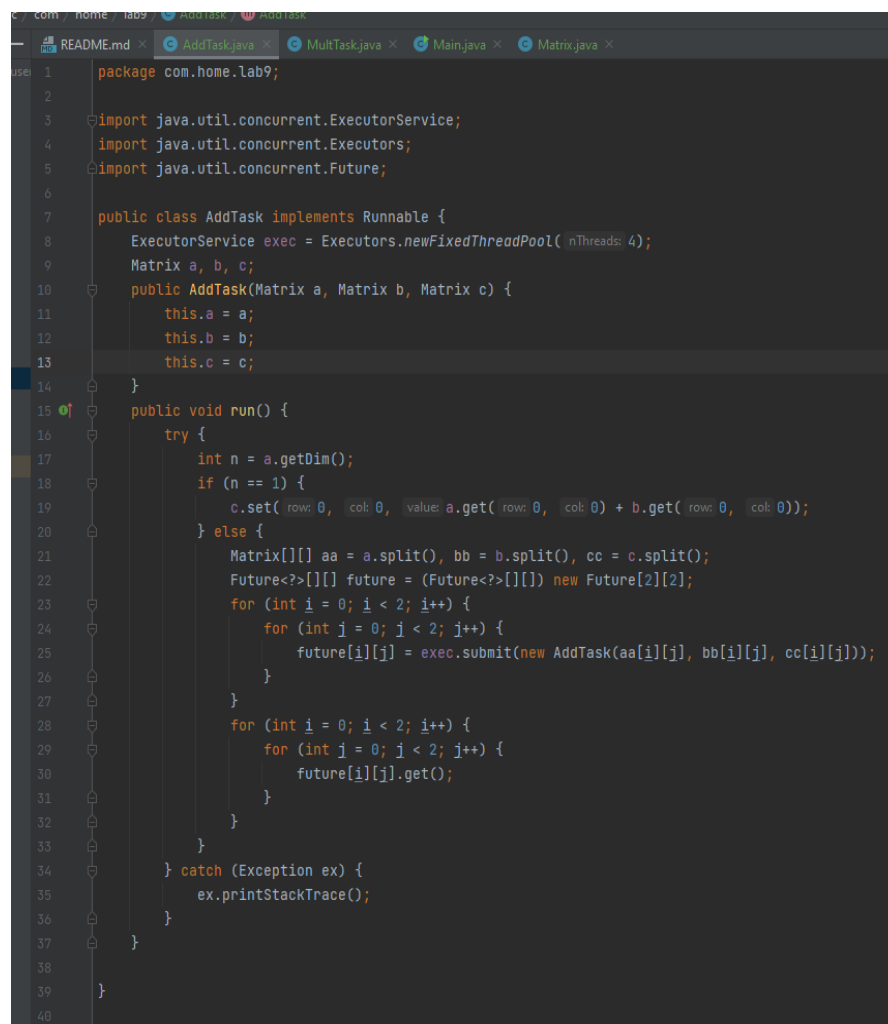


## Laborator 10

În acest laborator am creat un nou proiect Java unde am implementat codul pentru adunarea matricelor utilizând metoda divide-et-impera din cursul 9, iar pe baza clasei AddTask am mai implementat o clasă numită MultTask care ne ajută să multiplicăm două matrice.

Mai jos atașez screenshot atât cu implementarea, cât și cu output-ul rezultat în urma rularii programului principal.

### Clasa AddTask:



```
1 package com.home.lab9;
2
3 import java.util.concurrent.ExecutorService;
4 import java.util.concurrent.Executors;
5 import java.util.concurrent.Future;
6
7 public class AddTask implements Runnable {
8     ExecutorService exec = Executors.newFixedThreadPool("nThreads: 4");
9     Matrix a, b, c;
10    public AddTask(Matrix a, Matrix b, Matrix c) {
11        this.a = a;
12        this.b = b;
13        this.c = c;
14    }
15    public void run() {
16        try {
17            int n = a.getDim();
18            if (n == 1) {
19                c.set( row: 0, col: 0, value: a.get( row: 0, col: 0) + b.get( row: 0, col: 0));
20            } else {
21                Matrix[][] aa = a.split(), bb = b.split(), cc = c.split();
22                Future<>[][] future = (Future<>[][] new Future[2][2]);
23                for (int i = 0; i < 2; i++) {
24                    for (int j = 0; j < 2; j++) {
25                        future[i][j] = exec.submit(new AddTask(aa[i][j], bb[i][j], cc[i][j]));
26                    }
27                }
28                for (int i = 0; i < 2; i++) {
29                    for (int j = 0; j < 2; j++) {
30                        future[i][j].get();
31                    }
32                }
33            }
34        } catch (Exception ex) {
35            ex.printStackTrace();
36        }
37    }
38
39 }
40
```

