**Міністерство освіти і науки України**

**Національний технічний університет України**

**«Київський політехнічний інститут імені Ігоря Сікорського»**

**Факультет інформатики та обчислювальної техніки**

**Кафедра обчислювальної техніки**

**Лабораторна робота №6**

з дисципліни

«Об’єктно орієнтоване програмування»

на тему

“Побудування програмної системи з множини об’єктів,

керованих повідомленнями”

Виконав: Перевірив:

Студент групи ІМ-22 Порєв В.М.

Кушнір Микола Миколайович

номер у списку групи: 13

Київ 2023

**Мета**

Отримати вміння та навички використовувати засоби обміну інформацією та запрограмувати взаємодію незалежно працюючих програмних компонентів.

**Завдання**

**1.** Створити у середовищі MS Visual Studio C++ проект Win32 з ім’ям **Lab6**.

**2.** Написати вихідні тексти усіх програм-компонентів згідно варіанту завдання.

**3.** Скомпілювати вихідні тексти і отримати виконувані файли програм.

**4.** Перевірити роботу програм. Налагодити взаємодію програм.

**5.** Проаналізувати та прокоментувати результати та вихідні тексти програм.

**6.** Оформити звіт.

**Умови завдання за варіантом 1 *(Ж mod 4 = 13 mod 4 = 1)*:**

|  |  |  |
| --- | --- | --- |
| Програма **Lab6** | Програма **Object2** | Програма **Object3** |
| **1.** Користувач вводить значення ***n***, ***Min***, ***Max*** у діалоговому вікні  **2.** Програма викликає програми **Object2**, **Object3** і забезпечує обмін повідомленнями для передавання та отримання інформації | **1.** Створює матрицю ***n×n*** цілих (*int*) чисел у діапазоні ***Min – Max***  **2.** Показує числові значення у власному головному вікні  **3.** Записує дані в *Clipboard Windows* у текстовому форматі | **1.** Зчитує дані з *Clipboard Windows*  **2.** Відображає значення детермінанту матриці у власному головному вікні |

**Вихідні тексти файлів програм**

**Lab6.kt**

package com.oop.lab6  
  
import android.content.BroadcastReceiver  
import android.content.Context  
import android.content.Intent  
import android.content.IntentFilter  
import android.os.Bundle  
import android.os.Handler  
import android.os.Looper  
import android.widget.Button  
import android.widget.Toast  
import androidx.appcompat.app.AppCompatActivity  
class Lab6 : AppCompatActivity() {  
 private val object2PackageName = "com.oop.object2"  
 private val object3PackageName = "com.oop.object3"

private val object2SignalHandler = object : BroadcastReceiver() {  
 override fun onReceive(context: Context?, intent: Intent?) {  
 intent?.*let* **{** if (**it**.*action* == "OBJECT2\_SEND\_SIGNAL") {  
 val signal = **it**.getStringExtra("SIGNAL")  
 if (signal == "TASK\_END\_SUCCESS") {  
 val object3LaunchDelay = 200L  
 Handler(Looper.getMainLooper()).postDelayed(**{** launchAppWithSignal(object3PackageName, "START")  
 **}**, object3LaunchDelay)  
 } else {  
 showToast("Сталась помилка виконання $object2PackageName")  
 }  
 }  
 **}** }  
 }

override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 setContentView(R.layout.*main\_activity*)  
 val intentFilter = IntentFilter("OBJECT2\_SEND\_SIGNAL")  
 registerReceiver(object2SignalHandler, intentFilter, *RECEIVER\_EXPORTED*)

val btnCompleteTask: Button = findViewById(R.id.*complete\_task*)  
 btnCompleteTask.setOnClickListener **{** val dialog = Dialog()  
 dialog.setOnConfirmListener **{** data **->** launchAppWithData(object2PackageName, data)  
 **}** dialog.show(*supportFragmentManager*, "DIALOG")  
 **}** }

override fun onDestroy() {  
 super.onDestroy()  
 unregisterReceiver(object2SignalHandler)  
 }

private fun launchAppWithData(packageName: String, data: IntArray) {  
 *packageManager*.getLaunchIntentForPackage(packageName)?.*apply* **{** putExtra("DATA", data)  
 startActivity(this)  
 **}** ?: showToast("Програму $packageName не знайдено")  
 }

private fun launchAppWithSignal(packageName: String, signal: String) {  
 *packageManager*.getLaunchIntentForPackage(packageName)?.*apply* **{** putExtra("SIGNAL", signal)  
 startActivity(this)  
 **}** ?: showToast("Програму $packageName не знайдено")  
 }

private fun showToast(text: String) {  
 *with*(Toast(this)) **{** setText(text)  
 *duration* = Toast.*LENGTH\_SHORT* show()  
 **}** }  
}

