Міністерство освіти і науки України

Національний технічний університет України

«Київський політехнічний інститут ім. Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

**Лабораторна робота №2**

З дисципліни «Паралельне програмування»

**Потоки в мові Java**

ВИКОНАВ:

Студент ІІІ курсу ФІОТ

Групи ІО-91

Діденко В.В.

ПЕРЕВІРИВ:

Корочкін О.В.

Київ 2021 р.

**Варіант завдання:**

F1: A = B + C + D\*(MD\*ME)

F2: MG = SORT(MF - MH \* MK)

F3: S = SORT(O\*MO)\*(MS \*MT)

**Лістинг програми:**

Lab2.java

import java.util.Scanner;  
  
*/\*\*  
 \* Parallel programming.  
 \* Labwork 2. Java.  
 \*  
 \* Didenko Vladyslav  
 \* IO-91  
 \*  
 \* F1 A = B + C + D \* (MD \* ME)  
 \* F2 MG = sort(MF - MH \* MK)  
 \* F3 S = sort(O \* MO) \* (MS \* MT)  
 \*/*public class Lab2 extends Thread {  
 public static void main(String[] args) {  
 new Lab2().start();  
 }  
  
 @Override  
 public void run() {  
 setName("Lab2");  
 Scanner scanner = new Scanner(System.*in*);  
 System.*out*.println("Enter matrix shape: ");  
 int N = scanner.nextInt();  
 System.*out*.println("Lab2 started!");  
  
 Data data = new Data(N);  
  
 Func1 func1 = new Func1("Func1", Thread.*NORM\_PRIORITY*, data);  
 Func2 func2 = new Func2("Func2", Thread.*MAX\_PRIORITY*, data);  
 Func3 func3 = new Func3("Func3", Thread.*MIN\_PRIORITY*, data);  
  
// func2.inputFill();  
  
 func1.start();  
 func2.start();  
 func3.start();  
  
 try {  
 func1.join();  
 func2.join();  
 func3.join();  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 System.*out*.println("Lab2 finished");  
  
 }  
}

Data.java

import java.util.Arrays;  
import java.util.Objects;  
import java.util.Random;  
import java.util.Scanner;  
  
public class Data {  
 private int n;  
  
 public Data(int n) {  
 this.n = n;  
 }  
  
 public int getN() {  
 return n;  
 }  
  
 public void setN(int n) {  
 this.n = n;  
 }  
  
 // F1 A = B + C + D \* (MD \* ME)  
 public int[] func1(int[] b, int[] c, int[] d, int[][] md, int[][] me) {  
 return vectorAdd(Objects.*requireNonNull*(vectorAdd(b, c)), vectorMatrixMult(d, matrixMult(md, me)));  
 }  
  
 // F2 MG = sort(MF - MH \* MK)  
 public int[][] func2(int[][] mf, int[][] mh, int[][] mk) {  
 return matrixSort(Objects.*requireNonNull*(matrixDiff(mf, matrixMult(mh, mk))));  
 }  
  
 // F3 S = sort(O \* MO) \* (MS \* MT)  
 public int[] func3(int[] o, int[][] mo, int[][] ms, int[][] mt) {  
 return vectorMatrixMult(Objects.*requireNonNull*(vectorSort(vectorMatrixMult(o, mo))), matrixMult(ms, mt));  
 }  
  
 public int[] vectorInput() {  
 int[] input = new int[n];  
 Scanner scanner = new Scanner(System.*in*);  
 for (int i = 0; i < input.length; i++) {  
 input[i] = scanner.nextInt();  
 }  
 return input;  
 }  
  
 public int[] vectorRandom() {  
 int[] randVector = new int[n];  
 Random random = new Random();  
 for (int i = 0; i < randVector.length; i++) {  
 randVector[i] = random.nextInt(10);  
  
 }  
 return randVector;  
 }  
  
 public void vectorOutput(int[] vector) {  
 for (int i1 : vector) {  
 System.*out*.println(i1);  
 }  
 }  
  
 public int[][] matrixInput() {  
 int[][] matrix = new int[n][n];  
 Scanner scanner = new Scanner(System.*in*);  
 for (int i = 0; i < matrix.length; i++) {  
 for (int j = 0; j < matrix[i].length; j++) {  
 matrix[i][j] = scanner.nextInt();  
 }  
 }  
 return matrix;  
 }  
  
 public int[][] matrixNum(int num) {  
 int[][] matrix = new int[n][n];  
 for (int i = 0; i < matrix.length; i++) {  
 for (int j = 0; j < matrix[i].length; j++) {  
 matrix[i][j] = num;  
 }  
 }  
 return matrix;  
 }  
  
