Лабораторная работа № 3

«Кнопки и обработка событий»

Продолжительность выполнения лабораторной работы: 2 ак. часа.

Целью лабораторной работы является знакомство с различными классами кнопок, представленными на Android, а также обработкой событий.

Теоретическая часть

Виджет — это объект View, который служит интерфейсом для взаимодействия с пользователем. Фактически, виджеты — это элементы управления. В Android имеется набор готовых виджетов, таких как кнопки, переключатели, текстовые поля и т.д., с помощью которых можно быстро сформировать пользовательский интерфейс приложения.

Текстовые поля в Android представлены двумя классами: **TextView** и **EditText**. TextView предназначен для отображения текста без возможности его редактирования пользователем, а также может использоваться как элемент для отображения текстовых данных в контейнерных виджетах для отображения списков. EditText ─ это поле для ввода текста пользователем. EditText является потомком TextView.

Ширина и высота TextView и EditText задаются как и для любого объекта View с помощью атрибутов android:layout_width и android:layout_height. Если необходимо обращаться к этим элементам в программном коде, например, для изменения их свойств или получения введенного текста, то необходимо задать идентификатор android:id. Для задания текста, отображаемого элементом, используется атрибут android:text. В него может передаваться как сам текст в фиксированном виде, так и ссылка на **строковый ресурс**.

Некоторые объекты (изображения, строковые константы, цвета, анимация, стили и т.д.) принято хранить за пределами исходного кода в виде отдельных файлов ресурсов. Их легче поддерживать, обновлять и редактировать. Строковые константы хранятся в файле strings.xml в папке res/values.

Листинг 1. Строковые ресурсы strings.xml

```
<resources>
    <string name="app_name">App Name</string>
</resources>
```

По умолчанию в файле strings.xml создаётся корневой элемент resources, внутри которого размещается единственный элемент string, содержащий имя приложения. В атрибуте name элемента string указывается имя строкового ресурса. Все другие строковые ресурсы должны размещаться внутри корневого элемента resources.

Для задания текста, отображаемого элементом TextView или EditText (или другими потомками TextView, например, кнопкой Button), в качестве ссылки на строковый ресурс, в xml-коде компоновки в атрибут android:text передаётся ссылка в следующем виде:

android:text="@string/string_name"

Для тех же целей в программном java-коде необходимо найти элемент с помощью метода findViewById по его идентификатору и изменить текст методом setText. Например:

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

text.setText(R.string.text1);

В таблице 1 представлены часто использующиеся xml-атрибуты TextView и java-методы, позволяющие задать те же свойства.

Таблица 1. Соответствие XML-атрибутов и методов в классах представлений.

XML-атрибут	**Java-метод**	**Назначение**
android:text	setText	Назначить текст
android:textColor	setTextColor	Назначить цвет текста
android:textColorHighlight	setHighlightColor	Назначить цвет выделения текста
android:textSize	setTextSize	Назначить размер текста
android:textStyle	setTypeface (null, int)	Назначить стиль текста (жирный, курсив, нормальный)
android:typeface	setTypeface (Typeface)	Назначить тип шрифта (с засечками, без засечек, моноширинный)

Основной метод класса EditText — getText (), который возвращает текст, содержащийся в элементе EditText.

Для отображения графики предназначен виджет ImageView. Как и элемент TextView, который является базовым виджетом для текстового наполнения

пользовательского интерфейса, ImageView является базовым элементом для графического наполнения и может использоваться в контейнерных виджетах для отображения графики.

Класс ImageView может загружать изображения из различных источников, как из ресурсов приложения, так и из внешних источников. В этом классе существует несколько методов для загрузки изображений:

- setImageResource (int resId) — загружает изображение по его идентификатору ресурса;
- setImageURI (Uri uri) — загружает изображение по его URI;
- setImageBitmap (Bitmap bitmap) — загружает растровое изображение.

