

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 “ОСНОВИ ПАРАЛЕЛЬНОГО ПРОГРАМУВАННЯ ”**

**НА ТЕМУ: “ПОТОКИ В МОВІ JAVA”**

**Виконав:**

Студент ІІІ курсу ФІОТ

групи ІО-82

Шендріков Євгеній

Номер у списку - 25

**Перевірив:**

Доцент Корочкін О. В.

м. Київ – 2020 р.

**Мета**

Вивчення засобів мови Java для роботи с потоками.

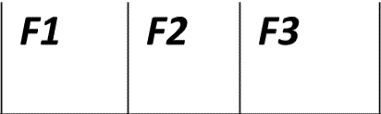
**Завдання**

Розробити програму яка містить ***паралельні потоки*** (задачі), кожна із яких реалізовує функцію F1, F2, F3 із **Додатку Б** згідно отриманому варіанту.

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

В потоках використати методи **sleep()** та **join()**.

**Варіант завдання**



1.21 D = SORT(A)+SORT(B)+SORT(C)\*(MA\*ME)

2.27 MF = (MG\*MH)\*TRANS(MK)

3.30 S = (MO\*MP)\*V+t\*MR\*(O+P)

**Висновки**

Проблемою при тестуванні програмної частини, як і в першій лабораторній, стало введення початкових даних з клавіатури, при якому програма поводила себе незвично.

Візьмемо наступну ситуацію: припустимо, що N=3, користувач обрав введення з клавіатури, і завданням є для T1 ввести всі елементи як 1, для T2 – як 2, для T3 – як 3. Якщо з цією проблемою вводу з клавіатури нічого не зробити, то при вводі значень, якщо 3 потоки одночасно запускаються, буде незрозуміло, що куди вводити, буде боротьба за спільні ресурси і все закінчиться хаосом, так само і з виводом.

Описана проблема як для введення, так і для виведення результату є загальною проблемою паралельного програмування. Ця проблема більш відома як *завдання взаємного виключення* і полягає вона в тому, що під час виконання декількох паралельних процесів може виникнути одночасне звернення до спільного ресурсу і таке звернення призводить до конфлікту процесів, як в описаному прикладі.

Для вирішення цієї проблеми необхідно блокувати процес, який звертається до спільного ресурсу, який в цей момент вже використовується іншим процесом. А розблокування процесу відбувається одразу ж після звільнення спільного ресурсу.

Для реалізації цього підходу використовуються спеціальні програмні засоби, такі як семафори, мютекси, критичні секції і т.д. Тому в цій лабораторній була зроблена спроба обробки вхідних даних з консолі за допомогою семафорів, що дозволило виконати необхідне завдання щодо встановлення з клавіатури всіх елементів для T1 як 1, для T2 як 2 і для T3 як 3 (приклад роботи в Додатку А).

Семафори в Java – це конструкція синхронізації потоків, яка контролює доступ до загального ресурсу за допомогою лічильників. Загалом, щоб використовувати семафор, потік, якому потрібен доступ до загального ресурсу, намагається отримати дозвіл. Якщо значення лічильника більше нуля, то потік отримує доступ до ресурсу, при цьому лічильник зменшується на одиницю. Після закінчення роботи з ресурсом потік звільняє семафор, і лічильник збільшується на одиницю. Якщо ж лічильник дорівнює нулю, то потік блокується і чекає, поки не отримає дозвіл від семафора. За замовчуванням всім очікуючим потокам надається дозвіл в невизначеному порядку.

Кожен потік, який хоче використовувати спільний ресурс, повинен спочатку викликати функцію **acquire()** перед зверненням до ресурсу для отримання блокування. Коли потік завершує роботу з ресурсом, він повинен викликати **release()** для зняття блокування.

При N > 10 можливість вводити дані з клавіатури не передбачена.

Отже, в даній лабораторній роботі було розглянуто принцип побудови паралельних програм на основі мови Java. Було використано семафори для обробки вводу даних з клавіатури, при генерації рандомізованих значень чи встановлення всіх вхідних значень як 1 – семафори не використовуються.

Було зроблено аналіз проблем, які виникли під час виконання, а також запропоновано шляхи їх рішення, перевірено працездатність програми і програмним шляхом оброблено виключні ситуації.

Кінцева мета роботи досягнута.