 public int[][] matrixOne() {  
 return matrixNum(1);  
 }  
  
 public int[] vectorNum(int num) {  
 int[] vector = new int[n];  
 for (int i = 0; i < vector.length; i++) {  
 vector[i] = num;  
 }  
 return vector;  
 }  
  
 public int[] vectorOne() {  
 return vectorNum(1);  
 }  
  
 public int[][] matrixRandom() {  
 int[][] randMatrix = new int[n][n];  
 Random random = new Random();  
 for (int i = 0; i < randMatrix.length; i++) {  
 for (int j = 0; j < randMatrix[i].length; j++) {  
 randMatrix[i][j] = random.nextInt(60);  
 }  
 }  
 return randMatrix;  
 }  
  
 public void matrixOutput(int[][] matrix) {  
 for (int[] row : matrix) {  
 System.*out*.println(Arrays.*toString*(row));  
 }  
 }  
  
 private int findMax(int[] array) {  
 int[] res = Arrays.*copyOf*(array, n);  
 Arrays.*sort*(res);  
 return res[res.length - 1];  
 }  
  
 private int[] vectorSort(int[] a) {  
 Arrays.*sort*(a);  
 return a;  
 }  
  
 private int[] vectorAdd(int[] first, int[] second) {  
 if (first.length != n || second.length != n) {  
 return null;  
 }  
 int[] res = new int[n];  
 for (int i = 0; i < n; i++) {  
 res[i] = first[i] + second[i];  
 }  
 return res;  
 }  
  
 private int[] vectorDiff(int[] a, int[] b) {  
 if (a.length != n || b.length != n) {  
 return null;  
 }  
 int[] c = new int[n];  
 for (int i = 0; i < n; i++) {  
 c[i] = a[i] - b[i];  
 }  
 return c;  
 }  
  
 private int[] vectorMult(int[] a, int[] b) {  
 if (b.length != n || a.length != n) {  
 return null;  
 }  
 int[] c = new int[n];  
 for (int i = 0; i < n; i++) {  
 c[i] = a[i] \* b[i];  
 }  
 return c;  
 }  
  
 private int[] vectorMatrixMult(int[] a, int[][] ma) {  
 if (a.length != n || ma.length != n) {  
 return null;  
 }  
 int[] c = new int[n];  
 for (int i = 0; i < n; i++) {  
 for (int j = 0; j < n; j++) {  
 c[i] += a[j] \* ma[i][j];  
 }  
 }  
 return c;  
 }  
  
 private int[][] matrixMult(int[][] ma, int[][] mb) {  
 if (ma.length != n || mb.length != n) {  
 return null;  
 }  
 int[][] c = new int[n][n];  
 for (int i = 0; i < n; i++) {  
 for (int j = 0; j < n; j++) {  
 for (int k = 0; k < n; k++) {  
 c[i][j] += ma[i][k] \* mb[k][j];  
 }  
 }  
 }  
 return c;  
 }  
  
 private int[][] matrixAdd(int[][] ma, int[][] mb) {  
 if (ma.length != n || mb.length != n) {  
 return null;  
 }  
 int[][] c = new int[n][n];  
 for (int i = 0; i < n; i++) {  
 for (int j = 0; j < n; j++) {  
 c[i][j] += ma[i][j] + mb[i][j];  
 }  
 }  
 return c;  
 }  
  
 private int[][] matrixDiff(int[][] ma, int[][] mb) {  
 if (ma.length != n || mb.length != n) {  
 return null;  
 }  
 int[][] c = new int[n][n];  
 for (int i = 0; i < n; i++) {  
 for (int j = 0; j < n; j++) {  
 c[i][j] -= ma[i][j] + mb[i][j];  
 }  
 }  
 return c;  
 }  
  
 private int[] intVectorMult(int a, int[] b) {  
 int[] c = new int[n];  
 for (int i = 0; i < n; i++) {  
 c[i] = a \* b[i];  
 }  
 return c;  
 }  
  
 private int[][] intMatrixMult(int a, int[][] b) {  
 int[][] c = new int[n][n];  
 for (int i = 0; i < n; i++) {  
 for (int j = 0; j < n; j++) {  
 c[i][j] = a \* b[i][j];  
 }  
 }  
 return c;  
 }  
  
 private int[][] matrixTransp(int[][] ma) {  
 int buf;  
 for (int i = 0; i < ma.length; i++) {  
 for (int j = 0; j <= i; j++) {  
 buf = ma[i][j];  
 ma[i][j] = ma[j][i];  
 ma[j][i] = buf;  
 }  
 }  
 return ma;  
 }  
  
 private int[][] matrixSort(int[][] ma) {  
 for (int[] m : ma) {  
 Arrays.*sort*(m);  
 }  
 return ma;  
 }  
  
