Національний університет «Львівська Політехніка»

Інститут комп’ютерних технологій, автоматики та метрології

Кафедра електронних обчислювальних машин



Звіт

Про виконання лабораторної роботи №1

З дисципліни «Кросплатформлені засоби програмування»

**Виконав:**

студент групи КІ-305

Панасюк Т.Ю.

**Перевірив:**

Доцент кафедри ЕОМ

Іванов Ю. С.

Львів – 2023

**Тема:** Дослідження базових конструкцій мови Java.

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

**Завдання:**

1. Написати та налагодити програму на мові Java згідно варіанту. Програма має

задовольняти наступним вимогам:

• програма має розміщуватися в загальнодоступному класі Lab1ПрізвищеГрупа;

• програма має генерувати зубчатий масив, який міститиме лише заштриховані області квадратної матриці згідно варіанту;

• розмір квадратної матриці і символ-заповнювач масиву вводяться з клавіатури;

• при не введені або введенні кількох символів-заповнювачів відбувається коректне переривання роботи програми;

• сформований масив вивести на екран і у текстовий файл;

• програма має володіти коментарями, які дозволять автоматично згенерувати документацію до розробленої програми.

2. Автоматично згенерувати документацію до розробленої програми.

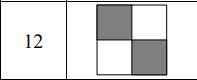
3. Завантажити код на GitHub згідно методичних вказівок по роботі з GitHub.

4. Скласти звіт про виконану роботу з приведенням тексту програми, результату її виконання та фрагменту згенерованої документації та завантажити його у ВНС.

5. Дати відповідь на контрольні запитання.

**Завдання згідно варіанту:**

**Варіант: 21**



**Виконання:**

**Код програми:**

import java.io.\*;

import java.util.\*;

public class app {

private static final int MIN\_MATRIX\_SIZE = 3;

private static final String OUTPUT\_FILE = "Result.txt";

public static void main(String[] args) {

try (Scanner scanner = new Scanner(System.in);

BufferedWriter writer = new BufferedWriter(new FileWriter(OUTPUT\_FILE))) {

int size = getInputSize(scanner);

String fillCharacter = getInputFillCharacter(scanner);

char[][] matrix = generateMatrix(size, fillCharacter);

displayMatrix(matrix);

saveMatrixToFile(matrix, writer);

} catch (IOException e) {

System.err.println("An error occurred while writing to the file: " + e.getMessage());

}

}

private static int getInputSize(Scanner scanner) {

int size = 0;

while (true) {

try {

System.out.print("Enter size: ");

size = scanner.nextInt();

if (size <= MIN\_MATRIX\_SIZE) {

System.out.println("The size of the matrix must be greater than " + MIN\_MATRIX\_SIZE);

continue;

}

break;

} catch (InputMismatchException e) {

System.out.println("Invalid input. Please enter an integer.");

scanner.nextLine();

}

}

return size;

}

private static String getInputFillCharacter(Scanner scanner) {

String fillCharacter;

while (true) {

System.out.print("Enter fill character: ");

fillCharacter = scanner.next();

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

System.out.println("Fill character must be a single character.");

} else {

break;

}

}

return fillCharacter;

}

private static char[][] generateMatrix(int size, String fillCharacter) {

char[][] matrix = new char[size][];

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

matrix[i] = new char[size - i];

Arrays.fill(matrix[i], fillCharacter.charAt(0));

}

for (int j = 0; j < matrix[4].length; j++) {

matrix[4][j] = j % 2 == 0 ? fillCharacter.charAt(0) : '\*';

}

return matrix;

}

private static void displayMatrix(char[][] matrix) {

for (char[] row : matrix) {

for (int skip = 0; skip < matrix.length - row.length; skip++) {

System.out.print("\t");

}

for (char cell : row) {

System.out.print(cell + "\t");

}

System.out.println();

}

}

private static void saveMatrixToFile(char[][] matrix, BufferedWriter writer) throws IOException {

for (char[] row : matrix) {

for (int skip = 0; skip < matrix.length - row.length; skip++) {

writer.write("\t");

}

for (char cell : row) {

writer.write(cell + "\t");

}

writer.newLine();

