МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЗАПОРІЗЬКА ПОЛІТЕХНІКА»

кафедра програмних засобів

Звіт

з лабораторної роботи № 3

з дисципліни «Мобільні операційні системи»

на тему:

«Механізм роботи додатків. Створення інтерфейсу користувача.  
Alert діалоги та Toast повідомлення»

Виконав

ст. гр. КНТ-137 В.В. Козлов

Прийняв

старший викладач Є.М. Федорченко

1. Мета роботи

Ознайомитись із механізмами роботи мобільних застосунків написаних в інтегрованому середовищі розробки Android Studio. Створити мобільний застосунок, що реалізує інтерфейс користувача з кнопками і полями вводу та використовує Alert діалоги та Toast повідомлення.

1. Завдання на лабораторну роботу

Ознайомитись з основними теоретичними відомостями.

Створити мобільний застосунок, що реалізує інтерфейс користувача з кнопками і полями вводу.

Використати Alert діалоги та Toast повідомлення.

Відповісти на контрольні запитання.

Оформити звіт.

1. Теоретичні відомості

Візуально зрозумілий та привабливий інтерфейс сприяє більш приємній та ефективній роботі з програмою. Він має бути реалізований інтуїтивно зрозумілим для користувача – непрофесіонала в комп’ютерній галузі.

При розробці графічного інтерфейсу за мету має визначатися створення функціонального, проте простого, неперенавантаженого зайвою інформацією, інтерфейсу. Інтерфейс, створений за такими принципами сприятиме зручному використанню програми, та підвищить конкурентоздатність системи.

В процесі розробки графічного користувацького інтерфейсу перевага має віддаватися спокійним, однотонним відтінкам, які привертають увагу користувача. Для полегшення процесу візуального сприйняття, основні елементи управління можуть мати закруглені кути, так як вони не просто приємні для погляду, а й полегшують сприйняття графіки та обробку інформації людиною.

* 1. Alert діалоги

Alert діалог надає можливість створення звичайних повідомлень або повідомлень про помилки у вигляді окремих екранів.

Використання класу Alert в мобільних додатках обумовлено виникненням різних позаштатних ситуацій. В основному Alert діалоги застосовуються для створення екрану, який інформує користувача про помилку, що сталася в додатку або будь-якому іншому повідомленні інформаційного, характеру. Екран, визначений класом Alert може містити строкове повідомлення про помилку, що сталася, або текстовий рядок із заданим зображенням.

* 1. Toast повідомлення

Спливаюче повідомлення (Toast Notification) є повідомленням, яке з’являється на поверхні вікна програми, заповнюючи певну кількість простору, необхідного для повідомлення. При цьому поточна діяльність додатка залишається працездатною для користувача. Протягом декількох секунд повідомлення плавно закривається.

Спливаюче повідомлення також може бути створено службою, яка працює у фоновому режимі. Як правило, спливаюче повідомлення використовується для показу коротких текстових повідомлень.

1. Результати виконання роботи

Візуально зрозумілий та привабливий інтерфейс сприяє більш приємній та ефективній роботі з будь-якою програмою. Він має бути реалізований інтуїтивно зрозумілим для користувача.

* 1. Розробка графічного інтерфейсу користувача в середовищі Android Studio

Для розробки графічного інтерфейсу мобільного додатку необхідно відкрити XML файл з відповідним макетом інтерфейсу в створеному проекті.

На рисунку 4.1 наведено зображення інтерфейсу розробки графічного інтерфейсу користувача (представлення з файла «activity\_main.xml»).

