**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**Національний Технічний Університет України**

**«Київський Політехнічний Інститут»**

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

**Лабораторна робота № 1**

з дисципліни *«****Паралельні обчислення****»*

*Варіант 2*

**Виконала:**

студентка 4-го курсу,

групи ІП-74

Герасименко Ю.К.

**Перевірив:**

старший викладач Долголенко О. М.

Київ — 2021

Приклад input.txt файлу Приклад output.txt файлу - у прикріпленних

0.5 - B

0.7 - E

1.0 - MC

0.1 - MZ

0.7 - MM

0.1 - MX

0.9 - MD

0.4 - MT

0.7 - ME

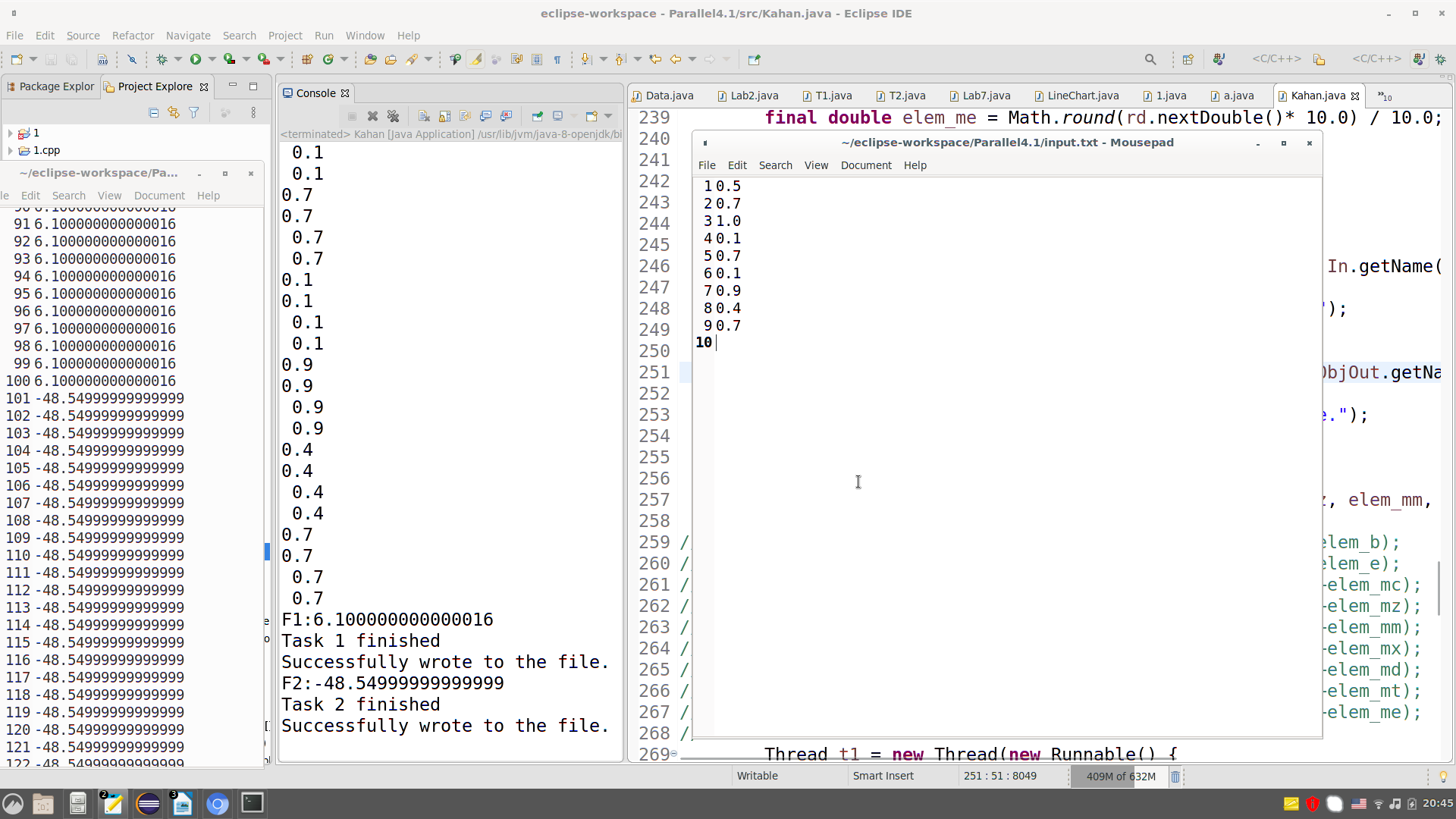
У зіп файлі міститься проект.

