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»**

Виконала:

студентка групи ІВ-71

Молчанова В.С.

Перевірив:

Долголенко О.М.

Київ

2019 р.

**Завдання:**

F1: A = B\*MIN(C) \*(MA\*MD+MD)

F2: MK = MA\*(MA\*MZ) + TRANS(MB)

F3: O = SORT(R + S)\*(MT\*MP)

**Лістинг програми:**

**main.java**

import Data.\*;

public class Main{

public static void main(String[] args) {

F1 f1 = new F1("Task 1", Thread.NORM\_PRIORITY);

F2 f2 = new F2("Task 2", Thread.MIN\_PRIORITY);

F3 f3 = new F3("Task 3", Thread.MAX\_PRIORITY);

f1.start();

f2.start();

f3.start();

try {

f1.join();

f2.join();

f3.join();

System.out.println();

System.out.println(f1.getResult());

System.out.println(f2.getResult());

System.out.println(f3.getResult());

} catch (InterruptedException e) {

e.printStackTrace();

}

}

public void run() {

}

}

**Data.java**

package Data;

public class Data {

public static int func1(Vector A, Vector B, Matrix MA, Matrix MB) {

return max(multiply(MA, MB)) \* multiply(A, B);

}

public static Matrix func2(Matrix MG, Matrix MH, Matrix MK) {

return multiply(max(MG), multiply(MH, MK));

}

public static Matrix func3(Vector O, Vector P, Matrix MR, Matrix MS) {

return multiply(sort(add(O, P)), transpose(multiply(MR, MS)));

}

public static class Vector {

public int Length;

private int[] \_vector;

public Vector(int n) {

\_vector = new int[n];

Length = n;

}

public Vector(int[] vector) {

\_vector = vector;

Length = vector.length;

}

public void fillByOrder() {

for (var i = 0; i < Length; i++)

{

\_vector[i] = i;

}

}

public void fillWithOnes() {

for (var i = 0; i < Length; i++)

{

\_vector[i] = 1;

}

}

@Override

public String toString() {

StringBuffer result = new StringBuffer("");

for (int i = 0; i < Length; i++)

{

var str = String.format("%3d ",\_vector[i]);

result.append(str);

}

return result+"\n";

}

public int getAt(int index) {

return \_vector[index];

}

}

public static class Matrix {

public int RowNumber;

public int ColumnNumber;

private Vector[] \_matrix;

public Matrix(int n, int m) {

RowNumber = n;

ColumnNumber = m;

\_matrix = new Vector[n];

for (var i = 0; i < n; i++)

{

\_matrix[i] = new Vector(m);

}

}

public Matrix(Vector[] matrix) {

\_matrix = matrix;

RowNumber = matrix.length;

ColumnNumber = matrix[0].Length;

}

public void fillByOrder() {

for(var i = 0; i < RowNumber; i++)

for (var j = 0; j < ColumnNumber; j++)

{

\_matrix[i].\_vector[j] = ColumnNumber \* i + j;

}

}

public void fillWithOnes() {

for(var i = 0; i < RowNumber; i++)

for (var j = 0; j < ColumnNumber; j++)

{

\_matrix[i].\_vector[j] = 1;

}

}

@Override

public String toString() {

StringBuffer result = new StringBuffer("");

for (int i = 0; i < RowNumber; i++)

{

result.append(\_matrix[i]);

}

return result.toString();

}

public int getAt(int n, int m) {

return \_matrix[n].\_vector[m];

}

public void setAt(int n, int m, int value) {

\_matrix[n].\_vector[m] = value;

}

}

public static Vector sort(Vector v){

int[] newVector = v.\_vector.clone();

for (int i = 0; i < v.Length; ++i) {

for (int j = 0; j < v.Length - i - 1; ++j) {

if (newVector[j] > newVector[j + 1]) {

int t = newVector[j];

newVector[j] = newVector[j + 1];

newVector[j + 1] = t;

}

}

}

return new Vector(newVector);

}

public static Matrix sort(Matrix m){

Vector[] newMatrix = new Vector[m.RowNumber];

for (int i = 0; i < m.RowNumber; i++) {

newMatrix[i] = sort(m.\_matrix[i]);

}

return new Matrix(newMatrix);

}

public static int max(Vector v){

int max = v.\_vector[0];

for (int i = 1; i < v.Length; i++)

{

if (v.\_vector[i] > max) max = v.\_vector[i];

}

return max;

}

public static int min(Vector v){

int min = v.\_vector[0];

for (int i = 1; i < v.Length; i++)

{

if (v.\_vector[i] < min) min = v.\_vector[i];

}

return min;

}

public static int max(Matrix m) {

int max = max(m.\_matrix[0]);

for (int i = 1; i < m.RowNumber; i++)

{

var currentVectorMax = max(m.\_matrix[i]);

if (currentVectorMax > max) max = currentVectorMax;

}

return max;

}

public static int min(Matrix m) {

int min = min(m.\_matrix[0]);

for (int i = 1; i < m.RowNumber; i++)

{

var currentVectorMin = min(m.\_matrix[i]);

if (currentVectorMin < min) min = currentVectorMin;

