МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»

Кафедра ЕОМ

ЗВІТ ДО ЛАБОРАТОРНОЇ РОБОТИ №1

з дисципліни Кросплатформні засоби програмування

з теми: «Дослідження базових конструкцій мови Java»

Виконав:

Студент групи КІ-301
Кравчук Ілля Миколайович

Роботу прийняв:

Олексів М.В.

Львів – 2025

ЛАБОРАТОРНА РОБОТА №1 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.

Порядок виконання:
1. Вивчення теоретичних відомостей: Ознайомлено з принципами автоматичного документування за допомогою утиліти javadoc (коментарі між / і /, дескриптори @param, @return, @author тощо). Вивчено основні типи даних Java (boolean, char, byte, short, int, long, float, double), синтаксис оголошення змінних (тип назва [= значення];) та масивів (тип[] назва = new тип[розмір];). Розглянуто особливості масивів: динамічне виділення пам’яті, ініціалізація нулями, індексація з 0, незмінний розмір після створення.
2. Підготовка середовища: Використано IDE (наприклад, IntelliJ IDEA) для написання коду в пакеті a1, класі LAB_01. Імпортовано пакети: java.io.FileWriter, java.io.IOException, java.util.Scanner для вводу/виводу.
3. Розробка програми: Створено консольну програму для введення розміру матриці (int size) та символу-заповнювача (char fillChar). Згенеровано двовимірний масив char[][] (квадратна матриця розміром size x size). Заповнено матрицю за шаблоном: верхня ліва чверть та нижня права чверть — символом fillChar, інші позиції — пробілом (' '). Використано вкладені цикли for для генерації та виведення. Додано запис масиву у файл "output.txt" за допомогою FileWriter. Оброблено помилку вводу (IOException).
4. Додавання документації: Додано Javadoc-коментарі до класу, методів та загальні коментарі з дескрипторами (@author, @version, @param). Згенеровано HTML-документацію командою: javadoc -d doc a1.LAB_01.
5. Компіляція та виконання: Скомпільовано код (javac LAB_01.java), запущено (java a1.LAB_01). Тестові дані: розмір=4, символ=''. Результати: виведено матрицю на екран, записано у файл.
6. Аналіз результатів: Перевірено коректність заповнення масиву (шаблон чвертей), обробку помилок (довжина символу=1). Документація згенерована успішно (файли index.html, LAB_01.html з описами).
7. Складання звіту: Зафіксовано скріншоти, пояснення коду та результатів. Захист: демонстрація запуску програми та документації.

ТЕОРЕТИЧНІ ВІДОМОСТІ

Автоматичне документування:
При автоматичній генерації документації використовується утиліта javadoc, яка аналізує вміст між / і / та генерує документацію у форматі .html. Коментарі починаються з описового тексту, за яким ідуть дескриптори (@param, @return тощо). Дозволяється використання HTML-тегів.

Основні типи даних Java:

Тип	Розмір, байти	Діапазон значень	Приклад запису
boolean	1	true, false	true
char	2	…\uFFFF	'A'
byte	1	-128...127	15
short	2	-32768…32767	15
int	4	-2^31...2^31-1	15
long	8	-2^63...2^63-1	15L
float	4	±3.4E+38	15.0F
double	8	±1.79E+308	15.0

РЕЗУЛЬТАТИ ВИКОНАННЯ

Таблиця оголошення змінних та масивів у програмі:

Змінна/Масив	Тип	Призначення	Ініціалізація/Приклад
size	int	Розмір матриці	int size = scanner.nextInt();
input	String	Введений символ (тимчасово)	String input = scanner.next();
fillChar	char	Символ-заповнювач	char fillChar = input.charAt(0);
jaggedArray	char[][]	Двовимірний масив (матриця)	char[][] array = new char[size][size];
scanner	Scanner	Для вводу з консолі	Scanner scanner = new Scanner(System.in);

