Лабораторна робота №4. Робота з компонентами UI

Мета: вивчити методику роботи з елементами графічного інтерфейсу та макетами.

Теоретичні відомості

Макети

Layout представляє собою структуру, яка групує об’єкти графічного інтерфейсу. Android проронує декілька типів Layout, які відрізняються розташуванням компонентів (рис. 1).

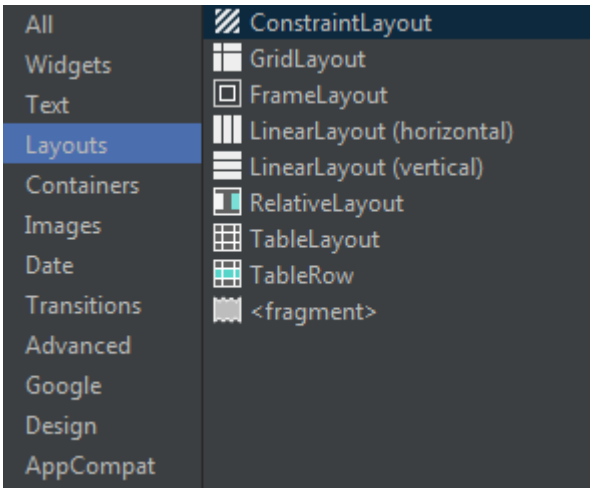

Рис. 1

За замовченням, при створенні нового проекту визначається тип **ConstraintLayout** (**https://developer.android.com/training/constraint-layout/index.html**).

ConstraintLayout дозволяє створювати складний графічний інтерфейс з ієрархією елементів інтерфейсу. Цей макет є більш зручним варіантом макету **RelativeLayout**, він доступний у версіях Android з 2.9. Однією з особливостей роботи з ConstraintLayout є вбудованість його в редактор макетів (Layout Editor). Тобто, налаштування цього макету можна повністю виконувати у редакторі макетів замість редагування XML файлів.

Для визначення позиції віджета, необхідно визначити хоча б одне обмеження (Constraint) по горизонталі та по вертикалі. Кожне обмеження визначає зв’язок або вирівнювання віджета з іншими віджетами, макетом (parent layout) або невидимими лініями (invisible guideline), які використовуються при побудові інтерфейса. Так, якщо в редакторі макету віджети не мають зв’язків та обмежень, під час запуску цього макету у додатку,

всі елементи інтерфейсу будуть відображені у верхніьому лівому куті макету. Тому, у прикладі (рис. 2) необхідно додати вертикальний зв'язок елементу **C** з елементом **A.**

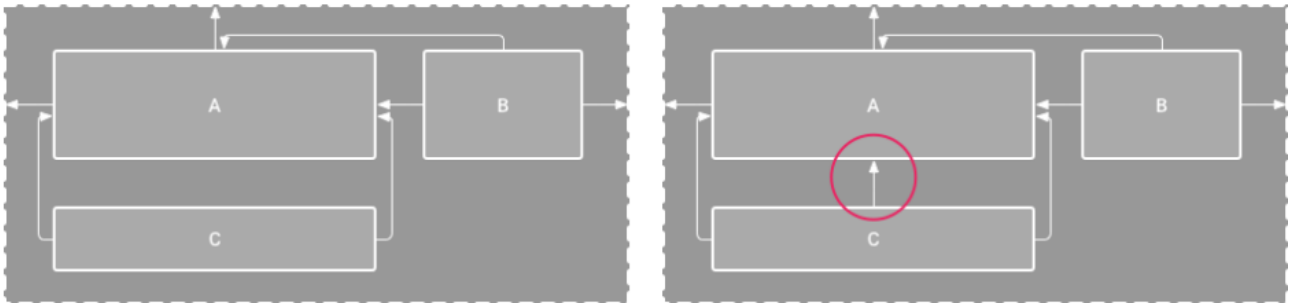

Рис. 2

Перетворення у ConstraintLayout

Про популярність макету свідчить можливість перетворення з інших типів макетів (рис. 3).

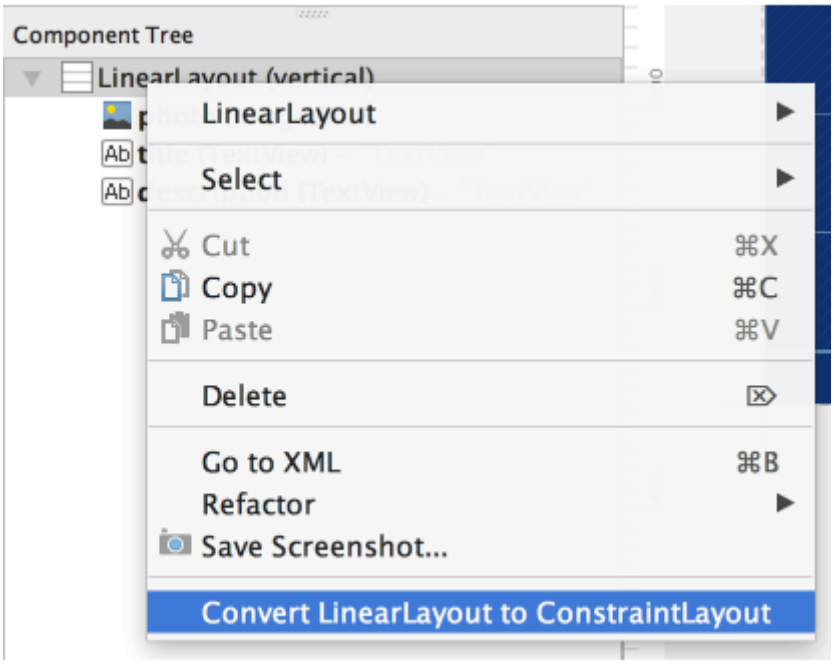

Рис. 3

Додавання обмеження.

