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:
Студент групи ІМ-31
Максимовський Назар
номер у списку групи: 13

Перевірив:
Порєв В.М.

Київ 2024

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?.takeIf { it.action == "OBJECT2_SEND_SIGNAL"
}?.getStringExtra("SIGNAL")?.let { signal ->
                if (signal == "TASK_END_SUCCESS") {
                    Handler(Looper.getMainLooper()).postDelayed({
                        launchAppWithSignal(object3PackageName, "START")
                    }, 200L)
                } else showToast("Сталась помилка виконання
$object2PackageName")
            }
        }
    }

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.main_activity)
        registerReceiver(object2SignalHandler,
IntentFilter("OBJECT2_SEND_SIGNAL"), RECEIVER_EXPORTED)

        findViewById<Button>(R.id.complete_task).setOnClickListener {
            Dialog().apply {
                setOnConfirmListener { data ->
launchAppWithData(object2PackageName, data) }
                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) {
        Toast.makeText(this, text, 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 view = layoutInflater.inflate(R.layout.dialog, null)
        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)

        return AlertDialog.Builder(requireActivity())
            .setTitle(R.string.dialog_header_text)
            .setView(view)
            .create().apply {
                setCancelable(false)
                setCanceledOnTouchOutside(false)

view.findViewById<Button>(R.id.dialog_cancel).setOnClickListener { dismiss()
}

view.findViewById<Button>(R.id.dialog_okay).setOnClickListener {
                    validateInput(nEditText.text.toString(),
minEditText.text.toString(), maxEditText.text.toString())?.let {
                        dismiss()
                        Handler(Looper.getMainLooper()).postDelayed({
onConfirmListener(it) }, 100L)
                    }
                }
            }
    }

    private fun validateInput(n: String, min: String, max: String): IntArray?
{
        val parsedN = n.toIntOrNull().takeIf { it != null && it > 0 } ?:
return showError("n має бути більше 0")
        val parsedMin = min.toIntOrNull() ?: return showError("Min не є
числом")
        val parsedMax = max.toIntOrNull() ?: return showError("Max не є
числом")
        if (parsedMin > parsedMax) return showError("Min не може бути більшим
за Max")
        return intArrayOf(parsedN, parsedMin, parsedMax)
    }

    private fun showError(message: String): IntArray? {
        Toast.makeText(requireActivity(), message, Toast.LENGTH_SHORT).show()
        return null
    }

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

Object2.kt

```kotlin
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() {
        intent.getIntArrayExtra("DATA")?.let { (n, min, max) ->
            try {
                val matrix = generateMatrix(n, min, max)
                showMatrix(matrix)
                writeToClipboard(serializeMatrix(matrix))
                sendTaskEndingSignal(0)
            } catch (e: Exception) {
                sendTaskEndingSignal(1)
            }
        }
    }

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

    private fun showMatrix(matrix: Array<IntArray>) {
        matrixTextView.text = matrix.joinToString("\n\n") { row ->
            row.joinToString(" ") { value ->
                val prefix = if (value >= 0) " " else ""
                val spaces = when (abs(value)) {
                    in 0..9 -> "      "
                    in 10..99 -> "    "
                    in 100..999 -> "  "
                    else -> " "
                }
                "$prefix$value$spaces"
            }.trimEnd()
        }
    }

    private fun serializeMatrix(matrix: Array<IntArray>) =
matrix.joinToString("\n") {
        it.joinToString("\t")
    }
```