Приклад результату для size=4, fillChar='':

```
* *
*
  * *
    * *
```

Код програми:

```
package a1;

import java.io.FileWriter;
import java.io.IOException;
import java.util.Scanner;

public class LAB_01 {


   public static void main(String[] args) {
      Scanner scanner = new Scanner(System.in);

      // Розмір матриці
      System.out.print("Введіть розмір квадратної матриці: ");
      int size = scanner.nextInt();

      // Символ-заповнювач
      System.out.print("Введіть символ-заповнювач: ");
      String input = scanner.next();

      // Перевірка символу-заповнювача
      if (input.length() != 1) {
         System.out.println("Помилка: потрібно ввести рівно один символ!");
         scanner.close();
         return;
      }

      char fillChar = input.charAt(0);
      scanner.close();

      // Генерація зубчастого масиву
      char[][] jaggedArray = generateJaggedArray(size, fillChar);

      // Виведення масиву на екран
      System.out.println("\nЗгенерований зубчатий масив:");
      printArray(jaggedArray);

      // Запис масиву у файл
      writeArrayToFile(jaggedArray, "output.txt");

      System.out.println("Масив також записано у файл output.txt");
   }

   /**
    * Генерує квадратну матрицю з шаблонним заповненням.
    * @param size розмір матриці
    * @param fillChar символ для заповнення
    * @return заповнена матриця char[][]
    */
   private static char[][] generateJaggedArray(int size, char fillChar) {
      char[][] array = new char[size][size]; // Квадратна матриця

      for (int i = 0; i < size; i++) {
         for (int j = 0; j < size; j++) {
            // Чи поточна позиція належить до заштрихованої області
            boolean isShaded;

            if (i < size / 2) {
               // Верхня половина матриці
```

```
                if (j < size / 2) {
                    // Ліва верхня частина - заштрихована
                    isShaded = true;
                } else {
                    // Права верхня частина - не заштрихована
                    isShaded = false;
                }
            } else {
                // Нижня половина матриці
                if (j < size / 2) {
                    // Ліва нижня частина - не заштрихована
                    isShaded = false;
                } else {
                    // Права нижня частина - заштрихована
                    isShaded = true;
                }
            }

            // Заповнюємо символом або пробілом
            array[i][j] = isShaded ? fillChar : ' ';
        }
    }

    return array;
}

/**
 * Виводить матрицю на екран.
 * @param array матриця для виведення
 */
private static void printArray(char[][] array) {
    for (int i = 0; i < array.length; i++) {
        for (int j = 0; j < array[i].length; j++) {
            System.out.print(array[i][j]);
        }
        System.out.println();
    }
}

/**
 * Записує матрицю у файл.
 * @param array матриця для запису
 * @param filename ім'я файлу
 * @throws IOException у разі помилки запису
 */
private static void writeArrayToFile(char[][] array, String filename) {
    try (FileWriter writer = new FileWriter(filename)) {
        for (int i = 0; i < array.length; i++) {
            for (int j = 0; j < array[i].length; j++) {
                writer.write(array[i][j]);
            }
            writer.write(System.lineSeparator());
        }
    } catch (IOException e) {
        System.out.println("Помилка при записі у файл: " + e.getMessage());
    }
}
}
```

1. Загальна структура та мета програми

Клас LAB_01 (пакет a1) призначений для створення консольної програми, яка:

- Приймає від користувача розмір квадратної матриці (size) та символ-заповнювач (fillChar).
- Генерує двовимірний масив char[][], заповнюючи його за шаблоном: верхня ліва та нижня права чверті матриці заповнюються введеним символом, інші позиції — пробілом.
- Виводить матрицю на екран і зберігає її у файл output.txt.
- Використовує базові конструкції Java: типи даних (int, char, String), масиви, цикли, умовні оператори та обробку винятків.

---

2. Імпорт бібліотек

```
import java.io.FileWriter;
import java.io.IOException;
import java.util.Scanner;
```

- Scanner: Використовується для зчитування даних із консолі (введення size і input).
- FileWriter та IOException: Необхідні для запису матриці у файл і обробки можливих винятків під час роботи з файлами.

