFormatException ex){
        numberField.setText("");
    }
}
lastOperation = op;
operationField.setText(lastOperation);
```

Тут отримуємо раніше введене число і введену операцію і передаємо їх в метод performOperation(). Так як в метод передається не просто рядок, а число Double, то нам треба перетворити рядок в чсло. І оскільки теоретично можуть бути введені нечислові символи, то для вилову виключення, яке може виникнути при перетворенні використовується конструкція try ... catch.

Крім того, так як роздільником цілої і дробової частини в Double в java є точка, то нам треба замінити кому на крапку, так як передбачається, що ми використовуємо як роздільник кому.

А методі performOperation()виконуємо власне операцію. При введенні першої операції, коли операнд ще не встановлений, ми просто встановлюємо операнд:

```
if(operand ==null){
    operand = number;
}
```

При введенні другої і наступних операцій застосовуємо попередню операцію, знак якої зберігається в змінної lastOperation, до операнду operand і другого числа, яке було введено в числове поле. Отриманий результат операції зберігаємо в змінної operand.

Спливаючі вікна. Toast

Для створення простих повідомлень в Android використовується клас **Toast** . Фактично Toast представляє спливаюче вікно з деяким текстом, яке відображається протягом деякого часу.

Об'єкт Toast не можна створити в коді розмітки xml, наприклад, в файл activity_main.xml. Toast можна використовувати тільки в коді java.

Так, визначимо в файлі розмітки activity_main.xml кнопку:

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
```

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

        <Button
            android:layout_width = "wrap_content"
            android:layout_height = "wrap_content"
            android:text = "Click"
            android:onClick = "onClick" />

    </android.support.constraint.ConstraintLayout>
```

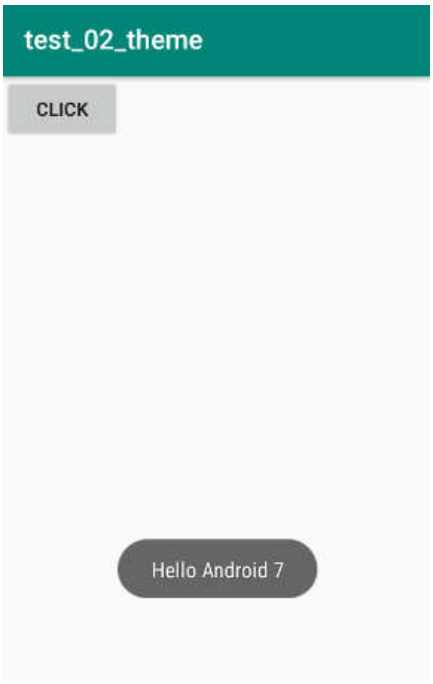
У кнопки встановлено обробник натискання - метод onClick. Визначимо його в коді MainActivity:

```
    package com.example.i.test_02_theme;

    import android.support.v7.app.AppCompatActivity;
    import android.os.Bundle;
    import android.view.View;
    import android.widget.Toast;

    public class MainActivity extends AppCompatActivity {

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

            setContentView (R.layout.activity_main);
        }

        public void onClick (View view) {
            Toast toast = Toast.makeText (this, "Hello Android 7",
Toast.LENGTH_LONG);
            toast.show ();
        }
    }
```

У обробнику відображається спливаюче вікно. Для його створення застосовується метод **Toast.makeText()**, в який передається три параметра: поточний контекст (поточний об'єкт activity), що відображається текст і час отобаження вікна.

Як часу показу вікна ми можемо використовувати цілочисельне значення - коліческо мілісекунд або вбудовані константи **Toast.LENGTH_LONG** (3500 мілісекунд) і **Toast.LENGTH_SHORT** (2000 мілісекунд).

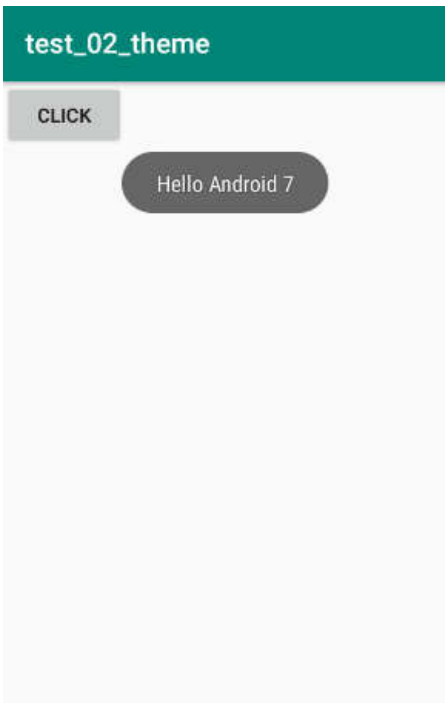
Для самого відображення вікна викликається метод **show()**.



За замовчуванням вікно відображається внизу інтерфейсу з центруванням по центру. Але ми можемо кастомізувати позиціонування вікна за допомогою методів **setGravity()** і **setMargin()** . Так, змінимо метод onClick:

```
    public void onClick (View view) {
        Toast toast = Toast.makeText (this, "Hello Android 7",
Toast.LENGTH_LONG);
        toast.setGravity(Gravity.TOP, 0, 160);
        toast.show ();
    }
```

Перший параметр методу setGravity вказує, в якій частині контейнера треба позиціонувати Toast, другий і третій параметр встановлюють відступи від цієї позиції по горизонталі і вертикалі відповідно:

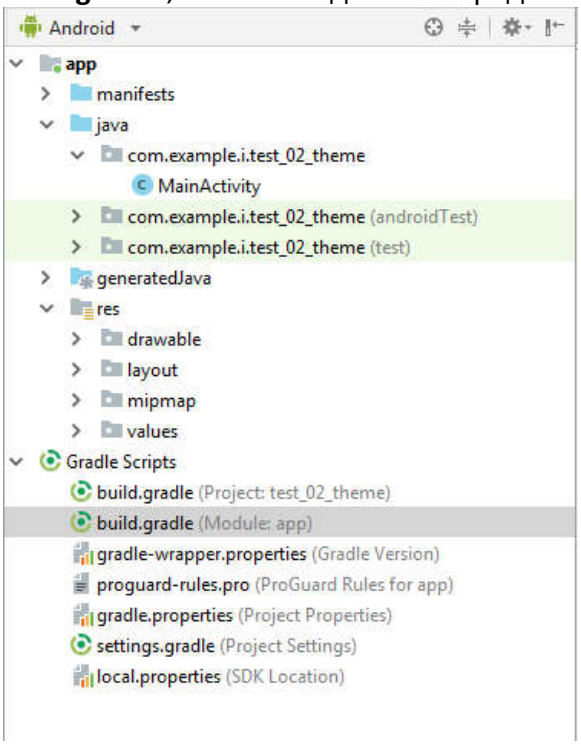


Метод **setMargin()** приймає два параметри: відступ від лівої межі контейнера в процентах від ширини контейнера і відступ від верхньої межі у відсотках від довжини контейнера.

Snackbar

Елемент **Snackbar** в деякому роді схожий на Toast: він також дозволяє виводити спливаючі повідомлення, але тепер повідомлення розтягуються по ширині екрану.

При створенні проекту по типу Empty Activity за замовчуванням функціональність Snackbar може бути недоступна, нам її треба буде додати. Для цього перейдемо до файлу **build.gradle** , який знаходиться всередині модуля app:



Він виглядає приблизно так:

```
apply plugin: 'com.android.application'

android {
    compileSdkVersion 27
    defaultConfig {
        applicationId "com.example.i.test_02_theme"
        minSdkVersion 24
        targetSdkVersion 27
        versionCode 1
        versionName "1.0"
        testInstrumentationRunner
"android.support.test.runner.AndroidJUnitRunner"
    }
    buildTypes {
        release {
            minifyEnabled false
            proguardFiles  getDefaultProguardFile('proguard-android.txt'),
'proguard-rules.pro'
        }
    }
}