**Clasa MultTask:**

```

1 package com.home.lab9;
2
3 import java.util.concurrent.ExecutorService;
4 import java.util.concurrent.Executors;
5 import java.util.concurrent.Future;
6
7 public class MultTask {
8     ExecutorService exec = Executors.newFixedThreadPool( nThreads: 4);
9     Matrix a, b, c;
10
11     public MultTask(Matrix a, Matrix b, Matrix c) {
12         this.a = a;
13         this.b = b;
14         this.c = c;
15     }
16
17     public void run() {
18         try {
19             int n = a.getDim();
20             if (n == 1) {
21                 c.set( row: 0, col: 0, value: a.get( row: 0, col: 0) * b.get( row: 0, col: 0));
22             } else {
23                 Matrix[][] aa = a.split(), bb = b.split(), cc = c.split();
24                 Future<>[][] future = (Future<>[][][]) new Future[2][2];
25                 for (int i = 0; i < 2; i++)
26                     for (int j = 0; j < 2; j++)
27                         future[i][j] = exec.submit(new AddTask(aa[i][j], bb[i][j], cc[i][j]));
28                 for (int i = 0; i < 2; i++) {
29                     for (int j = 0; j < 2; j++) {
30                         future[i][j].get();
31                     }
32                 }
33             }
34         } catch (Exception ex) {
35             ex.printStackTrace();
36         }
37     }
38
39 }
40

```

**Clasa Matrix:**

```

1 package com.home.lab9;
2
3 public class Matrix {
4     int dim;
5     double[][] data;
6     int rowDisplace;
7     int colDisplace;
8     Matrix(int d) {
9         dim = d;
10        rowDisplace = colDisplace = 0;
11        data = new double[d][d];
12    }
13    Matrix(double[][] matrix, int x, int y, int d) {
14        data = matrix;
15        rowDisplace = x;
16        colDisplace = y;
17        dim = d;
18    }
19    double get(int row, int col) {return data[row + rowDisplace][col + colDisplace];}
20    void set(int row, int col, double value) {data[row + rowDisplace][col + colDisplace] = value;}
21    public int getDim() {return dim;}
22    Matrix[][] split() {
23        Matrix[][] result = new Matrix[2][2];
24        int newDim = dim / 2;
25        result[0][0] = new Matrix(data, rowDisplace, colDisplace, newDim);
26        result[0][1] = new Matrix(data, rowDisplace, y: colDisplace + newDim, newDim);
27        result[1][0] = new Matrix(data, x: rowDisplace + newDim, colDisplace, newDim);
28        result[1][1] = new Matrix(data, x: rowDisplace + newDim, y: colDisplace + newDim, newDim);
29        return result;
30    }
31    public void printMatrix() {
32        for (int i = 0; i <= rowDisplace; i++) {
33            for (int j = 0; j <= colDisplace; j++) {
34                System.out.print(this.data[i][j] + " ");
35            }
36            System.out.println();
37        }
38    }
39 }
40

```

Draghici Andreea-Maria  
CR 3.1B

### Clasa Main:

```
1 package com.home.lab9;
2
3 public class Main {
4     public static void main(String[] argv) throws InterruptedException {
5         double[][] a = {
6             {1.0, 1.0},
7             {1.0, 1.0},
8         };
9         double[][] b = {
10            {1.0, 1.0},
11            {1.0, 1.0},
12        };
13        double[][] c = {
14            {0.0, 0.0},
15            {0.0, 0.0},
16        };
17        Matrix firstMatrix = new Matrix(a, x: 1, y: 1, d: 1);
18        Matrix secondMatrix = new Matrix(b, x: 1, y: 1, d: 1);
19        Matrix thirdMatrix = new Matrix(c, x: 1, y: 1, d: 1);
20
21        firstMatrix.printMatrix();
22        System.out.println();
23
24        AddTask addTask = new AddTask(firstMatrix, secondMatrix, thirdMatrix);
25        Thread thread = new Thread(addTask);
26        thread.start();
27        thread.join();
28
29        thirdMatrix.printMatrix();
30        System.out.println();
31
32        MultTask multTask = new MultTask(firstMatrix, secondMatrix, thirdMatrix);
33        Thread thread1 = new Thread(String.valueOf(multTask));
34        thread1.start();
35        thread1.join();
36
37        thirdMatrix.printMatrix();
38    }
39
40 }
```

### Output:

```
Main (1) x
C:\Users\user\.jdk\corretto-16.0.2
1.0 1.0
1.0 1.0

0.0 0.0
0.0 2.0

0.0 0.0
0.0 2.0

Process finished with exit code 0
```

### **Observatii:**

O matrice este reprezentata prin clasa Matrix care contine dimensiunea, deplasamentul pe linii si pe coloane al elementului din stanga sus al matricei, un tablou bidimensional cu elementele matricei, o metoda pentru determinarea dimensiunii, cat si metoda pentru citirea/scierea unui element.

Aceasta matrice este descompusa folosind metoda split() in patru submatrice patrute de dimensiuni egale. Pentru adunarea matricelor se realizeaza adunarea concurrent, pe fiecare bloc, iar pentru realizarea calculelor se foloseste o rezerva de fire.

Dupa modelul clasei AddTask s-a implementat si clasa MultTask, care realizeaza inmultirea celor doua matrice.