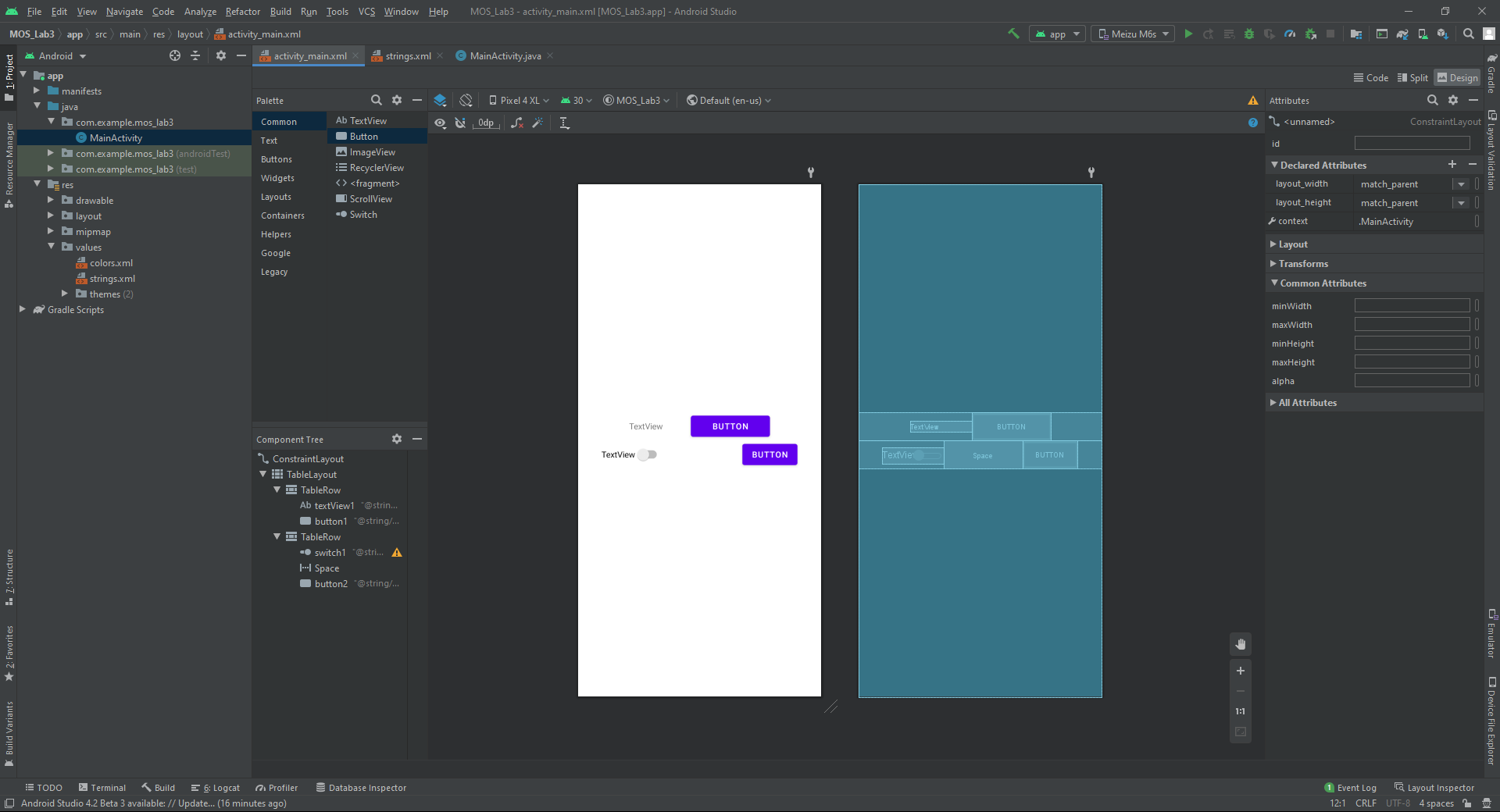


Рисунок 4.1 – Інтерфейс розробки графічного інтерфейсу користувача

Якщо проводити аналогію з Windows, то додаток складається з вікон, які називаються Activity. У конкретний момент часу зазвичай відображається одне Activity і займає весь екран, а додаток перемикається між ними. Як приклад можна розглянути поштовий додаток. У ньому одне Activity – це список листів, друге – перегляд листів, третє – налаштування ящика. При роботі користувач перемикається між ними.

Вміст Activity формується з різних компонентів, які називають View та віджети (рис. 4.2). Віджет – це об’єкт View, який служить інтерфейсом для взаємодії з користувачем. Прості об’єкти View є елементами управління (кнопки, текстові поля, прапорці, перемикачі, списки тощо) за допомогою яких користувач взаємодіє з програмою. Більшість візуальних елементів, що успадковано від класу View, розташовується в пакеті android.widget.

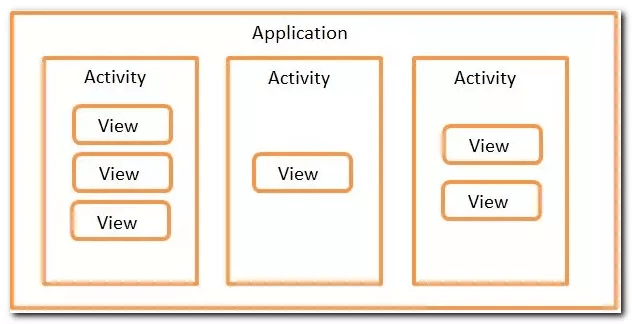


Рисунок 4.2 – Структура Android додатку

Необхідно зауважити, що зазвичай графічний інтерфейс користувача являє собою ієрархію об’єктів android.view.View та android.view.ViewGroup (рис. 4.3). Кожен об’єкт ViewGroup представляє собою контейнер, який містить і впорядковує дочірні об’єкти View.

Найпоширеніший приклад ViewGroup – це Layout. Layout буває різних типів і відповідає за те, як будуть розташовані його дочірні елементи на екрані (таблицею, рядком, стовпцем тощо).

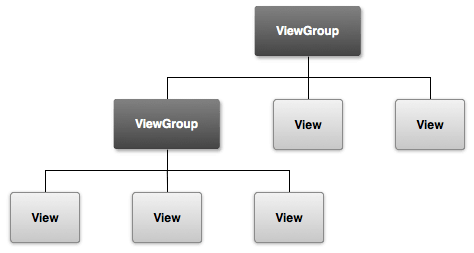


Рисунок 4.3 – Ієрархія об’єктів графічного інтерфейсу користувача

При визначенні інтерфейсу є три стратегії:

* Створити елементи управління програмно в коді Java;
* Оголосити елементи інтерфейсу в XML файлі;
* Додати елементи за допомогою графічного інтерфейсу розробки.

В панелі «Palette» середовища Android Studio (рис. 4.4) розташовано всі компоненти користувацького інтерфейсу які можна додати до мобільного додатку (список можна розширити, завантаживши необхідні додаткові елементи). Панель складається з таких вкладок: Common (найчастіше використовувані), Text (текстові елементи), Buttons (різні види кнопок), Widgets (різні віджети), Layout (макети розташування елементів) тощо.

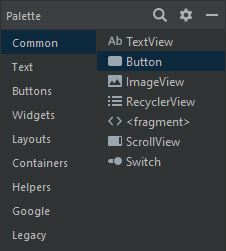


Рисунок 4.4 – Панель «Palette» в середовищі Android Studio

В панелі «Component Tree» представлена ієрархія View компонентів екрану. На рисунку 4.5 наведено зображення ієрархії з корневим елементом ConstraintLayout, в який вкладено TableLayout з дочірніми елементами.

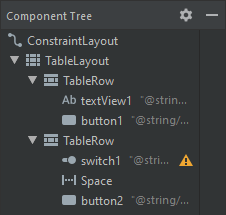


Рисунок 4.5 – Панель «Component Tree» в Android Studio

В панелі «Attributes» середовища Android Studio (рис. 4.6) наведено всі доступні параметри вибраного компонента View інтерфейсу користувача. Встановлюючи різні значення параметрів можна керувати виглядом та поведінкою вибраного елемента.