При додаванні нового віджета на ConstraintLayout він відображається у рамці з квадратними маркерами зміни розміру по кутах та круговим маркером обмежень на кожній стороні.

Для створення нового обмеження необхідно обрати необхідний маркер обмежень на стороні рамці віджета та протягнути його до краю іншого віджета, границі макета або невидимій лінії.

При створенні нових обмежень дотримуються наступних правил.

1. Кожний віджет повинен мати хоча б два обмеження: по горизонталі та по вертикалі.
2. Можна створити обмеження тільки між відповідними маркерами обмежень різних віджетів. Наприклад, лівий або правий маркер одного віджета можна поєднати з лівим або правим маркером іншого віджета.
3. Один маркер обмежень можна використати для одного обмеження.

Наприклад, у макеті, зображеному на рис. 4 елемент **B** буде завжди праворуч від елементу **A**, елемент **C** буде розташовуватись нижче за елемент **A.** Однак, обмеження не визначає вирівнювання елементів.

Додавання вирівнювання.

Для додавання вирівнювання відносно віджета, необхідно задати обмеження відповідно з лівої, правої або обох сторін (рис. 4)

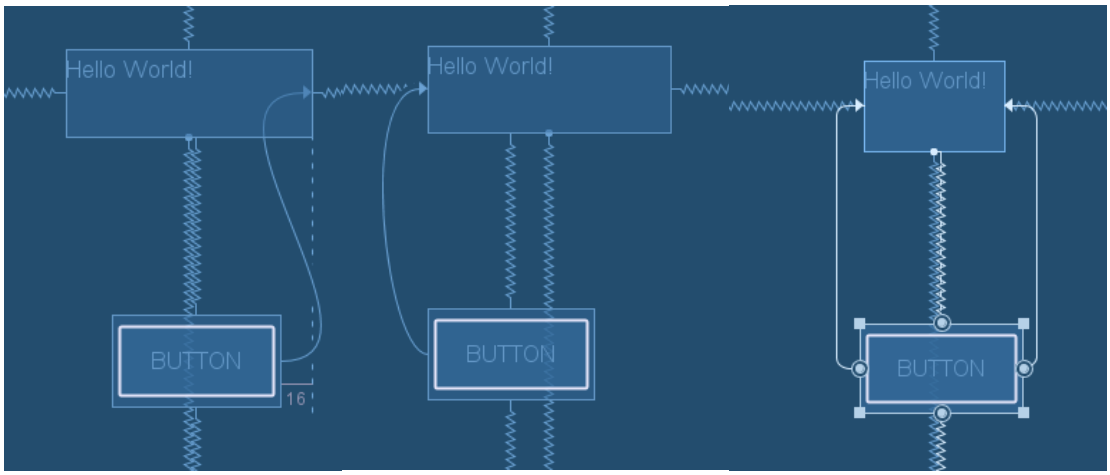

Рис. 4

Вирівнювання за базовою лінією.

Використовується для вирівнювання віджетів за рівнем тексту (рис. 5).

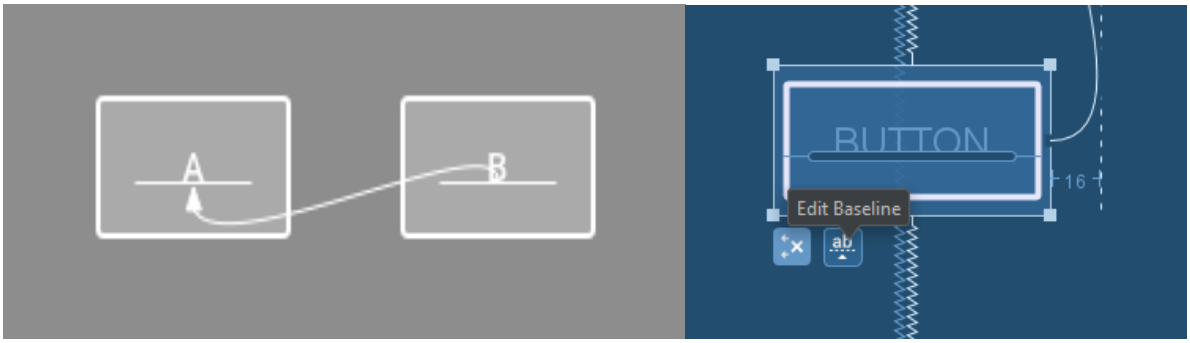

Рис. 5

Вирівнювання за лінією.