}

Func.java

public abstract class Func extends Thread {  
 protected Data data;  
 protected boolean filled;  
  
 public Func(String name, int priority, Data data) {  
 setName(name);  
 setPriority(priority);  
 this.data = data;  
 }  
  
 public Data getData() {  
 return data;  
 }  
  
 public void setData(Data data) {  
 this.data = data;  
 }  
  
 public abstract void randomFill();  
  
 public abstract void inputFill();  
  
 public abstract void oneFill();  
  
 @Override  
 public void run() {  
 if (!filled) oneFill();  
 }  
}

Func1.java

import java.util.Arrays;  
  
public class Func1 extends Func {  
 private int[] B;  
 private int[] C;  
 private int[] D;  
 private int[][] MD;  
 private int[][] ME;  
  
 public Func1(String name, int priority, Data data) {  
 super(name, priority, data);  
 }  
  
 public void randomFill() {  
 B = data.vectorRandom();  
 C = data.vectorRandom();  
 D = data.vectorRandom();  
 MD = data.matrixRandom();  
 ME = data.matrixRandom();  
 filled = true;  
 }  
  
 public void inputFill() {  
 System.*out*.println("Print B");  
 B = data.vectorInput();  
 System.*out*.println("Print C");  
 C = data.vectorInput();  
 System.*out*.println("Print D");  
 D = data.vectorInput();  
 System.*out*.println("Print MD");  
 MD = data.matrixInput();  
 System.*out*.println("Print ME");  
 ME = data.matrixInput();  
 filled = true;  
 }  
  
 public void oneFill() {  
 B = data.vectorOne();  
 C = data.vectorOne();  
 D = data.vectorOne();  
 MD = data.matrixOne();  
 ME = data.matrixOne();  
 filled = true;  
 }  
  
 // A = B + C + D \* (MD \* ME)  
 @Override  
 public void run() {  
 System.*out*.println("Func 1 started");  
 super.run();  
 try {  
 int[] result = data.func1(B, C, D, MD, ME);  
 *sleep*(100);  
 System.*out*.println("Func1 result: " + Arrays.*toString*(result));  
 System.*out*.println("Func 1 finished");  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 }  
}

Func2.java

import java.util.Arrays;  
  
public class Func2 extends Func {  
  
 private int[][] MF;  
 private int[][] MH;  
 private int[][] MK;  
  
 public Func2(String name, int priority, Data data) {  
 super(name, priority, data);  
 }  
  
 public void randomFill() {  
 MF = data.matrixRandom();  
 MH = data.matrixRandom();  
 MK = data.matrixRandom();  
 filled = true;  
 }  
  
 public void inputFill() {  
 System.*out*.println("Print MF");  
 MF = data.matrixInput();  
 System.*out*.println("Print MH");  
 MH = data.matrixInput();  
 System.*out*.println("Print MK");  
 MK = data.matrixInput();  
 filled = true;  
 }  
  
 public void oneFill() {  
 MF = data.matrixOne();  
 MH = data.matrixOne();  
 MK = data.matrixOne();  
 filled = true;  
 }  
  
 // MG = sort(MF - MH \* MK)  
 @Override  
 public void run() {  
 System.*out*.println("Func 2 started");  
 super.run();  
 try {  
 int[][] result = data.func2(MF, MH, MK);  
 *sleep*(100);  
 System.*out*.println("Func2 result: " + Arrays.*deepToString*(result));  
 System.*out*.println("Func 2 finished");  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 }  
}

Func3.java

import java.util.Arrays;  
  
public class Func3 extends Func {  
 private int[] O;  
 private int[][] MO;  
 private int[][] MS;  
 private int[][] MT;  
  
 public Func3(String name, int priority, Data data) {  
 super(name, priority, data);  
 }  
  
 public void randomFill() {  
 O = data.vectorRandom();  
 MO = data.matrixRandom();  
 MS = data.matrixRandom();  
 MT = data.matrixRandom();  
 filled = true;  
 }  
  
 public void inputFill() {  
 System.*out*.println("Print O");  
 O = data.vectorInput();  
 System.*out*.println("Print MO");  
 MO = data.matrixInput();  
 System.*out*.println("Print MS");  
 MS = data.matrixInput();  
 System.*out*.println("Print MT");  
 MT = data.matrixInput();  
 filled = true;  
 }  
  
 public void oneFill() {  
 O = data.vectorOne();  
 MO = data.matrixOne();  
 MS = data.matrixOne();  
 MT = data.matrixOne();  
 filled = true;  
 }  
  
 // S = sort(O \* MO) \* (MS \* MT)  
 @Override  
 public void run() {  
 System.*out*.println("Func 3 started");  
 super.run();  
 try {  
 int[] result = data.func3(O, MO, MS, MT);  
 *sleep*(100);  
 System.*out*.println("Func3 result: " + Arrays.*toString*(result));  
 System.*out*.println("Func 3 finished");  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 }  
}

Результат виконання програми:

