

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет “Львівська політехніка”
Інститут післядипломної освіти



ЗВІТ

Про виконання лабораторної роботи №1
«Створення класів. Оголошення об’єктів. Доступ до змінних та
методів класів через об’єкт. Реалізація конструкторів з
параметрами»
з дисципліни «Об’єктно-орієнтоване програмування»

Виконав:
слухач групи ПЗС-11
Гринчук Тарас

Прийняла:
доц. Кортєєва Т.О.

« » _____ 2014 р.

Σ _____

Тема роботи: Створення класів. Оголошення об'єктів. Доступ до змінних та методів класів через об'єкт. Реалізація конструкторів з параметрами.

1. Завдання

Написати програму алгоритмічною мовою C++ згідно з завданням, отриманим від викладача за табл. 1: задану прямокутну матрицю $A=\{a_{ij}\}$ відсортувати за вказаним алгоритмом; для відсортованої матриці знайти значення функції $F(f_i(a_{ij}))$; алгоритм сортування оформити у вигляді функції-члена; обчислення $f_i(a_{ij})$ оформити у вигляді функції-члена; елементи матриці вводити з клавіатури; програма повинна вивести на екран відсортовану матрицю, всі значення $f_i(a_{ij})$ та значення функції $F(f_i(a_{ij}))$.

Використати клас двовірного масиву та функції-члени.

Варіант 5. Впорядкувати елементи стовпців матриці за зростанням їх значень методом бульбашки. $f_i(a_{ij})$ -середнє арифметичне значення елементів у кожному рядку матриці; $F(f_i(a_{ij}))$ -добуток $f_i(a_{ij})$.

2. Блок-схеми основної програми та окремих функцій

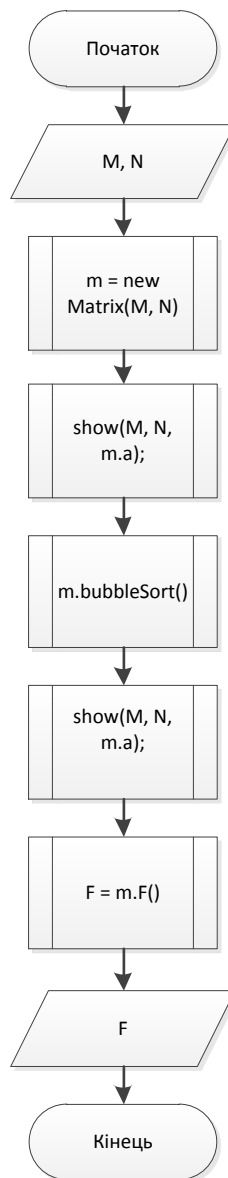


Рис. 2.1. Блок-схема основної програми

Блок-схема конструктора класу матриці **class Matrix{...}** (розмірності m на n):
public Matrix(int m, int n), зображена на рис. 2.2.

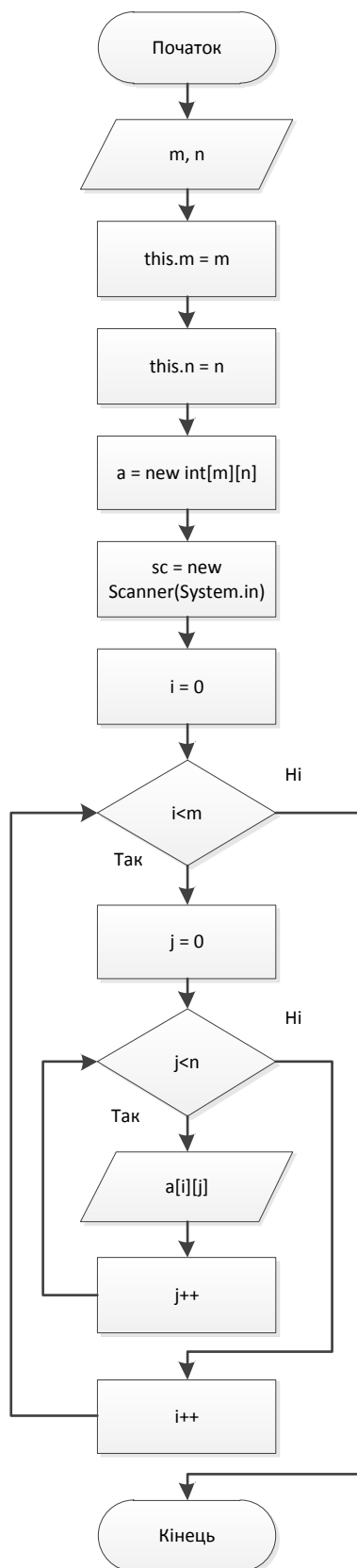


Рис. 2.2. Конструктор **public Matrix(int m, int n)**

Блок-схема сортування елементів стовбців "методом бульбашки", зображена на рис. 2.3.