**Лістинг програми**

**Lab2.java**

/\*----------------------------------------------------  
 | Labwork #2 |  
 ----------------------------------------------------  
 | Author | Jack (Yevhenii) Shendrikov |  
 | Group | IO-82 |  
 | Variant | #25 |  
 | Date | 18.09.2020 |  
 ----------------------------------------------------  
 | Function 1 | D = SORT(A)+SORT(B)+SORT(C)\*(MA\*ME) |  
 | Function 2 | MF = (MG\*MH)\*TRANS(MK) |  
 | Function 3 | S = (MO\*MP)\*V+t\*MR\*(O+P) |  
 ----------------------------------------------------  
\*/  
  
import java.util.Scanner;  
import java.util.concurrent.Semaphore;  
  
public class Lab2 extends Thread{  
  
 @Override  
 public void run(){  
 setName("Lab2");  
 Scanner scanner = new Scanner(System.*in*);  
 int N;  
  
 //---------------------------- Input Handler ---------------------------  
 // header  
 System.*out*.print("----------------------------------------------------\n" +  
 "| Function 1 | D = SORT(A)+SORT(B)+SORT(C)\*(MA\*ME) |\n" +  
 "| Function 2 | MF = (MG\*MH)\*TRANS(MK) |\n" +  
 "| Function 3 | S = (MO\*MP)\*V+t\*MR\*(O+P) |\n" +  
 "----------------------------------------------------\n\n" +  
 "!!! Note that if the value of N > 10 -> the result will not be displayed !!!\n" +  
 "!!! If you enter N <= 0 - execution will be terminated !!!\n\n" + "Enter N: ");  
  
 // check for int value of N, else N = 3  
 if (scanner.hasNextInt()){  
 N = scanner.nextInt();  
 } else{  
 System.*out*.println("\n!!! You should enter data of type int, N will be taken as 3 !!!\n");  
 N = 3;  
 }  
  
 // check for positive value of N  
 if (N <= 0) throw new ArithmeticException("Restart the program and enter N > 0.");  
  
  
  
 if (N > 10){  
 throw new IllegalArgumentException("If you want to enter a value from the keyboard - enter N <= 10.");  
 }  
  
 System.*out*.print("\n!!! Enter All Values From The Keyboard !!!\n");  
  
 //------------------------- Main Body ----------------------------------  
 System.*out*.println("\nLab2 started!\n");  
  
 Data data = new Data(N);  
 Semaphore sem = new Semaphore(1);  
  
 Func1 T1 = new Func1("Func1", Thread.*NORM\_PRIORITY*, data, sem);  
 Func2 T2 = new Func2("Func2", Thread.*MIN\_PRIORITY*, data, sem);  
 Func3 T3 = new Func3("Func3", Thread.*MAX\_PRIORITY*, data, sem);  
  
 T1.start();  
 T2.start();  
 T3.start();  
  
 try {  
 T1.join();  
 T2.join();  
 T3.join();  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
  
 System.*out*.println("Lab2 finished.");  
 }  
  
 public static void main(String[] args){  
 new Lab2().start();  
 }  
}

**Func1.java**

import java.util.concurrent.Semaphore;  
  
public class Func1 extends Thread {  
  
 private Data data;  
 private Semaphore sem;  
  
 // конструктор класу Func1  
 Func1(String name, int priority, Data data, Semaphore sem){  
 setName(name);  
 setPriority(priority);  
 this.data = data;  
 this.sem = sem;  
 }  
  
 // D = SORT(A)+SORT(B)+SORT(C)\*(MA\*ME)  
 @Override  
 public void run(){  
 System.*out*.println("Func 1 started.");  
 try {  
 int[] A, B, C;  
 int[][] MA, ME;  
  
 // Generate Input Values  
 *sleep*(50);  
 System.*out*.println("Func 1 is waiting for a permit.");  
  
 sem.acquire();  
 *sleep*(100);  
  
 System.*out*.println("Func 1 gets a permit.\n");  
 A = data.vectorInput('A'); B = data.vectorInput('B'); C = data.vectorInput('C');  
 MA = data.matrixInput("MA"); ME = data.matrixInput("ME");  
  
 System.*out*.println("Func 1 releases the permit.");  
 sem.release();  
  
 // Calculate The Result  
 int[] result = data.func1(A, B, C, MA, ME);  
 *sleep*(100);  
  