dependencies {
    implementation fileTree(dir: 'libs', include: ['*.jar'])
```

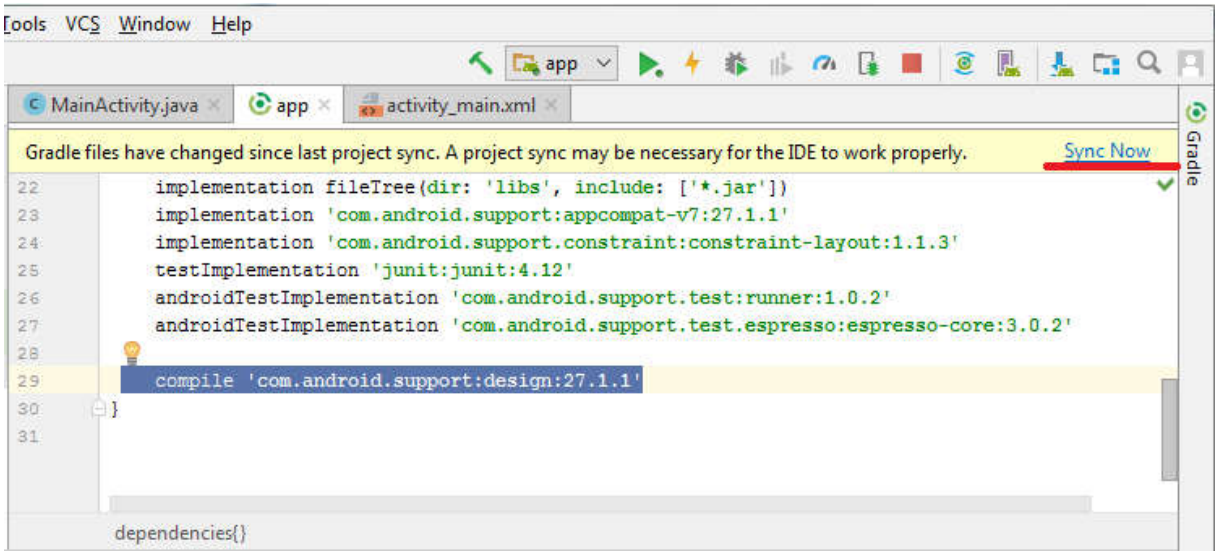
```
        implementation 'com.android.support:appcompat-v7:27.1.1'
        implementation              'com.android.support.constraint:constraint-
layout:1.1.3'
        testImplementation 'junit:junit:4.12'
        androidTestImplementation 'com.android.support.test:runner:1.0.2'
        androidTestImplementation  'com.android.support.test.espresso:espresso-
core:3.0.2'
    }
```

В кінець секції **dependencies** додамо потрібний нам пакет:

```
    dependencies {
        implementation fileTree(dir: 'libs', include: ['*.jar'])
        implementation 'com.android.support:appcompat-v7:27.1.1'
        implementation              'com.android.support.constraint:constraint-
layout:1.1.3'
        testImplementation 'junit:junit:4.12'
        androidTestImplementation 'com.android.support.test:runner:1.0.2'
        androidTestImplementation  'com.android.support.test.espresso:espresso-
core:3.0.2'

        compile 'com.android.support:design:27.1.1'
    }
```

Слід зазначити, що для компіляції в даному випадку застосовується API 27 (compileSdkVersion 27), тому і у пакета версія починається з цього числа: 25.1.1.

Після зміни файлу вгорі в Android Studio з'явиться повідомлення про необхідність виконати синхронізацію, і для цього натиснемо на посилання **Sync Now**:



Після завершення синхронізації змінимо файл **activity_main.xml**:

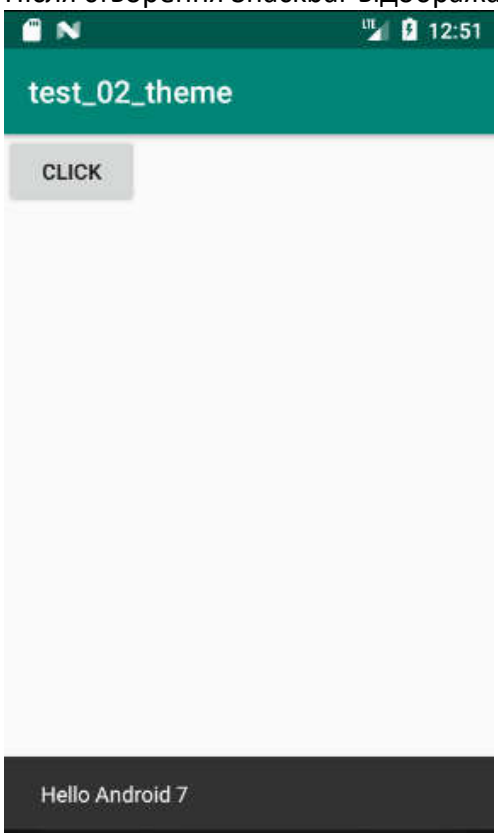
```
    <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
        android:layout_width="match_parent"
        android:layout_height="match_parent">

        <Button
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="CLICK"
            android:onClick="onClick"/>
    </RelativeLayout>
```

Тут визначена кнопка, після натискання на яку буде оображаться повідомлення.
І також змінимо клас **MainActivity**:

```
package com.example.i.test_02_theme;

import android.os.Bundle;
import android.support.design.widget.Snackbar;
import android.support.v7.app.AppCompatActivity;
import android.view.View;

public class MainActivity extends AppCompatActivity {

    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
    public void onClick(View view){
        Snackbar.make(view, "Hello Android 7", Snackbar.LENGTH_LONG)
                .show();
    }
}
```

Snackbar створюється за допомогою методу **make ()**, в який передаються три параметра: об'єкт View, до якого прикріплюється виспливающее оощеніе, саме повідомлення в вигляді рядка і параметр, який вказує, скільки буде відображатися повідомлення. Останній параметр може приймати числове значення - кількість мілісекунд, або одну з трьох констант: **Snackbar.LENGTH_INDEFINITE** (відображення протягом невизначеного періоду часу), **Snackbar.LENGTH_LONG** (довгий відображення) або **Snackbar.LENGTH_SHORT** (недовгий відображення).

Після створення Snackbar відображається за допомогою методу **show**:



При цьому на відміну від Toast ми не можемо вплинути на позицію повідомлення, воно відображається в нижній частині екрана і займає всю нижню частину.

Checkbox

Елементи Checkbox є прапорці, які можуть перебувати в зазначеному і невідзначеними стані. Прапорці дозволяють виробляти множинний вибір з кількох значень. Отже, визначимо в файлі розмітки *activity_main.xml*кілька прапорців:

```
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/activity_main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="16dp">

    <TextView android:id="@+id/selection"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:textSize="26sp" />
    <CheckBox android:id="@+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/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 натискання на прапорець. Визначимо обробник натискання в коді MainActivity:

```
package com.example.i.test_02_theme;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.CheckBox;
import android.widget.TextView;

public class MainActivity extends AppCompatActivity {

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

    public void onCheckboxClicked(View view) {
        // Отримуємо прапорець
        CheckBox language = (CheckBox) view;
        // Отримуємо, чи відмічений даний прапорець
        boolean checked = language.isChecked();

        TextView selection = (TextView) findViewById(R.id.selection);
        // Дивимось який саме з прапорців відмічений
```

```
            switch(view.getId()) {
                case R.id.java:
                    if (checked){
                        selection.setText("Java");
                    }
                    break;
                case R.id.javascript:
                    if (checked)
                        selection.setText("JavaScript");
                    break;
            }
        }
    }
```

Як параметр 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.

Правда, якщо нам просто треба взяти текст з обраного прапорця, то необов'язково в даному випадку використовувати конструкцію switch, так як ми можемо скоротити весь код наступним чином:

```
    public void onCheckboxClicked(View view) {
        // Отримуємо прапорець
        CheckBox language = (CheckBox) view;
        // Отримуємо, чи відмічений даний прапорець
        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 відображати обидва виділених елемента:

Програмне встановлення ширини і висоти

Якщо елемент, наприклад, той же TextView створюється в коді java, то для установки висоти і ширини можна використовувати метод **setLayoutParams ()**. Так, змінимо код MainActivity:

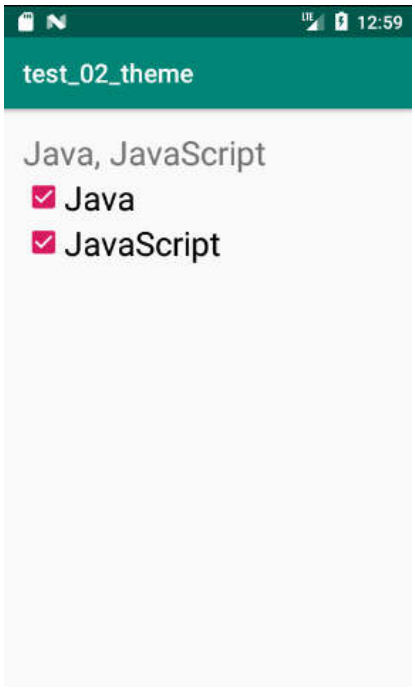
```
package com.example.ozm.helloworld;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.ViewGroup;
import android.widget.RelativeLayout;
import android.widget.TextView;

