Міністерство освіти і науки України
Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра обчислювальної техніки

Лабораторна робота ЛР1.3
ПОТОКИ В МОВІ JAVA
з дисципліни
Програмне забезпечення високопродуктивних комп'ютерних систем

Виконав
Студент групи ІМ-13
Котенко Ярослав Олегович

Перевірив
доц. Корочкін О.В.

Київ 2024

Завдання

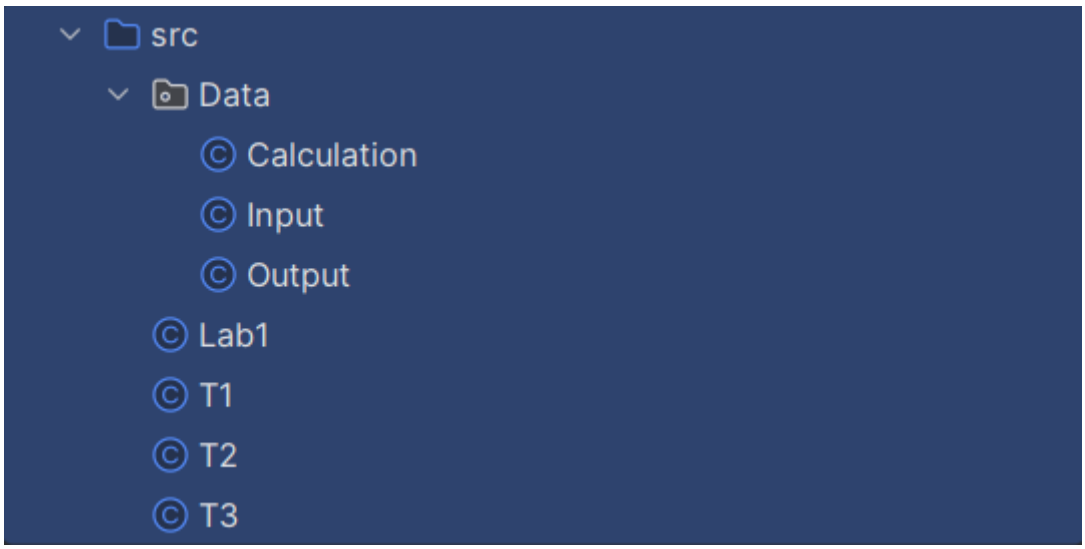
Розробити програму, яка містить паралельні потоки , що реалізують відповідну функцію F1, F2, F3 з Додатку Б згідно отриманому варіанту. В потоках використати метод join().

F1: (1.19) d = MAX(B + C) + MIN(A + B*(MA*ME))

F2: (2.27) MF = (MG*MH)*TRANS(MK)

F3: (3.28) s = MAX(S*MO) + MIN(MT*MS + MP)

Структура програми



Код програми

Основні файли (Файли де описуються потоки)

Файл: **Lab1**

```
// Лабораторна робота ЛР1.3 (Java)
// F1: (1.19)  d = MAX(B + C) + MIN(A + B*(MA*ME))
// F2: (2.27)  MF = (MG*MH)*TRANS(MK)
// F3: (3.28)  s = MAX(S*MO) + MIN(MT*MS + MP)
// Котенко Я. О. ІМ-13
// Дата 17.02.2024

import java.util.Scanner;

public class Lab1 {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("Enter the dimensions of vectors and matrices: ");
        final int N = scanner.nextInt();

        T1 t1 = new T1(N);
        T2 t2 = new T2(N);
```

```
        T3 t3 = new T3(N);
        t1.start();
        t2.start();
        t3.start();

        try {
            t1.join();
            t2.join();
            t3.join();
        } catch (InterruptedException e) {
            System.out.println(e);
        }

        System.out.println("All threads have finished. Exiting main
thread.");
    }
}
```