 // Output  
 System.*out*.println("Func 1 is waiting for a permit.");  
 sem.acquire();  
  
 System.*out*.println("Func 1 gets a permit.\n");  
 System.*out*.print("Func 1 result:\n");  
 data.vectorOutput(result, 'D');  
  
 System.*out*.println("Func 1 releases the permit.");  
 sem.release();  
  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
  
  
 System.*out*.println("Func 1 finished.\n");  
 }  
}

**Func2.java**

import java.util.concurrent.Semaphore;  
  
public class Func2 extends Thread {  
  
 private Data data;  
 private Semaphore sem;  
  
 // конструктор класу Func2  
 Func2(String name, int priority, Data data, Semaphore sem){  
 setName(name);  
 setPriority(priority);  
 this.data = data;  
 this.sem = sem;  
 }  
  
  
 // MF = (MG\*MH)\*TRANS(MK)  
 @Override  
 public void run(){  
 System.*out*.println("Func 2 started.");  
 try {  
 int[][] MG, MH, MK;  
  
 // Generate Input Values  
 *sleep*(50);  
 System.*out*.println("Func 2 is waiting for a permit.");  
  
 sem.acquire();  
 *sleep*(100);  
  
 System.*out*.println("Func 2 gets a permit.\n");  
 MG = data.matrixInput("MG"); MH = data.matrixInput("MH"); MK = data.matrixInput("MK");  
  
 System.*out*.println("Func 2 releases the permit.");  
 sem.release();  
  
 // Calculate The Result  
 int[][] result = data.func2(MG, MH, MK);  
 *sleep*(100);  
  
 // Output  
 System.*out*.println("Func 2 is waiting for a permit.");  
 sem.acquire();  
  
 System.*out*.println("Func 2 gets a permit.\n");  
 System.*out*.print("Func 2 result:\n");  
 data.matrixOutput(result, "MF");  
  
 System.*out*.println("Func 2 releases the permit.");  
 sem.release();  
  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
  
 System.*out*.println("Func 2 finished.\n");  
 }  
}

**Func3.java**

import java.util.concurrent.Semaphore;  
  
public class Func3 extends Thread {  
  
 private Data data;  
 private Semaphore sem;  
  
 // конструктор класу Func3  
 Func3(String name, int priority, Data data, Semaphore sem){  
 setName(name);  
 setPriority(priority);  
 this.data = data;  
 this.sem = sem;  
 }  
  
 // S = (MO\*MP)\*V+t\*MR\*(O+P)  
 @Override  
 public void run() {  
 System.*out*.println("Func 3 started.");  
 try {  
 int t;  
 int[] V, O, P;  
 int[][] MO, MP, MR;  
  
 // Generate Input Values  
  
 *sleep*(50);  
 System.*out*.println("Func 3 is waiting for a permit.");  
  
 sem.acquire();  
 *sleep*(100);  
  
 System.*out*.println("Func 3 gets a permit.\n");  
 t = data.numInput('t');  
 V = data.vectorInput('V'); O = data.vectorInput('O'); P = data.vectorInput('P');  
 MO = data.matrixInput("MO"); MP = data.matrixInput("MP"); MR = data.matrixInput("MR");  
  
 System.*out*.println("Func 3 releases the permit.");  
 sem.release();  
  
 // Calculate The Result  
 int[] result = data.func3(t, V, O, P, MO, MP, MR);  
 *sleep*(100);  
  
 // Output  
 System.*out*.println("Func 3 is waiting for a permit.");  
 sem.acquire();  
  
 System.*out*.println("Func 3 gets a permit.\n");  
 System.*out*.print("Func 3 result:\n");  
 data.vectorOutput(result, 'S');  
  
 System.*out*.println("Func 3 releases the permit.");  
 sem.release();  
  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
  
 System.*out*.println("Func 3 finished.\n");  
 }  
}

**Data.java**

import java.util.Arrays;  
import java.util.Random;  
import java.util.Scanner;  
  
class Data {  
 private int N;  
  
 Data(int N){  
 this.N = N;  
 }  
  
 int getN(){  
 return N;  
 }  
  
 // ------------------- Create All-Ones Matrix And Vector -------------------  
 int[][] allOnesMatrix(){  
 int[][] MA = new int[N][N];  
 for (int i = 0; i < MA.length; i++) {  
 for (int j = 0; j < MA[i].length; j++) {  
 MA[i][j] = 1;  
 }  
 }  
 return MA;  
 }  
  
 int[] allOnesVector(){  
 int[] A = new int[N];  
 for (int i = 0; i < A.length; i++) {  
 A[i] = 1;  
 }  
 return A;  
 }  
  
