Лабораторна робота 5.

Доступ та керування елементами екрана з коду. Опрацювання подій. ViewBinding.

**Завдання**

* Створити проект застосунку з простим функціоналом та ViewBinding.
* Додати обробку кліку елемента та будь-яку дію по кліку, яку видно при роботі застосунку.
* Додати різну обробку кліку для декількох кнопок або декількох
* візульних елементів.
* Додати обробку будь-якої іншої події (не кліку) для будь-якого візуального елемента.
* Додати обробку зміну тексту (before,after, onTextChanged).
* Винести обробку подій в інший, окремий клас.

**Хід роботи**

Створення проекту Android застосунку з простим функціоналом починається з відкриття Android Studio і вибору шаблону для нового проекту. При створенні проекту важливо ввімкнути ViewBinding, що дозволить спростити роботу з елементами інтерфейсу в коді. Після цього додаємо необхідні залежності в build.gradle файл для активації ViewBinding.

Далі переходимо до розробки інтерфейсу користувача. В activity\_main.xml розміщуємо кілька візуальних елементів, таких як кнопки, текстові поля та зображення. Наприклад, додаємо кнопку для зміни кольору тексту, кнопку для зміни числового значення, TextView для відображення тексту, ImageView, який можна перетягувати, та EditText для введення тексту.

Після налаштування інтерфейсу, переходимо до реалізації обробки подій в MainActivity. В MainActivity створюємо екземпляр класу ViewBinding для доступу до елементів інтерфейсу. Додаємо обробники кліків для кнопок, які змінюють колір тексту та числове значення. Наприклад, обробник кліку для кнопки зміни кольору може змінювати колір TextView при кожному натисканні.

Також реалізуємо обробку іншої події, наприклад, перетягування ImageView. Для цього встановлюємо обробник подій дотику для ImageView, який дозволяє користувачу перетягувати зображення по екрану. Це додає інтерактивності та покращує користувацький досвід.

Ще одне завдання – реалізація обробки зміни тексту в EditText. Додаємо TextWatcher до EditText, щоб відслідковувати зміни тексту до, під час та після введення користувачем. Це дозволяє динамічно реагувати на введення користувача, наприклад, відображати підказки або виконувати валідацію введених даних.

Для поліпшення структури коду та зручності обробки подій, виносимо обробку подій в окремий клас. Це дозволяє розділити логіку обробки подій та логіку активності, що сприяє кращій підтримці та розширюваності коду. Створюємо клас, який імплементує необхідні інтерфейси обробки подій, та використовуємо його екземпляр в MainActivity для делегування обробки подій.

Завершивши всі ці кроки, запускаємо проект на емуляторі або фізичному пристрої, щоб перевірити роботу всіх функцій. Переконуємося, що обробка кліків, перетягування та зміни тексту працюють належним чином і взаємодія з користувачем відбувається без помилок. Це дозволяє нам впевнитися, що всі завдання виконані правильно, і застосунок готовий до використання.

Повний код програми:

package com.lab\_5

import android.annotation.SuppressLint

import android.content.Context

import android.graphics.Color

import android.text.Editable

import android.text.TextWatcher

import android.util.Log

import android.view.MotionEvent

import android.widget.Button

import android.widget.EditText

import android.widget.ImageView

import android.widget.TextView

import android.widget.Toast

import kotlin.random.Random

class EventHandler {

fun changeColorOnClick(button: Button) {

button.setOnClickListener {

val randomColor = Color.argb(255, Random.nextInt(256), Random.nextInt(256), Random.nextInt(256))

button.setBackgroundColor(randomColor)

}

}

fun changeNumberOnClick(button: Button) {

button.setOnClickListener {

val randomNumber = Random.nextInt(10)

button.text = randomNumber.toString()

}

}

fun showMessageOnClick(textView: TextView, message: String, context: Context) {

textView.setOnClickListener {

// Створення і показ системного повідомлення (Toast)

Toast.makeText(context, message, Toast.LENGTH\_SHORT).show()

}

}

@SuppressLint("ClickableViewAccessibility")

fun makeImageDraggable(imageView: ImageView) {

imageView.setOnTouchListener { view, event ->

when (event.action) {

MotionEvent.ACTION\_DOWN -> {

Log.d("TouchEvent", "ACTION\_DOWN: ${event.rawX}, ${event.rawY}")

true

}

MotionEvent.ACTION\_MOVE -> {

Log.d("TouchEvent", "ACTION\_MOVE: ${event.rawX}, ${event.rawY}")

imageView.x = event.rawX - view.width / 2

imageView.y = event.rawY - view.height / 2

true

}

MotionEvent.ACTION\_UP -> {

Log.d("TouchEvent", "ACTION\_UP")

true

}

else -> false

}

}

}

fun handleEditTextChanges(editText: EditText, textView: TextView) {

editText.addTextChangedListener(object : TextWatcher {

override fun beforeTextChanged(s: CharSequence?, start: Int, count: Int, after: Int) {

// Операції, які виконуються перед зміною тексту

}

override fun onTextChanged(s: CharSequence?, start: Int, before: Int, count: Int) {

// Операції, які виконуються під час зміни тексту

val newText = s?.toString() ?: ""

textView.text = newText // Відображення тексту у TextView

}

override fun afterTextChanged(s: Editable?) {

// Операції, які виконуються після зміни тексту

}

})

}

}

package com.lab\_5

import android.os.Bundle

import android.text.Editable

import android.text.TextWatcher

import android.widget.Button

import android.widget.EditText

import android.widget.ImageView

import android.widget.TextView

import androidx.appcompat.app.AppCompatActivity

import com.lab\_5.databinding.ActivityMainBinding

class MainActivity : AppCompatActivity() {

private lateinit var binding: ActivityMainBinding

private val eventHandler = EventHandler()

override fun onCreate(savedInstanceState: Bundle?) {

super.onCreate(savedInstanceState)

binding = ActivityMainBinding.inflate(layoutInflater)

setContentView(binding.root)

val colorButton: Button = binding.colorButton

val changeNumberButton: Button = binding.changeNumberButton

val textView: TextView = binding.textView

val draggableImageView: ImageView = binding.draggableImageView

val editText: EditText = binding.editText

// Додайте обробку змін тексту у EditText

eventHandler.handleEditTextChanges(editText, textView)

// Обробка кліку для зміни коліру кнопки

eventHandler.changeColorOnClick(colorButton)

// Обробка кліку для зміни цифри на кнопці

eventHandler.changeNumberOnClick(changeNumberButton)

// Обробка кліку для відображення системного повідомлення

eventHandler.showMessageOnClick(textView, "Це текстове повідомлення!", this)

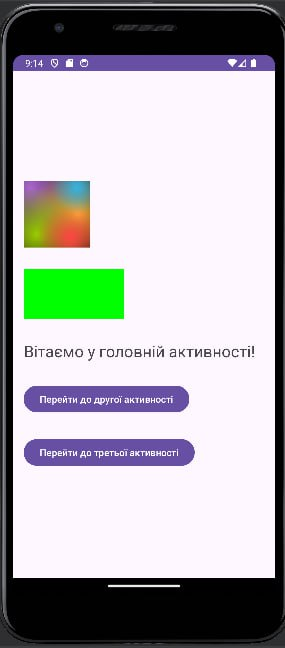
// Обробка переміщення тексту по тапу

eventHandler.makeImageDraggable(draggableImageView)

}

}

**Результати роботи**



Додаток після запуску



Друга активність



Третя активність