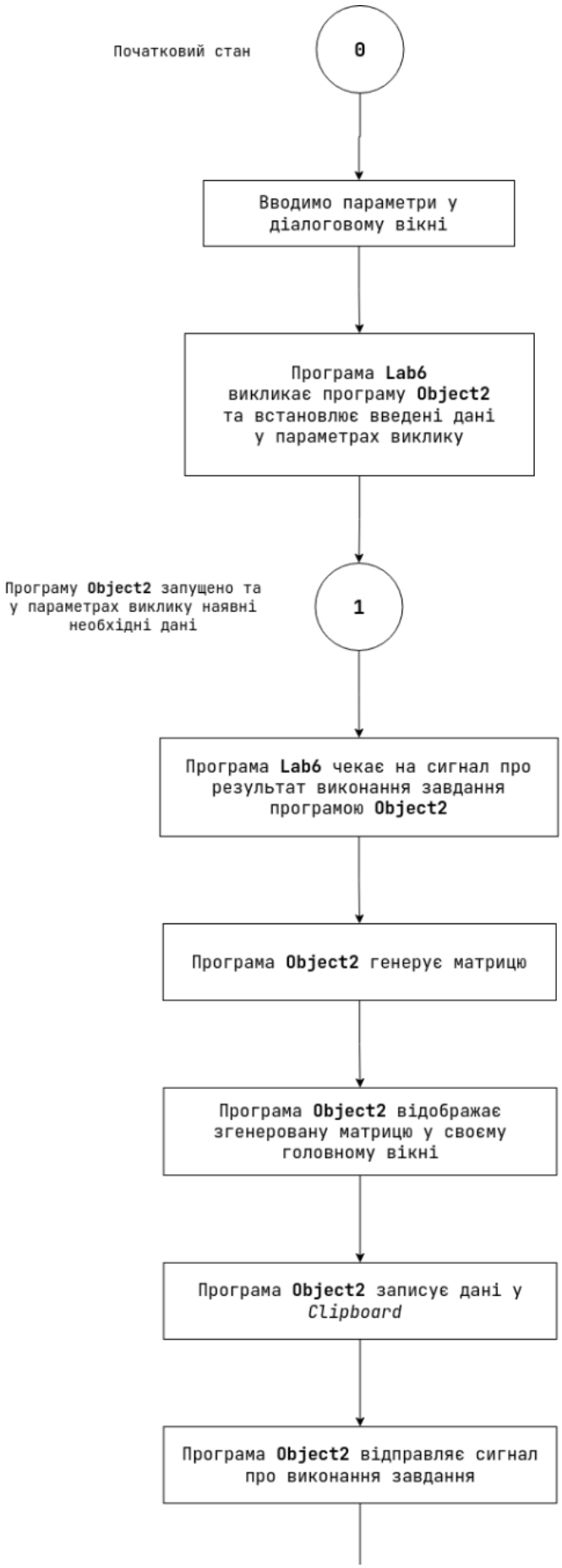
```
    private fun writeToClipboard(data: String) {
        (getSystemService(CLIPBOARD_SERVICE) as
ClipboardManager).setPrimaryClip(
            ClipData.newPlainText("MATRIX", data)
        )
    }

    private fun sendTaskEndingSignal(status: Int) {
        sendBroadcast(Intent("OBJECT2_SEND_SIGNAL").apply {
            putExtra("SIGNAL", if (status == 0) "TASK_END_SUCCESS" else
"TASK_END_FAILURE")
        })
    }
}
```