  
 // ---------------- Create Random Matrix, Vector And Number ----------------  
 int[][] randomMatrix(){  
 int[][] MA = new int[N][N];  
 Random random = new Random();  
 for (int i = 0; i < MA.length; i++) {  
 for (int j = 0; j < MA[i].length; j++) {  
 MA[i][j] = random.nextInt(10);  
 }  
 }  
 return MA;  
 }  
  
 int[] randomVector(){  
 int[] A = new int[N];  
 Random random = new Random();  
 for (int i = 0; i < A.length; i++) {  
 A[i] = random.nextInt(10);  
 }  
 return A;  
 }  
  
 int randomNum(){  
 Random random = new Random();  
 return random.nextInt(10);  
 }  
  
  
 // ---------- Data Entry Handler For Matrices, Vectors And Numbers ---------  
 int[][] matrixInput(String name){  
 System.*out*.println("Enter the " + N\*N + " elements of the Matrix " + name + ":");  
 int[][] MA = new int[N][N];  
 Scanner scanner = new Scanner(System.*in*);  
 for (int i = 0; i < MA.length; i++) {  
 for (int j = 0; j < MA[i].length; j++) {  
 System.*out*.print(name + "[" + i + "][" + j + "] = ");  
 MA[i][j] = scanner.nextInt();  
 }  
 }  
 return MA;  
 }  
  
 int[] vectorInput(char name){  
 System.*out*.println("Enter the " + N + " elements of the Vector " + name + ":");  
 int[] input = new int[N];  
 Scanner scanner = new Scanner(System.*in*);  
 for (int i = 0; i < input.length; i++) {  
 System.*out*.print(name + "[" + i + "] = ");  
 input[i] = scanner.nextInt();  
 }  
 return input;  
 }  
  
 int numInput(char name){  
 System.*out*.print("Enter number " + name + " = ");  
 Scanner scanner = new Scanner(System.*in*);  
 return scanner.nextInt();  
 }  
  
  
 // ------------- Print Matrix, Vector And Number Into Console --------------  
 void matrixOutput(int[][] MA, String name){  
 System.*out*.println("\tMatrix " + name + ":");  
 for (int[] i : MA) {  
 System.*out*.print("\t\t");  
 for (int j : i) {  
 System.*out*.print(j + " ");  
 }  
 System.*out*.println();  
 }  
 }  
  
 void vectorOutput(int[] A, char name){  
 System.*out*.print("\tVector " + name + ": ");  
 for (int i : A) {  
 System.*out*.print(i + " ");  
 }  
 System.*out*.println();  
 }  
  
 void numOutput(int a, char name){  
 System.*out*.print("\tNumber " + name + ": " + a + "\n");  
 }

// Sort Vector  
 private int[] sortVector(int[] A){  
 Arrays.*sort*(A);  
 return A;  
 }  
  
  
 // Calculates Sum Of 2 Vectors  
 private int[] sumVectors(int[] A, int[] B){  
 int[] C = new int[N];  
 for (int i = 0; i < N ; i++){  
 C[i] = A[i] + B[i];  
 }  
 return C;  
 }  
  
  
 // Transposing Matrix  
 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;  
 }  
  
  
 // Multiply 2 Matrices  
 private int[][] matrixMult(int[][] MA, int[][] MB){  
 int[][] MC = 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++){  
 MC[i][j] += MA[i][k] \* MB[k][j];  
 }  
 }  
 }  
 return MC;  
 }  
  
  
 // Multiply Matrix And Vector  
 private int[] vectorMatrixMult(int[] A, int[][] MA){  
 int[] B = new int[N];  
 for (int i = 0; i < N ; i++){  
 for (int j = 0; j < N ; j++){  
 B[i] += A[j] \* MA[i][j];  
 }  
 }  
 return B;  
 }  
  
  
 // Multiply Integer And Matrix  
 private int[] intVectorMult(int a, int[] A) {  
 int[] B = new int[N];  
 for (int i = 0; i < N ; i++){  
 B[i] = a \* A[i];  
 }  
 return B;  
 }  
  