Для вирівнювання елементів інтерфейсу можна використовувати вертикальні або горизонтальні лінії (guideline). Ці лінії невидимі для користувача. Для додавання лінії необхідно викликати відповідний інструмент з Toolbar (рис. 6).

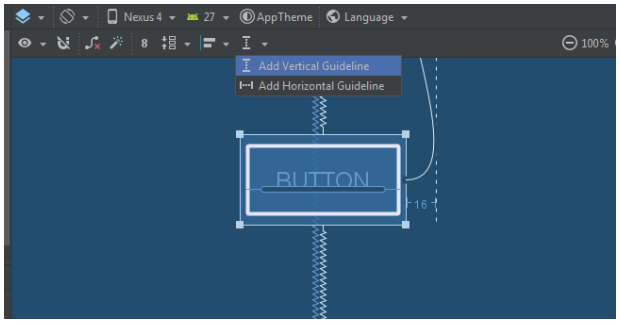

Рис. 6

Регулювання зміщення обмежень.

При додаванні нового віджета та встановлення вирівнювання (наприклад, обмежень ліворуч і праворуч), віджет вирівнюється між двома обмеженнями з співвідношенням 50/50% за замовченням. Це відношення змінюється у вікні Attributes (рис. 7).

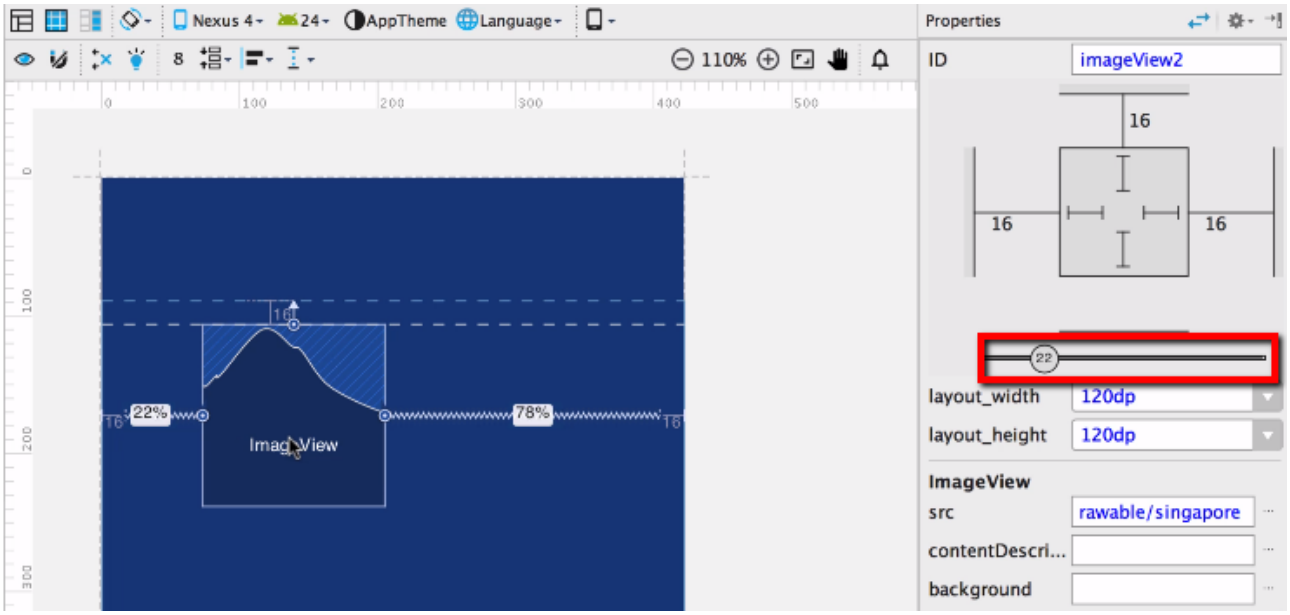

Рис. 7

Створення ланцюгів між віджетами.

Ланцюг – це група віджетів, які поєднані між собою двонаправленими обмеженями (рис. 8).

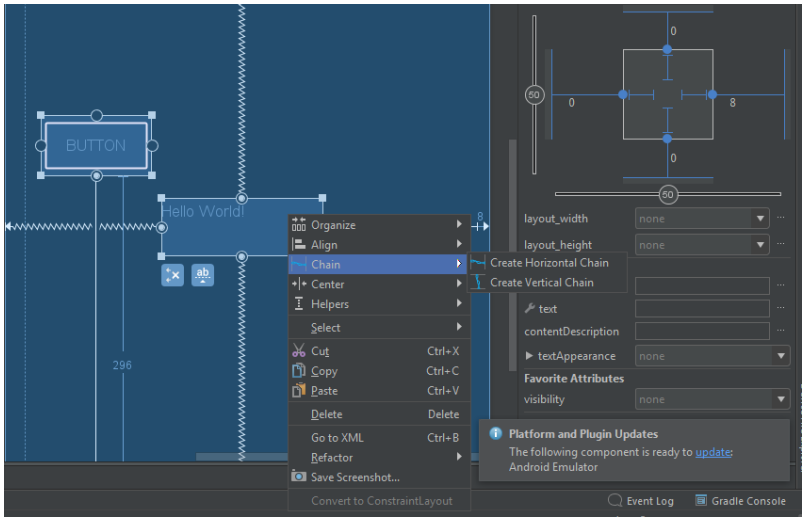

Рис. 8

Ланцюг дозволяє вирівнювати групу елементів вертикально або горизонтально.

