

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

**Український державний університет науки і технологій**

Кафедра «Комп’ютерні інформаційні технології»

**Лабораторна робота №15  
з дисципліни «Об’єктно орієнтованне програмування»**

## на тему:

**«Робота з файловими потоками Java»**

Виконав:

студент гр. ПЗ2011

Кулик С. В.

Прийняв:

Андрющенко В. О.

Дніпро, 2022

**Тема:** Робота з файловими потоками Java.

**Мета:** Отримати навички використання файлових потоків Java.

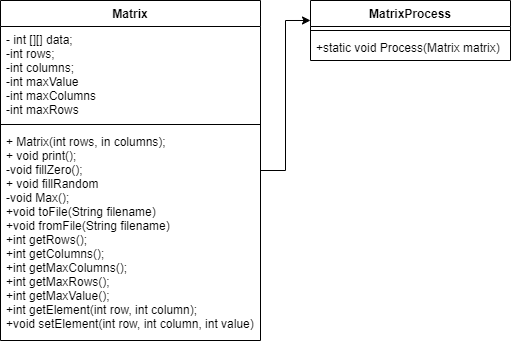
**Завдання**

Написати об'єктно-орієнтовану програму на мові Java, яка використовує файлові потоки для збереження та зчитування матриці з файлу. Індивідуальні завдання взяти з попередньої лабораторної роботи.

**Індивідуальне завдання**

Дана дійсна матриця. Переставляючи її рядки і стовпці, домогтися того, щоб найбільший елемент (один з них) виявився в верхньому лівому кутку.

**Діаграма класів**



*Рис. 1 – Діаграма класів*

**Опис файлових потоків, які використовуються у програмі**

***Байтові потоки***

Потоки байтів Java використовуються для здійснення введення і виведення 8-бітних байтів. Незважаючи на безліч класів, пов'язаних з потоками байтів, найбільш поширене використання таких класів: FileInputStream та FileOutputStream.

Потік FileInputStream – це потік, який використовується Java для читання даних з файлу. Об'єкти можуть бути створені за допомогою ключового слова new, доступні кілька типів конструкторів.

Потік FileOutputStream – це потік, який використовується в Java для створення файлу та наступного запису до нього. Потік створить файл у разі його відсутності перед його відкриттям для виведення.

***Символьні потоки***

Потоки байтів у Java дозволяють зробити введення та виведення 8-бітних байтів, у той час як потоки символів використовуються для введення та виведення 16-бітного юнікоду. Незважаючи на безліч класів, пов'язаних із потоками символів, найбільш поширене використання таких класів: FileReader та FileWriter. Незважаючи на той факт, що внутрішній FileReader використовує FileInputStream, і FileWriter використовує FileOutputStream, основна відмінність полягає в тому, що FileReader здійснює зчитування двох байтів у конкретний момент часу, у той час як FileWriter робить запис двох байтів за той же час.

**Опис патерну Декоратор та його застосування в роботі з файлами**

Патерн Декоратор дозволяє створили невеликі класи-обертки, які додають нової поведінки до або після основної роботи вкладеної бізнес-логіки.

Перша обгортка шифрує та розшифрує дані, а друга – стискає та розпакує їх.

Ми можемо використовувати обгортки як окремо, так і всі разом, обернувши один декоратор іншим.

**Код програми**

**Файл Main.java**

package com.company;  
  
import java.util.Scanner;  
  
public class Main {  
  
 public static void main(String[] args) {  
 Matrix matrix = new Matrix(0, 0);  
 //меню користувача для обробки матриці  
 while (true) {  
 System.out.println("Виберіть дію: ");  
 System.out.println("1. Введення матриці");  
 System.out.println("2. Заповнити матрицю випадковими значеннями");  
 System.out.println("3. Записати матрицю в файл");  
 System.out.println("4. Взяти матрицю з файлу");  
 System.out.println("5. Виведення матриці");  
 System.out.println("6. Обробка матриці");  
 System.out.println("7. Вихід");  
 int choice = Integer.parseInt(new Scanner(System.in).nextLine());  
 switch (choice) {  
 case 1:  
 System.out.println("Введіть кількість рядків:");  
 int rows = Integer.parseInt(new Scanner(System.in).nextLine());  
 System.out.println("Введіть кількість стовпчиків:");  
 int columns = Integer.parseInt(new Scanner(System.in).nextLine());  
 matrix = new Matrix(rows, columns);  
 break;  
 case 2:  
 matrix.fillRandom();  
 break;  
 case 3:  
 System.out.println("Введіть ім'я файлу:");  
 String fileName = new Scanner(System.in).nextLine();  
 matrix.toFile(fileName);  
 break;  
 case 4:  
 System.out.println("Введіть ім'я файлу:");  
 fileName = new Scanner(System.in).nextLine();  
 matrix.fromFile(fileName);  
 break;  
 case 5:  
 matrix.print();  
 break;  
 case 6:  
 MatrixProcess.Process(matrix);  
 break;  
 case 7:  
 System.exit(0);  
 break;  
 default:  
 System.out.println("Введено невірне значення");  
 }  
 }  
 }  
}