Ресурсы изображения хранятся в папке res/drawable. Для загрузки ресурса изображения в коде, можно передать идентификатор ресурса в виде параметра метода. Например, с помощью метода setImageResource () можно указать использование виджетом ImageView ресурса res/drawable/image.png:

Листинг 2. Задание источника изображения в ImageView в коде

```
ImageView myImage = (ImageView) findViewById(R.id.imageview);
myImage.setImageResource(R.drawable.image);
```

При добавлении в компоновку элемента ImageView можно указать ссылку на источник изображения в атрибуте android:src, например:

Листинг 3. Задание источника изображения в ImageView в xml

```
<ImageView
    android:id="@+id/imageview"
    android:src="@drawable/image"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"/>
```

Для организации взаимодействия виджетов с пользователем необходимо определить обработчик события и зарегистрировать его для данного элемента. Класс View содержит коллекцию вложенных интерфейсов, которые называются On…Listener (), в каждом из которых объявлен единственный абстрактный метод. Этот метод необходимо переопределить в вашем классе. Его будет вызывать система Android, когда пользователь будет взаимодействовать с экземпляром View (например,

с кнопкой), к которому был подсоединен слушатель события. Класс View содержит несколько вложенных интерфейсов, в том числе:

- OnClickListener — слушатель события нажатия;
- OnLongClickListener — слушатель события длинного нажатия;
- OnFocusChangeListener — слушатель события изменения фокуса ввода;
- OnKeyListener — слушатель события нажатия аппаратной кнопки;
- OnTouchListener — слушатель события касания и перемещения пальца.

Если требуется, например, чтобы кнопка (с идентификатором myButton) получила уведомление о нажатии пользователем на неё, необходимо в классе активити реализовать интерфейс OnClickListener и определить его метод обратного вызова onClick, куда будет помещен код обработки нажатия кнопки (код поменяет текст в TextView с идентификатором myText на «Вы нажали на кнопку»), и зарегистрировать слушатель события с помощью метода setOnClickListener (листинг 4).

Листинг 4. Обработка нажатия на кнопку.

```
Button myButton = (Button)findViewById(R.id.myButton);
TextView myText = (TextView)findViewById(R.id.myText);
myButton.setOnClickListener(new View.OnClickListener()
{
    public void onClick(View v)
    {
        myText.setText("Вы нажали на кнопку");
    }
});
```

Динамическое создание элементов управления

В некоторых случаях во время выполнения программы нам может потребоваться создание компоновки и элементов управления в коде программы.

В программном коде при создании компоновки используются те же принципы, как и при создании компоновки в xml-файле. Платформа Android в пакете android.widget предоставляет набор классов для создания компоновок способом, аналогичным объявлению в xml-файле.

Для создания корневого элемента разметки создаётся экземпляр класса разметки, в конструктор которого передается контекст приложения. Контекст приложения —

это набор информации о среде, в которой выполняется приложение. В классе окна приложения контекст приложения можно получить вызовом метода getApplicationContext (). Например, создать линейную компоновку можно следующим образом:

LinearLayout layout = new LinearLayout(getApplicationContext());

В классе, реализующем Activity, необязательно явно вызывать метод getApplicationContext () для получения контекста приложения, вместо этого можно использовать ссылку на данный экземпляр класса this, который уже содержит контекст приложения.

Далее задаются параметры компоновки, путём создания экземпляра класса LinearLayout.LayoutParams и передачи его в метод setLayoutParams () следующим образом:

LinearLayout.LayoutParams params = new LinearLayout.LayoutParams(

LayoutParams.MATCH_PARENT, LayoutParams.MATCH_PARENT));

layout.setLayoutParams (params);

Для задания способа расположения дочерних элементов в линейной компоновке (вертикально или горизонтально) есть метод setOrientation (), параметром которого служат две целочисленные константы, HORIZONTAL (0) и VERTICAL (1).