Інструмент Infer Constraints (рис. 9) дозволяє створювати обмеження автоматично, розташувавши необхідні елементи графічного інтерфейсу на дизайнері.

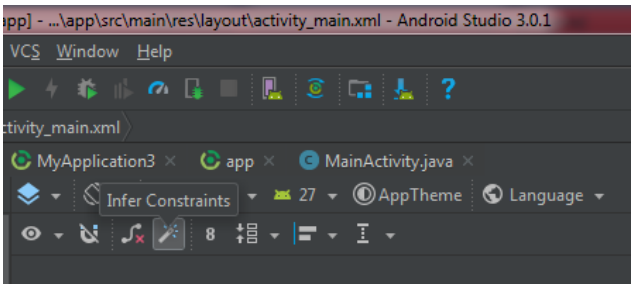

Рис. 9

RelativeLayout – це такий тип макету, в якому всі елементи розташовуються на відносних один до одного позиціях. Наприклад, один віджет може розташовуватися зліва чи знизу порівняно з іншим віджетом.

Деякі властивості позицій, які можуть визначатися при створенні RelativeLayout наведені нижче.

Розташування відносно батьківського елемента (значенням атрибуту є true/false):

- **android:layout_alignParentTop** – верхня границя елемента розташовується на верхній границі RelativeLayout;
- **android:layout_alignParentLeft** – ліва границя елемента розташовується на лівій границі RelativeLayout;
- **android:layout_alignParentRight** – права границя елемента розташовується на правій границі RelativeLayout;

- **android:layout_alignParentBottom** – нижня границя елемента розташовується на нижній границі RelativeLayout;
- **android:layout_centerInParent** – елемент розташовується в центрі батьківського по горизонталі и вертикалі;
- **android:layout_centerHorizontal** – елемент розташовується в центрі батьківського по горизонталі;
- **android:layout_centerVertical** – елемент розташовується в центрі батьківського по вертикалі.

Розташування відносно інших елементів (значенням атрибуту є id іншого елемента):

- **android:layout_above** – елемент розташовується зверху від іншого елемента;
- **android:layout_toLeftOf** – елемент розташовується ліворуч від іншого елемента;
- **android:layout_toRightOf** – елемент розташовується праворуч від іншого елемента;
- **android:layout_below** – елемент розташовується знизу від іншого елемента;
- **android:layout_alignBaseline** – базова лінія елемента вирівнюється по базовій лінії іншого елемента;
- **android:layout_alignTop** – верхня границя елемента вирівнюється по верхній границі іншого елемента;
- **android:layout_alignLeft** – ліва границя елемента вирівнюється по лівій границі іншого елемента;
- **android:layout_alignRight** – права границя елемента вирівнюється по правій границі іншого елемента;
- **android:layout_alignBottom** – нижня границя елемента вирівнюється по нижній границі іншого елемента.

Приклад xml файла RelativeLayout:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingLeft="16dp"
    android:paddingRight="16dp" >
    <EditText
        android:id="@+id/name"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="@string/reminder" />
    <Spinner
        android:id="@+id/dates"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_below="@id/name"
        android:layout_alignParentLeft="true"
        android:layout_toLeftOf="@+id/times" />
    <Spinner
        android:id="@id/times"
        android:layout_width="96dp"
        android:layout_height="wrap_content"
        android:layout_below="@id/name"
        android:layout_alignParentRight="true" />
    <Button
        android:layout_width="96dp"
        android:layout_height="wrap_content"
        android:layout_below="@id/times"
        android:layout_alignParentRight="true"
        android:text="@string/done" />
</RelativeLayout>
```

Рис. 10

LinearLayout. Макет для одного чи декількох елементів в лінію, горизонтально або вертикально. Для вибору орієнтації використовується атрибут **android:orientation** з можливими значеннями «horizontal»\«vertical» (рис. 11).

```
<?xml version="1.0" encoding="utf-8"?>
<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:paddingBottom="16dp"
    android:paddingLeft="16dp"
    android:paddingRight="16dp"
    android:paddingTop="16dp"
    tools:context="com.example.alex.lab6.MainActivity"
    android:orientation="vertical">

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="New Button"
        android:id="@+id/button"
        android:layout_weight="0" />

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="New Button"
        android:id="@+id/button2"
        android:layout_weight="2" />

</LinearLayout>
```

Рис. 11

TableLayout. Макет для розташування елементів у вигляді таблиці. Ряди таблиці задаються за допомогою елемента TableRow (тег TableRow в xml). Кількість стовпців в таблиці дорівнює максимальній кількості стовпців у рядках. Комірки таблиці можна залишати пустими.

Приклад. xml файл, в якому визначається таблиця 3x3 по 3 кнопки в ряд (рис. 12).

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

    <TableRow
        android:layout_width="match_parent"
        android:layout_height="match_parent">

        <Button
            android:text="New Button"
            android:id="@+id/button6"
            android:layout_column="0" />

        <Button
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="New Button"
            android:id="@+id/button7"
```

```
            android:layout_column="1" />

        <Button
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="New Button"
            android:id="@+id/button8"
            android:layout_column="2" />
    </TableRow>

    <TableRow
        android:layout_width="match_parent"
        android:layout_height="match_parent">

        <Button
            android:text="New Button"
            android:id="@+id/button9"
            android:layout_column="0" />

        <Button
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="New Button"
            android:id="@+id/button10"
            android:layout_column="1"
            android:layout_marginLeft="10dp"
            android:paddingTop="20dp"/>

        <Button
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="New Button"
            android:id="@+id/button11"
            android:layout_column="2" />
    </TableRow>

    <TableRow
        android:layout_width="match_parent"
        android:layout_height="match_parent">

        <Button
            android:text="New Button"
            android:id="@+id/button12"
            android:layout_column="0" />

        <Button
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="New Button"
            android:id="@+id/button13"
            android:layout_column="1" />

        <Button
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="New Button"
            android:id="@+id/button14"
            android:layout_column="2" />
    </TableRow>
</TableLayout>
```

Рис. 12

FrameLayout. Макет для відображення одного елемента. Всі елементи, які додаються на макет розташовуються у верхньому лівому куті екрана .

Приклад.

```
<?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">

    <FrameLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_gravity="center_horizontal">

        <Button
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@string/new_button1"
            android:id="@+id/button3"
            android:layout_gravity="bottom|left"/>

        <Button
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="New Button4"
            android:id="@+id/button4"
            android:layout_gravity="bottom|right"/>

        <Button
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="New Button3"
```

```
            android:id="@+id/button5"
            android:layout_gravity="center"/>

        <Button
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="New Button"
            android:id="@+id/button15"
            android:layout_gravity="top|left"/>

        <Button
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="New Button"
            android:id="@+id/button16"
            android:layout_gravity="top|right"/>
    </FrameLayout>
</LinearLayout>
```

Компоненти графічного інтерфейсу.

Прапорець (CheckBox). Елемент **CheckBox** дозволяє користувачу обрати один чи декілька об'єктів з множини (рис. 13).

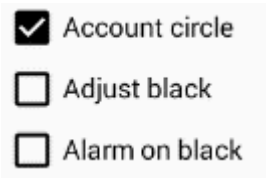

Рис. 13

Для того, щоб користувач міг обрати декілька об'єктів, потрібно оброблювати подію натиснення кожного з елементів **CheckBox.**

Обробка натискання на CheckBox. Для обробки події натискання може використовуватися атрибут **android:onClick**, який задається в xml файлі макета. Значення атрибута – ім'я метода обробника події натискання.

Наприклад, два елемента **CheckBox** описуються в xml файлі (рис. 14).

```
<CheckBox android:id="@+id/checkbox_meat"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/meat"
    android:onClick="onCheckboxClicked"/>
<CheckBox android:id="@+id/checkbox_cheese"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/cheese"
    android:onClick="onCheckboxClicked"/>
```

Рис. 14

Визначення метода відбувається **onCheckboxClicked** в java-класі (рис. 15)

```
public void onCheckboxClicked(View view) {
    // Is the view now checked?
    boolean checked = ((CheckBox) view).isChecked();

    // Check which checkbox was clicked
    switch(view.getId()) {
        case R.id.checkbox_meat:
            if (checked)
                // Put some meat on the sandwich
            else
                // Remove the meat
            break;
        case R.id.checkbox_cheese:
            if (checked)
                // Cheese me
            else
                // I'm lactose intolerant
            break;
        // TODO: Veggie sandwich
    }
}
```

Рис. 15

Приклад 1. Потрібно розробити додаток, який при натисканні кнопки циклічно змінює картинки, які обрані за допомогою **CheckBox**. Можлива реалізація виглядає наступним чином.

