Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»



Звіт

З лабораторної роботи №1

З дисципліни «Кросплатформні засоби програмування»

На тему:

**ДОСЛІДЖЕННЯ БАЗОВИХ КОНСТРУКЦІЙ МОВИ JAVA**

Виконав: Тупісь М.Н.

ст. гр.KI-304

Прийняв:

Іванов Ю.С.

Львів – 2025

**Мета**: ознайомитися з базовими конструкціями мови Java та оволодіти навиками написання й автоматичного документування простих консольних програм мовою Java.

**Теоретичні відомості**

Масив – структура даних, що зберігає набір значень однакового типу. Пам’ять підмасив виділяється у керованій кучі. При завершенні життєвого циклу масиву пам’ять, якувін займав, вивільняється збирачем сміття. Доступ до елементів масиву здійснюється задопомогою індексів. Індексація масивів у Java починається з 0. Для створення масиву уJava необхідно оголосити змінну-масив та ініціалізувати її. При створенні за допомогоюоператора new масиву чисел всі його елементи ініціалізуються нулями (масиви типуboolean ініціалізуються значеннями false, масиви об’єктів ініціалізуються значеннямиnull). Після створення масиву змінити його розмір неможливо.

Java дозволяє створювати і анонімні масиви (без іменні). При створенні анонімного масиву відбувається виділення необхідної кількості пам’яті для збереження елементів масиву та ініціалізація масиву значеннями зі списку ініціалізації.

Анонімні масиви корисні тоді, коли необхідно ініціалізувати існуючий масив новими значеннями без створення нової змінної, коли необхідно повернути з методу масив нульової довжини (пустий масив), або коли в метод необхідно передати масив наперед відомих значень.

Завдяки тому, що багатовимірні масиви у Java реалізуються як множина одновимірних масивів, стає можливим реалізувати багатовимірні масиви з різною кількістю елементів у межах виміру. Синтаксис оголошення зубчатого масиву нічим не відрізняється від синтаксису оголошення звичайного багатовимірного масиву. Різниця є лише у способі ініціалізації, де використовується виділення пам’яті під різну кількість елементів у межах виміру.

Розмір масиву зберігається у властивості length.Копіювання масивів не можна здійснити звичайним присвоюванням однієї змінної-масиву іншій. У цьому випадку обидві змінні-масиви посилатимуться на одну і ту самуобласть пам’яті, тобто фізично на один і той самий масив. Для коректного копіювання8масивів слід скористатися методом copyOf класу Arrays. Цей метод створює копіюмасиву, що переданий через перший параметр методу, у пам’яті та повертає посилання нанього. Кількість елементів масиву, що підлягають копіюванню, передається через другийпараметр методу.

Хід роботи

Завдання: Написати та налагодити програму на мові Java згідно варіанту.

Варіант: 3



Лістинг програми мовою Java:

import java.io.FileWriter;

import java.io.IOException;

import java.util.Scanner;

public class Lab1TupisMaksym {

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);

        try {

            // --- Введення розміру матриці ---

            System.out.print("Введіть розмір квадратної матриці (n > 0): ");

            int size = scanner.nextInt();

            if (size <= 0) {

                System.out.println("Помилка: розмір матриці має бути додатнім числом.");

                return; // Завершення програми

            }

            scanner.nextLine(); // Споживаємо залишок рядка після nextInt()

            // --- Введення символу-заповнювача ---

            System.out.print("Введіть один символ-заповнювач: ");

            String inputChar = scanner.nextLine();

            // Перевірка коректності введеного символу

            if (inputChar.length() != 1) {

                System.out.println("Помилка: потрібно ввести рівно один символ.");

                return; // Завершення програми

            }

            char fillChar = inputChar.charAt(0);

            // --- Генерація та вивід масиву ---

            char[][] shadedArray = generateShadedArray(size, fillChar);

