Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

Основи паралельного програмування

Лабораторна робота №2

**«Потоки в мові JAVA»**

Виконав:

студент групи ІВ-82

Троценко Д.А.

Перевірив:

Доцент Корочкін О. В.

Київ

2020 р.

**Завдання:**

**F1: d = (A\*((B+C)\*(MA\*ME)))**

**F2: MF = MIN(MH)\*MK\*ML**

**F3: O = MAX(MP\*MR)\*V**

**Переделать выводы.**

**Выводы не обнаружены.**

**Следующая лаба на C#**

**Мета роботи:** вивчення засобів мови Java для роботи с потоками.

**Виконання роботи:** Розробив програму, яка містить параллельні потоки, що реалізують відповідну функцію F1, F2, F3.

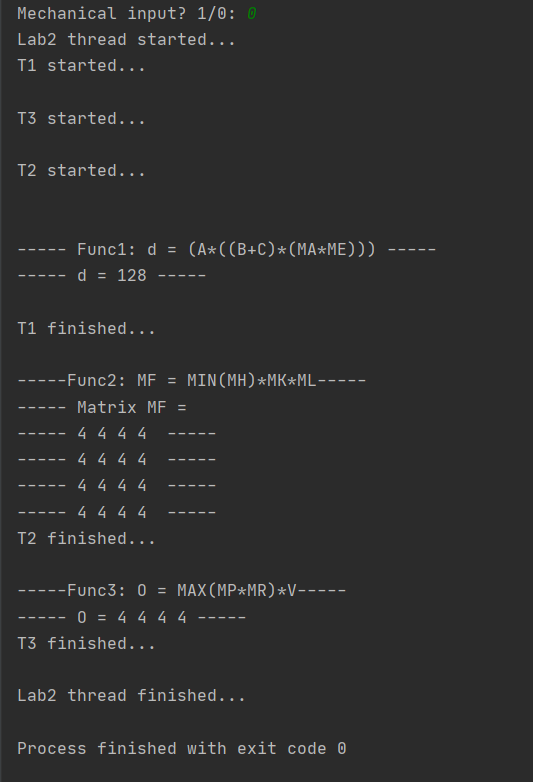
Вимоги що до створення потоків і завдання дослідження особливості виконання паралельної програми визначені в лабораторної роботі 1.

В потоках використати методи sleep() і join().

**Висновки:**

1. Розробив програму за допомогою мови програмування Java. Були створені потоки T1-T3 за допомогою наслідування від класу Thread.
2. Був виконаний ручний та автоматичний ввід. Про ручний ввід описано в **6** пункті.
3. Реалізовані методи run та застосовані семафори для виконання вводу з клавіатури. Проблеми, які виникали в минулій лабораторній роботі, а саме запуск потоків та вивід без оператора delay(в АДА)(Thread.sleep() в Java) не виникло.
4. Встановив пріорітети для задач T1-T3 за допомогою метода set\_priority(int) відповідно 1-3-5. Для того, щоб вивід не змішувався був застосований метод Thread.sleep(int). За допомогою нього глушили потоки та виводили на экран.
5. Для того, щоб поток Lab2(Main program) розпочинався та закінчувався останнім, я застосував метод join() для того, щоб головний поток закінчувався лише після інших потоків.
6. Для того, щоб ввести дані вручну, я застосував семафор, для того щоб зупинити потоки та чекати, поки не буде виповнений ввід.

**Результат виконання программи для автоматичного вводу**



**Лістинг програми:**

**Lab\_2.java**

/\*  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*Main programm\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
\*\*\*\*\* Parallel programming basics.  
\*\*\*\*\* Labwork 2. Threads in Java  
\*\*\*\*\* Trotsenko Daniil  
\*\*\*\*\* IV-82  
\*\*\*\*\* 1.10.2020  
\*\*\*\*\* Func1: d = (A\*((B+C)\*(MA\*ME)))  
\*\*\*\*\* Func2: MF = MIN(MH)\*MK\*ML  
\*\*\*\*\* Func3: O = MAX(MP\*MR)\*V  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
\*/  
  
package com.company;  
  
  
import java.util.Scanner;  
import java.util.concurrent.Semaphore;  
  