Файл: **T1**

```
import Data.Calculation;
import Data.Input;
import Data.Output;

import java.util.Scanner;

public class T1 extends Thread {

    //    1.19 d = MAX(B + C) + MIN(A + B*(MA*ME))
    private int N = 0;
    private double d;
    private double[] A;
    private double[] B;
    private double[] C;

    private double[][] MA;
    private double[][] ME;
    private int range = 1000;

    public T1(int N) {
        this.N = N;
    }

    public void run() {
        System.out.println("T1 is started");
        if (this.N < 20) {
            initData();
            setAllParamsByKeyboard();
            this.d = Calculation.F1(this.A, this.B, this.C, this.MA, this.ME);
            System.out.println("Result of first expression | T1: " + this.d);
        } else {
            Scanner scanner = new Scanner(System.in);
            System.out.println("\n" + "Choose the data generation method:\n" + "1 -
Setting all data elements to a specified value (e.g., 1, 2, or 3)\n" + "2 - Using a
random value generator. | T1");
            int generateMethod = scanner.nextInt();
            if (generateMethod == 1) {
                initData();
                double[][] vectors = setVectorsBySpecifiedValue();
                double[][][] matrices = setMatrixBySpecifiedValue();
                writeDataToFiles(vectors[0], vectors[1], vectors[2], matrices[0],
matrices[1]);
                readAndSetDataFromFile();
                this.d = Calculation.F1(this.A, this.B, this.C, this.MA, this.ME);
```

```
                Output.writeResultToFile("resultF1.txt", this.d);
            } else if (generateMethod == 2) {
                initData();
                double[][] vectors = setVectorsByRandomGeneration(this.range);
                double[][][] matrices = setMatrixByRandomGeneration(this.range);
                writeDataToFiles(vectors[0], vectors[1], vectors[2], matrices[0],
matrices[1]);
                readAndSetDataFromFile();
                this.d = Calculation.F1(this.A, this.B, this.C, this.MA, this.ME);
                Output.writeResultToFile("resultF1.txt", this.d);
            } else {
                throw new IllegalArgumentException("Unknown generation method!");
            }
        }
        System.out.println("T1 is finished");
    }

    private void initData() {
        this.A = new double[this.N];
        this.B = new double[this.N];
        this.C = new double[this.N];
        this.MA = new double[this.N][this.N];
        this.ME = new double[this.N][this.N];
    }

    private void setAllParamsByKeyboard() {
        this.A = Input.setVectorByKeyboard(this.A, "A", this.N, "T1");
        this.B = Input.setVectorByKeyboard(this.B, "B", this.N, "T1");
        this.C = Input.setVectorByKeyboard(this.C, "C", this.N, "T1");
        this.MA = Input.setMatrixByKeyboard(this.MA, "MA", this.N, "T1");
        this.ME = Input.setMatrixByKeyboard(this.ME, "ME", this.N, "T1");
    }

    private double[][] setVectorsByRandomGeneration(int range) {
        double[] vectorA = Input.setVectorByRandomValues(this.A, this.N, range);
        double[] vectorB = Input.setVectorByRandomValues(this.B, this.N, range);
        double[] vectorC = Input.setVectorByRandomValues(this.C, this.N, range);
        return new double[][]{vectorA, vectorB, vectorC};
    }

    private double[][][] setMatrixByRandomGeneration(int range) {
        double[][] matrixMA = Input.setMatrixByRandomValues(this.MA, this.N, range);
        double[][] matrixME = Input.setMatrixByRandomValues(this.ME, this.N, range);
        return new double[][][]{matrixMA, matrixME};
    }

    private double[][] setVectorsBySpecifiedValue() {
        double[] vectorA = Input.setVectorBySpecifiedValue(this.A, this.N, 1);
        double[] vectorB = Input.setVectorBySpecifiedValue(this.B, this.N, 1);
        double[] vectorC = Input.setVectorBySpecifiedValue(this.C, this.N, 1);
        return new double[][]{vectorA, vectorB, vectorC};
    }

    private double[][][] setMatrixBySpecifiedValue() {
        double[][] matrixMA = Input.setMatrixBySpecifiedValue(this.MA, this.N, 1);
        double[][] matrixME = Input.setMatrixBySpecifiedValue(this.ME, this.N, 1);
        return new double[][][]{matrixMA, matrixME};
    }

    private void writeDataToFiles(double[] vectorA, double[] vectorB, double[]
vectorC, double[][] matrixMA, double[][] matrixME) {
        Input.writeVectorToFile("Vector_A.txt", vectorA);
        Input.writeVectorToFile("Vector_B.txt", vectorB);
        Input.writeVectorToFile("Vector_C.txt", vectorC);
        Input.writeMatrixToFile("Matrix_MA.txt", matrixMA);
        Input.writeMatrixToFile("Matrix_ME.txt", matrixME);
    }

    private void readAndSetDataFromFile() {
        this.A = Input.readVectorFromFile("Vector_A.txt", this.N);
        this.B = Input.readVectorFromFile("Vector_B.txt", this.N);
```

```
        this.C = Input.readVectorFromFile("Vector_C.txt", this.N);
        this.MA = Input.readMatrixFromFile("Matrix_MA.txt", this.N);
        this.ME = Input.readMatrixFromFile("Matrix_ME.txt", this.N);
    }
}
```