ВАРІАНТ 2

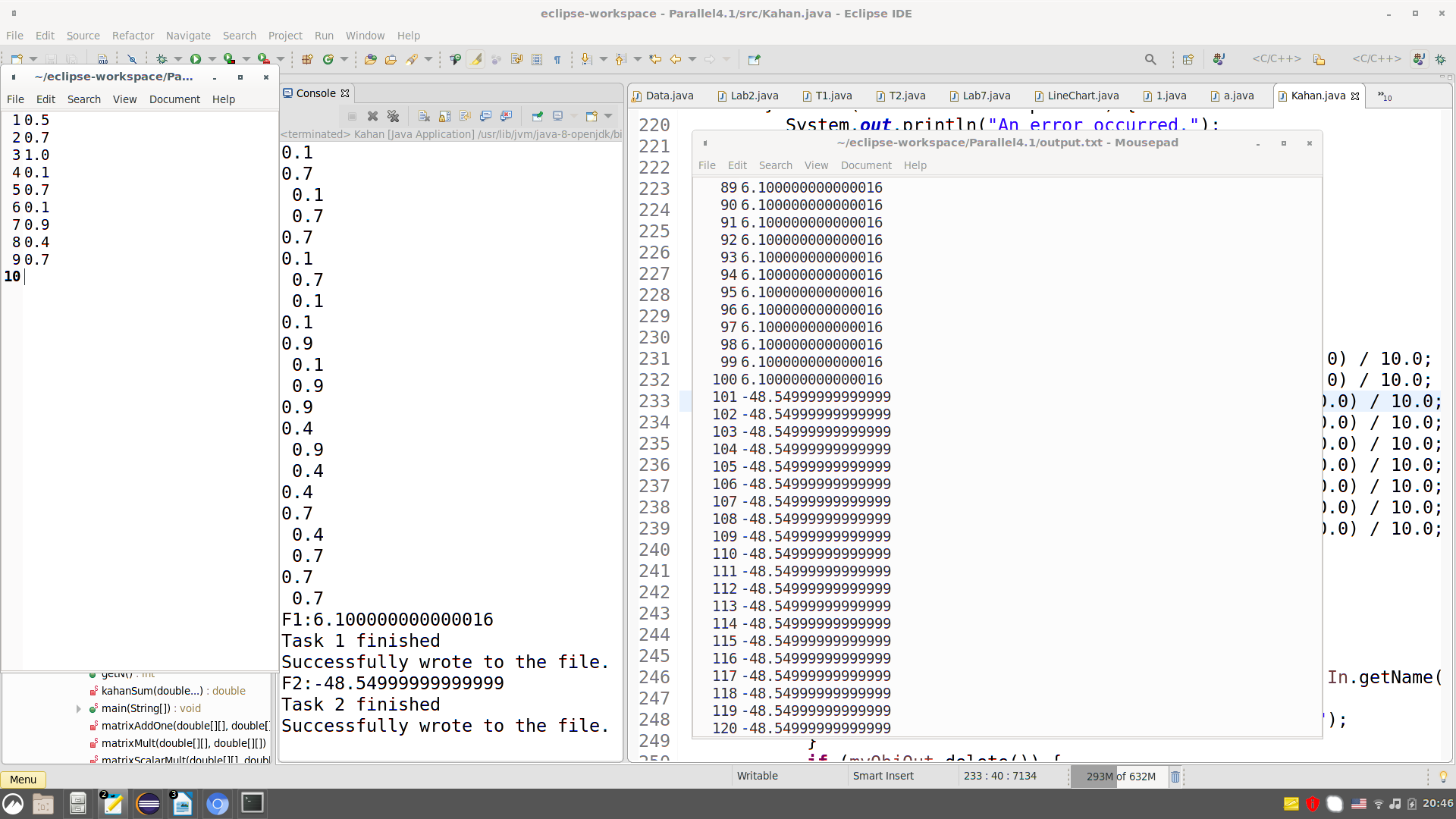
D=В\*(МС+MZ)-E\*MM+МX;