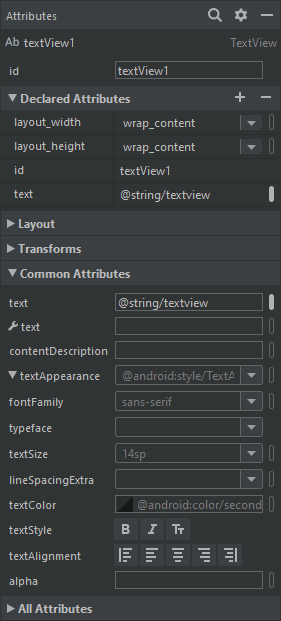


Рисунок 4.6 – Панель «Attributes» в середовищі Android Studio

* 1. Прив’язка до дій над елементами керування

Яскравим прикладом елемента керування є кнопка. Кнопка – це один з найпоширеніших елементів управління в програмуванні. Успадковується від TextView і є базовим класом для класу СompoundButton. Від класу CompoundButton в свою чергу успадковуються такі елементи як CheckBox, ToggleButton та RadioButton.

В Android для кнопки використовується клас android.widget.Button. На кнопці розташовується текст і на кнопку потрібно натиснути, щоб отримати результат.

У студії кнопка представлена компонентом Button в розділі Widgets. Управляти розміром шрифту, кольором тексту і іншими властивостями можна за допомогою атрибута textAppearance, який задіює системні стилі. Список, що випадає даного властивості містить величезний перелік варіантів. Також ви можете вручну задати конкретні індивідуальні настройки через окремі властивості.

Якщо кнопку (або інший елемент керування) було просто розміщено на екрані, то під час натискання нічого не станеться. Необхідно написати код, який буде виконуватися при натисканні. Існує кілька способів обробки натискань на кнопку.

* + 1. Перший спосіб – атрибут onClick

Відносно новий спосіб, спеціально розроблений для Android – використовувати атрибут onClick в коді XML представлення:

android:onClick="functionName"

Ім’я для події можна вибрати довільне. Далі потрібно прописати в класі активності придумане ім’я методу, який буде обробляти натискання. Метод повинен бути відкритим (public) і з одним параметром, що використовує об’єкт View. Сам метод необхідно помістити в клас активності:

public void functionName(View view) {

// Код обробки натискання кнопки

}

Коли користувач натисне на кнопку, то буде викликано метод functionName().

* + 1. Другий спосіб – метод setOnClickListener()

Більш традиційний спосіб в Java – за допомогою метода setOnClickListener(), який прослуховує натискання на кнопку.

Припустимо, що в інтерфейс додано кнопку з іменем buttonName. Тоді в коді класу активності оголосимо її звичайним способом:

Button buttonName = (Button) findViewById(R.id.buttonName);

Наступний крок – написання методу для обробки натискання. В даному куску кода до кнопки приєднується створювана функція яку буде викликано під час натискання кнопки:

buttonName.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

// Код обробки натискання кнопки

}

});

Як варіант, можна винести код для OnClickListener в окреме місце, це зручно, коли кнопок на екрані кілька і такий підхід дозволить упорядкувати код:

OnClickListener myButtonClickListener = new OnClickListener() {

@Override

public void onClick(View v) {

// Код обробки натискання кнопки

}

};

buttonName.setOnClickListener(myButtonClickListener);

* + 1. Третій спосіб – інтерфейс OnClickListener

Третій спосіб схожий на другий і також є традиційним для Java. Кнопка встановлює собі обробника за допомогою метода setOnClickListener (View.OnClickListener I). В такому випадку підійде будь-який об’єкт з інтерфейсом View.OnClickListener. Можна вказати, що клас Activity буде використовувати інтерфейс View.OnClickListener. Для цього імплементуємо OnClickListener до класу:

public class ClassName extends AppCompatActivity implements OnClickListener

Назва класу буде підкреслена хвилястою червоною рискою, натисніть слово public і дочекайтеся появи червоної лампочки, виберіть варіант «Implement methods». З’явиться діалогове вікно з виділеним методом onClick. Вибираємо його і в коді з’явиться заготовка для натискання кнопки:

@Override

public void onClick(View v) {

// Код обробки натискання кнопки

}

Метод буде реалізовано не в окремому об’єкті-обробнику, а в Activity, який і буде виступати оброблювачем. Залишається тільки в методі onCreate() призначити обробник кнопці. Це буде об’єкт this, тобто поточний об’єкт вибраної активності:

Button buttonName = (Button) findViewById(R.id.buttonName);

buttonName.setOnClickListener(this);

* + 1. Поради та коментарі

При використанні вище зазначених методів важливо перед виконанням коду імпортувати необхідні Java пакети.

Коли є одна кнопка у вікні, то буде один метод, дві кнопки – два метода і так далі. Якщо є декілька кнопок, то не обов’язково для кожної прописувати свій метод, можна обійтися і одним, а вже в самому методі розділяти код за ідентифікатором кнопки.

Якщо уважно подивитися на код в попередніх прикладах, то можна побачити, що в методі присутній параметр View, який і дозволяє визначити, для якої кнопки призначений шматок коду:

public void onMyButtonClick(View view)

{

switch(view.getId()) {

case R.id.buttonName1:

// Код обробки натискання кнопки buttonName1

break;

case R.id.buttonName2:

// Код обробки натискання кнопки buttonName2

break;

}

}

Іноді потрібно зробити кнопку недоступною і активувати її за певних умов. За допомогою XML не можна зробити кнопку недоступною (немає відповідного атрибута). Це можна зробити програмно за допомогою метода setEnabled():

buttonName.setEnabled(false);

Окрім обробників натискання Click існують інші обробники як для кнопки та інших елементів керування.

* + 1. Приклади інших обробників подій

Якщо уважно подивитися за поведінкою кнопки, то можна побачити, що код спрацьовує в той момент, коли користувач відпускає палець з кнопки. Для звичайних додатків це цілком нормально, але для ігор на швидкість такий підхід може виявитися занадто повільним. В таких випадках краще обробляти ситуацію не з натисканням кнопки, а з її торканням.

В Android є відповідний слухач OnTouchListener():

Button buttonName = (Button) findViewById(R.id.buttonName);

buttonName.setOnTouchListener(new OnTouchListener() {

@Override

public boolean onTouch(View v, MotionEvent event) {

if (event.getAction() == MotionEvent.ACTION\_DOWN) {

// Код обробки натискання кнопки buttonName

}

return false;

}

});

У метода onTouch() є параметр MotionEvent, який дозволяє більш тонко визначати торкання екрана. Якщо станеться подія, коли користувач торкнувся екрану, то йому буде відповідати константа ACTION\_DOWN. Відповідно, якщо користувач прибере палець, то потрібно використовувати константу ACTION\_UP.

Для елемента EditText існує свій обробник TextChangedListener():

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

EditTextName.addTextChangedListener(new TextWatcher() {

@Override

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

// Код, що виконається перед зміною значення в полі вводу

}

@Override

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

// Код, що виконається під час зміни значення в полі вводу

}

@Override

public void afterTextChanged(Editable s) {

// Код, що виконається після зміни значення в полі вводу

}

});

Для всіх елементів керування можна знайти різні обробники подій згідно зі специфікацією елементів.

* 1. Alert діалоги та Toast повідомлення

Для створення Toast повідомлення необхідно ініціалізувати об’єкт Toast за допомогою метода Toast.makeText(), а потім викликати метод show() для відображення на екрані:

Toast toast = Toast.makeText(getApplicationContext(), "повідомлення", Toast.LENGTH\_SHORT);

toast.show();

Для створення Alert діалогу необхідно створити об’єкт класу AlertDialog.Builder, передавши як параметр посилання на активність. Потім, використовуючи методи класу Builder спроектувати структуру діалогового вікна. За допомогою метода AlertDialog.Builder.create() створити об’єкт класу AlertDialog й викликати метод show() для відображення на екрані:

AlertDialog.Builder builder = new AlertDialog.Builder(this);

builder.setMessage("Повідомлення")

.setCancelable(false)

.setPositiveButton("Так", new DialogInterface.OnClickListener() {

public void onClick(DialogInterface dialog, int id) {

// Код обробки натискання кнопки «Так»

cancel();

}

})

.setNegativeButton("Ні", new DialogInterface.OnClickListener() {

public void onClick(DialogInterface dialog, int id) {

// Код обробки натискання кнопки «Ні»

cancel();

}

});

AlertDialog alert = builder.create();

alert.show();

* 1. Розробка та демонстрація мобільного додатку

В якості кольорової палети для розробки інтерфейсу було використано Flat кольори, які зазвичай використовуються для розробки приємного для погляду Flat дизайна (рис. 4.7).



Рисунок 4.7 – Одна з багатьох палет Flat кольорів

В результаті компонування елементів керування (кнопки, поля для вводу та відображення тексту) в контейнерах Layout було отримано ієрархію елементів, яку зображено на рисунку 4.8.

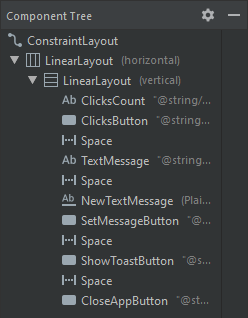


Рисунок 4.8 – Ієрархія View компонентів розробленого додатку

На рисунку 4.9 наведено зображення розробленого інтерфейсу додатку в середовищі Android Studio.