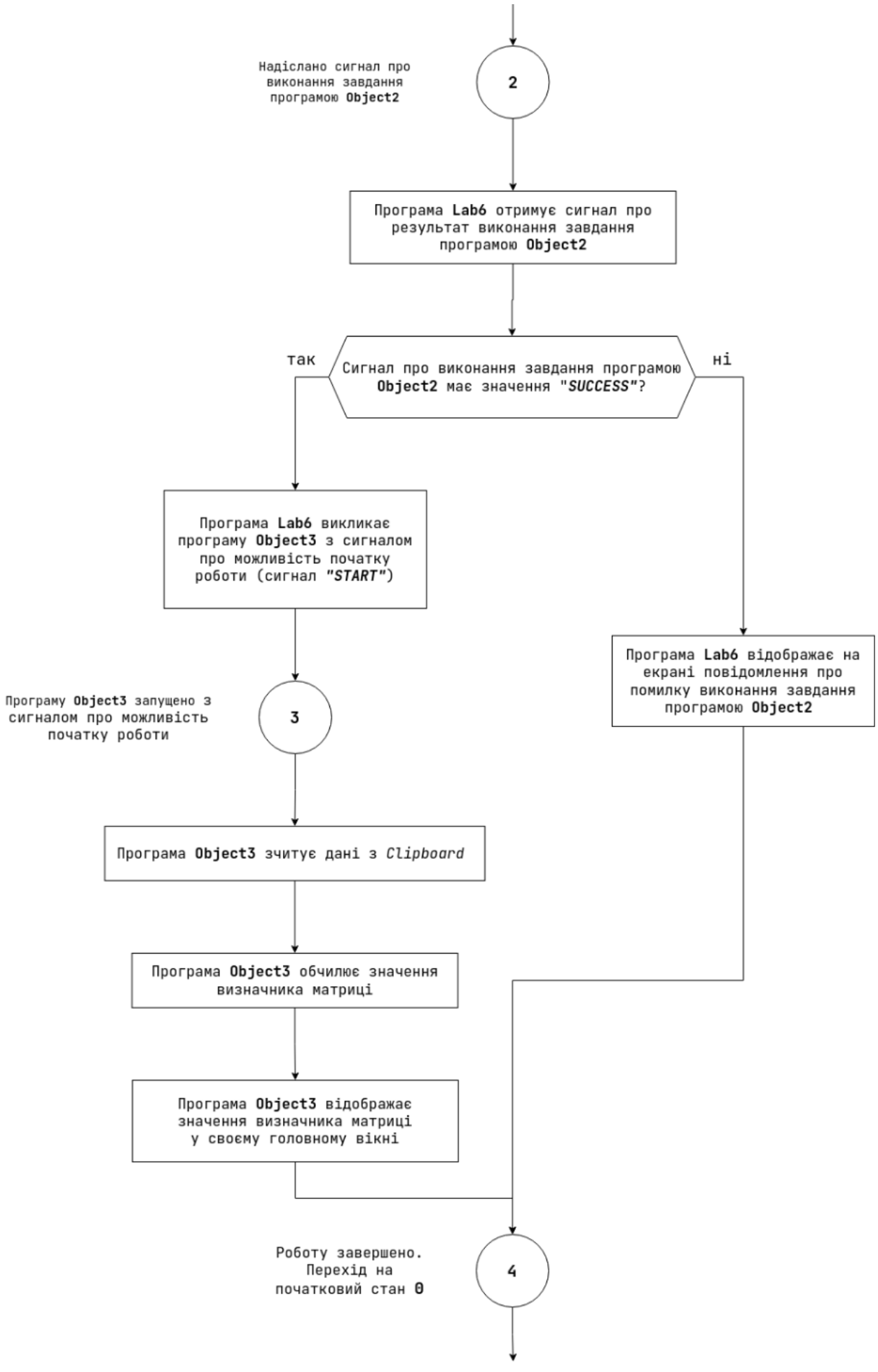
Object3.kt

```
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 kotlin.math.abs
import kotlin.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) {
            handleNewIntent()
            wasNewIntent = false
        }
    }

    private fun handleNewIntent() {
        if (intent.getStringExtra("SIGNAL") == "START") {
            val data = readFromClipboard()
            if (data.isNotEmpty()) {
                val determinant =
calculateDeterminant(deserializeMatrix(data))
                determinantTextView.text = determinant.toString()
            } else {
```