Файл activity_mail.xml.

```
<?xml version=»1.0» encoding=»utf-8»?>
<TableLayout 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:paddingBottom=»@dimen/activity_vertical_margin»
    android:paddingLeft=»@dimen/activity_horizontal_margin»
    android:paddingRight=»@dimen/activity_horizontal_margin»
    android:paddingTop=»@dimen/activity_vertical_margin»
    tools:context=»com.example.alex.lab10.MainActivity»>

    <TextView
        android:id=»@+id/textView»
        android:layout_width=»wrap_content»
        android:layout_height=»wrap_content»
        android:text=»@string/change_images» />

    <ImageView
        android:layout_width=»wrap_content»
        android:layout_height=»wrap_content»
        android:id=»@+id/imageView»
        android:layout_marginTop=»26dp»
        android:src=»@drawable/ic_account_circle_black_48dp»
        android:layout_below=»@+id/textView»
        android:layout_toEndOf=»@+id/textView» />
```

```
<CheckBox
    android:layout_width=»wrap_content»
    android:layout_height=»wrap_content»
    android:text=»@string/account_circle»
    android:id=»@+id/checkBox»
    android:checked=»true»
    android:layout_alignBaseline=»@+id/button»
    android:layout_alignBottom=»@+id/button»
    android:layout_alignEnd=»@+id/imageView»
    android:onClick=»checkImage»/>

<CheckBox
    android:layout_width=»wrap_content»
    android:layout_height=»wrap_content»
    android:text=»@string/adjust_black»
    android:id=»@+id/checkBox2»
    android:checked=»false»
    android:layout_below=»@+id/checkBox»
    android:layout_alignRight=»@+id/imageView»
    android:onClick=»checkImage»/>

<CheckBox
    android:layout_width=»wrap_content»
    android:layout_height=»wrap_content»
    android:text=»@string/alarm_on_black»
    android:id=»@+id/checkBox3»
    android:checked=»false»
    android:layout_below=»@+id/checkBox2»
    android:layout_alignRight=»@+id/imageView»
    android:onClick=»checkImage»/>

<CheckBox
    android:layout_width=»wrap_content»
    android:layout_height=»wrap_content»
    android:text=»@string/assinstant_black»
    android:id=»@+id/checkBox4»
    android:checked=»false»
    android:layout_below=»@+id/checkBox3»
    android:layout_alignRight=»@+id/imageView»
    android:onClick=»checkImage»/>

<CheckBox
    android:layout_width=»wrap_content»
    android:layout_height=»wrap_content»
    android:text=»Brihtness»
    android:id=»@+id/checkBox5»
    android:checked=»false»
    android:layout_below=»@+id/checkBox4»
    android:layout_alignRight=»@+id/imageView»
    android:onClick=»checkImage»/>

<TableRow
    android:layout_width=»match_parent»
    android:layout_height=»match_parent»></TableRow>

<TableRow
    android:layout_width=»match_parent»
    android:layout_height=»match_parent»>

    <Button
        android:id=»@+id/button»
        android:text=»@string/change_image»
        android:textColor=»@color/colorAccent»
        android:layout_below=»@+id/checkBox5»
        android:layout_toEndOf=»@+id/imageView»
        android:layout_column=»25»
```

```
            android:layout_gravity=»center_horizontal» />
    </TableRow>
</TableLayout>
```

Файл MainActivity.java

```
package com.example.alex.lab4;

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

import java.util.ArrayList;

public class MainActivity extends AppCompatActivity {

    ArrayList images = new ArrayList();
    Button myButton1;
    ImageView imageView;
    CheckBox image1,image2,image3,image4,image5;
    int count=0;


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        imageView=(ImageView) findViewById(R.id.imageView);
        CheckBox image1=(CheckBox) findViewById(R.id.checkBox);
        CheckBox image2=(CheckBox) findViewById(R.id.checkBox2);
        CheckBox image3=(CheckBox) findViewById(R.id.checkBox3);
        CheckBox image4=(CheckBox) findViewById(R.id.checkBox4);
        CheckBox image5=(CheckBox) findViewById(R.id.checkBox5);


        image1.setChecked(true);
        images.add(R.drawable.ic_account_circle_black_48dp);
        imageView.setImageDrawable(getResources().getDrawable((int)
images.get(0)));

        myButton1 = (Button) findViewById(R.id.button);
        myButton1.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                count=(count+1)%5;
                imageView.setImageDrawable(getResources().getDrawable((int)
images.get(count)));
            }

        });
    }
    public void checkImage(View v){
        boolean ch = ((CheckBox) v).isChecked();
        switch (v.getId()){
            case R.id.checkBox:
                if (ch)
                    images.add(R.drawable.ic_account_circle_black_48dp);
                else
                    images.remove(R.drawable.ic_account_circle_black_48dp);
                break;
            case R.id.checkBox2:
                if (ch)
```

```
                images.add(R.drawable.ic_adjust_black_48dp);
            else
                images.remove(R.drawable.ic_adjust_black_48dp);
            break;
        case R.id.checkBox3:
            if (ch)
                images.add(R.drawable.ic_alarm_on_black_48dp);
            else
                images.remove(R.drawable.ic_alarm_on_black_48dp);
            break;
        case R.id.checkBox4:
            if (ch)
                images.add(R.drawable.ic_assistant_black_48dp);
            else
                images.remove(R.drawable.ic_assistant_black_48dp);
            break;
        case R.id.checkBox5:
            if (ch)
                images.add(R.drawable.ic_brightness_4_black_48dp);
            else
                images.remove(R.drawable.ic_brightness_4_black_48dp);
            break;

    }
  }
}
```

Перемикач (RadioGroup, RadioButton). **RadioButton** дозволяє користувачу вибрати один об’єкт з множини. Для групування декількох елементів **RadioButton**, один з яких повинен бути ввімкнено, використовується **RadioGroup** (рис. 16).