**Dialog.kt** *(частина програми Lab6)*

package com.oop.lab6  
  
import android.app.AlertDialog  
import android.app.Dialog  
import android.os.Bundle  
import android.os.Handler  
import android.os.Looper  
import android.view.View  
import android.widget.Button  
import android.widget.EditText  
import android.widget.Toast  
import androidx.fragment.app.DialogFragment

class Dialog: DialogFragment() {  
 private lateinit var onConfirmListener: (IntArray) -> Unit

override fun onCreateDialog(savedInstanceState: Bundle?): Dialog {  
 val builder = AlertDialog.Builder(requireActivity())  
 builder.setTitle(R.string.*dialog\_header\_text*)  
 val view: View = *layoutInflater*.inflate(R.layout.*dialog*, null)  
 builder.setView(view)

val nEditText: EditText = view.findViewById(R.id.*enter\_n*)  
 val minEditText: EditText = view.findViewById(R.id.*enter\_min*)  
 val maxEditText: EditText = view.findViewById(R.id.*enter\_max*)  
 val btnCancel: Button = view.findViewById(R.id.*dialog\_cancel*)  
 btnCancel.setOnClickListener **{** dismiss() **}**

val btnOkay: Button = view.findViewById(R.id.*dialog\_okay*)  
 btnOkay.setOnClickListener **{** val n = nEditText.*text*.toString()  
 val min = minEditText.*text*.toString()  
 val max = maxEditText.*text*.toString()  
 val validationResult = validateInput(n, min, max)  
 if (validationResult != null) {  
 dismiss()  
 val delayToDismiss = 100L  
 Handler(Looper.getMainLooper()).postDelayed(**{** onConfirmListener(validationResult)  
 **}**, delayToDismiss)  
 }  
 **}**

return builder.create().*apply* **{** setCancelable(false)  
 setCanceledOnTouchOutside(false)  
 **}** }

private fun validateInput(n: String, min: String, max: String): IntArray? {  
 val parsedN = parseToInt("n", n) ?: return null  
 if (parsedN <= 0) {  
 showToast("n має бути більше 0")  
 return null  
 }  
 val parsedMin = parseToInt("Min", min) ?: return null  
 val parsedMax = parseToInt("Max", max) ?: return null  
 if (parsedMin > parsedMax) {  
 showToast("Min не може бути більшим за Max")  
 return null  
 }  
 return *intArrayOf*(parsedN, parsedMin, parsedMax)  
 }

private fun parseToInt(key: String, value: String): Int? {  
 return try {  
 value.*toInt*()  
 } catch (\_: Exception) {  
 if (value == "") showToast("Поле $key порожнє")  
 else showToast("$key не є числом")  
 null  
 }  
 }

private fun showToast(text: String) {  
 *with*(Toast(requireActivity())) **{** setText(text)  
 *duration* = Toast.*LENGTH\_SHORT* show()  
 **}** }

fun setOnConfirmListener (listener: (IntArray) -> Unit) {  
 onConfirmListener = listener  
 }  
}

**Object2.kt**

package com.oop.object2  
  
import android.content.ClipData  
import android.content.ClipboardManager  
import android.content.Intent  
import android.os.Bundle  
import android.widget.TextView  
import androidx.appcompat.app.AppCompatActivity  
import kotlin.math.abs  
  
class Object2 : AppCompatActivity() {  
 private lateinit var matrixTextView: TextView

override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 setContentView(R.layout.*main\_activity*)

matrixTextView = findViewById(R.id.*matrix*)  
 handleIntent()  
 }

override fun onNewIntent(intent: Intent?) {  
 super.onNewIntent(intent)  
 setIntent(intent)  
 handleIntent()  
 }

private fun handleIntent() {  
 val data = *intent*.getIntArrayExtra("DATA")  
 data?.*let* **{** val (n, min, max) = **it** var taskEndingStatus = 0  
 try {  
 val matrix = generateMatrix(n, min, max)  
 showMatrix(matrix)  
 val strMatrix = serializeMatrix(matrix)  
 writeToClipboard(strMatrix)  
 } catch (e: Exception) {  
 taskEndingStatus = 1  
 }  
 sendTaskEndingSignal(taskEndingStatus)  
 **}** }

private fun generateMatrix(n: Int, min: Int, max: Int): Array<IntArray> {  
 return Array(n) **{** IntArray(n) **{** (min..max).*random*()  
 **}  
 }** }

private fun showMatrix(matrix: Array<IntArray>) {  
 val str = StringBuilder()  
 val n = matrix.size  
 (0 *until* n).*forEach* **{** i **->** (0 *until* n - 1).*forEach* **{** j **->** val value = matrix[i][j]  
 val prefix = if (value >= 0) " " else ""  
 val spaces = when (*abs*(value)) {  
 in 0..9 -> " "  
 in 10..99 -> " "  
 in 100..999 -> " "  
 else -> " "  
 }  
 str.append("$prefix$value$spaces")  
 **}** str.append("${matrix[i][n - 1]}\n\n")  
 **}** matrixTextView.*text* = str.*dropLast*(2).toString()  
 }