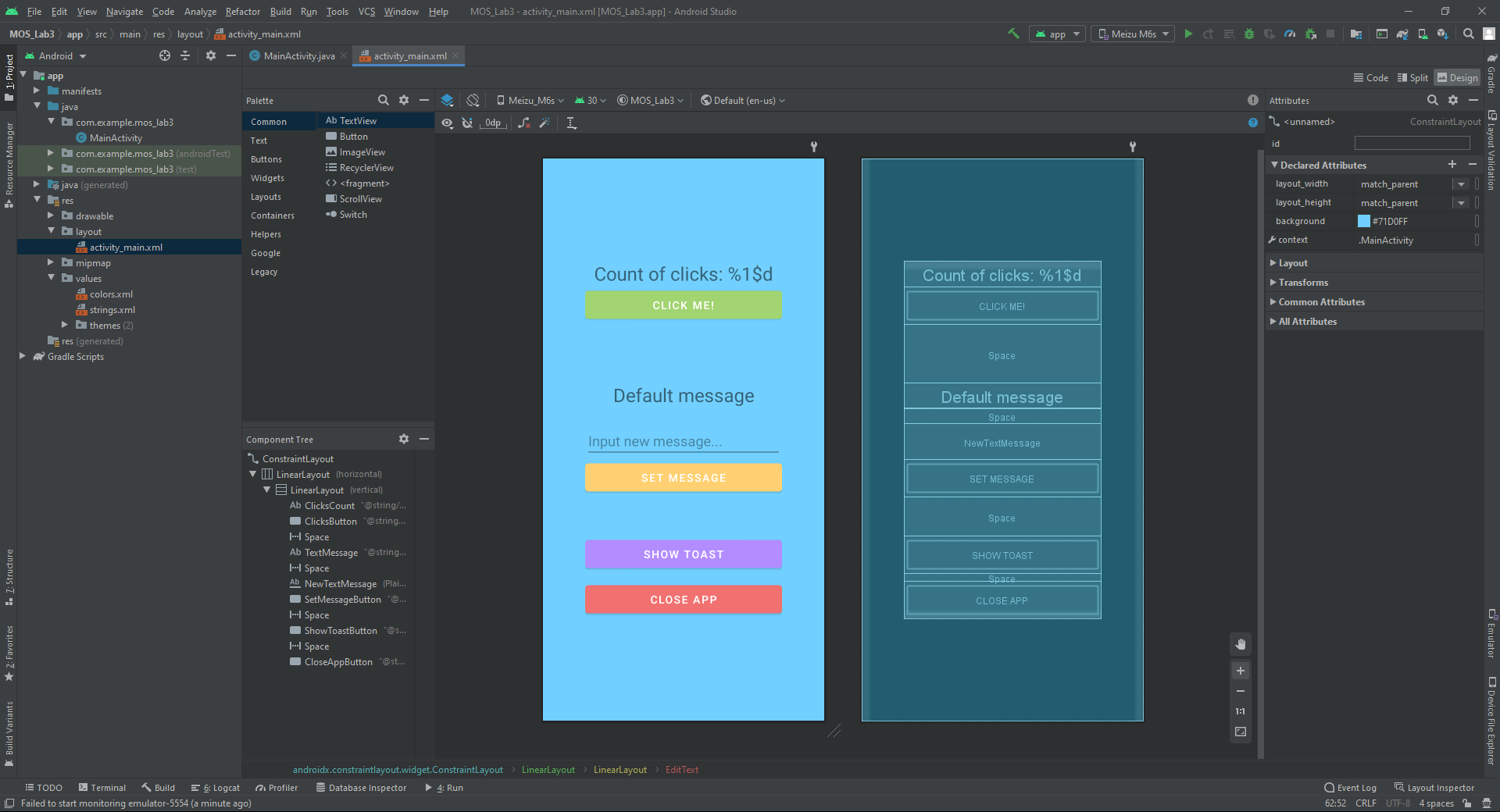


Рисунок 4.9 – Розроблений інтерфейс додатку в середовищі Android Studio

Після успішної компіляції мобільний додаток запущено в емуляторі Android пристрою (рис. 4.10).

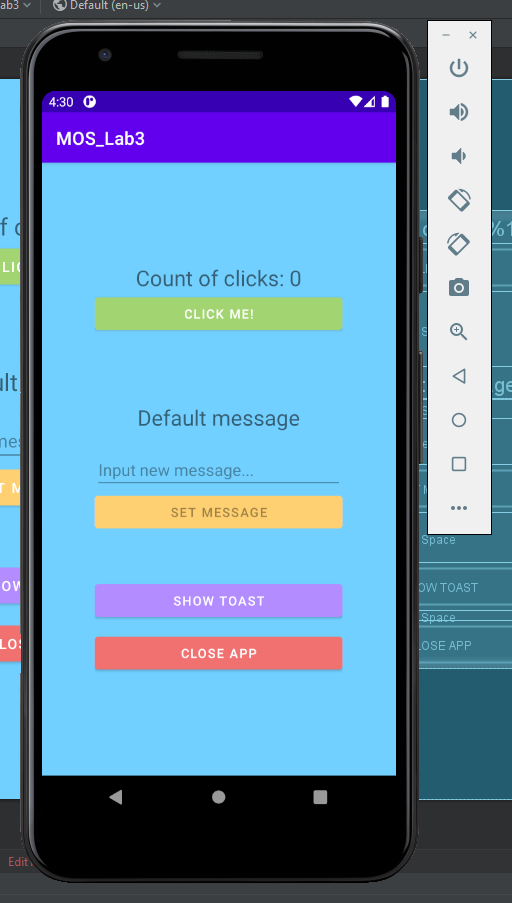


Рисунок 4.10 – Розроблений мобільний додаток запущений в емуляторі

При натисканні на кнопку «Click me!» відбувається інкрементація значення в полі «Count of clicks» на одиницю (рис. 4.11).

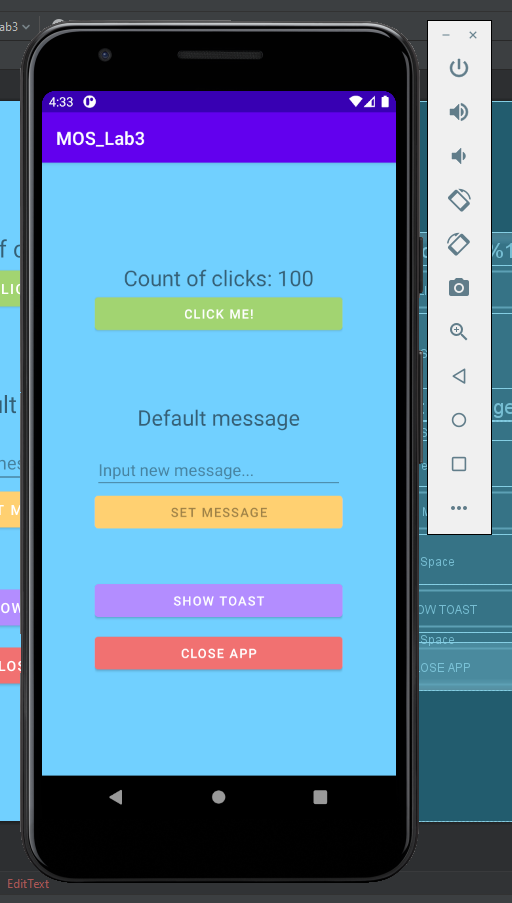


Рисунок 4.11 – Демонстрація натискання на кнопку «Click me!»

В поле вводу можна ввести текст як з комп’ютера так і за допомогою клавіатури пристрою в емуляторі (рис. 4.12).