Файл: **T2**

```
import Data.Calculation;
import Data.Input;
import Data.Output;

import java.util.Scanner;

public class T2 extends Thread {

    // MF = (MG*MH)*TRANS(MK)
    private int N = 0;
    private double[][] MF;
    private double[][] MG;
    private double[][] MH;
    private double[][] MK;
    private int range = 1000;

    public T2(int N) {
        this.N = N;
    }

    public void run() {
        System.out.println("T2 is started");
        if (this.N < 20) {
            initData();
            setAllParamsByKeyboard();
            this.MF = Calculation.F2(this.MG, this.MH, this.MK);
            System.out.println("Res T2: ");
            Output.printMatrix(this.MF, this.N, "MF", "T2");
        } else {
            Scanner scanner = new Scanner(System.in);
            System.out.println("\n" + "Choose the data generation method:\n" + "1 -
Setting all data elements to a specified value (e.g., 1, 2, or 3)\n" + "2 - Using a
random value generator. | T2");
            int generateMethod = scanner.nextInt();
            if (generateMethod == 1) {
                initData();
                double[][][] matrices = setMatrixBySpecifiedValue();
                writeDataToFiles(matrices[0], matrices[1], matrices[2]);
                readAndSetDataFromFile();
                this.MF = Calculation.F2(this.MG, this.MH, this.MK);
                Output.writeResultToFile("resultF2.txt", this.MF);
            } else if (generateMethod == 2) {
                initData();
                double[][][] matrices = setMatrixByRandomGeneration(this.range);
                writeDataToFiles(matrices[0], matrices[1], matrices[2]);
                readAndSetDataFromFile();
                this.MF = Calculation.F2(this.MG, this.MH, this.MK);
                Output.writeResultToFile("resultF2.txt", this.MF);
            } else {
                throw new IllegalArgumentException("Unknown generation method!");
            }
        }
        System.out.println("T2 is finished");
    }

    private void initData() {
        this.MF = new double[this.N][this.N];
        this.MG = new double[this.N][this.N];
        this.MH = new double[this.N][this.N];
        this.MK = new double[this.N][this.N];
    }
```

```
    private void setAllParamsByKeyboard() {
        this.MG = Input.setMatrixByKeyboard(this.MG, "MG", this.N, "T2");
        this.MH = Input.setMatrixByKeyboard(this.MH, "MH", this.N, "T2");
        this.MK = Input.setMatrixByKeyboard(this.MK, "MK", this.N, "T2");
    }

    private double[][][] setMatrixBySpecifiedValue() {
        double[][] matrixMG = Input.setMatrixBySpecifiedValue(this.MG, this.N, 2);
        double[][] matrixMH = Input.setMatrixBySpecifiedValue(this.MH, this.N, 2);
        double[][] matrixMK = Input.setMatrixBySpecifiedValue(this.MK, this.N, 2);
        return new double[][][]{matrixMG, matrixMH, matrixMK};
    }

    private double[][][] setMatrixByRandomGeneration(int range) {
        double[][] matrixMG = Input.setMatrixByRandomValues(this.MG, this.N, range);
        double[][] matrixMH = Input.setMatrixByRandomValues(this.MH, this.N, range);
        double[][] matrixMK = Input.setMatrixByRandomValues(this.MK, this.N, range);
        return new double[][][]{matrixMG, matrixMH, matrixMK};
    }

    private void writeDataToFiles(double[][] matrixMG, double[][] matrixMH, double[][]
matrixMK) {
        Input.writeMatrixToFile("Matrix_MG.txt", matrixMG);
        Input.writeMatrixToFile("Matrix_MH.txt", matrixMH);
        Input.writeMatrixToFile("Matrix_MK.txt", matrixMK);
    }

    private void readAndSetDataFromFile() {
        this.MG = Input.readMatrixFromFile("Matrix_MG.txt", this.N);
        this.MH = Input.readMatrixFromFile("Matrix_MH.txt", this.N);
        this.MK = Input.readMatrixFromFile("Matrix_MK.txt", this.N);
    }

}
```