Рис.16

Обробки натискання RadioButton. Використовується підхід, аналогічний до **CheckBox** (використання атрибута **android:onClick**) (рис. 17).

```
<?xml version="1.0" encoding="utf-8"?>
<RadioGroup xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical">
    <RadioButton android:id="@+id/radio_pirates"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/pirates"
        android:onClick="onRadioButtonClicked"/>
    <RadioButton android:id="@+id/radio_ninjas"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/ninjas"
        android:onClick="onRadioButtonClicked"/>
</RadioGroup>
```

```java
public void onRadioButtonClicked(View view) {
    // Is the button now checked?
    boolean checked = ((RadioButton) view).isChecked();

    // Check which radio button was clicked
    switch(view.getId()) {
        case R.id.radio_pirates:
            if (checked)
                // Pirates are the best
            break;
        case R.id.radio_ninjas:
            if (checked)
                // Ninjas rule
            break;
    }
}
```

Рис. 17

Кнопка-перемикач **(ToggleButton)**. Використовується для вибору одного з двох можливих налаштувань\станів (рис. 18). Перемикання кнопки виконується за допомогою методів **CompoundButton.setChecked()** або **CompoundButton.toggle()**.

Рис. 18

Обробка натискання **ToggleButton** за допомогою методів **CompoundButton.OnCheckedChangeListener**, **setOnCheckedChangeListener** (рис. 19).

```java
ToggleButton toggle = (ToggleButton) findViewById(R.id.togglebutton);
toggle.setOnCheckedChangeListener(new CompoundButton.OnCheckedChangeListener() {
    public void onCheckedChanged(CompoundButton buttonView, boolean isChecked) {
        if (isChecked) {
            // The toggle is enabled
        } else {
            // The toggle is disabled
        }
    }
});
```

Рис. 19

Контейнер ScrollView. При великому обсязі інформації, яку потрібно помістити на екран пристрою використовується віджет **ScrollView** – контейнерний елемент, який наслідується від **ViewGroup**.

Методи **scrollBy()** і **scrollTo()** дозволяють програмно прокручувати контент. Наприклад, можна організувати автоматичне прокручування під час читання.

Приклад 2. Потрібно реалізувати додаток відображення великого тексту і прокрутки його за натисканням кнопки «Up» догори і кнопки «Down» донизу. При ввімкненій кнопці-перемикачі прокрутка кнопками працює, інакше – ні.

Можлива реалізація виглядає наступним чином.

Файл activity_mail.xml.

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout 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:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="com.example.alex.lab103.MainActivity">

    <ScrollView
        android:layout_width="match_parent"
        android:layout_height="200dp"
        android:id="@+id/scrollView"
        android:layout_alignParentStart="true"
        android:layout_marginStart="23dp"
        android:layout_marginTop="36dp"
        android:scrollbars="vertical"
        android:layout_alignParentEnd="false" >

        <TextView
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:textAppearance="?android:attr/textAppearanceLarge"
            android:text="Large Text"
            android:id="@+id/textView2" />

    </ScrollView>
    <ToggleButton
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="New ToggleButton"
        android:id="@+id/toggleButton"
        android:layout_alignParentBottom="true"
        android:layout_alignEnd="@+id/scrollView"
        android:checked="false" />

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/to_begin"
        android:id="@+id/button"
        android:layout_alignTop="@+id/toggleButton"
        android:layout_alignStart="@+id/scrollView" />

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/down"
        android:id="@+id/button2"
```

```
        android:layout_above="@+id/button"
        android:layout_alignStart="@+id/scrollView" />

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/up"
        android:id="@+id/button3"
        android:layout_above="@+id/button2"
        android:layout_alignStart="@+id/scrollView" />


</RelativeLayout>
```

Файл MainActivity.java.

```
package com.example.alex.lab103;

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

public class MainActivity extends AppCompatActivity {
    TextView myText;
    Button onTop,up,down;
    ToggleButton mySwitch;
    ScrollView myScroll;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        myText=(TextView) findViewById(R.id.textView2);
        onTop=(Button) findViewById(R.id.button);
        up=(Button) findViewById(R.id.button2);
        down=(Button) findViewById(R.id.button3);
        mySwitch=(ToggleButton) findViewById(R.id.toggleButton);
        myScroll=(ScrollView) findViewById(R.id.scrollView);
        myText.setText(R.string.my_text);

        mySwitch.setOnCheckedChangeListener(new
CompoundButton.OnCheckedChangeListener() {
            @Override
            public void onCheckedChanged(CompoundButton buttonView, boolean
isChecked) {
                if (isChecked) {
                    up.setOnClickListener(new View.OnClickListener() {
                        @Override
                        public void onClick(View v) {
                            myScroll.scrollBy(0, +20);
                        }

                    });

                    down.setOnClickListener(new View.OnClickListener() {
                        @Override
                        public void onClick(View v) {
                            myScroll.scrollTo(0, -20);
                        }
```

```
                    });

                }

            }
        });

        onTop.setOnClickListener(   new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                myScroll.scrollTo(0, 0);
            }

        });

    }
}
```

Список, що розкривається (**Spinner**). За допомогою цього віджета користувач може вибрати одне значення з множини. Натиснення на **Spinner** активує спадне меню з можливими значеннями, з яких користувач може вибрати одне (рис. 20).

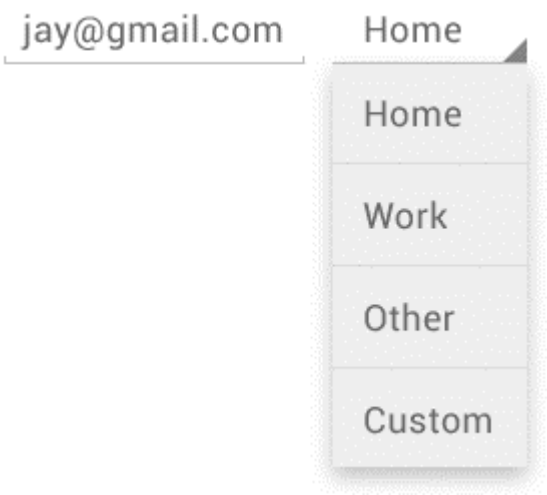

Рис. 20

Після об’явлення **Spinner** в xml файлі (рис. 21), потрібно визначити в java-класі **SpinnerAdapter** для налаштування спадного меню.

```
<Spinner
    android:id="@+id/planets_spinner"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content" />
```

Рис. 21

Множина можливих виборів для **Spinner** може бути визначена у якому-небудь ресурсі, але має бути обов’язково визначена в **SpinnerAdapter** за допомогою **ArrayAdapter**, якщо використовуються масиви, чи **CursorAdapter**, у разі використання бази даних.

В даному контексті, адаптер (**SpinnerAdapter, ArrayAdapter, CursorAdapter**) – це об’єкт, призначення якого прив’язати дані до елемента графічного інтерфейса\елемента керування.

Наприклад, якщо множина можливих значень для **Spinner** відома заздалегідь, її можна визначити як строковий масив у файлі string.xml (рис. 22).

```xml
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string-array name="planets_array">
        <item>Mercury</item>
        <item>Venus</item>
        <item>Earth</item>
        <item>Mars</item>
        <item>Jupiter</item>
        <item>Saturn</item>
        <item>Uranus</item>
        <item>Neptune</item>
    </string-array>
</resources>
```

Рис. 22

З масивом описаним на рис. 22, можна використовувати наступний код у файлі java-класу відповідного activity (рис. 23).

```java
Spinner spinner = (Spinner) findViewById(R.id.spinner);
// Create an ArrayAdapter using the string array and a default spinner layout
ArrayAdapter<CharSequence> adapter = ArrayAdapter.createFromResource(this,
        R.array.planets_array, android.R.layout.simple_spinner_item);
// Specify the layout to use when the list of choices appears
adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
// Apply the adapter to the spinner
spinner.setAdapter(adapter);
```

Рис. 23

Метод **createFromResource()** дозволяє створити **ArrayAdapter** зі строкового масива. Третій аргумент метода **createFromResource()** визначає layout у якому будуть відображатися елементи для вибору. Макет **android.R.layout.simple_spinner_item** є вбудованим в платформу, так само, як і **android.R.layout.simple_spinner_dropdown_item**, який використовується для представлення спадного меню з варіантами вибору.

Відповідь на вибір зі списку. Коли користувач вибере один з варіантів, які пропонуються у спадному меню, об’єкт **Spinner** отримує подію **on-item-selected**. Обробка цієї події може виконуватись наступним чином (рис. 24).

```
public class SpinnerActivity extends Activity implements OnItemSelectedListener {
    ...

    public void onItemSelected(AdapterView<?> parent, View view,
            int pos, long id) {
        // An item was selected. You can retrieve the selected item using
        // parent.getItemAtPosition(pos)
    }

    public void onNothingSelected(AdapterView<?> parent) {
        // Another interface callback
    }
}
```

```
Spinner spinner = (Spinner) findViewById(R.id.spinner);
spinner.setOnItemSelectedListener(this);
```

Рис. 24

Для визначення обробника події вибору застосовується обробник **AdapterView.OnItemSelectedListener** і методи **onItemSelected()** та **onNothingSelected()**.

ProgressBar (рис. 25) використовується для візуалізації стану процесу, що виконується. Під час виконання деякого процесу, користувачу необхідно відображати стан його виконання.

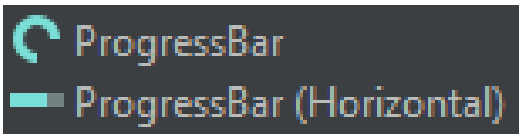

Рис. 25

Основні методи компонента ProgressBar:

setProgress() – встановлює значення індикатора прогресу;

getProgress() – повертає поточне значення індикатора прогресу;

incrementProgressBy() встановлює величину дискретизації прирощення значення індикатора;

setMax() — встановлює максимальне значення індикатора прогресу.

http://developer.alexanderklimov.ru/android/views/progressbar.php

Завдання до лабораторної роботи

1. Протестуйте програму з прикладу 1 та виправіть знайдені помилки. При необхідності використовуйте режим Debug.
2. Розширити можливості налаштування логічної гри з лабораторної роботи №3 за допомогою елементів графічного інтерфейсу **CheckBox**, **RadioButton**, **ProgressBar**, **Spinner**, **ToggleButton**. Для вирівнювання віджетів можна використовувати різні типи layout.

Контрольні запитання

1. Які чином виконується робота з елементом **ProgressBar**?
2. Яким чином оброблюється натискання на елементи віджетів **CheckBox, RadioButton**?
3. Яким чином оброблюється вибір одного варіанту з множини можливих в елементі **Spinner**?