}

return min;

}

public static Vector add(Vector v1, Vector v2) {

var result = new Vector(v1.Length);

for (var i = 0; i < v1.Length; i++)

{

result.\_vector[i]=v1.\_vector[i] + v2.\_vector[i];

}

return result;

}

public static Vector multiply(int a, Vector v) {

var result = new Vector(v.Length);

for (var i = 0; i < v.Length; i++)

{

result.\_vector[i] = v.\_vector[i] \* a;

}

return result;

}

public static Vector multiply(Vector v, int a) { return multiply(a, v);}

public static Matrix multiply(Vector v, Matrix m){

var matrixFromVector = new Matrix(new Vector[]{v});

return multiply(matrixFromVector, m);

}

public static int multiply(Vector v1, Vector v2) {

var result = 0;

for (var i = 0; i < v1.Length; i++)

{

result += v1.\_vector[i] \* v2.\_vector[i];

}

return result;

}

public static Matrix add(Matrix m1, Matrix m2) {

var result = new Matrix(m1.RowNumber, m1.ColumnNumber);

for (var i = 0; i < m1.RowNumber; i++)

{

result.\_matrix[i] = add(m1.\_matrix[i], m2.\_matrix[i]);

}

return result;

}

public static Matrix multiply(int a, Matrix m) {

var result = new Matrix(m.RowNumber, m.ColumnNumber);

for (var i = 0; i < m.RowNumber; i++)

{

result.\_matrix[i] = multiply(a, m.\_matrix[i]);

}

return result;

}

public static Matrix multiply(Matrix m, int a) { return multiply(a, m);}

public static Matrix multiply(Matrix m1, Matrix m2) {

var result = new Matrix(m1.RowNumber, m2.ColumnNumber);

var transposedMatrix2 = transpose(m2);

for (var i = 0; i < m1.RowNumber; i++)

for (var j = 0; j < transposedMatrix2.RowNumber; j++)

{

result.\_matrix[i].\_vector[j] = multiply(m1.\_matrix[i], transposedMatrix2.\_matrix[j]);

}

return result;

}

public static Matrix transpose(Matrix m){

var result = new Matrix(m.ColumnNumber, m.RowNumber);

for (var i = 0; i < m.ColumnNumber; i++)

for (var j = 0; j < m.RowNumber; j++)

{

result.\_matrix[i].\_vector[j] = m.\_matrix[j].\_vector[i];

}

return new Matrix(result.\_matrix);

}

}

**F1.java**

package Data;

import Data.Data.\*;

public class F1 extends Thread {

int result;

public F1(String name, int priority) {

setName(name);

setPriority(priority);

}

@Override

public void run() {

try {

System.out.println(getName()+" started");

var A = new Vector(5);

var B = new Vector(5);

var MA = new Matrix(5, 3);

var MB = new Matrix(3, 5);

A.fillWithOnes();

B.fillWithOnes();

MA.fillWithOnes();

MB.fillWithOnes();

result = Data.func1(A, B, MA, MB);

sleep(500);

System.out.println(getName()+" ended");

}

catch (InterruptedException e) {

e.printStackTrace();

}

}

public String getResult()

{

return function+"\n"+result+"\n";

}

}

**F2.java**

package Data;

import Data.Data.\*;

public class F2 extends Thread {

Matrix result;

public F2(String name, int priority) {

setName(name);

setPriority(priority);

}

@Override

public void run() {

try {

System.out.println(getName()+" started");

var MG = new Matrix(6, 6);

var MH = new Matrix(5, 4);

var MK = new Matrix(4, 5);

MG.fillWithOnes();

MH.fillWithOnes();

MK.fillWithOnes();

result = Data.func2(MG, MH, MK);

sleep(500);

System.out.println(getName()+" ended");

}

catch (InterruptedException e) {

e.printStackTrace();

}

}

public String getResult()

{

return function+"\n"+result;

}

}

**F3.java**

package Data;

import Data.Data.\*;

public class F3 extends Thread {

Matrix result;

public F3(String name, int priority) {

setName(name);

setPriority(priority);

}

@Override

public void run() {

try {

System.out.println(getName()+" started");

var O = new Vector(5);

var P = new Vector(5);

O.fillWithOnes();

P.fillWithOnes();

var MR = new Matrix(5, 3);

var MS = new Matrix(3, 5);

MR.fillWithOnes();

MS.fillWithOnes();

result = Data.func3(O, P , MR, MS);

sleep(500);

System.out.println(getName()+" ended");

}

catch (InterruptedException e) {

e.printStackTrace();

}

}

public String getResult()

{

return function+"\n"+result;

}

private String function = "F3: T = SORT(O+P)\*TRANS(MR\* MS)";

}

**Результат роботи:**

Task 3 started

Task 1 started

Task 2 started

Task 3 ended

Task 1 ended

Task 2 ended

F1:

12 12 12 12 12

F2:

4 4 4 4 4

4 4 4 4 4

4 4 4 4 4

4 4 4 4 4

4 4 4 4 4

F3:

30 30 30 30 30