  
 // F1 -> D = SORT(A)+SORT(B)+SORT(C)\*(MA\*ME)  
 int[] func1(int[] A, int[] B, int[] C, int[][] MA, int[][] ME){  
 return sumVectors(sumVectors(sortVector(A), sortVector(B)),  
 vectorMatrixMult(sortVector(C), matrixMult(MA, ME)));  
 }  
  
 // F2 -> MF = (MG\*MH)\*TRANS(MK)  
 int[][] func2(int[][] MG, int[][] MH, int[][] MK){  
 return matrixMult(matrixMult(MG, MH), matrixTransp(MK));  
 }  
  
 // F3 -> S = (MO\*MP)\*V+t\*MR\*(O+P)  
 int[] func3(int t, int[] V, int[] O, int[] P, int[][] MO, int[][] MP, int[][] MR){  
 return sumVectors((vectorMatrixMult(V, matrixMult(MO, MP))),  
 intVectorMult(t, vectorMatrixMult(sumVectors(O, P), MR))  
 );  
 }  
}

**Приклад роботи програми**

----------------------------------------------------

| Function 1 | D = SORT(A)+SORT(B)+SORT(C)\*(MA\*ME) |

| Function 2 | MF = (MG\*MH)\*TRANS(MK) |

| Function 3 | S = (MO\*MP)\*V+t\*MR\*(O+P) |

----------------------------------------------------

!!! Note that if the value of N > 10 -> the result will not be displayed !!!

!!! If you enter N <= 0 - execution will be terminated !!!

Enter N: 2

Enter All Values From The Keyboard.

Lab2 started!

Func 1 started.

Func 2 started.

Func 3 started.

Func 3 is waiting for a permit.

Func 1 is waiting for a permit.

Func 2 is waiting for a permit.

Func 3 gets a permit.

Enter number t = 3

Enter the 2 elements of the Vector V:

V[0] = 3

V[1] = 3

Enter the 2 elements of the Vector O:

O[0] = 3

O[1] = 3

Enter the 2 elements of the Vector P:

P[0] = 3

P[1] = 3

Enter the 4 elements of the Matrix MO:

MO[0][0] = 3

MO[0][1] = 3

MO[1][0] = 3

MO[1][1] = 3

Enter the 4 elements of the Matrix MP:

MP[0][0] = 3

MP[0][1] = 3

MP[1][0] = 3

MP[1][1] = 3

Enter the 4 elements of the Matrix MR:

MR[0][0] = 3

MR[0][1] = 3

MR[1][0] = 3

MR[1][1] = 3

Func 3 releases the permit.

Func 3 is waiting for a permit.

Func 2 gets a permit.

Enter the 4 elements of the Matrix MG:

MG[0][0] = 2

MG[0][1] = 2

MG[1][0] = 2

MG[1][1] = 2

Enter the 4 elements of the Matrix MH:

MH[0][0] = 2

MH[0][1] = 2

MH[1][0] = 2

MH[1][1] = 2

Enter the 4 elements of the Matrix MK:

MK[0][0] = 2

MK[0][1] = 2

MK[1][0] = 2

MK[1][1] = 2

Func 2 releases the permit.

Func 2 is waiting for a permit.

Func 1 gets a permit.

Enter the 2 elements of the Vector A:

A[0] = 1

A[1] = 1

Enter the 2 elements of the Vector B:

B[0] = 1

B[1] = 1

Enter the 2 elements of the Vector C:

C[0] = 1

C[1] = 1

Enter the 4 elements of the Matrix MA:

MA[0][0] = 1

MA[0][1] = 1

MA[1][0] = 1

MA[1][1] = 1

Enter the 4 elements of the Matrix ME:

ME[0][0] = 1

ME[0][1] = 1

ME[1][0] = 1

ME[1][1] = 1

Func 1 releases the permit.

Func 3 gets a permit.

Func 3 result:

Vector S: 216 216

Func 3 releases the permit.

Func 3 finished.

Func 2 gets a permit.

Func 2 result:

Matrix MF:

32 32

32 32

Func 2 releases the permit.

Func 2 finished.

Func 1 is waiting for a permit.

Func 1 gets a permit.

Func 1 result:

Vector D: 6 6

Func 1 releases the permit.

Func 1 finished.

Lab2 finished.

Process finished with exit code 0