public class MainActivity extends AppCompatActivity {

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

        RelativeLayout relativeLayout = new RelativeLayout(this);
        TextView textView1 = new TextView(this);
        textView1.setText("Hello Android 7");
        textView1.setTextSize(26);

        // встановлюємо розміри
        textView1.setLayoutParams(new ViewGroup.LayoutParams
                (ViewGroup.LayoutParams.WRAP_CONTENT,
ViewGroup.LayoutParams.WRAP_CONTENT));
        // додаємо TextView в RelativeLayout
        relativeLayout.addView(textView1);
        setContentView(relativeLayout);
    }
}
```

У метод setLayoutParams()передається об'єкт **ViewGroup.LayoutParams**. Цей об'єкт ініціалізується двома параметрами: шириною і висотою. Для вказівки ширини і висоти можна використовувати одну з констант **ViewGroup.LayoutParams.WRAP_CONTENT** або **ViewGroup.LayoutParams.MATCH_PARENT**.

Також ми можемо передати точні значення або комбінувати типи значень:

```
package com.example.i.test_02_theme;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.CheckBox;
import android.widget.TextView;

public class MainActivity extends AppCompatActivity {

    @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);
    }
}
```



OnCheckedChangeListener

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

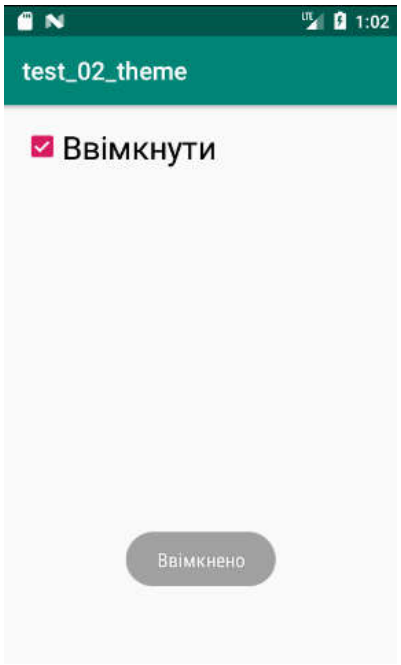
```
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/activity_main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="16dp">

    <CheckBox android:id="@+id/enabled"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:checked="true"
        android:text="Ввімкнути"
        android:textSize="26sp" />

</LinearLayout>
```

У коді MainActivity підключимо обробник зміни стану:

```
package com.example.i.test_02_theme;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.widget.CheckBox;
import android.widget.CompoundButton;
import android.widget.Toast;

public class MainActivity extends AppCompatActivity {

    @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 визначає один метод - **onCheckedChanged** . Перший параметр цього методу buttonView- сам змінений прапорець CheckBox. А другий параметр isCheckedвказує, відзначений прапорець.

При зміні стану прапорця буде виводитися у спливаючому вікні відповідне повідомлення:



ToggleButton

ToggleButton подібно елементу CheckBox може перебувати в двох станах: зазначеному і невідзначеними, причому для кожного стану ми можемо окремо встановити свій текст. Наприклад, визначимо наступний елемент ToggleButton:

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >

    <ToggleButton
        android:id="@+id/toggle"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textOn="Ввімкнено"
        android:textOff="Вимкнено"
        android:onClick="onToggleClicked"/>

</LinearLayout>
```

Атрибути android:textOni android:textOff задають текст кнопки в зазначеному і невідзначеними стані відповідно. І також, як і для інших кнопок, ми можемо обробити натискання на елемент за допомогою події onClick. В цьому випадку визначимо в класі Activity обробник події:

```
package com.example.i.test_02_theme;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Toast;
import android.widget.ToggleButton;

public class MainActivity extends AppCompatActivity {

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

        setContentView(R.layout.activity_main);
    }

    public void onToggleClicked(View view) {

        // чи ввімкнена кнопка
        boolean on = ((ToggleButton) view).isChecked();

        if (on) {
            // дії якщо кнопка ввімкнена
            Toast.makeText(this, "Світло ввімкнено", Toast.LENGTH_LONG).show();
        } else {
            // дії, якщо вимкнена
            Toast.makeText(this, "Світло вимкнено!", Toast.LENGTH_LONG).show();
        }
    }
}
```

Створити елемент ToggleButton в коді java:

```
package com.example.i.test_02_theme;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.LinearLayout;
import android.widget.Toast;
import android.widget.ToggleButton;

public class MainActivity extends AppCompatActivity {

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

        LinearLayout linearLayout = new LinearLayout(this);
        ToggleButton toggleButton = new ToggleButton(this);
        toggleButton.setTextOff("Вимкнено");
        toggleButton.setTextOn("Ввімкнено");
        toggleButton.setText("Вимкнено");
        toggleButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                boolean on = ((ToggleButton) view).isChecked();

                if (on) {
                    Toast.makeText(getApplicationContext(),        "Світло
ввімкнено", Toast.LENGTH_LONG).show();
                } else {
                    Toast.makeText(getApplicationContext(),        "Світло
вимкнено!", Toast.LENGTH_LONG).show();
                }
            }
        });
        linearLayout.addView(toggleButton, new LinearLayout.LayoutParams
                (LinearLayout.LayoutParams.WRAP_CONTENT,
LinearLayout.LayoutParams.WRAP_CONTENT));
```

```
        setContentView(linearLayout);
    }
}
```

RadioButton

Схожу з прапорцями 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"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >

    <TextView android:id="@+id/selection"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:textSize="26sp"
        />
    <RadioGroup xmlns:android="http://schemas.android.com/apk/res/android"
        android:id="@+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"
            android:onClick="onRadioButtonClicked"/>
        <RadioButton android:id="@+id/javascript"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="JavaScript"
            android:onClick="onRadioButtonClicked"/>
    </RadioGroup>

</LinearLayout>
```

Оскільки клас RadioGroupє 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:

```
package com.example.i.test_02_theme;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.RadioButton;
import android.widget.TextView;

public class MainActivity extends AppCompatActivity {

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

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

    public void onRadioButtonClicked(View view) {
        // якщо перемикач відмічений
        boolean checked = ((RadioButton) view).isChecked();
        TextView selection = (TextView) findViewById(R.id.selection);
        // Отримуємо натиснутий перемикач
        switch(view.getId()) {
            case R.id.java:
                if (checked){
                    selection.setText("Вибрано Java");
                }
                break;
            case R.id.javascript:
                if (checked){
                    selection.setText("Вибрано JavaScript");
                }
                break;
        }
    }
}
```

OnCheckedChangeListener

Крім обробки натискання на кожен окремий переключатель ми можемо в цілому повісити на весь 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"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >

    <TextView android:id="@+id/selection"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
```

```
            android:textSize="26sp"
            />
        <RadioGroup xmlns:android="http://schemas.android.com/apk/res/android"
            android:id="@+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"/>
            <RadioButton android:id="@+id/javascript"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="JavaScript"/>
        </RadioGroup>

    </LinearLayout>
```

Далі в коді MainActivity повісимо на об'єкт RadioGroup слухач **OnCheckedChangeListener**:

```
    package com.example.i.test_02_theme;

    import android.support.v7.app.AppCompatActivity;
    import android.os.Bundle;
    import android.view.View;
    import android.widget.RadioButton;
    import android.widget.RadioGroup;
    import android.widget.TextView;

    public class MainActivity extends AppCompatActivity {

        @Override
        protected void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            setContentView(R.layout.activity_main);
            // отримуємо об'єкт RadioGroup
            RadioGroup radGrp = (RadioGroup)findViewById(R.id.radios);
            // обробка перемикання стану перемикача
            radGrp.setOnCheckedChangeListener(new
RadioGroup.OnCheckedChangeListener() {
                @Override
                public void onCheckedChanged(RadioGroup arg0, int id) {
                    TextView          selection          =          (TextView)
findViewById(R.id.selection);
                    switch(id) {
                        case R.id.java:
                            selection.setText("Вибрано Java");
                            break;
                        case R.id.javascript:
                            selection.setText("Вибрано 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.