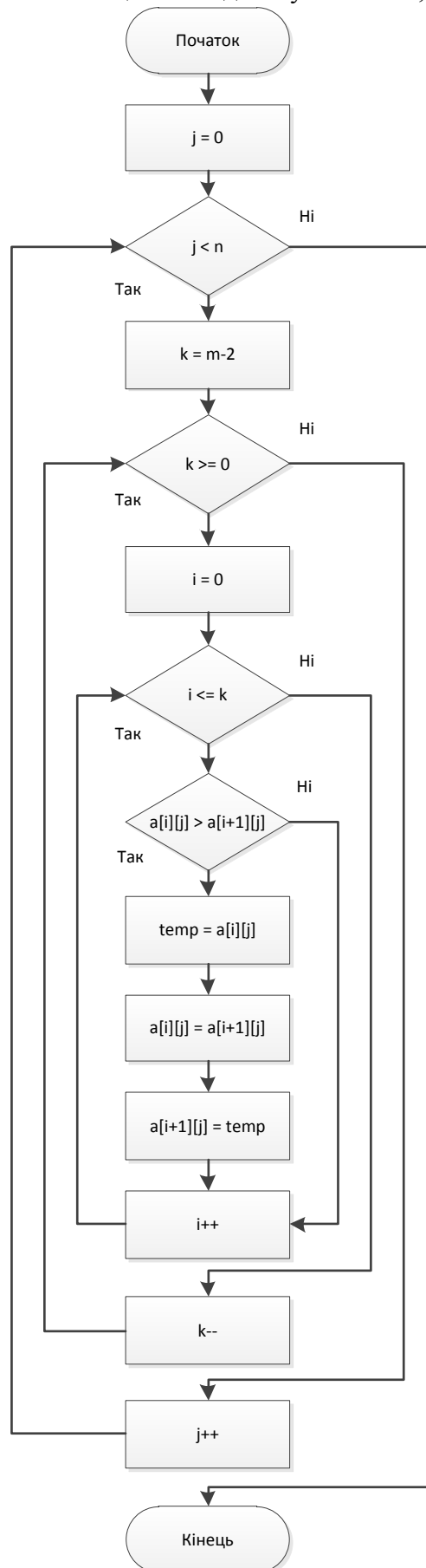


Рис. 2.3. Блок-схема функції-члену класу матриці: ***public void bubbleSort()***

Блок-схема функції-члену класу матриці **class Matrix{...}**, призначеної для обчислення середнього арифметичного значення елементів у i -го рядка матриці: **public double f(int i)** зображена на рис. 2.4. На рис. 2.5. зображено блок-схему функції-члена **public double F()**, призначеної для обчислення добутку значень $f(i)$ для всіх рядків матриці.

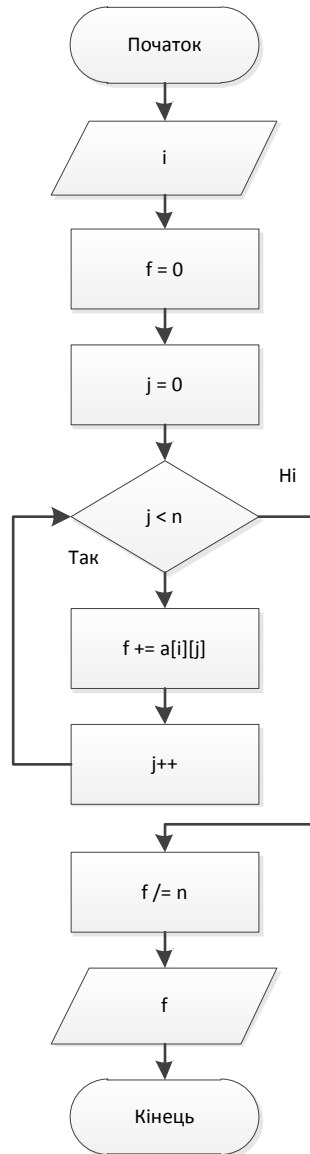


Рис. 2.4. Функція-член **double f(int i)**

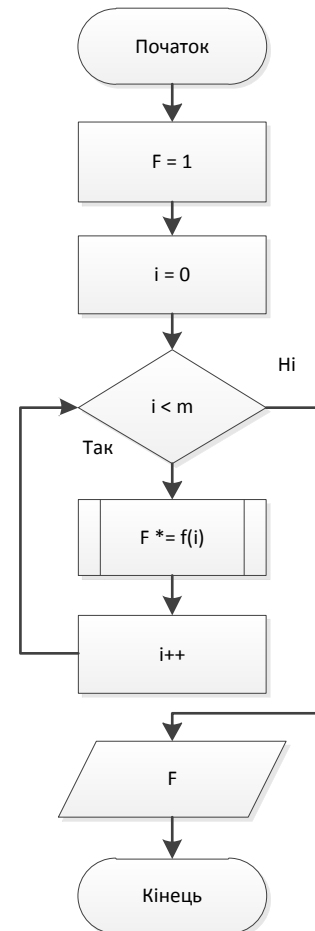


Рис. 2.5. Функція-член **double F()**

Блок-схема функції друку двовимірної матриці (розмірності m на n): **public void bubbleSort()**, зображена на рис. 2.6.

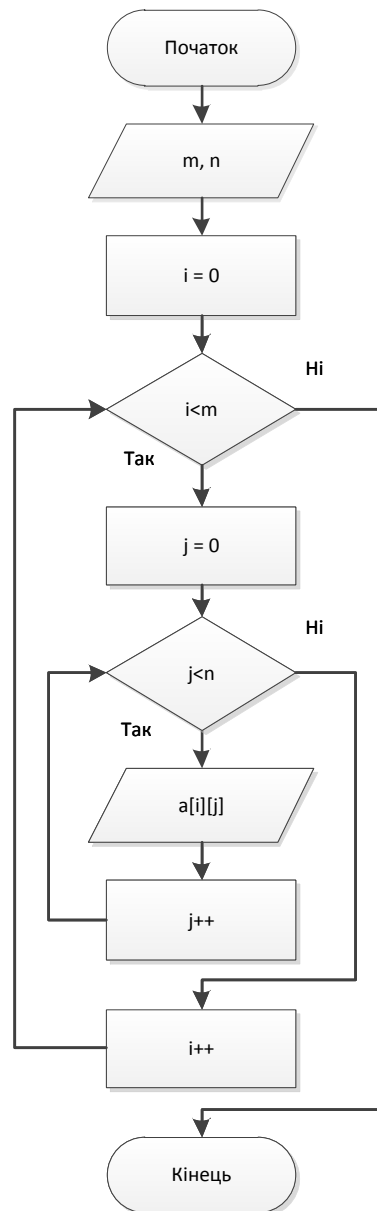


Рис. 2.6. Блок-схема функції друку двовимірної матриці:
public static void show(int m, int n, int[][] a)

3. Текст програми на мові програмування JAVA

```
package _oop_lab1;
import java.util.Scanner;

/**
 * Л/р № 1 (варіант 5)
 * Створення класів. Оголошення об'єктів. Доступ до змінних та методів
 * класів через об'єкт. Реалізація конструкторів з параметрами.
 * @author Taras
 */

//клас Матриці
class Matrix {
    //властивості класу
    public int m, n; //розмір масиву a
    public int[][] a;

    //конструктор класу
    public Matrix(int m, int n) {
        //збережемо розмір матриці у властивості класу
        this.m = m;
        this.n = n;
        //створимо масив розміром m на n
        a = new int[m][n];

        // створимо екземпляр класу Scanner для читання чисел з консолі
        Scanner sc = new Scanner(System.in);
        for(int i = 0; i < m; i++)
            for(int j = 0; j < n; ) {
                System.out.print("a[" + (i+1) + "][" + (j+1) + "]: ");
                // повертає істину, якщо з потоку можна зчитати ціле число
                if(sc.hasNextInt()) {
                    // зчитуємо ціле число і зберігаємо значення в елемент
масиву
                    a[i][j] = sc.nextInt();
                    j++;
                }
                else {
                    /* Якщо користувач ввів не ціле число виводимо відповідне
попередження,
                    * поки не буде введенно коректне значення
                    */
                    System.out.println("It's not integer value! Try
again...");
                    sc.next();
                }
            }
    }

    //сортування елементів стовбців матриці "методом бульбашки"
    public void bubbleSort() {
        //перебір всіх стовбців масиву
        for(int j = 0; j < n; j++)
```

```

        //сам "bubble sort"
        for(int k = m-2; k >= 0; k--)
            for(int i = 0; i <= k; i++)
                if(a[i][j] > a[i+1][j]) {
                    int temp = a[i][j];
                    a[i][j] = a[i+1][j];
                    a[i+1][j] = temp;
                }
    }

    //середнє арифметичне i-го рядка матриці
    public double f(int i){
        double f = 0;
        for(int j = 0; j < n; j++)
            f += a[i][j];

        //перехопимо помилку "ділення на нуль"
        try {
            f /= n;
        } catch (ArithmeticException e) {
            f = 0;
        }
        System.out.print("" + f + "\t");
        return f;
    }

    //добуток значень f(i) всіх рядків матриці
    public double F() {
        double F = 1;
        System.out.println("\nf():");
        for(int i = 0; i < m; i++)
            F *= f(i);
        return F;
    }
}

//головний клас пакету
public class _OOP_Lab1 {
    static final int M = 4; //константа кількості стрічок матриці
    static final int N = 3; //константа кількості стовбців матриці

    //виведення матриці розміром m на n на екран
    public static void show(int m, int n, int[][] a) {
        for(int i = 0; i < m; i++) {
            for(int j = 0; j < n; j++)
                System.out.print("" + a[i][j] + "\t");
            System.out.println();
        }
    }

    public static void main(String[] args) {
        //ініціалізація класу матриці
        Matrix m = new Matrix(M, N);
    }
}

```



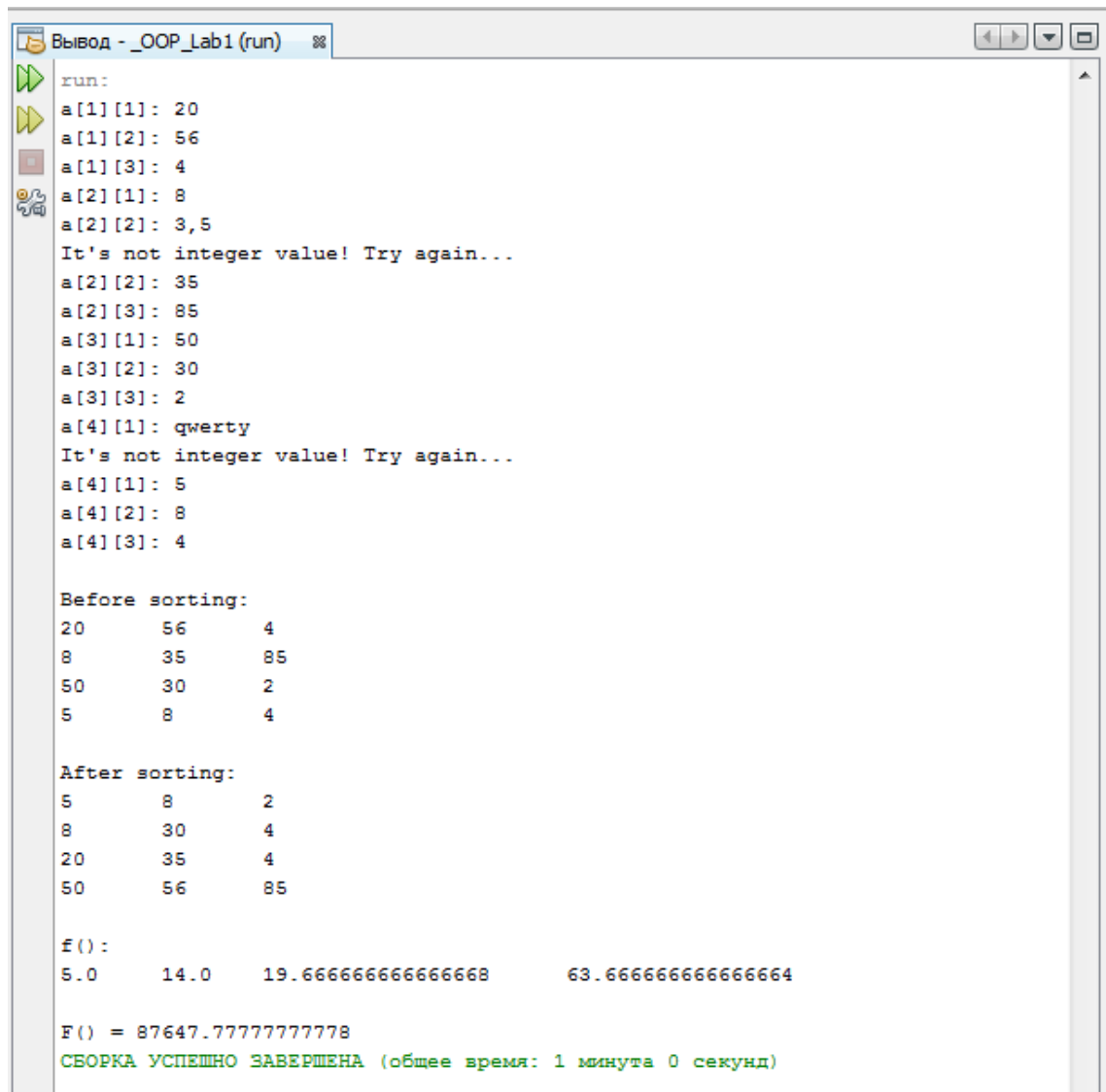
```

        System.out.println("\nBefore sorting:");
        //покажемо матрицю до сортування
        show(M, N, m.a);
        //Відсортуємо
        m.bubbleSort();
        System.out.println("\nAfter sorting:");
        //... і після сортування
        show(M, N, m.a);
        //обчислимо F()
        System.out.println("\n\nF() = " + m.F());
    }
}

```

4. Результат виконання програми

Запустимо програму на виконання (рис. 4.1). Введемо значення елементів вхідної матриці. В тому числі, спробуємо ввести дані, відмінні від цілочисельних.



```
Вывод - _OOP_Lab1 (run)
run:
a[1][1]: 20
a[1][2]: 56
a[1][3]: 4
a[2][1]: 8
a[2][2]: 3,5
It's not integer value! Try again...
a[2][2]: 35
a[2][3]: 85
a[3][1]: 50
a[3][2]: 30
a[3][3]: 2
a[4][1]: qwerty
It's not integer value! Try again...
a[4][1]: 5
a[4][2]: 8
a[4][3]: 4

Before sorting:
20      56      4
8        35     85
50       30      2
5         8       4

After sorting:
5         8       2
8        30      4
20       35      4
50       56     85

f():
5.0      14.0     19.666666666666668      63.666666666666664

F() = 87647.77777777778
СБОРКА УСПЕШНО ЗАВЕРШЕНА (общее время: 1 минута 0 секунд)
```

Рис. 4.1. Результат виконання програми

Як бачимо на рисунку, програма «перехоплює» не коректні дані, пропонуючи ввести ще раз цілочисельне значення елемента масиву. Елементи стовбців матриці відсортовані у зростаючому порядку. Значення функції $f(i)$ є також вірними, оскільки:

$$f(1) = \frac{5 + 8 + 2}{3} = 5$$
$$f(2) = \frac{8 + 30 + 4}{3} = 14$$
$$f(3) = \frac{20 + 35 + 4}{3} = 19,6(6)$$
$$f(4) = \frac{50 + 56 + 85}{3} = 63,6(6)$$

Функція $F()$ також повертає правильний результат, оскільки:

$$F = 5 \times 14 \times 19,6(6) \times 63,6(6) = 87647,7(7)$$

5. ВИСНОВКИ

На даній лабораторній роботі я навчився створювати класи, оголошувати об'єкти на мові програмування JAVA. Також вивчив особливості доступу до змінних та методів класів через об'єкт, реалізовувати конструктори класів з параметрами.