//Thread 1  
class T1 extends Thread {  
 private Semaphore semaphore;  
 private boolean mechanical;  
 private int d;  
 private int n = 4;  
 private Vector A = new Vector(n), B = new Vector(n), C = new Vector(n);  
 private Matrix MA = new Matrix(n), ME = new Matrix(n);  
  
 //--Func1: d = (A\*((B+C)\*(MA\*ME)))--  
 T1 (String name, Semaphore sem, boolean mechanical\_) {  
 super(name);  
 semaphore = sem;  
 mechanical = mechanical\_;  
 }  
 public void run() {  
 System.*out*.printf("%s started...", Thread.*currentThread*().getName());  
 System.*out*.println();  
 System.*out*.println();  
  
 try {  
 Thread.*sleep*(5);  
  
 if (mechanical) {  
 semaphore.acquire(1);  
  
 Scanner in = new Scanner(System.*in*);  
 System.*out*.print("Input N for T1 = ");  
 n = in.nextInt();  
  
 A = new Vector(n);  
 B = new Vector(n);  
 C = new Vector(n);  
 MA = new Matrix(n);  
 ME = new Matrix(n);  
 A.ReadVector("A");  
 B.ReadVector("B");  
 C.ReadVector("C");  
  
 MA.ReadMatrix("MA");  
 ME.ReadMatrix("ME");  
 }  
 d = A.MultiplicationVectors(B.SumVectors(C).MultiplicationVectorMatrix(MA.MultiplicationMatrices(ME)));  
 if (n <= 5) {  
 Thread.*sleep*(100);  
 System.*out*.println();  
 System.*out*.println("----- Func1: d = (A\*((B+C)\*(MA\*ME))) -----");  
 System.*out*.println("----- d = " + d + " -----");  
 System.*out*.println();  
 }  
 semaphore.release();  
 } catch (InterruptedException e) {  
 System.*out*.printf("Thread has been interrupted");  
 }  
 System.*out*.printf("%s finished...", Thread.*currentThread*().getName());  
 System.*out*.println();  
 System.*out*.println();  
 }  
}  
  
  
//Thread 2  
class T2 extends Thread {  
 private int n = 4;  
 private Semaphore semaphore;  
 private boolean mechanical;  
 private Matrix MF, MH = new Matrix(n), MK = new Matrix(n), ML = new Matrix(n);  
 T2 (String name, Semaphore sem, boolean mechanical\_) {  
 super(name);  
 semaphore = sem;  
 mechanical = mechanical\_;  
 }  
 // --Func2: MF = MIN(MH)\*MK\*ML--  
 public void run() {  
 System.*out*.printf("%s started...", Thread.*currentThread*().getName());  
 System.*out*.println();  
 System.*out*.println();  
 try {  
  
 Thread.*sleep*(5);  
  
 if (mechanical) {  
 semaphore.acquire(1);  
  
 Scanner in = new Scanner(System.*in*);  
 System.*out*.print("Input N for T2 = ");  
 n = in.nextInt();  
  
 MH = new Matrix(n); MK = new Matrix(n); ML = new Matrix(n);  
  
  
 MH.ReadMatrix("MH");  
 MK.ReadMatrix("MK");  
 ML.ReadMatrix("ML");  
 }  
 MF = MK.MultiplicationMatrices(ML).MatrixMultiplicationOnConstant(MH.MinOnMatrix());  
 if (n <= 5) {  
 Thread.*sleep*(800);  
  