DatePicker і TimePicker

DatePicker і TimePicker є елементи для вибору дати і часу відповідно. Отже, створимо розмітку для цих елементів:

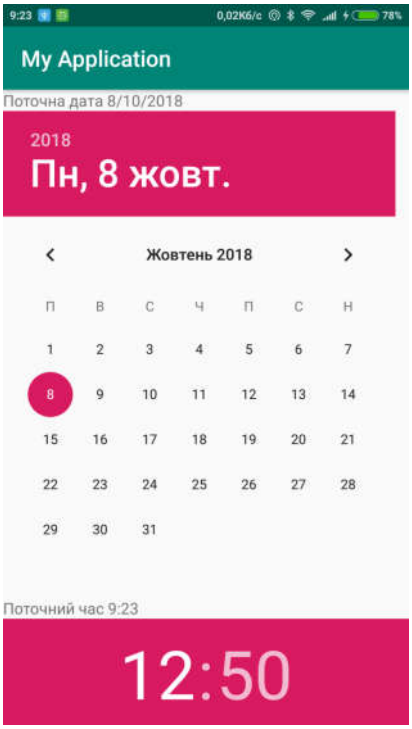
```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >

<TextView android:id = "@+id/dateDefault"
        android:layout_width = "match_parent"
        android:layout_height = "wrap_content" />

    <DatePicker android:id = "@+id/datePicker"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content" />

    <TextView android:id = "@+id/timeDefault"
        android:layout_width = "match_parent"
        android:layout_height = "wrap_content" />

    <TimePicker android:id = "@+id/timePicker"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content" />

</LinearLayout>
```

Тепер додамо в метод onCreateпоточної activity код, який встановить значення за замовчуванням для цих елементів:

```
package com.example.i.test_02_theme;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.widget.TextView;
import android.widget.DatePicker;
import android.widget.TimePicker;

public class MainActivity extends AppCompatActivity {

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

        TextView dateDefault = (TextView) findViewById (R.id.dateDefault);
        TextView timeDefault = (TextView) findViewById (R.id.timeDefault);

        DatePicker dp = (DatePicker) this.findViewById(R.id.datePicker);
        // Місяць починаючи з нуля. Для відображення додаємо 1.
        dateDefault.setText("Дата по замовчуванню " + dp.getDayOfMonth() +
"/" +
                (dp.getMonth() + 1) + "/" + dp.getYear());

        dp.init(2015, 02, 01, null);
```

```
        TimePicker tp = (TimePicker)this.findViewById(R.id.timePicker);
        java.util.Formatter timeF = new java.util.Formatter();
        timeF.format("Час по замовчуванню %d:%02d", tp.getHour(),
                tp.getMinute());
        timeDefault.setText(timeF.toString());
        tp.setIs24HourView(true);
        tp.setHour(new Integer(10));
        tp.setMinute(new Integer(10));
    }

}
```

Використовуючи метод dp.init(2015, 02, 01, null);встановлюємо дату за замовчуванням - 1 березня, так як відлік місяців йде з нуля. А перш ніж можна буде часу, вдаємося до форматування.



Цифровий і аналоговий годинник

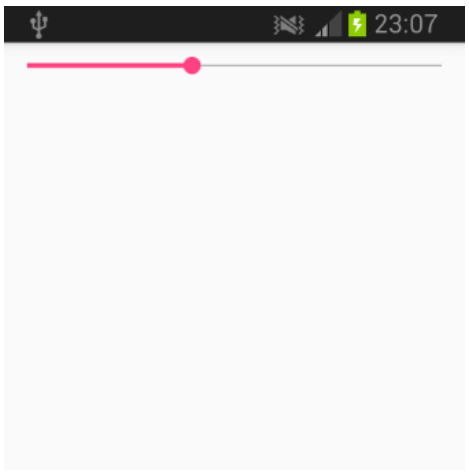
Крім елемента TimePicker Android підтримує для відображення часу такі елементи як аналоговий і цифровий годинник (елементи AnalogClock і DigitalClock):

```
<? Xml version = "1.0" encoding = "utf-8"?>
<LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:orientation = "vertical"
    android:layout_width = "match_parent"
    android:layout_height = "match_parent">
    <DigitalClock
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content" />
    <AnalogClock
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content" />

</ LinearLayout>
```