private fun serializeMatrix(matrix: Array<IntArray>): String {  
 val str = StringBuilder()  
 val n = matrix.size  
 (0 *until* n).*forEach* **{** i **->** (0 *until* n - 1).*forEach* **{** j **->** str.append("${matrix[i][j]}\t")  
 **}** str.append("${matrix[i][n - 1]}\n")  
 **}** return str.*dropLast*(1).toString()  
 }

private fun writeToClipboard(data: String) {  
 val manager = getSystemService(*CLIPBOARD\_SERVICE*) as ClipboardManager  
 val clipData = ClipData.newPlainText("MATRIX", data)  
 manager.setPrimaryClip(clipData)  
 }

private fun sendTaskEndingSignal(status: Int) {  
 Intent("OBJECT2\_SEND\_SIGNAL").*apply* **{** putExtra("SIGNAL",  
 if (status == 0) "TASK\_END\_SUCCESS"  
 else "TASK\_END\_FAILURE"  
 )  
 sendBroadcast(this)  
 **}** }  
}

**Object3.kt**

package com.oop.object3  
  
import android.content.ClipboardManager  
import android.content.Intent  
import android.os.Bundle  
import android.widget.TextView  
import android.widget.Toast  
import androidx.appcompat.app.AppCompatActivity  
import java.lang.Math.abs  
import java.lang.Math.round  
  
class Object3 : AppCompatActivity() {  
 private lateinit var determinantTextView: TextView

private var wasNewIntent = true

override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 setContentView(R.layout.*main\_activity*)

determinantTextView = findViewById(R.id.*determinant*)  
 }

override fun onNewIntent(intent: Intent?) {  
 super.onNewIntent(intent)  
 setIntent(intent)  
 wasNewIntent = true  
 }

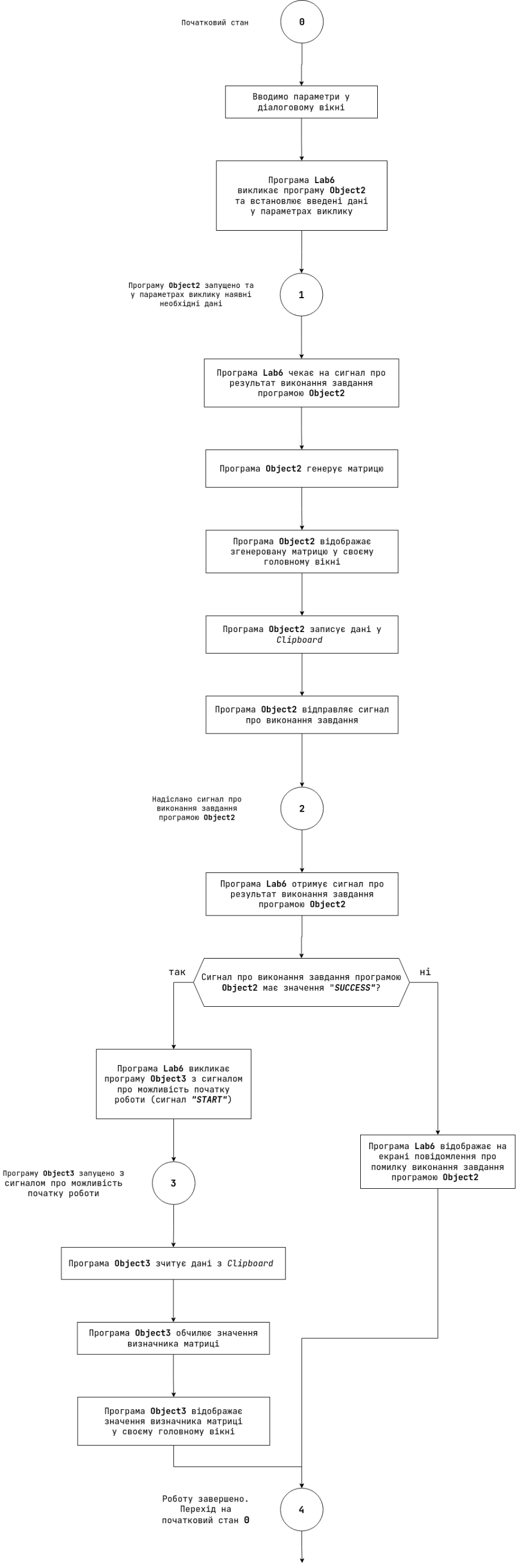
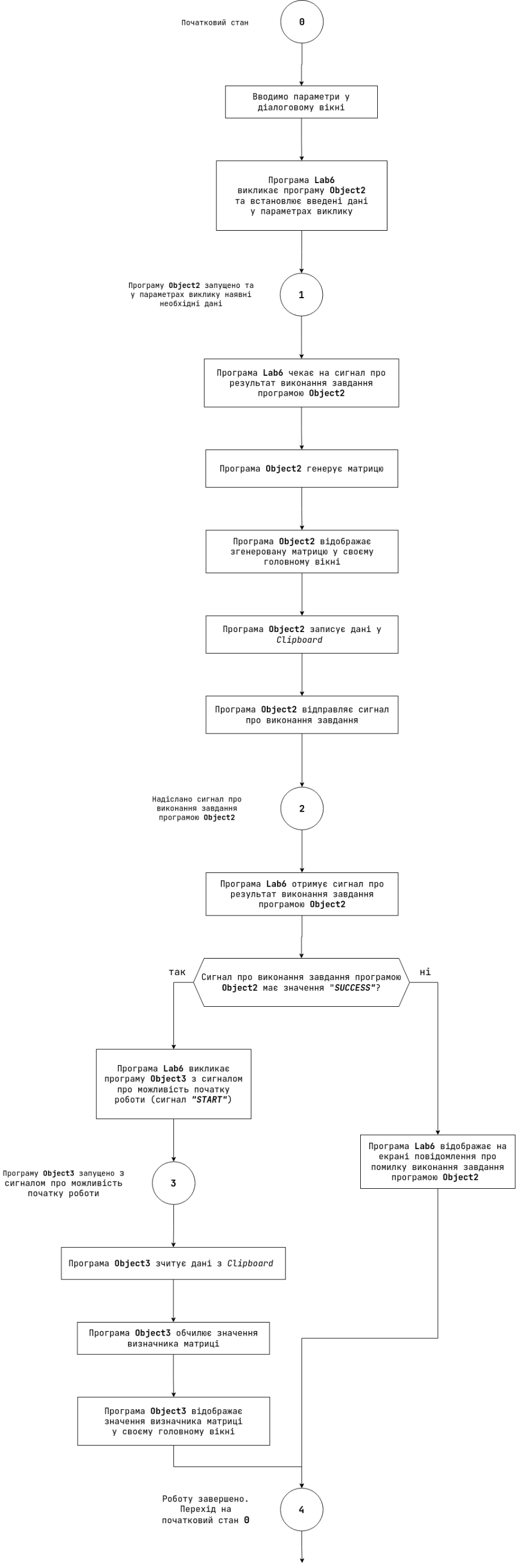
override fun onWindowFocusChanged(hasFocus: Boolean) {  
 super.onWindowFocusChanged(hasFocus)  
 if (hasFocus && wasNewIntent) {  
 if (intent.getStringExtra("SIGNAL") == "START") {  
 val data = readFromClipboard()  
 if (data != "") {  
 val matrix = deserializeMatrix(data)  
 val determinant = calculateDeterminant(matrix)  
 determinantTextView.*text* = determinant.toString()  
 } else {  
 with(Toast(this)) **{** setText("Виникла помилка читання з буфера обміну")  
 duration = Toast.*LENGTH\_LONG* show()  
 **}** }  
 }  
 wasNewIntent = false  
 }  
 }