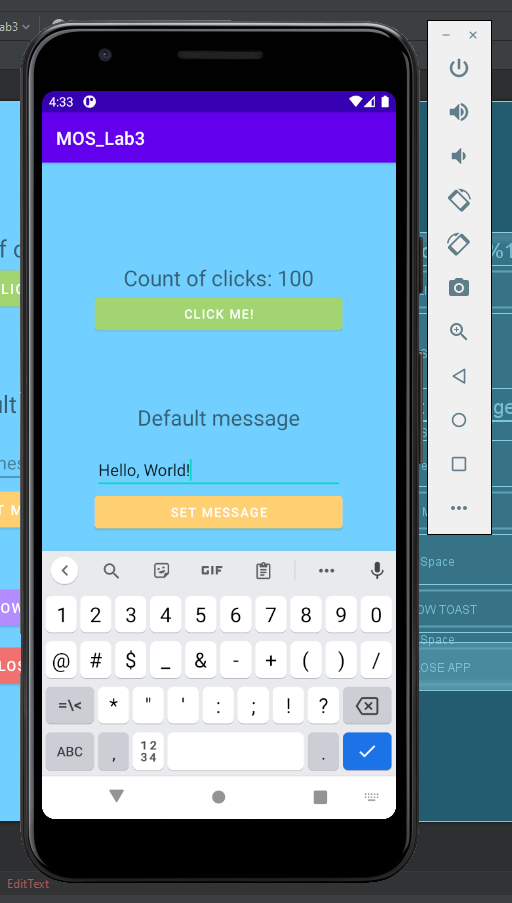


Рисунок 4.12 – Введення тексту з клавіатури в поле вводу

При натисканні на кнопку «Set message» пристрій запам’ятовує значення з поля вводу (рис. 4.13). Кнопка «Set message» стає недоступною (сірий колір та невідповідає на клік) при невалідному значенні в полі вводу.

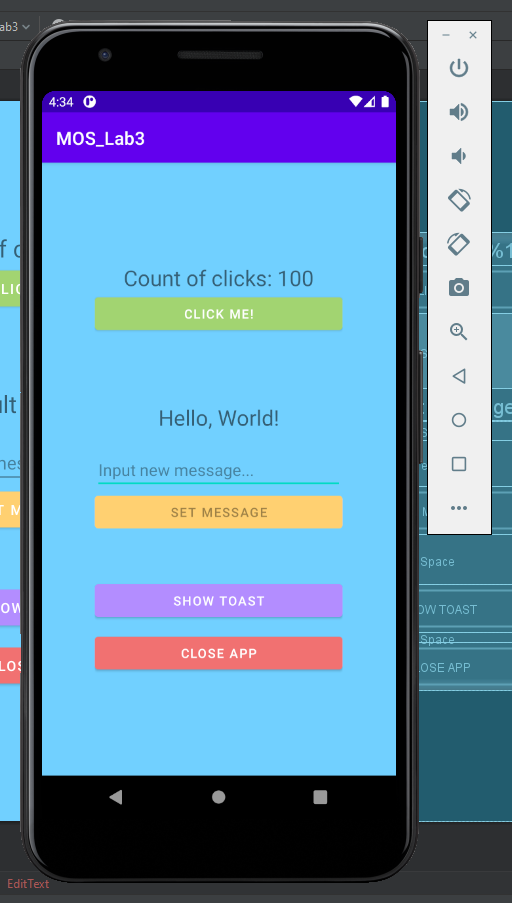


Рисунок 4.13 – Демонстрація натискання на кнопку «Set message»

При натисканні на кнопку «Show Toast» з’являється Toast повідомлення на екрані пристрою (рис. 4.14).

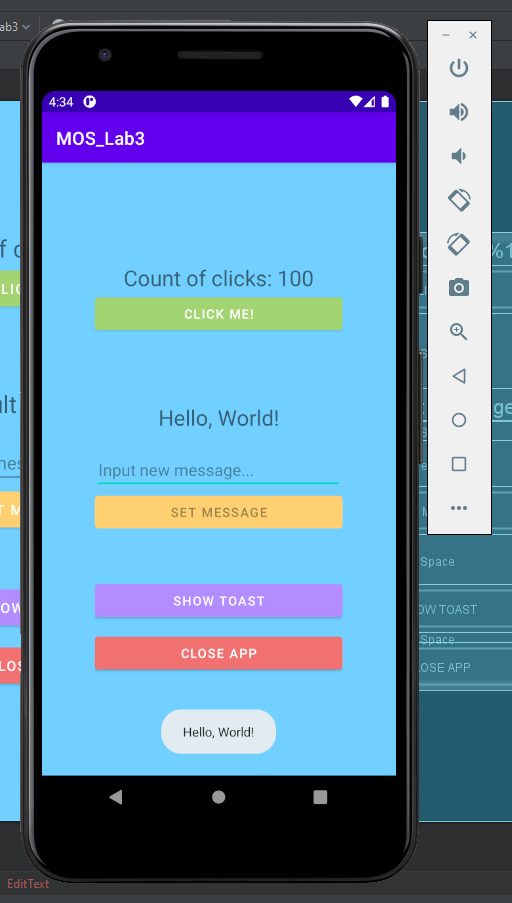


Рисунок 4.14 – Відображення Toast повідомлення в додатку

При натисканні на кнопку «Close app» з’являється Alert діалог з кнопками «Yes» та «No». При натисканні на кнопку «Yes» відбувається закриття додатку, а при натисканні на кнопку «No» Alert діалог закривається й відновлюється робота додатку (рис. 4.15).