            System.out.println("\nЗгенерований зубчатий масив:");

            printArrayToConsole(shadedArray);

            writeArrayToFile(shadedArray, "output.txt");

            System.out.println("\nМасив успішно збережено у файл output.txt");

        } catch (Exception e) {

            System.out.println("Сталася помилка під час введення даних. Будь ласка, перевірте правильність вводу.");

        } finally {

            scanner.close(); // Закриваємо сканер для уникнення витоку ресурсів

        }

    }

    public static char[][] generateShadedArray(int size, char fillChar) {

        char[][] jaggedArray = new char[size][];

        int midRow = (size + 1) / 2; // Рядок, до якого формується трикутник

        for (int i = 0; i < size; i++) {

            if (i < midRow) {

                // Розраховуємо кількість символів у рядку, що зменшується

                // від size до 1 (для непарного розміру) або 2 (для парного)

                int count = size - 2 \* i;

                jaggedArray[i] = new char[count];

                for (int j = 0; j < count; j++) {

                    jaggedArray[i][j] = fillChar;

                }

            } else {

                // Решта рядків залишаються порожніми

                jaggedArray[i] = new char[0];

            }

        }

        return jaggedArray;

    }

    public static void printArrayToConsole(char[][] array) {

        int size = array.length;

        for (int i = 0; i < size; i++) {

            // Розраховуємо кількість відступів зліва, щоб вирівняти фігуру по центру

            int padding = (size - array[i].length) / 2;

            for (int k = 0; k < padding; k++) {

                System.out.print("  "); // Два пробіли для імітації одного символу

            }

            // Виводимо елементи рядка

            for (int j = 0; j < array[i].length; j++) {

                System.out.print(array[i][j] + " ");

            }

            System.out.println(); // Перехід на новий рядок

        }

    }

    public static void writeArrayToFile(char[][] array, String filename) {

        int size = array.length;

        try (FileWriter writer = new FileWriter(filename)) {

            for (int i = 0; i < size; i++) {

                // Логіка для запису аналогічна виводу на консоль

                int padding = (size - array[i].length) / 2;

                for (int k = 0; k < padding; k++) {

                    writer.write("  ");

                }

                for (int j = 0; j < array[i].length; j++) {

                    writer.write(array[i][j] + " ");

                }

                writer.write(System.lineSeparator()); // Додаємо символ нового рядка

            }

        } catch (IOException e) {

            System.out.println("Сталася помилка під час запису у файл: " + e.getMessage());

