

Національний технічний університет України  
«Київський політехнічний інститут»  
Факультет інформатики та обчислювальної техніки  
Кафедра обчислювальної техніки

Лабораторна робота №3

Java.

Виконав:  
студент групи ІП-32  
Ковальчук О. М.  
Перевірів:  
Корочкін О. В.

Київ - 2015

## Лабораторна робота №2. АДА. Задачі

**Мета роботи:** вивчення засобів мови Java для роботи з потоками

**Мова програмування:** Java

**Завдання:** Розробити програму, що містить паралельні потоки, кожен з яких реалізує функції F1, F2, F3 з лабораторної роботи №1. Вимоги до потоків такі ж, як в лабораторній роботі № 2. В потоках використовувати методи `sleep()` та `join()`.

**Функції:**

F1:  $C = A - B * (MA * MD)$

F2:  $o = \text{Min}(MK * MM)$

F3:  $T = (MS * MZ) * (W + X)$

### Лістинг програми

```
./edu/kpi/pp/threading/Tasks.java
 1 package edu.kpi.pp.threading;
 2
 3
 4 import edu.kpi.pp.data.Vector;
 5 import edu.kpi.pp.data.Matrix;
 6 import edu.kpi.pp.data.Functions;
 7
 8
 9 class TaskBase {
10
11     protected final int size;
12
13     TaskBase(int size) {
14         this.size = size;
15     }
16
17     protected void sleep(long millis) {
18         try {
19             Thread.sleep(millis);
20         } catch (InterruptedException ie) {
21             throw new RuntimeException("Error in thread: " + ie.getMessage());
22         }
23     }
24 }
25
26 class Task1 extends TaskBase implements Runnable {
27
28     public Task1(int size) {
29         super(size);
30     }
31
32     @Override
33     public void run() {
34         System.out.println("Task1 started");
35         this.sleep(500);
36
37         double[] a, b, c;
38         double[] [] ma, md;
39
40         a = Vector.ones(this.size);
41         b = Vector.ones(this.size);
```

```

42
43     ma = Matrix.ones(this.size);
44     md = Matrix.ones(this.size);
45
46     c = Functions.func1(a, b, ma, md);
47
48     if (this.size < 8) {
49         System.out.print(
50             Vector.toString(c)
51         );
52     }
53
54     System.out.println("Task1 finished");
55 }
56 }
57
58 class Task2 extends TaskBase implements Runnable {
59
60     public Task2(int size) {
61         super(size);
62     }
63
64     @Override
65     public void run() {
66         System.out.println("Task2 started");
67         this.sleep(500);
68
69         double[] [] mk, mm;
70         double o;
71
72         mk = Matrix.ones(this.size);
73         mm = Matrix.ones(this.size);
74
75         o = Functions.func2(mk, mm);
76
77         if (this.size < 8) {
78             System.out.printf("%10.2f%n", o);
79         }
80
81         System.out.println("Task2 finished");
82     }
83 }
84
85 class Task3 extends TaskBase implements Runnable {
86
87     public Task3(int size) {
88         super(size);
89     }
90
91     @Override
92     public void run() {
93         System.out.println("Task3 started");
94         this.sleep(500);
95
96         double[] w, x, t;
97         double[] [] ms, mz;
98
99         w = Vector.ones(this.size);
100        x = Vector.ones(this.size);
101
102        ms = Matrix.ones(this.size);
103        mz = Matrix.ones(this.size);
104
105        t = Functions.func3(ms, mz, w, x);
106
107        if (this.size < 8) {