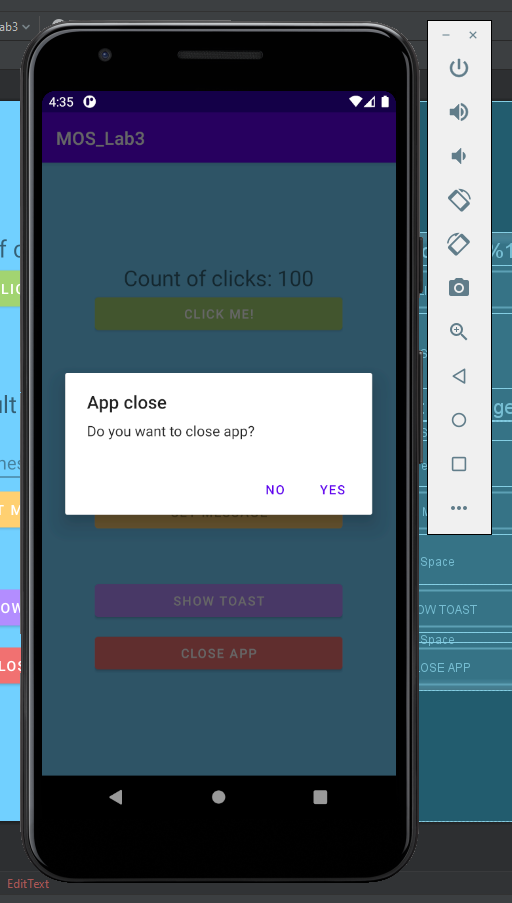


Рисунок 4.15 – Відображення Alert діалогу в додатку

В пунктах 4.4.1, 4.4.2, 4.4.3 та 4.4.4 наведено зміст основних файлів програми.

* + 1. AndroidManifest.xml

<?xml version="1.0" encoding="utf-8"?>  
<manifest xmlns:android="http://schemas.android.com/apk/res/android"  
 package="com.example.mos\_lab3">  
  
 <application  
 android:allowBackup="true"  
 android:icon="@mipmap/ic\_launcher"  
 android:label="@string/app\_name"  
 android:roundIcon="@mipmap/ic\_launcher\_round"  
 android:supportsRtl="true"  
 android:theme="@style/Theme.MOS\_Lab3">  
 <activity android:name=".MainActivity">  
 <intent-filter>  
 <action android:name="android.intent.action.MAIN" />  
  
 <category android:name="android.intent.category.LAUNCHER" />  
 </intent-filter>  
 </activity>  
 </application>  
  
</manifest>

* + 1. MainActivity.java

package com.example.mos\_lab3;  
  
import androidx.appcompat.app.AlertDialog;  
import androidx.appcompat.app.AppCompatActivity;  
  
import android.content.DialogInterface;  
import android.os.Bundle;  
import android.text.Editable;  
import android.text.TextWatcher;  
import android.view.MotionEvent;  
import android.view.View;  
import android.widget.Button;  
import android.widget.EditText;  
import android.widget.TextView;  
import android.widget.Toast;  
  
public class MainActivity extends AppCompatActivity {  
  
 int Clicks = 0;  
  
 TextView ClicksCount;  
 Button ClicksButton;  
 TextView TextMessage;  
 EditText NewTextMessage;  
 Button SetMessageButton;  
 Button ShowToastButton;  
 Button CloseAppButton;  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.activity\_main);  
  
 initElements();  
  
 ClicksCount.setText(getString(R.string.count\_of\_clicks, Clicks));  
 TextMessage.setText(getString(R.string.default\_message));  
 NewTextMessage.setText("");  
 SetMessageButton.setEnabled(false);  
  
 setClickListeners();  
 }  
  
 protected void initElements()  
 {  
 ClicksCount = (TextView) findViewById(R.id.ClicksCount);  
 ClicksButton = (Button) findViewById(R.id.ClicksButton);  
 TextMessage = (TextView) findViewById(R.id.TextMessage);  
 NewTextMessage = (EditText) findViewById(R.id.NewTextMessage);  
 SetMessageButton = (Button) findViewById(R.id.SetMessageButton);  
 ShowToastButton = (Button) findViewById(R.id.ShowToastButton);  
 CloseAppButton = (Button) findViewById(R.id.CloseAppButton);  
 }  
  
 protected void setClickListeners()  
 {  
 ClicksButton.setOnTouchListener(new View.OnTouchListener() {  
 @Override  
 public boolean onTouch(View v, MotionEvent event) {  
 if (event.getAction() == MotionEvent.ACTION\_DOWN) {  
 incrementClicks();  
 }  
 return false;  
 }  
 });  
  
 NewTextMessage.addTextChangedListener(new TextWatcher() {  
 @Override  
 public void beforeTextChanged(CharSequence s, int start, int count, int after) {  
 ;  
 }  
  
 @Override  
 public void onTextChanged(CharSequence s, int start, int before, int count) {  
 ;  
 }  
  
 @Override  
 public void afterTextChanged(Editable s) {  
 SetMessageButton.setEnabled(s.toString().length() != 0 && !s.toString().equals(TextMessage.getText().toString()));  
 }  
 });  
  
 SetMessageButton.setOnClickListener(buttonClickListener);  
 ShowToastButton.setOnClickListener(buttonClickListener);  
 CloseAppButton.setOnClickListener(buttonClickListener);  
 }  
  
 View.OnClickListener buttonClickListener = new View.OnClickListener() {  
 @Override  
 public void onClick(View view) {  
 switch(view.getId()) {  
 case R.id.SetMessageButton:  
 setNewMessage();  
 break;  
 case R.id.ShowToastButton:  
 showToastMessage();  
 break;  
 case R.id.CloseAppButton:  
 showAlertDialog();  
 break;  
 }  
 }  
 };  
  
 protected void incrementClicks()  
 {  
 Clicks += 1;  
 ClicksCount.setText(getString(R.string.count\_of\_clicks, Clicks));  
 }  
  
 protected void setNewMessage()  
 {  
 if(NewTextMessage.getText().toString().length() != 0) {  
 TextMessage.setText(NewTextMessage.getText().toString());  
 NewTextMessage.setText("");  
 }  
 }  
  
 protected void showToastMessage()  
 {  
 Toast.makeText(this, TextMessage.getText().toString(), Toast.LENGTH\_SHORT).show();  
 }  
  
 protected void showAlertDialog()  
 {  
 AlertDialog.Builder builder = new AlertDialog.Builder(this);  
 builder.setMessage("Do you want to close app?")  
 .setCancelable(false)  
 .setPositiveButton("Yes", new DialogInterface.OnClickListener() {  
 public void onClick(DialogInterface dialog, int id) {  
 finish();  
 }  
 })  
 .setNegativeButton("No", new DialogInterface.OnClickListener() {  
 public void onClick(DialogInterface dialog, int id) {  
 dialog.cancel();  
 }  
 });  
 AlertDialog alert = builder.create();  
 alert.setTitle("App close");  
 alert.show();  
 }  
}