Файл: **T3**

```
import Data.Calculation;
import Data.Input;
import Data.Output;

import java.util.Scanner;

public class T3 extends Thread {
    //MAX(S*MO) + MIN(MT*MS + MP)
    private int N = 0;
    private double s;
    private double[] S;
    private double[][] MO;
    private double[][] MT;
    private double[][] MS;
    private double[][] MP;
    private int range = 1000;

    public T3(int N) {
        this.N = N;
    }

    public void run() {
        System.out.println("T3 is started");
        if (this.N < 20) {
            setAllParamsByKeyboard();
            this.s = Calculation.F3(this.S, this.MO, this.MT, this.MS, this.MP);
            System.out.println("Result of third expression | T3: " + this.s);
        } else {
            Scanner scanner = new Scanner(System.in);
            System.out.println("\n" + "Choose the data generation method:\n" + "1 -
Setting all data elements to a specified value (e.g., 1, 2, or 3)\n" + "2 - Using a
```

```
random value generator. | T3");
            int generateMethod = scanner.nextInt();
            if (generateMethod == 1) {
                initData();
                double[] vectorS = setVectorSbySpecifiedValue();
                double[][][] matrices = setMatrixBySpecifiedValue();
                writeDataToFiles(vectorS, matrices[0], matrices[1], matrices[2],
matrices[3]);
                readAndSetDataFromFile();
                this.s = Calculation.F3(this.S, this.MO, this.MT, this.MS, this.MP);
                Output.writeResultToFile("resultF3.txt", this.s);
            } else if (generateMethod == 2) {
                initData();
                double[] vectorS = setVectorSbyRandomGeneration(this.range);
                double[][][] matrices = setMatrixByRandomGeneration(this.range);
                writeDataToFiles(vectorS, matrices[0], matrices[1], matrices[2],
matrices[3]);
                readAndSetDataFromFile();
                this.s = Calculation.F3(this.S, this.MO, this.MT, this.MS, this.MP);
                Output.writeResultToFile("resultF3.txt", this.s);
            } else {
                throw new IllegalArgumentException("Unknown generation method!");
            }
        }
        System.out.println("T3 is finished");
    }

    private void initData() {
        this.S = new double[this.N];
        this.MO = new double[this.N][this.N];
        this.MT = new double[this.N][this.N];
        this.MS = new double[this.N][this.N];
        this.MP = new double[this.N][this.N];
    }

    private void setAllParamsByKeyboard() {
        initData();
        this.S = Input.setVectorByKeyboard(this.S, "S", this.N, "T3");
        this.MO = Input.setMatrixByKeyboard(this.MO, "MO", this.N, "T3");
        this.MT = Input.setMatrixByKeyboard(this.MT, "MT", this.N, "T3");
        this.MS = Input.setMatrixByKeyboard(this.MS, "MS", this.N, "T3");
        this.MP = Input.setMatrixByKeyboard(this.MP, "MP", this.N, "T3");
    }

    private double[] setVectorSbyRandomGeneration(int range) {
        return Input.setVectorByRandomValues(this.S, this.N, range);
    }

    private double[] setVectorSbySpecifiedValue() {
        return Input.setVectorBySpecifiedValue(this.S, this.N, 3);
    }

    private double[][][] setMatrixBySpecifiedValue() {
        double[][] matrixMO = Input.setMatrixBySpecifiedValue(this.MO, this.N, 3);
        double[][] matrixMT = Input.setMatrixBySpecifiedValue(this.MT, this.N, 3);
        double[][] matrixMS = Input.setMatrixBySpecifiedValue(this.MS, this.N, 3);
        double[][] matrixMP = Input.setMatrixBySpecifiedValue(this.MP, this.N, 3);
        return new double[][][]{matrixMO, matrixMT, matrixMS, matrixMP};
    }

    private double[][][] setMatrixByRandomGeneration(int range) {
        double[][] matrixMO = Input.setMatrixByRandomValues(this.MO, this.N, range);
        double[][] matrixMT = Input.setMatrixByRandomValues(this.MT, this.N, range);
        double[][] matrixMS = Input.setMatrixByRandomValues(this.MS, this.N, range);
        double[][] matrixMP = Input.setMatrixByRandomValues(this.MP, this.N, range);
        return new double[][][]{matrixMO, matrixMT, matrixMS, matrixMP};
    }

    private void writeDataToFiles(double[] vectorS, double[][] matrixMO, double[][]
matrixMT, double[][] matrixMS, double[][] matrixMP) {
        Input.writeVectorToFile("Vector_S.txt", vectorS);
```

```
        Input.writeMatrixToFile("Matrix_MO.txt", matrixMO);
        Input.writeMatrixToFile("Matrix_MT.txt", matrixMT);
        Input.writeMatrixToFile("Matrix_MS.txt", matrixMS);
        Input.writeMatrixToFile("Matrix_MP.txt", matrixMP);
    }

    private void readAndSetDataFromFile() {
        this.S = Input.readVectorFromFile("Vector_S.txt", this.N);
        this.MO = Input.readMatrixFromFile("Matrix_MO.txt", this.N);
        this.MT = Input.readMatrixFromFile("Matrix_MT.txt", this.N);
        this.MS = Input.readMatrixFromFile("Matrix_MS.txt", this.N);
        this.MP = Input.readMatrixFromFile("Matrix_MP.txt", this.N);
    }
}
```

Допоміжні файли (Файли де виконуються всі допоміжні функції)

Файл: **Calculation**

```
package Data;

import java.util.Arrays;

public class Calculation {
    //    1.19 d = MAX(B + C) + MIN(A + B*(MA*ME))
    //    2.27 MF = (MG*MH)*TRANS(MK)
    //    3.28 MAX(S*MO) + MIN(MT*MS + MP)
    public static double F1(double[] A, double[] B, double[] C, double[][] MA,
double[][] ME) {
        return getMaximumFromVector(addTwoVectors(B, C)) +
getMinimumFromVector(addTwoVectors(A, multiplyVectorOnMatrix(B, multiplyTwoMatrix(MA,
ME))));
    }

    public static double[][] F2(double[][] MG, double[][] MH, double[][] MK) {
        return multiplyTwoMatrix(multiplyTwoMatrix(MG,MH), transposeMatrix(MK,
MK.length));
    }

    public static double F3(double[] S, double[][] MO, double[][] MT, double[][] MS,
double[][] MP) {
        return getMaximumFromVector(multiplyVectorOnMatrix(S, MO)) +
getMinimumFromMatrix(addTwoMartices(multiplyTwoMatrix(MT,MS), MP));
    }

    private static double[] addTwoVectors(double[] firstVector, double[] secondVector)
{
        double[] result = new double[firstVector.length];

        for (int i = 0; i < firstVector.length; i++) {
            result[i] = firstVector[i] + secondVector[i];
        }

        return result;
    }

    private static double[][] multiplyTwoMatrix(double[][] firstMatrix, double[][]
secondMatrix) {
        int N = firstMatrix.length;
        double[][] result = new double[N][N];

        for (int i = 0; i < N; i++) {
            for (int j = 0; j < N; j++) {
                for (int k = 0; k < N; k++) {
                    result[i][j] += firstMatrix[i][k] * secondMatrix[k][j];
                }
            }
        }
```

```
        return result;
    }

    private static double[] multiplyVectorOnMatrix(double[] vector, double[][] matrix)
{
        int N = vector.length;
        double[] result = new double[N];

        for (int i = 0; i < N; i++) {
            for (int j = 0; j < N; j++) {
                result[j] += vector[i] * matrix[i][j];
            }
        }

        return result;
    }

    private static double getMaximumFromVector(double[] vector) {
        return Arrays.stream(vector).max().getAsDouble();
    }

    private static double getMinimumFromVector(double[] vector) {
        return Arrays.stream(vector).min().getAsDouble();
    }

    private static double getMinimumFromMatrix(double[][] matrix) {
        double min = matrix[0][0];
        int length = matrix.length;

        for (int i = 0; i < length; i++) {
            for (int j = 0; j < length; j++) {
                if (matrix[i][j] < min) {
                    min = matrix[i][j];
                }
            }
        }
        return min;
    }
    private static double[][] transposeMatrix(double[][] matrix, int length) {
        double[][] transposedMatrix = new double[length][length];

        for (int i = 0; i < length; i++) {
            for (int j = 0; j < length; j++) {
                transposedMatrix[j][i] = matrix[i][j];
            }
        }

        return transposedMatrix;
    }

    private static double[][] addTwoMartices(double[][] firstMatrix, double[][]
secondMatrix) {
        int length = firstMatrix.length;

        double[][] sum = new double[length][length];

        for (int i = 0; i < length; i++) {
            for (int j = 0; j < length; j++) {
                sum[i][j] = firstMatrix[i][j] + secondMatrix[i][j];
            }
        }

        return sum;
    }
}
```