 System.*out*.println("-----Func2: MF = MIN(MH)\*MK\*ML-----");  
 MF.Print("MF");  
 }  
 semaphore.release();  
 } catch (InterruptedException e) {  
 System.*out*.printf("Thread has been interrupted");  
 }  
 System.*out*.printf("%s finished...", Thread.*currentThread*().getName());  
 System.*out*.println();  
 System.*out*.println();  
 }  
}  
// Thread 3  
class T3 extends Thread {  
 private int n = 4;  
 private Vector O, V = new Vector(n);  
 private Matrix MP = new Matrix(n), MR = new Matrix(n);  
 private Semaphore semaphore;  
 private boolean mechanical;  
 T3 (String name, Semaphore sem, boolean mechanical\_) {  
 super(name);  
 semaphore = sem;  
 mechanical = mechanical\_;  
 }  
 // --Func3: O = MAX(MP\*MR)\*V--  
 public void run() {  
 System.*out*.printf("%s started...", Thread.*currentThread*().getName());  
 System.*out*.println();  
 System.*out*.println();  
 try {  
 Thread.*sleep*(5);  
  
 if (mechanical) {  
 semaphore.acquire(1);  
  
 Scanner in = new Scanner(System.*in*);  
 System.*out*.print("Input N for T3 = ");  
 n = in.nextInt();  
  
 V = new Vector(n);  
 MP = new Matrix(n); MR = new Matrix(n);  
  
  
  
 V.ReadVector("V");  
  
  
 MP.ReadMatrix("MP");  
 MR.ReadMatrix("MR");  
 }  
 O = V.VectorMultiplicationConstant(MP.MultiplicationMatrices(MR).MaxOnMatrix());  
 if (n <= 5) {  
 Thread.*sleep*(1200);  
 System.*out*.println("-----Func3: O = MAX(MP\*MR)\*V-----");  
 O.Print("O");  
 }  
 semaphore.release();  
 } catch (InterruptedException e) {  
 System.*out*.printf("Thread has been interrupted");  
 }  
 System.*out*.printf("%s finished...", Thread.*currentThread*().getName());  
 System.*out*.println();  
 System.*out*.println();  
 }  
}  
//Main Thread  
public class Lab2 {  
 public static void main(String[] args) {  
 Scanner in = new Scanner(System.*in*);  
 System.*out*.print("Mechanical input? 1/0: ");  
 boolean mechanical = false;  
 int q = in.nextInt();  
 if (q == 1) {  
 mechanical = true;  
 System.*out*.println("OK");  
 }  
 System.*out*.println("Lab2 thread started...");  
 Semaphore sem = new Semaphore(1);  
 T1 t1 = new T1("T1", sem, mechanical);  
 T2 t2 = new T2("T2", sem, mechanical);  
 T3 t3 = new T3("T3", sem, mechanical);  
 t1.setPriority(1);  
 t2.setPriority(3);  
 t3.setPriority(5);  
 t1.start();  
 t2.start();  
 t3.start();  
 try {  
 t1.join();  
 t2.join();  
 t3.join();  
 } catch(InterruptedException e) {  
 System.*out*.printf("%s has been interrupted", Thread.*currentThread*().getName());  
 }  
 System.*out*.println("Lab2 thread finished...");  
 in.close();  
 }  
}

Vector.java

/\*  
 Class Vector  
\*/  
package com.company;  
import java.util.Scanner;  
public class Vector {  
 public void FillVectorZero() {  
 for (int i = 0; i < length; ++i) {  
 vector[i] = 0;  
 }  
 }  
 public void SetItem(int i, int key) {  
 vector[i] = key;  
 }  
 Vector (int n) {  
 vector = new int[n];  
 length = n;  
 for (int i = 0; i < length; ++i) {  
 vector[i] = 1;  
 }  
 }  
 public void Print(String name) {  
 System.*out*.print( "----- " + name + " = ");  
 for (int i = 0; i < length; ++i) {  
 System.*out*.print(vector[i] + " ");  
 }  
 System.*out*.println("-----");  
 }  
 Vector(int[] n, int length\_) {  
 length = length\_;  
 vector = new int[length\_];  
 for (int i = 0; i < length; ++i) {  
 vector[i] = n[i];  
 }  
 }  
 public int Length() {  
 return length;  
 }  
 public int GetItem(int i) {  
 return vector[i];  
 }  
 public void ReadVector(String name) {  
 System.*out*.print("Input " + name + " : ");  
  