MА = min(MD)\*(MT+MZ)-ME\*MM.

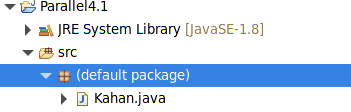
Скріншоти:1) Зображено перший запуск програми, де вхідні дані генеруються та записуються до файлу(input.txt). Звідти зчитуються обробляються 2 потоками на кожному з них окреме завдання.



2) У другий раз я заново не генерувала дані щоб перевірити що виконується для однакових даних програма вірно. Результати у вихідному файлі однакові з попереднім запуском. Для цього щоб протестувати це потрібно змінити флаг ifWriteToFile на false(перший раз true).



3) Як виглядає структура проекту.(Дивіться у прикріплених)



Запускала по декілька разів, кожен раз input i output створюються нові. Та результати записуються у файл та виводяться у консоль.

Код:

import java.util.ArrayList;

import java.util.Arrays;

import java.util.Random;

import java.io.FileWriter; // Import the FileWriter class

import java.io.IOException; // Import the IOException class to handle errors

import java.util.Scanner; // Import the Scanner class to read text files

import java.io.File; // Import the File class

import java.io.FileNotFoundException; // Import this class to handle erro

public class Kahan {

private static double[] vectorAddOrSubstract(double[] a, double[] b, char sign) {

if (a.length != n || b.length != n) {

return null;

}

double[] c = new double[n];

double[] kahanArray = new double[2];

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

if (sign == '+') {

// c[i] = a[i] + b[i];

kahanArray[0] = a[i];

kahanArray[1] = b[i];

c[i] = kahanSum(kahanArray);

}

if (sign == '-') {

// c[i] = a[i] - b[i];

kahanArray[0] = a[i];

kahanArray[1] = -b[i];

c[i] = kahanSum(kahanArray);

}

}

return c;

}

private static double[][] matrixAddOne(double[][] a, double[][] b, char sign) {

if (a.length != n || b.length != n) {

return null;

}

double[][] c = new double[n][n];

double[] kahanArray = new double[2];

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

for (int j = 0; j < n; j++) {

if (sign == '+') {

// c[i][j] = a[i][j] + b[i][j];

kahanArray[0] = a[i][j];

kahanArray[1] = b[i][j];

c[i][j] = kahanSum(kahanArray);

}

if (sign == '-') {

kahanArray[0] = a[i][j];

kahanArray[1] = -b[i][j];

c[i][j] = kahanSum(kahanArray);

}

}

}

return c;

}

private static double minMatrixEl(double[][] a) {

double c = a[0][0];

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

for (int j = 0; j < n; j++) {

if (a[i][j] < c)

c = a[i][j];

}

}

return c;

}

private static double[] vectorMatrixMult(double[] a, double[][] ma) {

double[] kahanArray = new double[n];

if (a.length != n || ma.length != n) {

return null;

}

double[] c = new double[n];

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

for (int j = 0; j < n; j++) {

kahanArray[j] = a[j] \* ma[j][i];

}

c[i] = kahanSum(kahanArray);

}

return c;

}

private static double[][] matrixScalarMult(double[][] a, double m) {

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

for (int j = 0; j < n; j++) {

a[i][j] \*= m;

}

}

return a;

}

private static double[][] matrixMult(double[][] ma, double[][] mb) {

if (ma.length != n || mb.length != n) {

return null;