private fun deserializeMatrix(str: String): Array<IntArray> {  
 val rows = str.split("\n")  
 val n = rows.size  
 val matrix = Array(n) **{** IntArray(n) **}** rows.forEachIndexed **{** i, row **->** val rowFields = row.split("\t")  
 rowFields.forEachIndexed **{** j, value **->** matrix[i][j] = value.toInt()  
 **}  
 }** return matrix  
 }

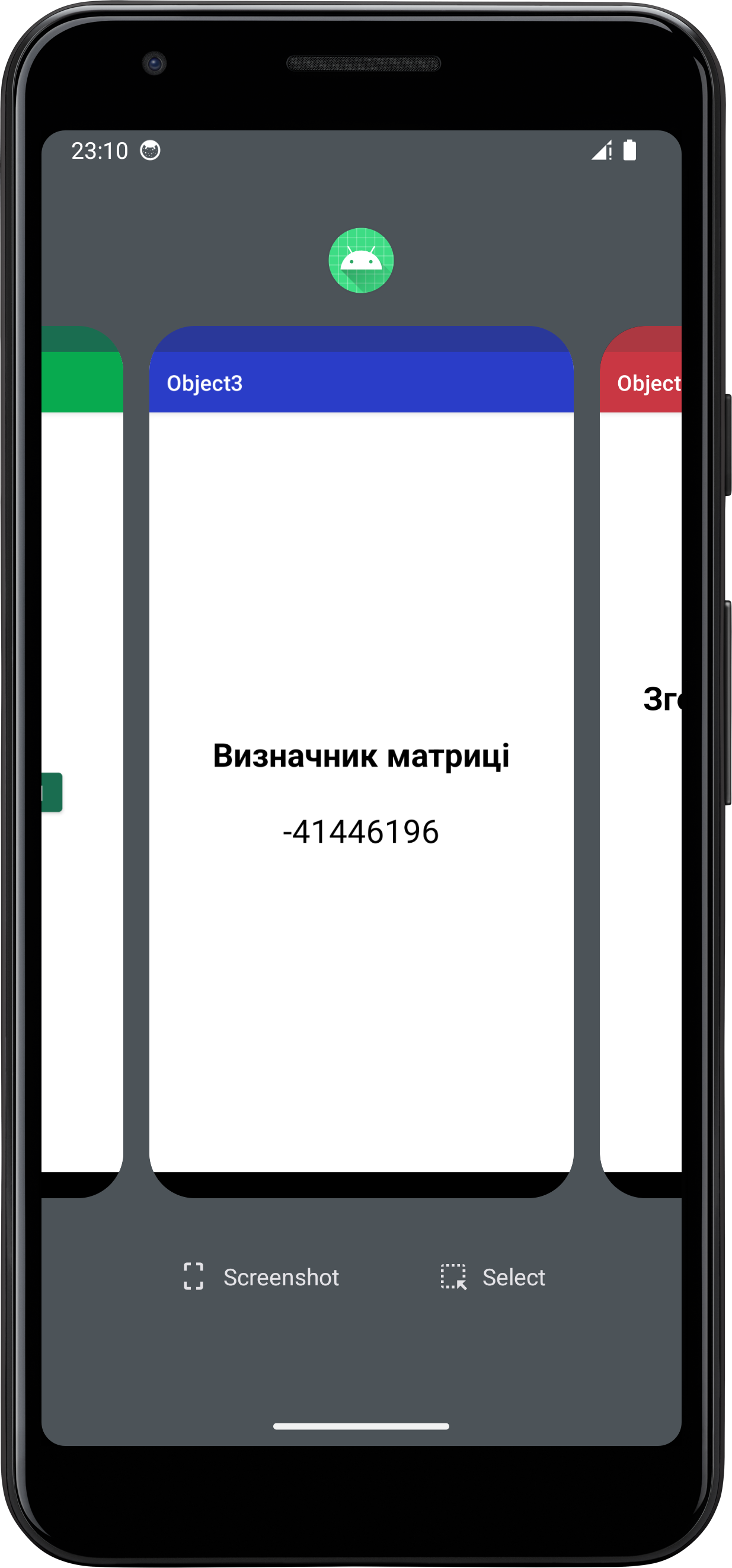
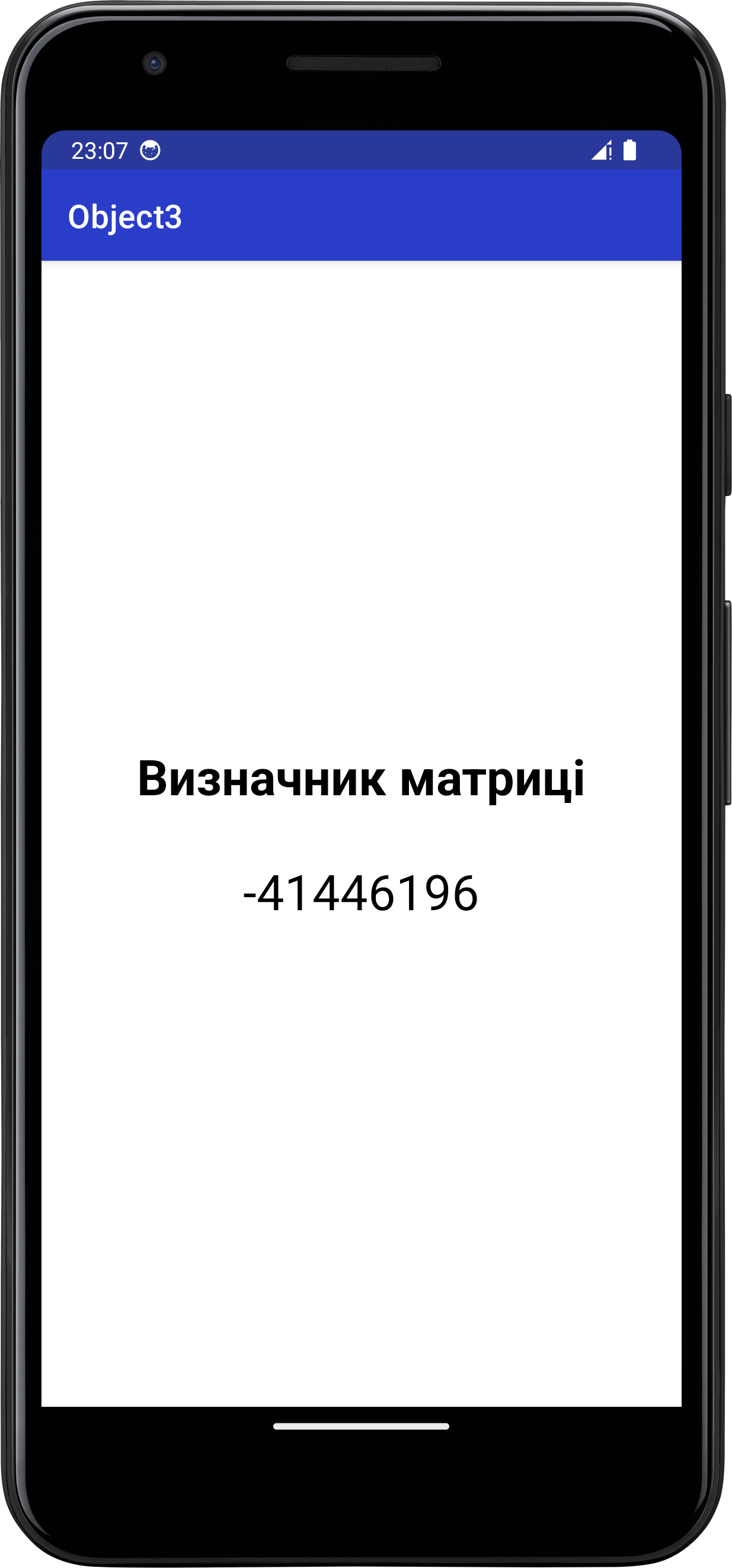
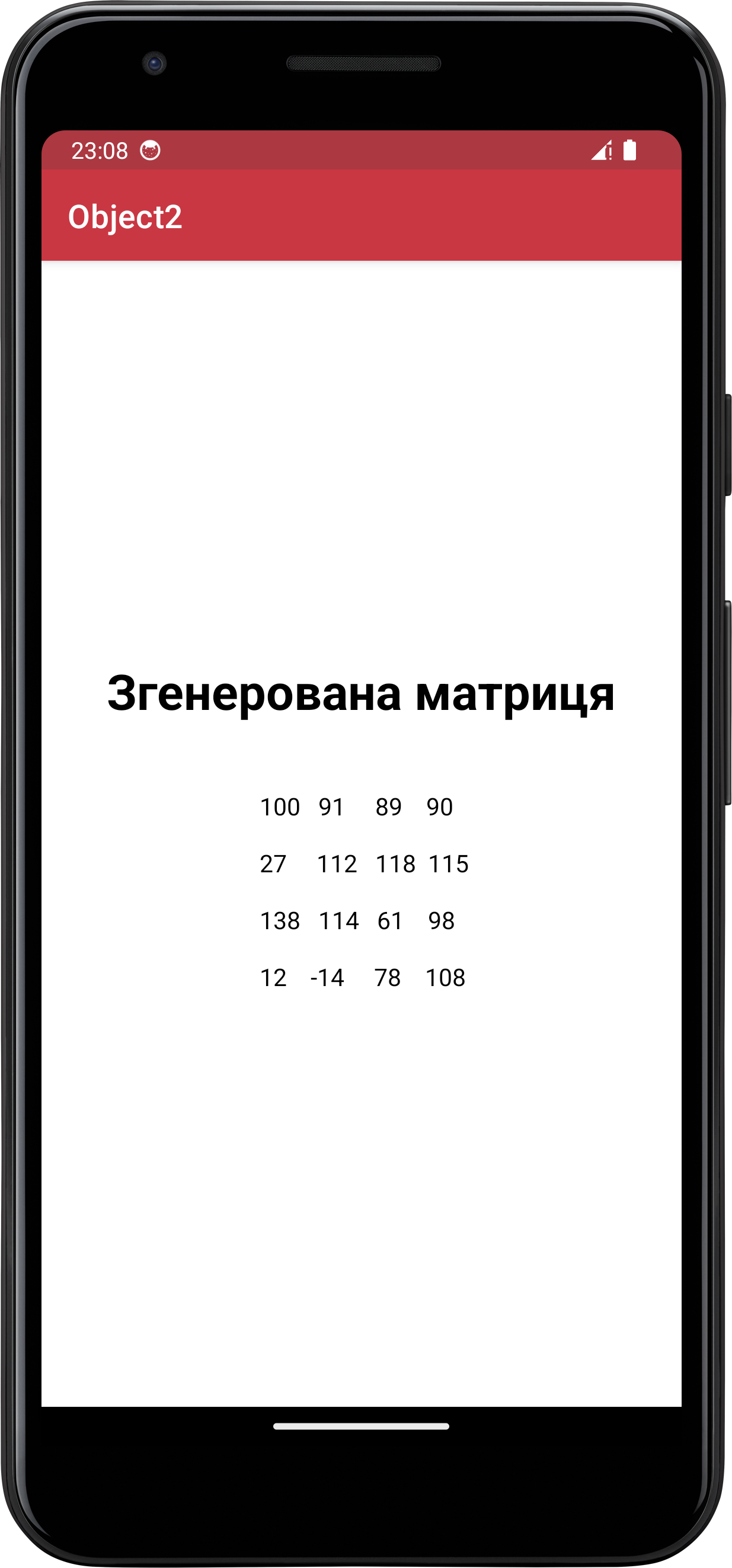
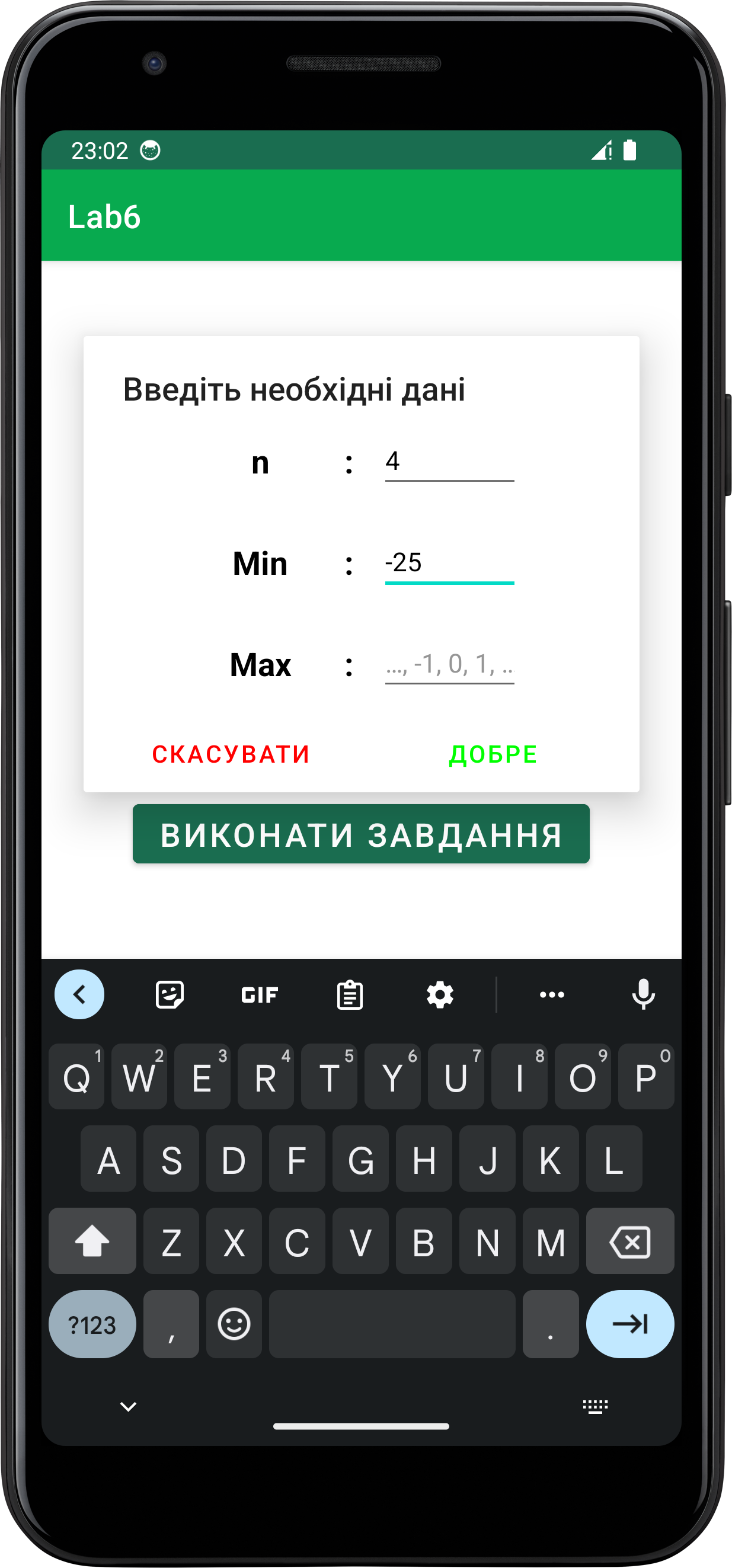
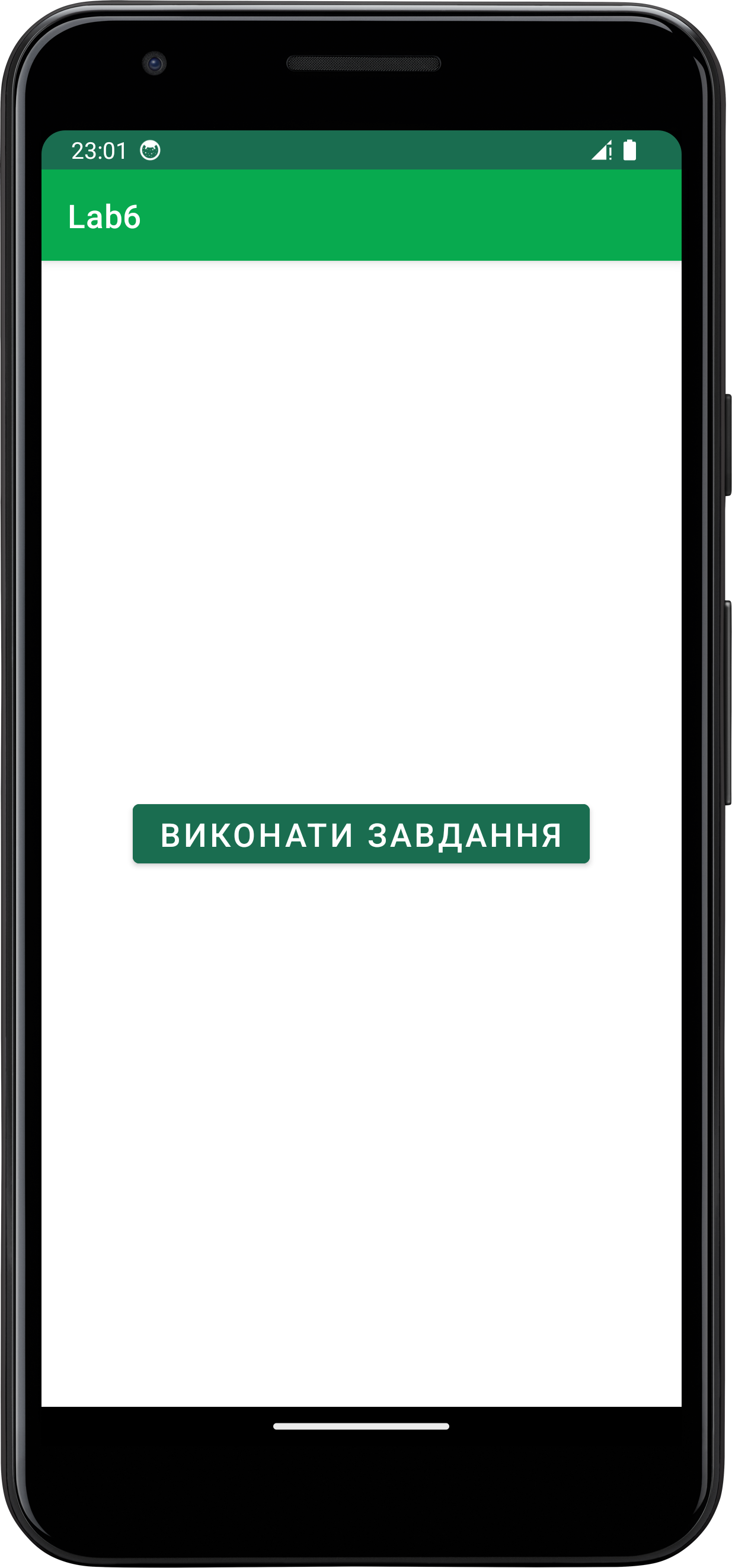
private fun calculateDeterminant(matrix: Array<IntArray>): Int {  
 val n = matrix.size  
 if (n == 1) return matrix[0][0]  