        }

    }

}

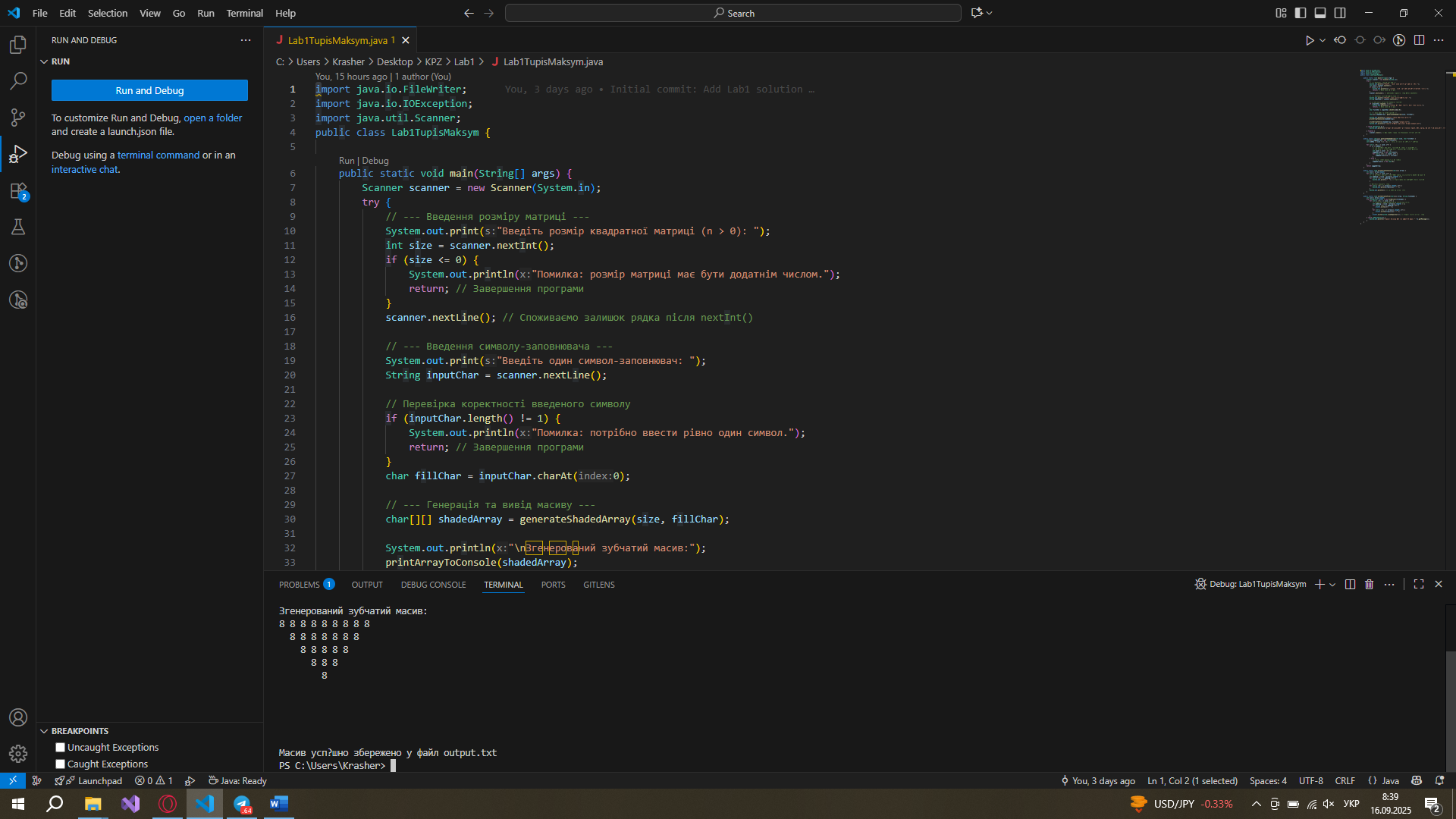


Рис.1 Приклад роботи програми

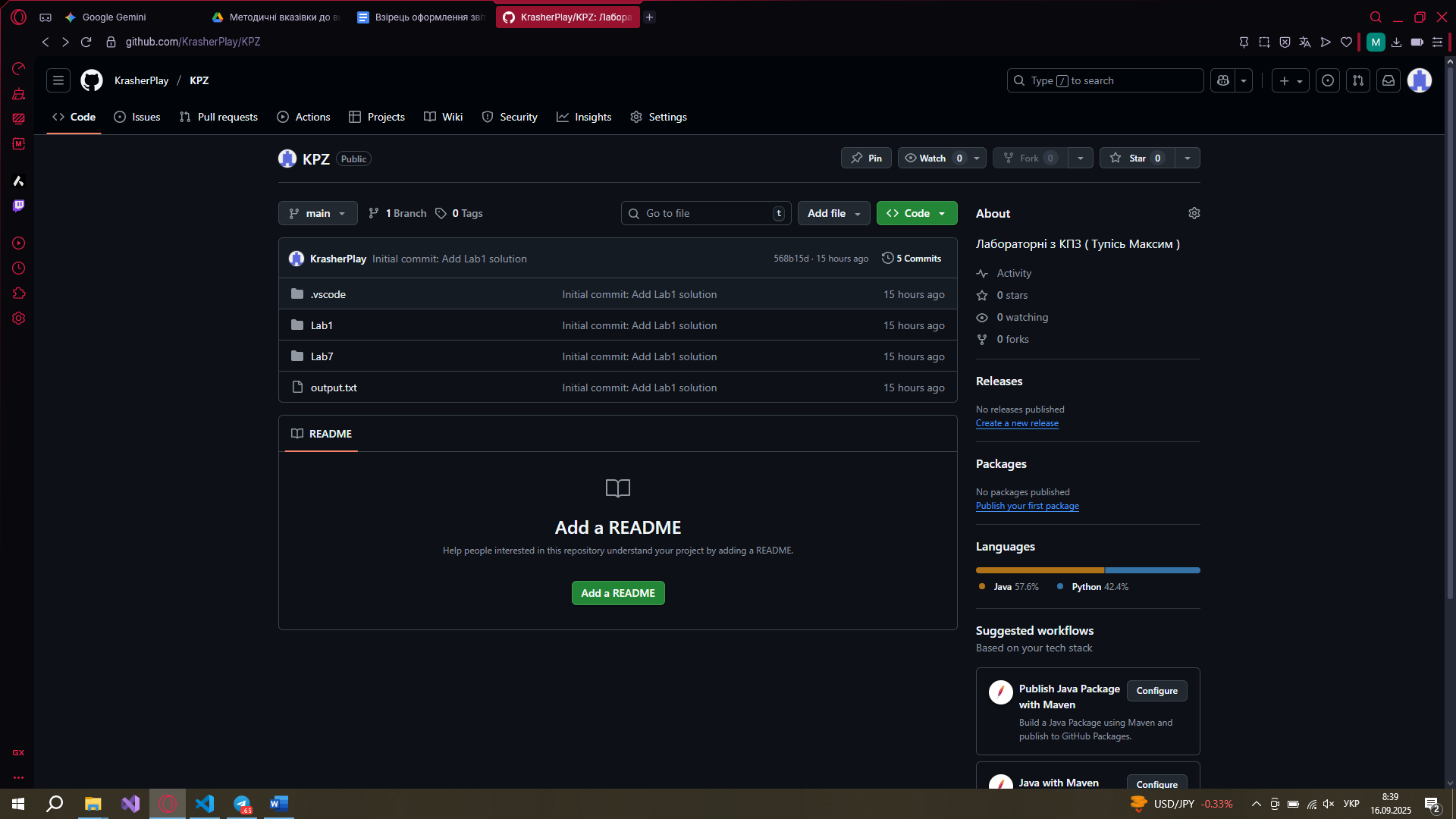


Рис.2 Git hub репозиторій з файлами програми

**Висновок:** У ході виконання лабораторної роботи було досліджено основні базові конструкції мови програмування Java. Зокрема, було розглянуто оголошення та використання змінних різних типів даних, оператори присвоєння, арифметичні та логічні операції, умовні конструкції (if, switch) та циклічні оператори (for, while, do-while). Практична реалізація прикладів дозволила закріпити знання щодо синтаксису мови та особливостей її застосування для розв’язання простих алгоритмічних задач.Окрему увагу було приділено роботі з масивами, у тому числі багатовимірними та так званими зубчастими матрицями (jagged arrays), які в Java реалізуються як масиви масивів. На практиці було досліджено способи їх ініціалізації, доступу до елементів та відмінності від класичних прямокутних матриць. Це дозволило зрозуміти гнучкість і потужність такої структури даних, а також її зручність для роботи з нерівномірними таблицями або структурами, що мають різну кількість елементів у рядках.Таким чином, у процесі роботи було сформовано базові практичні навички програмування на Java, отримано досвід використання основних операторів, циклів і умовних конструкцій, а також опановано принципи створення та застосування зубчастих матриць. Це є важливим кроком для подальшого вивчення об’єктно-орієнтованих можливостей мови та розробки більш складних програмних систем.