}

double[][] c = new double[n][n];

double[] kahanArray = new double[n];

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

for (int j = 0; j < n; j++) {

for (int k = 0; k < n; k++) {

kahanArray[k] = ma[i][k] \* mb[k][j];

}

c[i][j] = kahanSum(kahanArray);

}

}

return c;

}

private static double[] computeFirst(final int n, double[] B, double[] E, double[][] MC, double[][] MZ, double[][] MM, double[] M) {

return vectorAddOrSubstract(vectorMatrixMult(B, matrixAddOne(MC, MZ, '+')), vectorAddOrSubstract(M, vectorMatrixMult(E, MM), '-'), '+');

}

private static double[][] computeSecond(final int n, double[][] MD, double[][] MT, double[][] MZ, double[][] ME, double[][] MM) {

// return vectorMatrixMult(B, matrixAddOne(MC, MZ))-vectorAddOrSubstract(vectorMatrixMult(E, MM));

return matrixAddOne(matrixScalarMult(matrixAddOne(MT, MZ, '+'), minMatrixEl(MD)), matrixMult(ME, MM), '-');

}

private static int n = 100;

public static int getN() {

return n;

}

// Java program to illustrate the

// Kahan summation algorithm

// Function to implement the Kahan

// summation algorithm

private static double kahanSum(double... fa)

{

double sum = 0.0;

// Variable to store the error

double c = 0.0;

// Loop to iterate over the array

for (double f : fa) {

double y = f - c;

double t = sum + y;

// Algebraically, c is always 0

// when t is replaced by its

// value from the above expression.

// But, when there is a loss,

// the higher-order y is cancelled

// out by subtracting y from c and

// all that remains is the

// lower-order error in c

c = (t - sum) - y;

sum = t;

}

return sum;

}

public static void create(String filename) {

try {

File myObj = new File(filename);

if (myObj.createNewFile()) {

System.out.println("File created: " + myObj.getName());

} else {

System.out.println("File already exists.");

}

} catch (IOException e) {

System.out.println("An error occurred.");

e.printStackTrace();

}

}

private static void writeToFile(String fileName, double[] values) {

try {

File file = new File(fileName);

FileWriter myWriter = new FileWriter(file, true);

for (int i = 0; i < values.length; i++) {

myWriter.write(Double.toString(values[i])+"\n");

}

myWriter.close();

System.out.println("Successfully wrote to the file.");

} catch (IOException e) {

System.out.println("An error occurred.");

e.printStackTrace();

}

}

private static void writeToFileMatrix(String fileName, double[][] values) {

try {

File file = new File(fileName);

FileWriter myWriter = new FileWriter(file, true);

for (int i = 0; i < values.length; i++) {

for (int j = 0; j < values.length; j++) {

myWriter.write(Double.toString(values[i][j])+"\n");

}

myWriter.write("\n");

}

myWriter.write("\n");

myWriter.close();

System.out.println("Successfully wrote to the file.");

} catch (IOException e) {

System.out.println("An error occurred.");

e.printStackTrace();

}

}

private static final ArrayList<Double> readFromFile() {

ArrayList<Double> arrlist = new ArrayList<>();

try {

File myObj = new File("input.txt");

Scanner myReader = new Scanner(myObj);

while (myReader.hasNextLine()) {

String s = myReader.nextLine();

System.out.println(s);

arrlist.add(Double.parseDouble(s));

System.out.println(" "+arrlist.get(arrlist.size()-1));

}

myReader.close();

} catch (FileNotFoundException e) {

System.out.println("An error occurred.");

e.printStackTrace();

}

return arrlist;