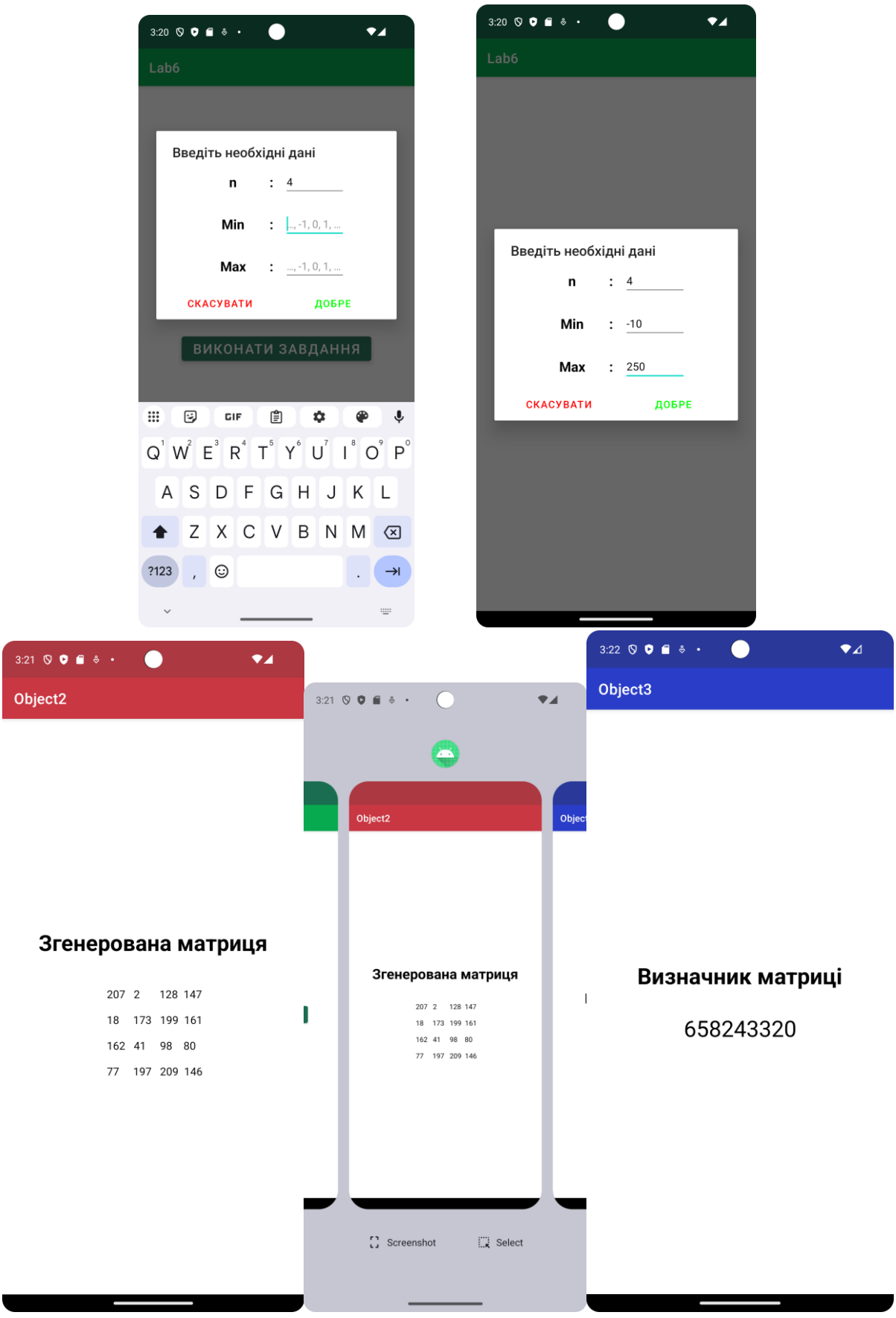
```
                showToast("Виникла помилка читання з буфера обміну")
            }
        }
    }

    private fun deserializeMatrix(str: String): Array<IntArray> =
str.split("\n").map { row ->
        row.split("\t").map { it.toInt() }.toIntArray()
    }.toTypedArray()

    private fun calculateDeterminant(matrix: Array<IntArray>): Int {
        val n = matrix.size
        if (n == 1) return matrix[0][0]
        val matrixDouble = matrix.map { row -> row.map { it.toDouble()
}.toDoubleArray() }.toTypedArray()
        for (i in 0 until n - 1) {
            val maxRow = (i until n).maxByOrNull { abs(matrixDouble[it][i]) }
?: i
            if (maxRow != i) matrixDouble.swapRows(i, maxRow)
            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]
                }
            }
        }
        return round(matrixDouble.fold(1.0) { acc, row -> acc *
row[matrixDouble.indexOf(row)] }).toInt()
    }

    private fun Array<DoubleArray>.swapRows(i: Int, j: Int) {
        val temp = this[i]
        this[i] = this[j]
        this[j] = temp
    }

    private fun readFromClipboard(): String {
        val manager = getSystemService(CLIPBOARD_SERVICE) as ClipboardManager
        return manager.primaryClip?.getItemAt(0)?.text?.toString() ?: ""
    }

    private fun showToast(message: String) {
        Toast.makeText(this, message, Toast.LENGTH_LONG).show()
    }
}
```

Схема послідовності надсилання-обробки повідомлень


Початковий стан
0
Вводимо параметри у діалоговому вікні
Програма Lab6 викликає програму Object2 та встановлює введені дані у параметрах виклику
Програму Object2 запущено та у параметрах виклику наявні необхідні дані
1
Програма Lab6 чекає на сигнал про результат виконання завдання програмою Object2
Програма Object2 генерує матрицю
Програма Object2 відображає згенеровану матрицю у своєму головному вікні
Програма Object2 записує дані у Clipboard
Програма Object2 відправляє сигнал про виконання завдання


Надіслано сигнал про виконання завдання програмою **Object2**
2
Програма **Lab6** отримує сигнал про результат виконання завдання програмою **Object2**
так
Сигнал про виконання завдання програмою **Object2** має значення "***SUCCESS***"?
ні
Програма **Lab6** викликає програму **Object3** з сигналом про можливість початку роботи (сигнал ***"START"***)
Програма **Lab6** відображає на екрані повідомлення про помилку виконання завдання програмою **Object2**
Програму **Object3** запущено з сигналом про можливість початку роботи
3
Програма **Object3** зчитує дані з *Clipboard*
Програма **Object3** обчилює значення визначника матриці
Програма **Object3** відображає значення визначника матриці у своєму головному вікні
Роботу завершено. Перехід на початковий стан **0**
4

Ілюстрації виконання програм


3:20
Lab6
Введіть необхідні дані
n : 4
Min : ..., -1, 0, 1, ...
Max : ..., -1, 0, 1, ...
СКАСУВАТИ
ДОБРЕ
ВИКОНАТИ ЗАВДАННЯ
3:20
Lab6
Введіть необхідні дані
n : 4
Min : -10
Max : 250
СКАСУВАТИ
ДОБРЕ
3:21
Object2
Згенерована матриця
207 2 128 147
18 173 199 161
162 41 98 80
77 197 209 146
3:21
Object2
Згенерована матриця
207 2 128 147
18 173 199 161
162 41 98 80
77 197 209 146
Screenshot
Select
3:22
Object3
Визначник матриці
658243320

Висновки

В процесі виконання цієї лабораторної роботи я опанував використання засобів обміну інформацією та забезпечив взаємодію автономно працюючих програмних компонентів на мові програмування Kotlin для платформи Android. Створена система включає головну програму (Lab6) та два незалежних компоненти (Object2 та Object3). Lab6 викликає ці компоненти та обробляє повідомлення, отримані від них. Незважаючи на залежність Object2 та Object3 від Lab6, кожен з цих компонентів може бути запущений користувачем вручну, що не впливає на подальші результати роботи. Обмін даними здійснюється за допомогою об'єкта-обгортки Intent, доступного в Android API, а також через буфер обміну для запису та читання тексту.