```

```

108         System.out.print(
109             Vector.toString(t)
110         );
111     }
112
113     System.out.println("Task3 finished");
114 }
115 }
116
117
118
119
./edu/kpi/pp/threading/TaskThreadGenerator.java
1  package edu.kpi.pp.threading;
2  import edu.kpi.pp.threading.Task1;
3  import edu.kpi.pp.threading.Task2;
4  import edu.kpi.pp.threading.Task3;
5
6  public class TaskThreadGenerator {
7      public static final byte TASK_FUNCTION_1 = 1;
8      public static final byte TASK_FUNCTION_2 = 2;
9      public static final byte TASK_FUNCTION_3 = 3;
10
11     private TaskThreadGenerator() {
12         // Do nothing
13     };
14
15     public static Thread getThread(byte taskId, int size) {
16         Thread taskThread = null;
17         switch (taskId) {
18             case TASK_FUNCTION_1:
19                 taskThread = new Thread(new Task1(size));
20                 break;
21             case TASK_FUNCTION_2:
22                 taskThread = new Thread(new Task2(size));
23                 break;
24             case TASK_FUNCTION_3:
25                 taskThread = new Thread(new Task3(size));
26                 break;
27             default:
28                 throw new RuntimeException("Invalid task id");
29         }
30         taskThread.setPriority(Thread.NORM_PRIORITY);
31         return taskThread;
32     }
33 }
./edu/kpi/pp/Lab3.java
1  package edu.kpi.pp;
2
3  import edu.kpi.pp.threading.TaskThreadGenerator;
4
5  /**
6   * Parallel programming
7   * Lab 3
8   *
9   * Functions:
10  * F1:  $C := A - B * (MA * MD)$ 
11  * F2:  $o := \text{Min}(MK * MM)$ 
12  * F3:  $T := (MS * MZ) * (W + X)$ 
13  *
14  * @since 2015-10-04
15  * @author Olexandr Kovalchuk
16  * IP-32
17  *
18  */
19  public class Lab3 {

```

```

20     public static void main(String[] args) throws InterruptedException {
21         int size = 4;
22         if (args.length > 0) {
23             size = Integer.parseInt(args[0]);
24         }
25
26         System.out.println("Lab3 started");
27
28         Thread[] threads = new Thread[3];
29         threads[0] = TaskThreadGenerator.getThread(TaskThreadGenerator.TASK_FUNCTION_1, size);
30         threads[1] = TaskThreadGenerator.getThread(TaskThreadGenerator.TASK_FUNCTION_2, size);
31         threads[2] = TaskThreadGenerator.getThread(TaskThreadGenerator.TASK_FUNCTION_3, size);
32
33         for (Thread t : threads) {
34             t.start();
35         }
36
37         for (Thread t : threads) {
38             try {
39                 t.join();
40             } catch (InterruptedException ie) {
41                 System.out.println("Error while joining " + t);
42             }
43         }
44
45         System.out.println("Lab3 finished");
46     }
47 }
48
./edu/kpi/pp/data/Functions.java
1  package edu.kpi.pp.data;
2
3
4  import edu.kpi.pp.data.Matrix;
5  import edu.kpi.pp.data.Vector;
6
7
8  public class Functions {
9      public static double[] func1(double[] a, double[] b, double[][] ma, double[][] md) {
10         return (
11             Vector.subtract(
12                 a,
13                 Vector.multiply(
14                     b,
15                     Matrix.multiply(ma, md)
16                 )
17             )
18         );
19     }
20
21     public static double func2(double[][] mk, double[][] mm) {
22         return (
23             Matrix.min(
24                 Matrix.multiply(mk, mm)
25             )
26         );
27     }
28
29     public static double[] func3(double[][] ms, double[][] mz, double[] w, double[] x) {
30         return (
31             Matrix.multiply(
32                 Matrix.multiply(ms, mz),
33                 Vector.add(w, x)
34             )
35         );
36     }

```

```

37  }
./edu/kpi/pp/data/Vector.java
1  package edu.kpi.pp.data;
2
3
4  import edu.kpi.pp.data.Matrix;
5
6
7  public class Vector {
8
9      public static double[] ones(int size) {
10         double[] vector = new double[size];
11         for (int i = 0; i < size; i++)
12             vector[i] = 1;
13         return vector;
14     }
15
16     // return a random m-by-n matrix with values between 0 and 1
17     public static double[] random(int size) {
18         double[] vector = new double[size];
19         for (int i = 0; i < size; i++)
20             vector[i] = Math.random();
21         return vector;
22     }
23
24     // return C = A + B
25     public static double[] add(double[] left, double[] right) {
26         double[] result = new double[left.length];
27         for (int i = 0; i < left.length; i++) {
28             result[i] = left[i] + right[i];
29         }
30         return result;
31     }
32
33     // return C = A - B
34     public static double[] subtract(double[] left, double[] right) {
35         double[] result = new double[left.length];
36         for (int i = 0; i < left.length; i++) {
37             result[i] = left[i] - right[i];
38         }
39         return result;
40     }
41
42     // matrix-vector multiplication ( $y = A * x$ )
43     public static double[] multiply(double[][] left, double[] right) {
44         return Matrix.multiply(left, right);
45     }
46
47     // vector-matrix multiplication ( $y = x^T A$ )
48     public static double[] multiply(double[] left, double[][] right) {
49         return Matrix.multiply(left, right);
50     }
51
52     public static String toString(double[] vector) {
53         StringBuffer sb = new StringBuffer();
54         for (double value: vector) {
55             sb.append(String.format("%10.2f,", value));
56         }
57         sb.append('\n');
58         return sb.toString();
59     }
60
61 }
./edu/kpi/pp/data/Matrix.java
1  package edu.kpi.pp.data;
2