}

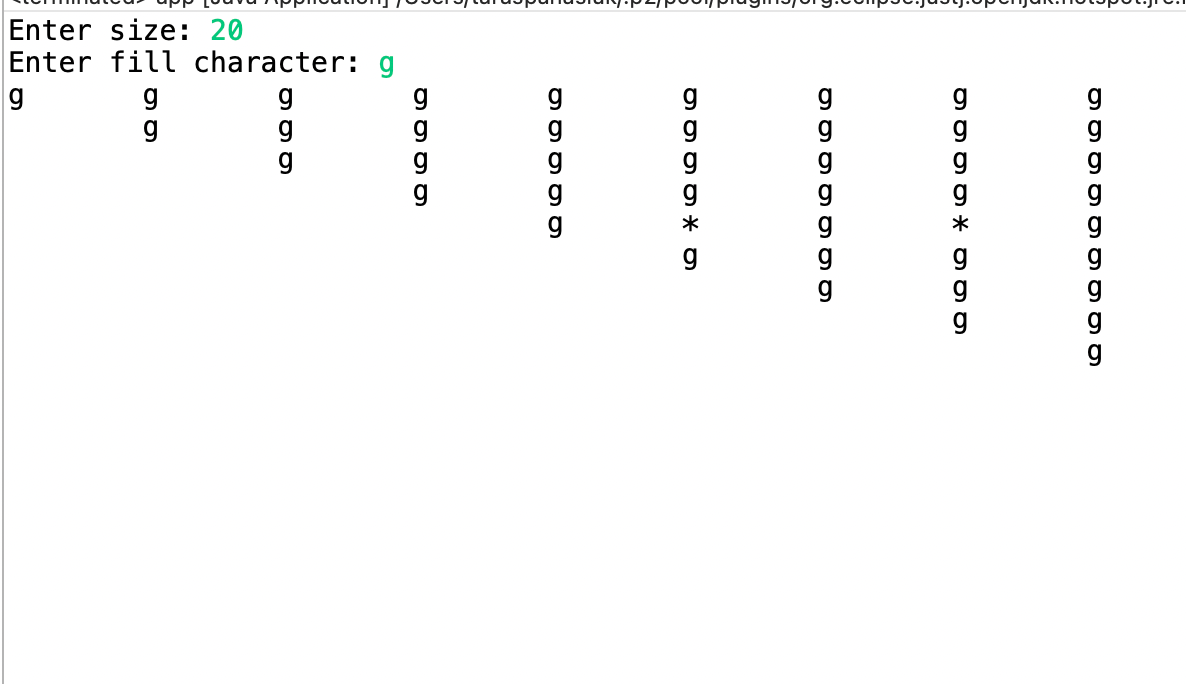
writer.flush();

System.out.println("Matrix saved to " + OUTPUT\_FILE);

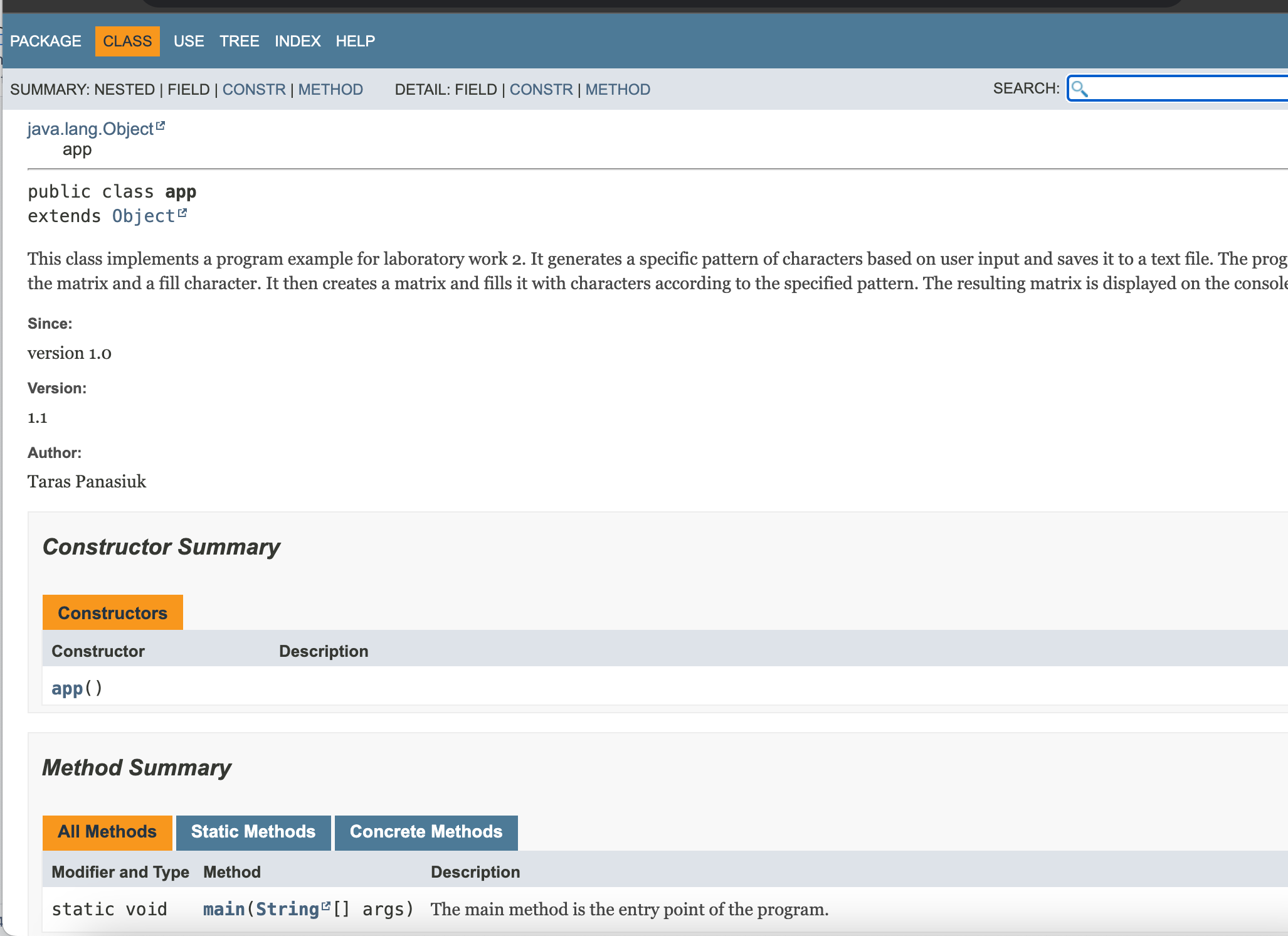
}

}

**Результат роботи програми у консолі та файлі:**



**Фрагмент генерації документації:**



**Контрольні запитання**

1. **Які дескриптори використовуються при коментуванні класів?**

**Відповідь:** Коментарі до класу мають бути розміщені після директив import безпосередньо перед визначенням класу. Найчастіше цей коментар має вигляд одного або кількох коротких речень:

/\*\*

Об’єкт класу <code>Person</code> описує особу.

Особа має властивості: ім’я, прізвище та стать.

\*/

1. **Які дескриптори використовуються при коментуванні методів?**

**Відповідь:** Коментарі до методів розташовуються безпосередньо перед методами, які вони описують. Крім дескрипторів загального призначення для коментування методів використовуються дескриптори:

• @param змінна опис

Цей дескриптор додає в опис методу розділ “parameters”. Опис цього елементу може складатися з кількох рядків та містити html-теги. Всі дескриптори @param, що відносяться до одного методу слід групувати разом.

• @return опис

Цей дескриптор додає в опис методу розділ “returns”. Опис цього елементу може складатися з кількох рядків та містити html-теги.

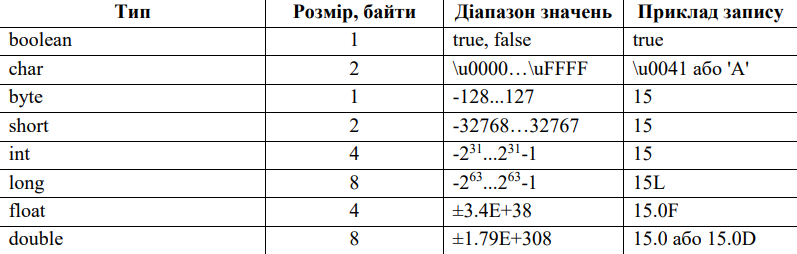
• @throws опис\_класу

Цей дескриптор додає в опис методу інформацію про класи об’єкти яких можуть генеруватися при виключних ситуаціях. Відомості про кожен клас слід описувати в окремому дескрипторі @throws.

1. **Як автоматично згенерувати документацію?**

**Відповідь:** Для генерування документації по пакету слід ввести в консолі ОС Windows: javadoc –d каталог\_doc ім’я\_пакету Опція –d каталог\_doc задає каталог, де слід розмістити згенеровану документація до пакету.

1. **Які прості типи даних підтримує Java?**

**Відповідь:** ****

1. **Як оголосити змінну-масив?**

**Відповідь:** Для створення масиву у Java необхідно оголосити змінну-масив та ініціалізувати її. При створенні за допомогою оператора new масиву чисел всі його елементи ініціалізуються нулями (масиви типу boolean ініціалізуються значеннями false, масиви об’єктів ініціалізуються значеннями null).

1. **Які керуючі конструкції підтримує Java?**

**Відповідь:**

1. Відповідь: if-else: Умовна конструкція для виконання коду в залежності від умови.
2. switch: Умовна конструкція для вибору зі списку варіантів в залежності від значення виразу.
3. for: Цикл для повторення блоку коду певну кількість разів або до виконання умови.
4. while: Цикл для повторення блоку коду доти, доки задана умова істинна.
5. do-while: Цикл для повторення блоку коду, який спочатку виконується, а потім перевіряється умова.
6. break: Використовується для виходу з поточного циклу або вибору зі switch.
7. continue: Використовується для переходу до наступної ітерації циклу без виконання залишку коду поточної ітерації.
8. try-catch: Використовується для обробки винятків. Код, який може викинути виняток, поміщається в блок try, і винятки можна обробити у блоках catch.
9. finally: Блок finally використовується для коду, який завжди виконується, незалежно від того, чи виникла помилка.
10. **В чому різниця між різними варіантами оператора for?**

**Відповідь:** оператор циклу for в стилі C/C++ та оператор циклу for з синтаксисом foreach.

У першому стилі оператора for ви маєте явно вказати ініціалізацію лічильника, логічну умову і модифікацію лічильника. Цей стиль корисний, коли вам потрібно повторювати код певну кількість разів або поки задана умова істинна. Ви маєте повний контроль над лічильником і ітераціями.

for (int i = 0; i < 100; i++)

System.out.println(i);

Другий стиль оператора for (відомий як foreach або enhanced for loop) використовується для ітерації через всі елементи набору даних, такі як масиви або колекції, без використання лічильника. Це зручно, коли вам потрібно перебрати всі елементи набору один за одним.

Приклад:

for (int elem: arr)

System.out.println(elem);

Основна відмінність полягає в тому, що ви не працюєте з лічильником (індексом) та не використовуєте логічну умову. Замість цього, ви маєте доступ до кожного елемента набору в змінній змінна для подальшого використання.

Отже, вибір між двома стилями for залежить від конкретного завдання. Якщо вам потрібно виконати певний код певну кількість разів або поки задана умова істинна, використовуйте перший стиль. Якщо вам потрібно перебрати всі елементи набору даних без потреби в лічильнику, використовуйте другий стиль (foreach).

1. **Як здійснити ввід з консолі?**

**Відповідь:** Для введення інформації з консолі необхідно створити об’єкт класу Scanner і зв’язати його з стандартним потоком вводу System.in, наприклад:

*Scanner in = new Scanner(System.in);*

Зробивши це ми отримаємо доступ до методів класу Scanner, які призначені для

введення даних простих типів і рядків:

- hasNext – перевіряє чи є доступні дані для введення;

- nextBoolean – вводить дані типу boolean;

- nextByte – вводить дані типу byte;

- nextShort – вводить дані типу short;

- nextInt – вводить дані типу int;

- nextLong – вводить дані типу long;

- nextFloat – вводить дані типу float;

- nextDouble – вводить дані типу double;

- nextLine – вводить весь рядок і представляє його як об’єкт класу String;

- next – вводить одне слово (токен) і представляє його як об’єкт класу String.

1. **Як здійснити ввід з текстового файлу?**

**Відповідь:** Для введення інформації з файлу необхідно підключити пакет java.io та створити об’єкт класу Scanner з об’єкту File:

Scanner fin = new Scanner(File("MyFile.txt"));

При компіляції цього рядка коду може виникнути виключна ситуація неіснування файлу з якого має проходити ввід даних (FileNotFoundExeption), яка розглядається компілятором серйозніше, ніж, скажімо, ділення на нуль. Тому, компілятор не дасть нам просто так скомпілювати цей рядок коду. Для коректної компіляції цього рядку коду необхідно або вказати, що наш метод може генерувати це виключення, або реалізувати код для перехоплення і обробки виключення. Для того, щоб вказати, що наш метод може генерувати це виключення, у оголошенні методу, де відбувається створення об’єкту класу Scanner з файлу, необхідно додати наступний код:

*throws FileNotFoundExeption*

1. **Як здійснити запис у текстовий файл?**

**Відповідь:** Для виведення інформації у текстовому вигляді у файл треба підключити пакет java.io та створити об’єкт класу PrintWriter в конструкторі якого необхідно вказати назву файлу, що відкривається на запис, наприклад:

*PrintWriter fout = new PrintWriter ("MyFile.txt");*

Зробивши це ми отримаємо доступ до методів класу PrintWriter, які призначені для виведення даних простих типів і рядків: print – виводить значення простих типів і рядків у текстовому вигляді; write – призначений для виводу даних типу char і String у текстовий файл.

При створенні об’єкту класу PrintWriter з іменем файлу, який не може бути створений буде згенероване виключення FileNotFoundExeption. Для його обробки необхідно виконати такі самі дії, що і в розглянутому вище способі введення даних з файлу.

**Висновок:** на цій лабораторній роботі я ознайомився з базовими конструкціями мови Java та оволодів навиками написання й автоматичного документування простих консольних програм мовою Java. Також написав код програми, що створює і заповнює зубчату матрицю згідно варіанту та записує її у текстовий файл.