---

3. Головний метод main

```
public static void main(String[] args) {
```

```
    Scanner scanner = new Scanner(System.in);
System.out.print("Введіть розмір квадратної матриці: ");
    int size = scanner.nextInt();
    System.out.print("Введіть символ-заповнювач: ");
    String input = scanner.next();
     if (input.length() != 1) {
    System.out.println("Помилка: потрібно ввести рівно один
символ!");
    scanner.close();
    return;
    }
    char fillChar = input.charAt(0);
    scanner.close();
    char[][] jaggedArray = generateJaggedArray(size, fillChar);
System.out.println("\n3генерований зубчатий масив:");
printArray(jaggedArray);
    writeArrayToFile(jaggedArray, "output.txt");
System.out.println("Масив також записано у файл output.txt");
    }
```

- **Пояснення:**
    - Створюється об’єкт Scanner для введення даних.
    - Користувач вводить size (тип int) — розмір матриці.
    - Вводиться input (тип String) — символ-заповнювач, який перевіряється на довжину (має бути 1 символ). Якщо умова не виконана, виводиться помилка, і програма завершується.
    - fillChar (тип char) отримується з першого символу input.
    - scanner.close() звільняє ресурси.
    - Викликаються методи: generateJaggedArray для створення матриці, printArray для виведення та writeArrayToFile для запису.
    - Виводиться повідомлення про успішний запис у файл.

---

4. Метод generateJaggedArray

```
private static char[][] generateJaggedArray(int size, char fillChar)
{
char[][] array = new char[size][size];
for (int i = 0; i < size; i++) {
for (int j = 0; j < size; j++) {
boolean isShaded;
if (i < size / 2) {
 if (j < size / 2) isShaded = true;
else isShaded = false;            }
else {
if (j < size / 2) isShaded = false;
else isShaded = true;            }
array[i][j] = isShaded ? fillChar : ' ';        }
 }
return array;
}
```

- **Пояснення:**
    - Створюється двовимірний масив char[][] розміром size x size, ініціалізований нулями (за замовчуванням для char — ``, але перезаписується).
    - Вкладені цикли for ітеруються по рядках (i) і стовпцях (j).
    - Умови визначають, чи поточна позиція належить до "заштрихованої" області:
        - Верхня половина (i < size / 2): ліва чверть (j < size / 2) — true, права — false.
        - Нижня половина (i >= size / 2): ліва чверть — false, права — true.
    - Значення array[i][j] присвоюється fillChar, якщо isShaded = true, інакше — пробіл ' '.
    - Повертається заповнений масив.

---

5. Метод printArray

```
private static void printArray(char[][] array) {
for (int i = 0; i < array.length; i++) {
for (int j = 0; j < array[i].length; j++) {
System.out.print(array[i][j]);
}
System.out.println();
}
}
```

- **Пояснення:**
    - Виводить матрицю на екран рядок за рядком.
    - Вкладені цикли ітеруються по всіх елементах масиву.
    - System.out.print() виводить символи без переходу на новий рядок, System.out.println() додає новий рядок після кожного рядка матриці.

---

6. Метод writeArrayToFile

```
private static void writeArrayToFile(char[][] array, String filename) {
try (FileWriter writer = new FileWriter(filename)) {
 for (int i = 0; i < array.length; i++) {
for (int j = 0; j < array[i].length; j++) {
writer.write(array[i][j]);
}
writer.write(System.lineSeparator());
}
} catch (IOException e) {
System.out.println("Помилка при записі у файл: " + e.getMessage());
}
}
```

- **Пояснення:**
    - Використовує try-with-resources для автоматичного закриття FileWriter.
    - Записує матрицю у файл filename (наприклад, "output.txt").

- Вкладені цикли ітеруються по елементах масиву, writer.write() записує кожен символ.
- System.lineSeparator() додає роздільник рядків (залежить від ОС).
- catch блокує виняток IOException і виводить повідомлення про помилку.

---

7. Приклад виконання

Для size = 4 і fillChar = '*' результат:

text

```
* *
*
  * *
    * *
```

Файл output.txt міститиме аналогічний вивід.

ВИСНОВОК

Виконання лабораторної роботи №1 дозволило ознайомитися з базовими конструкціями мови Java: типами даних (int, char, String), змінними, масивами, циклами, умовами та обробкою винятків. Було освоєно написання консольної програми з вводу/виводу, генерацією шаблонної матриці та записом у файл. Автоматична документація javadoc згенерована успішно, що полегшує розуміння коду. Результати коректні: матриця заповнена за шаблоном, помилки оброблені. Навички застосовуватимуться в подальших роботах.