Дочерние виджеты, например, кнопки, текстовые поля создаются аналогично корневой компоновке. Константы, которые будут использоваться в качестве идентификаторов объявляются в классе Activity:

private static final int ID_BUTTON = 101;

Листинг 5. Динамическое создание элементов управления.

```
package com.example.dynamicui;
import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.view.ViewGroup.LayoutParams;
import android.widget.Button;
import android.widget.LinearLayout;
import android.widget.TextView;
public class MainActivity extends AppCompatActivity implements
View.OnClickListener {
    private static final int ID_BUTTON1 = 101;
    private static final int ID_BUTTON2 = 102;
    private static final int ID_TEXT = 103;
```

```
    private TextView text;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        LinearLayout layout = new LinearLayout(this);
        layout.setOrientation(LinearLayout.VERTICAL);
        layout.setLayoutParams(new LinearLayout.LayoutParams(
                LayoutParams.MATCH_PARENT, LayoutParams.MATCH_PARENT));
        Button button1 = new Button(this);
        button1.setText("Кнопка 1");
        button1.setId(ID_BUTTON1);
        button1.setLayoutParams(new LinearLayout.LayoutParams(
                LayoutParams.WRAP_CONTENT, LayoutParams.WRAP_CONTENT));
        button1.setOnClickListener(this);
        Button button2 = new Button(this);
        button2.setText("Кнопка 2");
        button2.setId(ID_BUTTON2);
        button2.setLayoutParams(new LinearLayout.LayoutParams(
                LayoutParams.WRAP_CONTENT, LayoutParams.WRAP_CONTENT));
        button2.setOnClickListener(this);
        text = new TextView(this);
        text.setId(ID_TEXT);
        text.setText("Событие: ");
        text.setLayoutParams(new LinearLayout.LayoutParams(
                LayoutParams.WRAP_CONTENT, LayoutParams.WRAP_CONTENT));
        layout.addView(button1);
        layout.addView(button2);
        layout.addView(text);
        this.addContentView(layout, new LinearLayout.LayoutParams(
                LayoutParams.MATCH_PARENT, LayoutParams.MATCH_PARENT));
    }
    @Override
    public void onClick(View v) {
        switch (v.getId()) {
            case ID_BUTTON1:
                text.setText("Нажатие на первую кнопку"); break;
            case ID_BUTTON2:
                text.setText("Нажатие на вторую кнопку"); break; }
    }
}
```

Задание

В листинге 6 частично приведен код MainActivity.java из приложения, реализующего простой редактор текста, позволяющий изменить стиль, размер и тип шрифта для текста. Приложение представлено на рисунке 1.

Изучите код, представленный в листинге 6 и на его основе разработайте аналогичное приложение. В представленном коде имеются пропущенные фрагменты,

которые обозначены звездочками (***). Пропущено может быть произвольное количество строк.

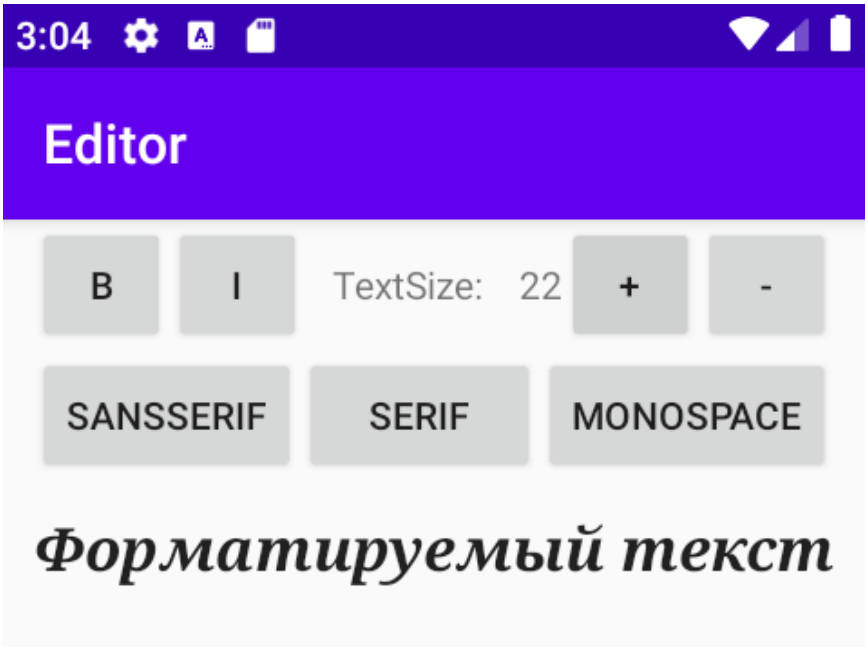

Рисунок 1. Простой текстовый редактор.