```

```

3
4 public class Matrix {
5
6     public static double[] [] ones(int size) {
7         return Matrix.ones(size, size);
8     }
9
10    public static double[] [] ones(int rows, int cols) {
11        double[] [] matrix = new double[rows][cols];
12        for (int r = 0; r < rows; r++)
13            for (int c = 0; c < cols; c++)
14                matrix[r][c] = 1;
15        return matrix;
16    }
17
18    // return a random m-by-n matrix with values between 0 and 1
19    public static double[] [] random(int rows, int columns) {
20        double[] [] matrix = new double[rows][columns];
21        for (int r = 0; r < rows; r++)
22            for (int c = 0; c < columns; c++)
23                matrix[r][c] = Math.random();
24        return matrix;
25    }
26
27    // return n-by-n identity matrix I
28    public static double[] [] identity(int size) {
29        double[] [] identityMatrix = new double[size][size];
30        for (int i = 0; i < size; i++)
31            identityMatrix[i][i] = 1;
32        return identityMatrix;
33    }
34
35    // return C = A + B
36    public static double[] [] add(double[] [] left, double[] [] right) {
37        int rows = left.length;
38        int cols = left[0].length;
39        double[] [] result = new double[rows][cols];
40        for (int r = 0; r < rows; r++)
41            for (int c = 0; c < cols; c++)
42                result[r][c] = left[r][c] + right[r][c];
43        return result;
44    }
45
46    // return C = A - B
47    public static double[] [] subtract(double[] [] left, double[] [] right) {
48        int rows = left.length;
49        int cols = left[0].length;
50        double[] [] result = new double[rows][cols];
51        for (int r = 0; r < rows; r++)
52            for (int c = 0; c < cols; c++)
53                result[r][c] = left[r][c] - right[r][c];
54        return result;
55    }
56
57    // return C = A * B
58    public static double[] [] multiply(double[] [] left, double[] [] right) {
59        int rLeft = left.length;
60        int cLeft = left[0].length;
61        int rRight = right.length;
62        int cRight = right[0].length;
63        if (cLeft != rRight) {
64            throw new RuntimeException("Illegal matrix dimensions.");
65        }
66        double[] [] result = new double[rLeft][cRight];
67        for (int i = 0; i < rLeft; i++)
68            for (int j = 0; j < cRight; j++)

```

```

69         for (int k = 0; k < cLeft; k++)
70             result[i][j] += left[i][k] * right[k][j];
71     return result;
72 }
73
74 // matrix-vector multiplication (y = A * x)
75 public static double[] multiply(double[][] left, double[] right) {
76     int rows = left.length;
77     int cols = left[0].length;
78     if (right.length != cols) {
79         throw new RuntimeException("Illegal matrix dimensions.");
80     }
81     double[] result = new double[rows];
82     for (int r = 0; r < rows; r++)
83         for (int c = 0; c < cols; c++)
84             result[r] += left[r][c] * right[c];
85     return result;
86 }
87
88 // vector-matrix multiplication (y = x^T A)
89 public static double[] multiply(double[] left, double[][] right) {
90     int rows = right.length;
91     int cols = right[0].length;
92     if (left.length != rows) {
93         throw new RuntimeException("Illegal matrix dimensions.");
94     }
95     double[] result = new double[cols];
96     for (int c = 0; c < cols; c++)
97         for (int r = 0; r < rows; r++)
98             result[c] += right[r][c] * left[r];
99     return result;
100 }
101
102 public static double min(double[][] matrix) {
103     double min = matrix[0][0];
104     for (int r = 0; r < matrix.length; r++) {
105         for (int c = 0; c < matrix[0].length; c++) {
106             if (matrix[r][c] < min) {
107                 min = matrix[r][c];
108             }
109         }
110     }
111     return min;
112 }
113
114 public static String toString(double[][] matrix) {
115     StringBuffer sb = new StringBuffer();
116     for (double[] row : matrix) {
117         for (double value : row) {
118             sb.append(String.format("%10.2f", value));
119         }
120         sb.append('\n');
121     }
122     return sb.toString();
123 }
124
125 }

```