 Scanner in = new Scanner(System.*in*);  
 for (int i = 0; i < length; ++i) {  
 vector[i] = in.nextInt();  
 }  
 }  
 public Vector SumVectors(Vector x) {  
 for (int i = 0; i < length; ++i) {  
 vector[i] += x.GetItem(i);  
 }  
 return new Vector(vector, length);  
 }  
 public int MultiplicationVectors (Vector x) {  
 int result = 0;  
  
 for (int i = 0; i < length; ++i) {  
 result += vector[i] \* x.GetItem(i);  
 }  
 return result;  
 }  
 public Vector MultiplicationVectorMatrix (Matrix m) {  
 int[] mas\_res = new int[length];  
 for (int i=0; i < length; ++i) {  
 mas\_res[i] = 0;  
 }  
 for (int i = 0; i < length; ++i) {  
 for (int j = 0; j < length; ++j) {  
 mas\_res[i] += vector[j] \* m.GetItem(i,j);  
 }  
 }  
 return new Vector(mas\_res, length);  
 }  
 public Vector VectorMultiplicationConstant(int k) {  
 Vector res = new Vector(length);  
 for (int i = 0; i < length; ++i) {  
 res.SetItem(i, res.GetItem(i)\*k);  
 }  
 return res;  
 }  
 private int[] vector;  
 private int length;  
}

Matrix.java

/\*  
 Class Matrix  
\*/  
  
package com.company;  
public class Matrix {  
 Matrix(int n) {  
 matrix = new Vector[n];  
 length = n;  
 for (int i = 0; i < n; ++i) {  
 matrix[i] = new Vector(n);  
 }  
 }  
 public void ReadMatrix(String name) {  
 System.*out*.println("Input matrix " + name);  
 for (int i = 0; i < length; ++i) {  
 matrix[i].ReadVector("Vector " + i);  
 }  
 }  
 public void Print(String name) {  
 System.*out*.println("----- Matrix " + name + " = ");  
 for (int i = 0; i < length; ++i) {  
 System.*out*.print("----- ");  
 for (int j = 0; j < length; ++j) {  
 System.*out*.print(matrix[i].GetItem(j) + " ");  
 }  
 System.*out*.print(" -----");  
 System.*out*.println();  
 }  
 }  
 public Matrix MultiplicationMatrices(Matrix m) {  
 int[][] matrices = new int[length][length];  
 for (int i = 0; i < length; ++i) {  
 for (int j = 0; j < length; ++j) {  
 matrices[i][j] = 0;  
 }  
 }  
 for (int i = 0; i < length; ++i) {  
 for (int j = 0; j < length; ++j) {  
 for (int p = 0; p < length; ++p) {  
 matrices[i][j] += matrix[i].GetItem(j) \* m.GetItem(p, j);  
 }  
 }  
 }  
 Matrix result = new Matrix(length);  
 for (int i = 0; i < length; ++i) {  
 for (int j = 0; j < length; ++j) {  
 result.SetItem(i, j, matrices[i][j]);  
 }  
 }  
 return result;  
 }  
 public Matrix MatrixMultiplicationOnConstant(int key) {  
 Matrix result = new Matrix(length);  
 for (int i = 0; i < length; ++i) {  
 for (int j = 0; j < length; ++j) {  
 result.SetItem(i, j, matrix[i].GetItem(j) \* key);  
 }  
 }  
 return result;  
 }  
 public int MaxOnMatrix() {  
 int max = matrix[0].GetItem(0);  
 for (int i = 0; i < length; ++i) {  
 for (int j = 0; j < length; ++j) {  
 if (matrix[i].GetItem(j) > max) {  
 max = matrix[i].GetItem(j);  
 }  
 }  
 }  
 return max;  
 }  
 public int MinOnMatrix() {  
 int min = matrix[0].GetItem(0);  
 for (int i = 0; i < length; ++i) {  
 for (int j = 0; j < length; ++j) {  
 if (matrix[i].GetItem(j) < min) {  
 min = matrix[i].GetItem(j);  
 }  
 }  
 }  
 return min;  
 }  
 public void SetItem(int i, int j, int key) {  
 matrix[i].SetItem(j, key);  
 }  
 public int GetItem(int i, int j) {  
 return matrix[i].GetItem(j);  
 }  
 private Vector[] matrix;  
 private int length;  
}