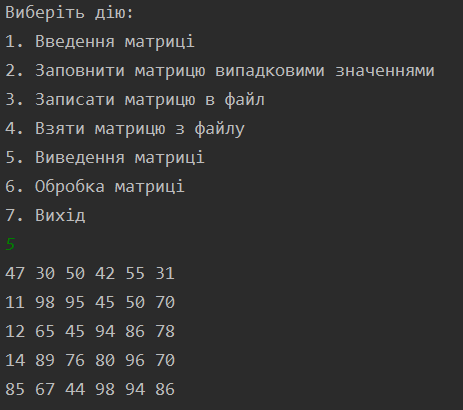
**Файл Matrix.java**

package com.company;  
import java.io.\*;  
  
public class Matrix  
{  
 private int[][] data;  
 private int rows;  
 private int columns;  
 private int MaxValue;  
 private int maxColumns;  
 private int maxRows;  
  
 public Matrix(int rows, int columns)  
 {  
 this.rows = rows;  
 this.columns = columns;  
 data = new int[rows][columns];  
 fillZero();  
 Max();  
 }  
  
 public void print()  
 {  
 for (int i = 0; i < rows; i++)  
 {  
 for (int j = 0; j < columns; j++)  
 {  
 System.out.print(data[i][j] + " ");  
 }  
 System.out.println();  
 }  
 }  
  
 private void fillZero()  
 {  
 for(int i = 0; i < rows; i++)  
 {  
 for(int j = 0; j < columns; j++)  
 {  
 data[i][j] = 0;  
 }  
 }  
 }  
  
 public void fillRandom()  
 {  
 for(int i = 0; i < rows; i++)  
 {  
 for(int j = 0; j < columns; j++)  
 {  
 data[i][j] = (int)(Math.random() \* 90 + 11);  
 }  
 }  
 }  
  
 private void Max()  
 {  
 int max = 0;  
 for (int i = 0; i < rows; i++)  
 {  
 for (int j = 0; j < columns; j++)  
 {  
 if (data[i][j] > max)  
 {  
 max = data[i][j];  
 maxColumns = j;  
 maxRows = i;  
 }  
 }  
 }  
 MaxValue = max;  
 }  
  
 public void toFile(String fileName)  
 {  
 try  
 {  
 FileWriter writer = new FileWriter(fileName);  
 //записать размерность матрицы  
 writer.write(rows + " " + columns + "\n");  
 for (int i = 0; i < rows; i++)  
 {  
 for (int j = 0; j < columns; j++)  
 {  
 writer.write(data[i][j] + " ");  
 }  
 writer.write("\n");  
 }  
 writer.close();  
 }  
 catch (IOException e)  
 {  
 System.out.println("Error");  
 }  
 }  
  
 public void fromFile(String fileName)  
 {  
 //считать размерность из файла  
 try  
 {  
 FileReader reader = new FileReader(fileName);  
 BufferedReader bufferedReader = new BufferedReader(reader);  
 String line = bufferedReader.readLine();  
 String[] size = line.split(" ");  
 rows = Integer.parseInt(size[0]);  
 columns = Integer.parseInt(size[1]);  
 data = new int[rows][columns];  
 for (int i = 0; i < rows; i++)  
 {  
 line = bufferedReader.readLine();  
 String[] str = line.split(" ");  
 for (int j = 0; j < columns; j++)  
 {  
 data[i][j] = Integer.parseInt(str[j]);  
 }  
 }  
 bufferedReader.close();  
 }  
 catch (IOException e)  
 {  
 System.out.println("Error");  
 }  
 }  
  
  
  
  
 public int getRows()  
 {  
 return rows;  
 }  
  
 public int getColumns()  
 {  
 return columns;  
 }  
  
 public int getMaxColumns()  
 {  
 Max();  
 return maxColumns;  
 }  
  
 public int getMaxRows()  
 {  
 Max();  
 return maxRows;  
 }  
  
 public int getMaxValue()  
 {  
 return MaxValue;  
 }  
 public int getElement(int row, int column)  
 {  
 return data[row][column];  
 }  
  
 public void setElement(int row, int column, int value)  
 {  
 data[row][column] = value;  
 }  
}

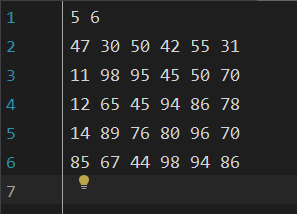
**Файл MatrixProcess.java**

package com.company;  
  
public class MatrixProcess  
{  
 public static void Process(Matrix matrix)  
 {  
 int maxRow = matrix.getMaxRows();  
 int maxCol = matrix.getMaxColumns();  
  
 for (int i = 0; i < matrix.getColumns(); i++)  
 {  
 int temp = matrix.getElement(0, i);  
 matrix.setElement(0, i, matrix.getElement(maxRow, i));  
 matrix.setElement(maxRow, i, temp);  
  
 }  
 for (int i = 0; i < matrix.getRows(); i++)  
 {  
 int temp = matrix.getElement(i, 0);  
 matrix.setElement(i, 0, matrix.getElement(i, maxCol));  
 matrix.setElement(i, maxCol, temp);  
 }  
 }  
}

**Приклад роботи програми**

****

*Рис. 2 – Матриця в програмі*

**

*Рис. 3 – Матриця в файлі*

Під час запису матриці в файл записується і розмірність файлу, а під час считування матриці з файлу, розмірність також считується та виділяється пам’ять під нову матрицю та заповнюється значеннями з файлу.

**Висновок**

Робота з файловими потоками є важливою і відносно не складною темою у мові java, а також часто використовується при написанні додатків або програм роботи с даними, які записуються або зчитуються з інших носіїв. Є можливість читання та запису по одному байту або по два.

Для полегшення роботи з файлами а також збільшення можливостей можливо застосовувати у розробці такий патерн як Декоратор.