Повзунок SeekBar

Елемент SeekBar виконує роль повзунка, тобто шкалу поділок, на якій ми можемо змінювати поточну оцінку. Визначимо SeekBar в розмітці layout:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <SeekBar
        android:id="@+id/seekBar"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:progress="20"
        android:max="50" />

</LinearLayout>
```



Серед функціоналу елемента SeekBar слід зазначити метод **setOnSeekBarChangeListener()**, який дозволяє встановити обробники подій зміни значення повзунка. Скористаємося цією функціональністю і визначимо в файлі layout наступний код:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <TextView
        android:id="@+id/txtView"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:textAlignment="center"
        android:layout_centerHorizontal="true"
        android:gravity="center_horizontal"
        android:textSize="30sp" />
    <SeekBar
        android:id="@+id/seekBar"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:progress="20"
        android:max="50" />

</LinearLayout>
```

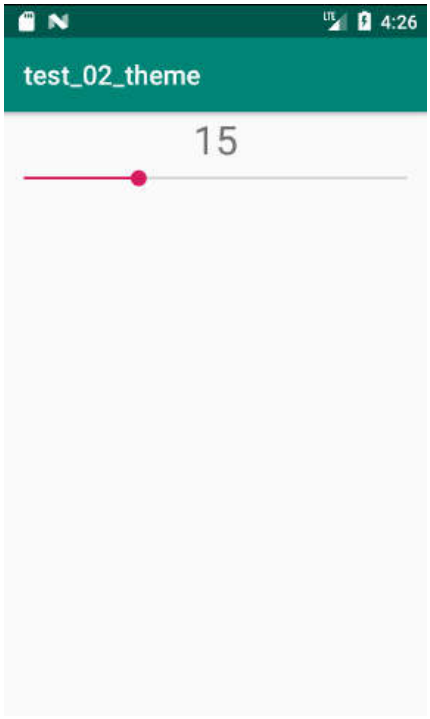
Тут визначено елемент TextView, який буде виводити поточне значення повзунка при його зміні.

І змінимо код MainActivity:

```
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.widget.SeekBar;
import android.widget.TextView;

public class MainActivity extends AppCompatActivity {

    TextView textView;
    SeekBar seekBar;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        seekBar = (SeekBar) findViewById(R.id.seekBar);
        textView = (TextView) findViewById(R.id.txtView);
        seekBar.setOnSeekBarChangeListener(new SeekBar.OnSeekBarChangeListener()
{
            @Override
            public void onProgressChanged(SeekBar seekBar, int progress, boolean
fromUser) {

                textView.setText(String.valueOf(progress));
            }

            @Override
            public void onStartTrackingTouch(SeekBar seekBar) {

            }

            @Override
            public void onStopTrackingTouch(SeekBar seekBar) {

            }
        });
    }
}
```

У метод setOnSeekBarChangeListener() передається об'єкт **SeekBar.OnSeekBarChangeListener**, який дозволяє встановити три методу-обробника:

- onProgressChanged: Спрацьовує при перетягуванні повзунка за шкалою. Рухаючись в метод параметр progress дозволяє отримати нове значення повзунка, яке в даному випадку передається в TextView для відображення на екрані
- onStartTrackingTouch: Спрацьовує при початку перетягування повзунка за шкалою
  onStopTrackingTouch: Спрацьовує при завершенні перетягування повзунка за шкалою



Також ми можемо отримати поточне значення повзунка, використавши метод **getProgress()**:

```
    public void onProgressChanged(SeekBar seekBar, int progress, boolean
fromUser) {

        textView.setText(String.valueOf(seekBar.getProgress()));
    }
```