Листинг 6. Фрагмент кода приложения.

```
package com.example.editor;
import androidx.appcompat.app.AppCompatActivity;
import android.graphics.Typeface;
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 implements
View.OnClickListener {
    private float mTextSize = 20;
    private EditText mEdit;
    private TextView tSize;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        mEdit = findViewById(R.id.edit_text);
        tSize = findViewById(R.id.size);
        Button buttonB = findViewById(R.id.button_b);
        Button buttonI = findViewById(R.id.button_i);
        Button buttonSans = findViewById(R.id.button_sans);
        ***
        buttonB.setOnClickListener(this);
        buttonI.setOnClickListener(this);
        buttonSans.setOnClickListener(this);
```

```
        ***
    }
    @Override
    public void onClick(View v) {
        switch (v.getId()) {
            case R.id.button_plus:
                if (mTextSize < 72) {
                    mTextSize += 2;
                    mEdit.setTextSize(mTextSize);
                    tSize.setText(String.format("%.0f", mTextSize));
                }
                break;
            case R.id.button_minus:
                if (mTextSize > 18) {
                    mTextSize-=2;
                    mEdit.setTextSize(mTextSize);
                    tSize.setText(String.format("%.0f",mTextSize));
                }
                break;
            case R.id.button_b:
                if (mEdit.getTypeface().getStyle()== Typeface.ITALIC)
                    mEdit.setTypeface(mEdit.getTypeface(),
Typeface.BOLD_ITALIC);
                else if (mEdit.getTypeface().getStyle()==
Typeface.BOLD_ITALIC)
                    mEdit.setTypeface(mEdit.getTypeface(),
Typeface.ITALIC);
                else if (mEdit.getTypeface().getStyle()== Typeface.BOLD)
                    mEdit.setTypeface(Typeface.create(mEdit.getTypeface(),
Typeface.NORMAL));
                else mEdit.setTypeface(mEdit.getTypeface(), Typeface.BOLD);
                break;
            case R.id.button_i: ***
            case R.id.button_sans:
                if (mEdit.getTypeface().getStyle()== Typeface.ITALIC)
                    mEdit.setTypeface(Typeface.SANS_SERIF,
Typeface.ITALIC);
                else if (mEdit.getTypeface().getStyle()==
Typeface.BOLD_ITALIC)
                    mEdit.setTypeface(Typeface.SANS_SERIF,
Typeface.BOLD_ITALIC);
                else if (mEdit.getTypeface().getStyle()== Typeface.BOLD)
                    mEdit.setTypeface(Typeface.SANS_SERIF, Typeface.BOLD);
                else mEdit.setTypeface(Typeface.SANS_SERIF,
Typeface.NORMAL);
                break;
            case R.id.button_serif:
                if (mEdit.getTypeface().getStyle()== Typeface.ITALIC)
                    mEdit.setTypeface(Typeface.SERIF, Typeface.ITALIC);
                else if (mEdit.getTypeface().getStyle()==
Typeface.BOLD_ITALIC)
```

```
                    mEdit.setTypeface(Typeface.SERIF,
Typeface.BOLD_ITALIC);
                else if (mEdit.getTypeface().getStyle()== Typeface.BOLD)
                    mEdit.setTypeface(Typeface.SERIF, Typeface.BOLD);
                else mEdit.setTypeface(Typeface.SERIF, Typeface.NORMAL);
                break;
            case R.id.button_monospace: ***
        }
    }
}
```

Контрольные вопросы:

1. Какой метод содержит интерфейс View.OnClickListener?
2. Каково назначение метода findViewById?
3. Каково назначение метода setTypeface?
4. Какие параметры передаются в метод setTypeface?
5. Каково назначение метода setTextSize?
6. Каково назначение метода setText?
7. Каково назначение метода getText?
8. Каково назначение метода getTypeface?
9. Каково назначение метода setOnClickListener?
10. Каково назначение метода setContentView?