 val matrixDouble = Array(n) **{** i **->** DoubleArray(n) **{** j **->** matrix[i][j].toDouble()  
 **}  
 }** for (i in 0 until n - 1) {  
 var maxRow = i  
 for (k in i + 1 until n) {  
 if (abs(matrixDouble[k][i]) > abs(matrixDouble[maxRow][i])) {  
 maxRow = k  
 }  
 }  
 if (maxRow != i) {  
 val temp = matrixDouble[maxRow]  
 matrixDouble[maxRow] = matrixDouble[i]  
 matrixDouble[i] = temp  
 }  
 if (matrixDouble[i][i] == 0.0) return 0  
 for (j in i + 1 until n) {  
 val factor = matrixDouble[j][i] / matrixDouble[i][i]  
 for (k in i + 1 until n) {  
 matrixDouble[j][k] -= factor \* matrixDouble[i][k]  
 }  
 }  
 }  
 var determinant = 1.0  
 for (i in 0 until n) {  
 determinant \*= matrixDouble[i][i]  
 }  
 return round(determinant).toInt()  
 }

private fun readFromClipboard(): String {  
 val manager = getSystemService(CLIPBOARD\_SERVICE) as ClipboardManager  
 val clipData = manager.*primaryClip* return clipData?.let **{** data **->** data.getItemAt(0)?.let **{** item **->** item.text.toString()  
 **}** ?: ""  
 **}** ?: ""  
 }  
}

**Схема послідовності надсилання-обробки повідомлень**



**Ілюстрації виконання програм**



**Висновки**

Під час виконання цієї лабораторної роботи я навчився використовувати засоби обміну інформацією та запрограмував взаємодію незалежно працюючих програмних компонентів на мові програмування ***Kotlin*** для платформи ***Android***. Створена система складається з головної програми (**Lab6**), та двох незалежних один від одного компонентів (**Object2** та **Object3)**. **Lab6** викликає та займається обробкою повідомлень, отриманих від інших програм. Незважаючи на залежність **Object2** та **Object3** від **Lab6** кожна з програм може бути викликана користувачем вручну без впливу на результат виконання роботи в майбутньому. Обмін даними здійснюється за допомогою oб’єкта-обгортки ***Intent***, що доступний у ***Android API***, а також запису тексту у буфер обміну та подальше читання з нього.