}

public static void main(String[] args) {

Random rd = new Random(); // creating Random object

boolean ifWriteToFile = true;

final double elem\_b = Math.round(rd.nextDouble()\* 10.0) / 10.0;

final double elem\_e = Math.round(rd.nextDouble()\* 10.0) / 10.0;

final double elem\_mc = Math.round(rd.nextDouble()\* 10.0) / 10.0;

final double elem\_mz = Math.round(rd.nextDouble()\* 10.0) / 10.0;

final double elem\_mm = Math.round(rd.nextDouble()\* 10.0) / 10.0;

final double elem\_mx = Math.round(rd.nextDouble()\* 10.0) / 10.0;

final double elem\_md = Math.round(rd.nextDouble()\* 10.0) / 10.0;

final double elem\_mt = Math.round(rd.nextDouble()\* 10.0) / 10.0;

final double elem\_me = Math.round(rd.nextDouble()\* 10.0) / 10.0;

if (ifWriteToFile) {

File myObjIn = new File("input.txt");

File myObjOut = new File("output.txt");

if (myObjIn.delete()) {

System.out.println("Deleted the file: " + myObjIn.getName());

} else {

System.out.println("Failed to delete the file.");

}

if (myObjOut.delete()) {

System.out.println("Deleted the file: " + myObjOut.getName());

} else {

System.out.println("Failed to delete the file.");

}

create("input.txt");

writeToFile("input.txt",

new double[]{elem\_b, elem\_e, elem\_mc, elem\_mz, elem\_mm, elem\_mx, elem\_md, elem\_mt, elem\_me});

}

// System.out.println("Each element of vector B is "+elem\_b);

// System.out.println("Each element of vector E is "+elem\_e);

// System.out.println("Each element of matrix MC is "+elem\_mc);

// System.out.println("Each element of matrix MZ is "+elem\_mz);

// System.out.println("Each element of matrix MM is "+elem\_mm);

// System.out.println("Each element of matrix MX is "+elem\_mx);

// System.out.println("Each element of matrix MD is "+elem\_md);

// System.out.println("Each element of matrix MT is "+elem\_mt);

// System.out.println("Each element of matrix ME is "+elem\_me);

//

Thread t1 = new Thread(new Runnable() {

public void run() {

System.out.println("Task 1 started");

final ArrayList<Double> arrlist = readFromFile();

double[] B = new double[n];

double[] E = new double[n];

double[] M = new double[n];

Arrays.fill(B, arrlist.get(0));

Arrays.fill(E, arrlist.get(1));

Arrays.fill(M, arrlist.get(5));

double[][] MC = new double[n][n];

double[][] MZ = new double[n][n];

double[][] MM = new double[n][n];

double[][] MX = new double[n][n];

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

Arrays.fill(MC[i], arrlist.get(2));

Arrays.fill(MZ[i], arrlist.get(3));

Arrays.fill(MM[i], arrlist.get(4));

Arrays.fill(MX[i], arrlist.get(5));

}

final double[] s = computeFirst(n, B, E, MC, MZ, MM, M);

synchronized (System.out) {

System.out.println("F1:"+ s[0]);

System.out.println("Task 1 finished");

writeToFile("output.txt", s);

}

}

});

Thread t2 = new Thread(new Runnable() {

public void run() {

System.out.println("Task 2 started");

final int n = getN();

final ArrayList<Double> arrlist = readFromFile();

double[][] MD = new double[n][n];

double[][] MT = new double[n][n];

double[][] MZ = new double[n][n];

double[][] ME = new double[n][n];

double[][] MM = new double[n][n];

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

Arrays.fill(MD[i], arrlist.get(6));

Arrays.fill(MT[i], arrlist.get(7));

Arrays.fill(MZ[i], arrlist.get(3));

Arrays.fill(ME[i], arrlist.get(8));

Arrays.fill(MM[i], arrlist.get(4));

}

final double[][] s = computeSecond(n, MD, MT, MZ, ME, MM);

synchronized (System.out) {

System.out.println("F2:"+ s[0][0]);

System.out.println("Task 2 finished");

writeToFileMatrix("output.txt", s);

}

}

});

t1.setPriority(Thread.MAX\_PRIORITY);

t2.setPriority(Thread.MIN\_PRIORITY);

t1.start();

t2.start();

}

}