# Darea de seamă la Java: Lucrare de laborator nr. 5.1

Tema: Programare paralela in java

Varianta: Nr. 16

Efectuat: Țurcanu Cristian, studentul grupei IA1901

Verificat: Epifanova Irina, lector universitar, magistru în informatică

## Formularea problemei:

Se dă matrice de tip double. De divizat matricea pe coloane, fiecare coloană fiind atribuită unui fir de execuție separat. Fiecare fir trebuie să găsească în coloana prelucrată numărul elementelor mai mari decît x, unde x – număr introdus de la tastatură. În main de afișat elementul maxim din rezultatele firelor. De utilizat clasa Thread.

## Codul sursă

DoubleMatrix.java

```
public class DoubleMatrix extends Thread {
    static double Matrix[][];
    static int rows, cols, globalGreaterThanX;
    int threadId;
    double x;


    static {
        rows = Helper.randomizer.nextInt(6) + 5;
        cols = Helper.randomizer.nextInt(6) + 5;
        Matrix = new double[rows][cols];

        for (int i = 0; i < rows; i++)
            for (int j = 0; j < cols; j++)
                Matrix[i][j] = Helper.randomizer.nextDouble() * 20 - 10;
    }

    static void PrintMatrix() {
        for (int i = 0; i < rows; i++)  {
            for (int j = 0; j < cols; j++) {
                Helper.print(Matrix[i][j] + " ");
            }

            Helper.println("");
        }
    }

    public DoubleMatrix(int id, double x) {
        this.threadId = id;
        this.x = x;
    }

    public void run() {
        int localGreaterThanX = 0;

        for (int i = 0; i < rows; i++)
            if (Matrix[i][threadId] > x)
                localGreaterThanX++;

        Helper.println("Thread " + threadId + " finished with result " +
localGreaterThanX);

        SumToGlobalGreaterThanX(localGreaterThanX);
    }

    private static synchronized void SumToGlobalGreaterThanX(int x) {
        globalGreaterThanX += x;
    }
}
```

Main.java

```
public class Main{
    public static void main(String[] args) {
        DoubleMatrix.PrintMatrix();

        Helper.print("X: ");
        double x = Helper.InputFloat(); //extends to double

        for (int j = 0; j < DoubleMatrix.cols; j++)
            (new DoubleMatrix(j, x)).start();

        try {
            Thread.sleep(1000);
        } catch (InterruptedException e) { }

        Helper.println("Elements greater than " + x + ": " +
DoubleMatrix.globalGreaterThanX);

    }
}
```

Helper.java

```
import java.util.Random;
import java.io.*;

public class Helper {
    public static Random randomizer = new Random();

    public static String InputString() {
        BufferedReader box = new BufferedReader(new InputStreamReader(System.in));

        String str = "";
        try {
            str = box.readLine();
        } catch (Exception e) {
            System.out.println("Failed row reading at Helper.InputString");
            System.out.println(e);
        }

        return str;
    }

    public static int InputInt() {
        boolean success = false;
        int result = 0;

        do {
            try {
                result = (Integer.valueOf(InputString())).intValue();
                success = true;
            } catch (Exception e) {
                System.out.println(e);
                System.out.print("Dati valoarea inca o data: ");
                success = false;
            }
        } while (!success);

        return result;
    }

    public static float InputFloat() {
        boolean success = false;
        float result = 0;

        do {
            try {
                result = (Float.valueOf(InputString())).floatValue();
                success = true;
            } catch (Exception e) {
                System.out.println(e);
                System.out.print("Dati valoarea inca o data: ");
                success = false;
            }
        } while (!success);

        return result;
    }

    public static int InputIntLimit(int min, int max) {
        int result = min - 1;

        do {
            result = InputInt();
            if (result < min || result > max) {
                println("Outside of limits [" + min + ", " + max + "], try
again");
            }
        } while (result < min || result > max);

        return result;
    }

    public static int InputIntLimit(int min) {
        return InputIntLimit(min, Integer.MAX_VALUE);
    }

    public static float InputFloatLimit(float min, float max) {
        float result = min - 1f;

        do {
            result = InputFloat();
            if (result < min || result > max) {
                println("Outside of limits [" + min + ", " + max + "], try
again");
            }
        } while (result < min || result > max);

        return result;
    }

    public static boolean InputBoolean() {
        boolean success = false;
        boolean result = false;
        do {
            try {
                result = Boolean.parseBoolean(InputString());
                success = true;
            } catch (Exception e) {
                println("Shit's fucked man");
                println(e.toString());
                println("Try again");
                success = false;
            }
        } while (!success);

        return result;
    }

    public static float InputFloatLimit(float min) {
        return InputFloatLimit(min, Float.MAX_VALUE);
    }

    public static void println(String text) {
        System.out.println(text);
    }

    public static void print(String text) {
        System.out.print(text);
    }

    public static String FileRead(String path) {
        String result = "";
        try {
            BufferedReader box = new BufferedReader(new FileReader(path));

            String line;
            while ((line = box.readLine()) != null) {
                result += line;
            }
            box.close();
        } catch (Exception e) {
            println("Failed file reading at Helper.FileRead");
            println(e.toString());
        }

        return result;
    }

    public static void FileWrite(String path, String text) {
        try {
            File file = new File(path);
            FileOutputStream fileOutputStream = new FileOutputStream(file);

            if (!file.exists()) {
                file.createNewFile();
            }

            byte b[] = text.getBytes();

            fileOutputStream.write(b);
            fileOutputStream.flush();
            fileOutputStream.close();
        } catch (Exception e) {
            println("Failed file writing at Helper.FileWrite");
            println(e.toString());
        }
    }
}
```

## Rezultatele rulării programului

```
-4.021475325157564 -7.146070018299664 8.176570641671756 9.245294681262223
7.686131820684221 5.290022837541246 -5.79491245530023
-7.076356823415528 -8.189164002506178 -4.044601784522468 4.7178140525285155
```

```
-5.711083542211308 0.47992650124647085 5.700144145457543
-4.824338728792683 -1.6493515666546266 -9.773338448750994 3.336296705813375
-4.629405520646646 -4.688368222448423 -4.76290978395272
-7.828990555869757 -5.925401582038649 -6.196707947339295 -6.627987231249857
2.2746053014945637 7.150114137394414 -1.0470703248930668
9.115073346388513 -9.098885348586096 -7.967281478684598 -7.524178935896472
-7.226228873221492 -3.0640661797722597 4.4850953663136846
5.754705597304419 -8.939043522672254 4.327260168211797 -2.8653777571476535
6.263712284567806 0.6101502495926425 4.165017183303812
-6.454809734538644 2.713313960286923 -6.97118409663219 5.402979214897215
-3.562857961464352 -5.062572547509876 -2.2305064438181743
-9.823134376784244 4.157504400402161 -9.186691406807538 7.399479520843851
1.2531918461050306 -4.384030984475875 4.3994216853501715
X: 0
Thread 4 finished with result 4
Thread 1 finished with result 2
Thread 0 finished with result 2
Thread 2 finished with result 2
Thread 5 finished with result 4
Thread 6 finished with result 4
Thread 3 finished with result 5
Elements greater than 0.0: 23
```