* + 1. activity\_main.xml

<?xml version="1.0" encoding="utf-8"?>  
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:background="#71D0FF"  
 tools:context=".MainActivity">  
  
 <LinearLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:baselineAligned="false"  
 android:gravity="center"  
 android:orientation="horizontal"  
 android:weightSum="10">  
  
 <LinearLayout  
 android:layout\_width="0dp"  
 android:layout\_height="wrap\_content"  
 android:layout\_weight="7"  
 android:orientation="vertical">  
  
 <TextView  
 android:id="@+id/ClicksCount"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:text="@string/count\_of\_clicks"  
 android:textAlignment="center"  
 android:textSize="24sp" />  
  
 <Button  
 android:id="@+id/ClicksButton"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:text="@string/click\_me"  
 app:backgroundTint="#A2D471" />  
  
 <Space  
 android:layout\_width="match\_parent"  
 android:layout\_height="75dp" />  
  
 <TextView  
 android:id="@+id/TextMessage"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:text="@string/default\_message"  
 android:textAlignment="center"  
 android:textSize="24sp" />  
  
 <Space  
 android:layout\_width="match\_parent"  
 android:layout\_height="20dp" />  
  
 <EditText  
 android:id="@+id/NewTextMessage"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:autofillHints=""  
 android:ems="10"  
 android:hint="@string/input\_new\_message"  
 android:inputType="textPersonName"/>  
  
 <Button  
 android:id="@+id/SetMessageButton"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:text="@string/set\_message"  
 app:backgroundTint="#FFD071" />  
  
 <Space  
 android:layout\_width="match\_parent"  
 android:layout\_height="50dp" />  
  
 <Button  
 android:id="@+id/ShowToastButton"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:text="@string/show\_toast"  
 app:backgroundTint="#B38DFF" />  
  
 <Space  
 android:layout\_width="match\_parent"  
 android:layout\_height="10dp" />  
  
 <Button  
 android:id="@+id/CloseAppButton"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:text="@string/close\_app"  
 app:backgroundTint="#F17171" />  
 </LinearLayout>  
 </LinearLayout>  
  
</androidx.constraintlayout.widget.ConstraintLayout>

* + 1. strings.xml

<resources>  
 <string name="app\_name">MOS\_Lab3</string>  
 <string name="count\_of\_clicks">Count of clicks: %1$d</string>  
 <string name="click\_me">Click me!</string>  
 <string name="default\_message">Default message</string>  
 <string name="input\_new\_message">Input new message...</string>  
 <string name="set\_message">Set message</string>  
 <string name="show\_toast">Show Toast</string>  
 <string name="close\_app">Close app</string>  
</resources>

1. Відповіді на контрольні запитання
   1. Дайте визначенн інтерфейсу користувача

Інтерфейс користувача або user interface – це засіб зручної взаємодії користувача з інформаційною системою. Сукупність засобів для обробки та відбиття інформації, якнайбільше пристосованих для зручності користувача; у графічних системах інтерфейс користувача, втілюється багатовіконним режимом, змінами кольору, розміру, видимості вікон, їхнім розташуванням, сортуванням елементів вікон, гнучкими налаштуваннями як самих вікон, так і окремих їх елементів, доступністю багатокористувацьких налаштувань.

* 1. Які основні класи ви знаєте для створення інтерфейсу користувача?

Серед основних класів для створення інтерфейсу користувача в інтегрованому середовищі розробки Android Studio можна виділити такі: Button (кнопка), CheckBox (кнопка з відміткою), RadioButton (радіокнопка), Switch (перемикач), TextView (поле відображення текста), EditText (поле вводу текста), LinearLayout (контейнер для елементів), Space (вільний простір).

* 1. Опишіть структуру Android додатку

Android додаток складається з вікон, які називаються Activity. У конкретний момент часу зазвичай відображається одне Activity і займає весь екран.

Вміст Activity формується з різних компонентів, які називають View та ViewGroup. Кожен об’єкт ViewGroup представляє собою контейнер, який містить і впорядковує дочірні об’єкти View. Прості об’єкти View є елементами управління (кнопки, текстові поля, прапорці, перемикачі, списки тощо) за допомогою яких користувач взаємодіє з програмою.

* 1. Поясніть відмінність між View та ViewGroup елементами інтерфейсу

Елементи інтерфейсу View – це прості об’єкти за допомогою яких користувач взаємодіє з програмою. Приклад елементів View: Button, TextView, EditText тощо.

Елементи ViewGroup – це контейнери, які містять дочірні прості елементи або інші контейнери та використовуються для організації впорядкованого відображення елементів. Приклад елементів ViewGroup: LinearLayout, TableLayout, Grid тощо.

* 1. Поясніть відмінність між Toast повідомленнями та Alert діалогами

Використання Alert діалогів в мобільних додатках обумовлено виникненням різних позаштатних ситуацій. При відображенні Alert діалогу призупиняється робота додатку, відкривається нове вікно й очікується відклик від користувача.

На відміну від Alert діалогу, Toast повідомлення з’являється на поверхні вікна програми. При цьому поточна діяльність додатка залишається працездатною, а саме повідомлення плавно закривається протягом декількох секунд і не вимагає жодної активності з боку користувача.

Висновки

Протягом виконання цієї лабораторної роботи ознайомився із механізмами роботи мобільних застосунків написаних в інтегрованому середовищі розробки Android Studio.

Розглянув різні підходи та засоби прив’язки до дій над елементами керування.

Розробив мобільний застосунок, що реалізує інтерфейс користувача з кнопками і полями вводу та використовує Alert діалоги та Toast повідомлення.