Файл: **Input**

```
package Data;

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

public class Input {
    public static double[] setVectorByKeyboard(double[] vector, String vectorName, int
length, String threadName) {
        Scanner scanner = new Scanner(System.in);
        for (int i = 0; i < length; i++) {
            System.out.printf("Enter element [%d] for vector %s | %s \n", i,
vectorName, threadName);
            vector[i] = scanner.nextDouble();
        }
        return vector;
    }

    public static double[][] setMatrixByKeyboard(double[][] matrix, String matrixName,
int length, String threadName) {
        Scanner scanner = new Scanner(System.in);
        for (int i = 0; i < length; i++) {
            for (int j = 0; j < length; j++) {
                System.out.printf("Enter element [%d][%d] for matrix %s| %s \n", i, j,
matrixName, threadName);
                matrix[i][j] = scanner.nextDouble();
            }
        }
        return matrix;
    }

    public static double[] setVectorByRandomValues(double[] vector, int length, int
range) {
        for (int i = 0; i < length; i++) {
            vector[i] = Math.round(Math.random() * range) / 100.0;
        }
        return vector;
    }

    public static double[][] setMatrixByRandomValues(double[][] matrix, int length,
int range) {
        for (int i = 0; i < length; i++) {
            for (int j = 0; j < length; j++) {
                matrix[i][j] = Math.round(Math.random() * range) / 100.0;
            }
        }
        return matrix;
    }

    public static double[] setVectorBySpecifiedValue(double[] vector, int length, int
value) {
        for (int i = 0; i < length; i++) {
            vector[i] = value;
        }
        return vector;
    }

    public static double[][] setMatrixBySpecifiedValue(double[][] matrix, int length,
int value) {
        for (int i = 0; i < length; i++) {
            for (int j = 0; j < length; j++) {
                matrix[i][j] = value;
            }
        }
        return matrix;
    }
```

```
    public static void writeVectorToFile(String filename, double[] vector) {
        try (FileWriter writer = new FileWriter(filename)) {
            StringBuilder content = new StringBuilder();
            for (double elem : vector) {
                content.append(elem + " ");
            }
            writer.write(String.valueOf(content));
        } catch (IOException e) {
            System.out.println("An error occurred while writing to the file " +
filename);
            System.out.println("Error: " + e);
        }
    }

    public static void writeMatrixToFile(String filename, double[][] matrix) {
        try (FileWriter writer = new FileWriter(filename)) {
            StringBuilder content = new StringBuilder();
            for (double[] array : matrix) {
                for (double elem : array) {
                    content.append(elem + " ");
                }
                content.append("\n");
            }
            writer.write(String.valueOf(content));
        } catch (IOException e) {
            System.out.println("An error occurred while writing to the file " +
filename);
            System.out.println("Error: " + e);
        }
    }

    public static double[] readVectorFromFile(String filename, int vectorLength) {
        double[] vector = new double[vectorLength];
        try (FileReader reader = new FileReader(filename)) {
            int character;
            StringBuilder content = new StringBuilder();
            while ((character = reader.read()) != -1) {
                content.append((char) character);
            }
            String[] values = content.toString().trim().split(" ");
            for (int i = 0; i < values.length; i++) {
                vector[i] = Double.parseDouble(values[i]);
            }
        } catch (IOException e) {
            System.out.println("An error occurred while reading from the file " +
filename);
            System.out.println("Error: " + e);
        }

        return vector;
    }

    public static double[][] readMatrixFromFile(String filename, int matrixLength) {
        double[][] matrix = new double[matrixLength][matrixLength];
        try (BufferedReader reader= new BufferedReader(new FileReader(filename))){
            String line;
            int row = 0;
            while ((line = reader.readLine()) != null && row < matrixLength) {
                String[] values = line.split("\\s+");
                for (int col = 0; col < Math.min(values.length, matrixLength); col++)
{
                    matrix[row][col] = Double.parseDouble(values[col]);
                }
                row++;
            }
        } catch (IOException e) {
            System.out.println("An error occurred while reading from the file " +
filename);
            System.out.println("Error: " + e);
        }
```

```
        return matrix;
    }
}
```

Файл: **Output**

```
package Data;

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

public class Output {
    public static void printVector(double[] vector, String vectorName, String thread)
{
        System.out.printf("vector %s: %s | %s\n", vectorName, Arrays.toString(vector),
thread);
    }

    public static void printMatrix(double[][] matrix, int length, String matrixName,
String thread) {
        System.out.printf("Matrix %s: | %s\n", matrixName, thread);
        for (double[] array : matrix) {
            for (double num : array) {
                System.out.print(num + " ");
            }
            System.out.println();
        }
    }

    public static void writeResultToFile(String filename, double result) {
        try (FileWriter writer = new FileWriter(filename)) {
            String strResult = "Result of expression: " + result;
            writer.write(String.valueOf(strResult));
        } catch (IOException e) {
            System.out.println("An error occurred while writing to the file " +
filename);
            System.out.println("Error: " + e);
        }
    }

    public static void writeResultToFile(String filename, double[][] result) {
        try (FileWriter writer = new FileWriter(filename)) {
            String strResult = "Result of expression: " + Arrays.deepToString(result);
            writer.write(String.valueOf(strResult));
        } catch (IOException e) {
            System.out.println("An error occurred while writing to the file " +
filename);
            System.out.println("Error: " + e);
        }
    }
}
```

Виконання лабораторної роботи

1. Вибір мови та засоби програмування потоків.

У даній лабораторній роботі було обрано мову програмування **Java** для реалізації потоків виконання, оскільки Java надає вбудовані зручні засоби для роботи з потоками. Використано клас **Thread** та методи **run(), start()** і **join()** для створення та керування потоками.

2. Проблема введення даних. Причина та шляхи подолання

Під час виконання лабораторної роботи виникла проблема з введенням даних, оскільки перший запит на введення виводився одночасно для кожного потоку. Це ускладнило виконання випадку з N = 4, оскільки не було можливості ввести кожному потоку лише значення 1, 2 або 3 відповідно.

Причина:

Це сталося через те, що кожен потік відділявся від основного потоку і виконував свій запит на введення даних одночасно з іншими потоками, не дозволяючи користувачеві визначити відповідне значення для кожного потоку в окремості.

```
C:\Users\yarik\.jdks\openjdk-21.0.2\bin\java.exe "-ja
Enter the dimensions of vectors and matrices:
4
T2 is started
T1 is started
T3 is started
Enter element [0][0] for matrix MG| T2
Enter element [0] for vector A | T1
Enter element [0] for vector S | T3
1
Enter element [1] for vector A | T1
```

Шляхи подолання:

На даний момент було знайдено одне рішення цієї проблеми - використання методу join для синхронізації потоків. Проте це рішення має свої обмеження, оскільки після синхронізації потоки вже не працюють паралельно, а виконуються послідовно.

```
        t1.start();
        t1.join();
        t2.start();
        t2.join();
        t3.start();
        t3.join();
```

```
Enter the dimensions of vectors and matrices:
4
T1 is started
Enter element [0] for vector A | T1
1
Enter element [1] for vector A | T1
1
Enter element [2] for vector A | T1
1
Enter element [3] for vector A | T1
```

Можливо в майбутньому, під час виконання наступних лабораторних робіт, будуть знайдені кращі методи для вирішення цієї проблеми.

3. Проблема виведення даних. Причина та шляхи подолання.

Додатково, виникла проблема з виведенням результатів, оскільки вони виводилися не всі разом після завершення останнього потоку, а по мірі завершення кожного потоку окремо. ?

Причина:

Це може відбуватися через те, що в паралельних програмах кожен потік може завершувати своє виконання в різний час, що призводить до несинхронізованого виведення результатів.

```
Enter element [1][1] for matrix ME| T1
3,4
Enter element [0][1] for matrix MS| T3
Res T2:
Matrix MF: | T2
1347.9209299999998 1119.21236
2319.4018 2113.6984
T2 is finished
5,6
Result of first expression | T1: 1510.9441920000002
T1 is finished
```

```
Enter element [1][0] for matrix MP| T3
7,8
Enter element [1][1] for matrix MP| T3
11,2
Result of third expression | T3: 200.57
T3 is finished
All threads have finished. Exiting main thre
```

Шляхи подолання:

Одним із рішень цієї проблеми може бути додавання методу **getResult** для кожного потоку, а потім, після завершення виконання, виведення всіх результатів за допомогою цього методу.

```
public void getResult() {
    System.out.println("Result of first expression | T1: " + this.d);
}
```

```
public void getResult() {
    System.out.println("Res T2: ");
    Output.printMatrix(this.MF, this.N, "MF", "T2");
}
```

```
public void getResult() {
    System.out.println("Result of third expression | T3: " + this.s);
}
```

```
T1 t1 = new T1(N);
T2 t2 = new T2(N);
T3 t3 = new T3(N);
t1.start();
t2.start();
t3.start();

try {
    t1.join();
    t2.join();
    t3.join();
} catch (InterruptedException e) {
    System.out.println(e);
}
t1.getResult();
t2.getResult();
t3.getResult();
```

```
3
T2 is finished
4
Enter element [1][1] for matrix ME| T1
5,67
Enter element [1][0] for matrix MS| T3
8,9
T1 is finished
```

```
T3 is finished
Result of first expression | T1: 970.4062
Res T2:
Matrix MF: | T2
1960.7651999999998 988.64186
3100.5588 1684.69554
Result of third expression | T3: 228.76999999999998
All threads have finished. Exiting main thread.
```

У поточній реалізації програми зроблено зміни, щоб під час виконання лише виводилось повідомлення про завершення кожного потоку, а результати виводилися після завершення роботи останнього потоку. Хоча це рішення дозволяє зменшити кількість виведених повідомлень та забезпечує вивід результатів у більш структурованому порядку, є сумніви щодо його оптимальності. В майбутніх лабораторних роботах можливо будуть вивчені та застосовані більш цікаві та ефективні методи вирішення цієї проблеми.

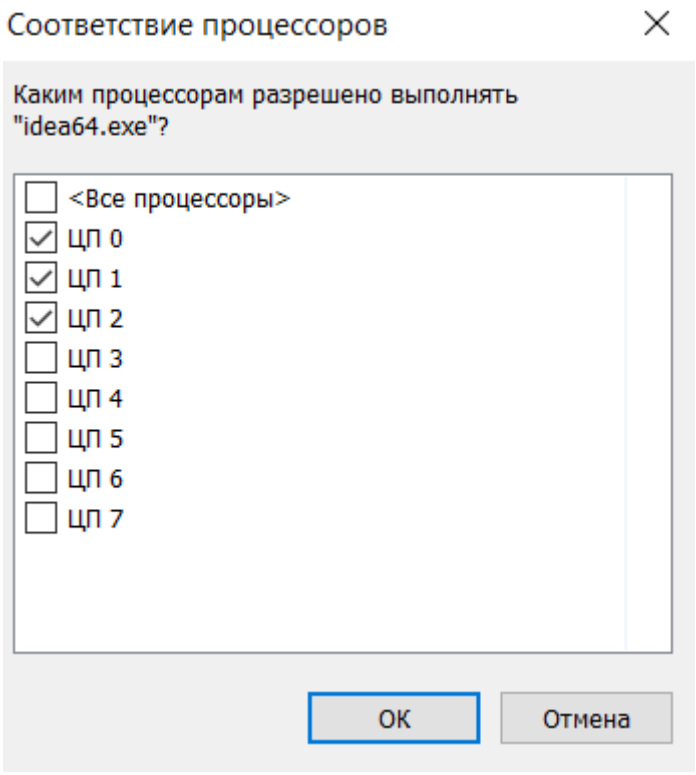
4. **Аналіз завантаження ядер процесора.**

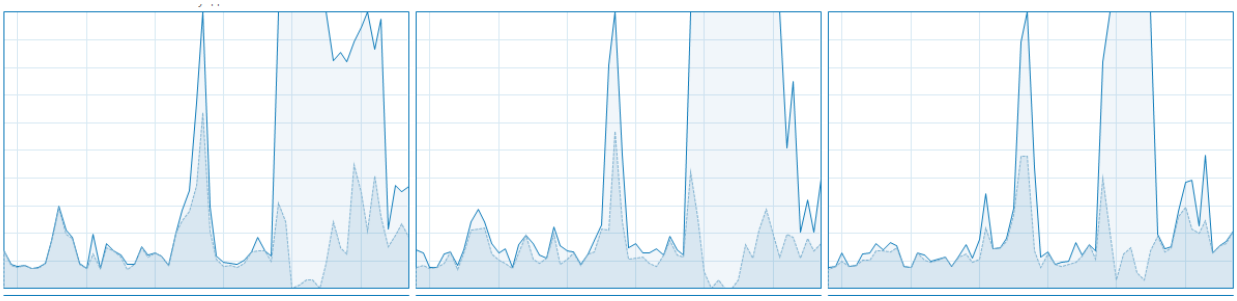
Опис комп'ютера:

AMD Ryzen 7 4700U with Radeon Graphics

Базовая скорость:	2,00 ГГц
Сокетов:	1
Ядра:	8
Логических процессоров:	8

Під час аналізу завантаження процесора було встановлено, що три процесора можуть бути використані для виконання програми IntelliJ IDEA. Після цього було запущено програму та введено значення N = 1000 з рандомним заповненням числами. Під час виконання програми було помічено раптове зростання навантаження на три ядра процесора. Однак, після завершення роботи програми, навантаження повернулося до попередніх значень.



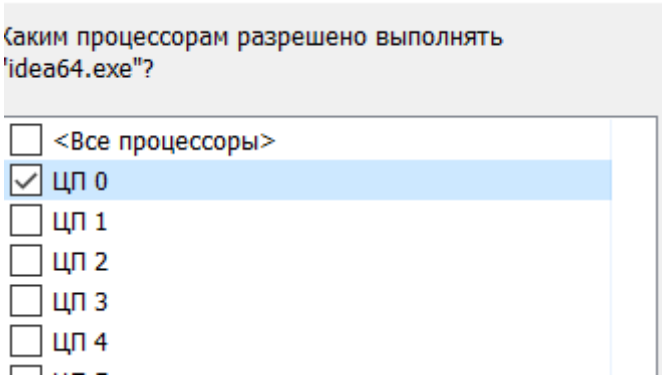


Для визначення часу виконання програми на одному ядрі та на чотирьох, було внесено невеликі зміни до головного файлу програми.

```
long startTime = System.currentTimeMillis();
Scanner scanner = new Scanner(System.in);
System.out.println("Enter the dimensions of vectors and matrices: ");
final int N = scanner.nextInt();

T1 t1 = new T1(N);
T2 t2 = new T2(N);
T3 t3 = new T3(N);
t1.start();
t2.start();
t3.start();

try {
    t1.join();
    t2.join();
    t3.join();
} catch (InterruptedException e) {
    System.out.println(e);
}
```

```
System.out.println("All threads have finished. Exiting main thread.");
long endTime = System.currentTimeMillis();
long totalTime = endTime - startTime;

System.out.println("Execution time of the program " + totalTime + " ms");
```

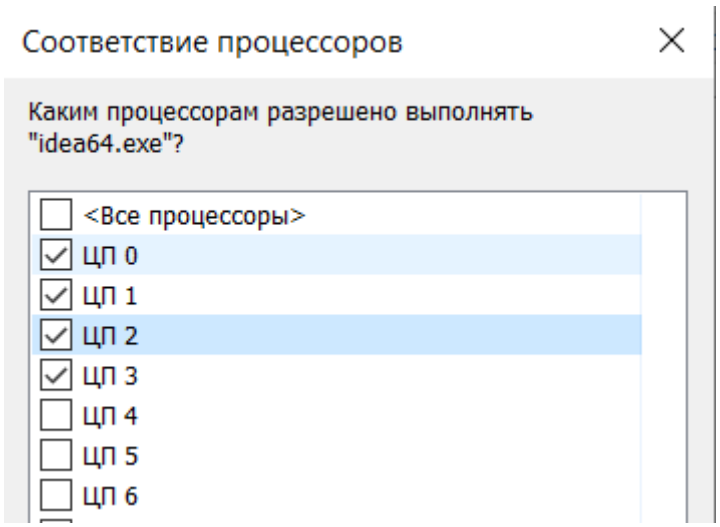
Час виконання програми на одному ядрі: **30192** мілісекунд ≈ **30,2** секунд.




```
Enter the dimensions of vectors and matrices:
1000
T1 is started

Choose the data generation method:
1 - Setting all data elements to a specified value (e.g., 1, 2, or 3)
2 - Using a random value generator. | T1
T2 is started
```

```
T3 is finished
T1 is finished
T2 is finished
All threads have finished. Exiting main thread.
Execution time of the program 30192 ms
```

Час виконання програми на чотирьох ядрах: 16854 мілісекунд ≈ 16.9 секунди.




```
Enter the dimensions of vectors and matrices:
1000
T1 is started
T2 is started
T3 is started
```

```
T3 is finished
T1 is finished
T2 is finished
All threads have finished. Exiting main thread.
Execution time of the program 16854 ms
```

Визначив коефіцієнт прискорення:

$K_y = T_1 / T_4 = 30.2 / 16.9 \approx 1.79$

Дивлячись на отриманий коефіцієнт прискорення, можна зробити висновок, що використання паралельних потоків дозволяє програмі працювати майже у два рази швидше, ніж у випадку використання лише одного потоку. Це підтверджує ефективність багатопоточного програмування в даному контексті і підкреслює значення оптимізації програм шляхом розпаралелювання обчислень.

Висновок

1. В рамках цієї лабораторної роботи було вибрано мову програмування Java для реалізації потоків виконання. Для створення та керування потоками використовувалися клас Thread разом із методами run(), start() і join().
2. Під час виконання лабораторної роботи виникла проблема з введенням даних через одночасний запит на введення для кожного потоку, ускладнюючи виконання випадку з N = 4. Це сталося через розділення кожного потоку від основного потоку, що призвело до конфлікту введення даних. Однак було знайдено рішення - використання методу join для синхронізації потоків, хоча це призводить до послідовного виконання потоків, замість паралельного.
3. Проблема з виведенням результатів у паралельних програмах виникає через асинхронне завершення потоків, що призводить до несинхронізованого виведення даних. Шляхи подолання цієї проблеми можуть включати в себе додавання методу getResult для кожного потоку та виведення всіх результатів після завершення їхньої роботи. Однак, існують сумніви щодо оптимальності цього підходу.
4. Отриманий коефіцієнт прискорення показує, що використання паралельних потоків у програмі значно збільшує її продуктивність. У даному випадку коефіцієнт прискорення становить близько 1.79, що свідчить про те, що програма працює майже у два рази швидше при використанні чотирьох ядер порівняно з одним. Такий результат підкреслює ефективність багатопоточного програмування в контексті даної задачі та важливість оптимізації програм за допомогою розпаралелювання обчислень.