Nama : La Ode Muhammad Gazali
NIM : 222212696
Kelas : 2KS2

TUGAS PRA-PERTEMUAN 12 PEMROGRAMAN BERORIENTASI OBJEK

Penugasan

Buatlah program sederhana untuk perkalian matriks tanpa multithreading dan dengan menggunakan multithreading ,kemudian jelaskan perbedaannya

Penyelesaian:

Untuk menghasilkan dan mencetak matriks otomatis secara random, perlu dibuat class matrix generator sebagai berikut:

MatrixGeneratorUtil.java

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to
change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Main.java to edit this
template
 */
package prapertemuan12;

/**
 *
 * @author U53R
 */
import java.util.Random;
public class MatrixGeneratorUtil {
    public static int[][] generateMatrix(int rows, int columns) {

        // output array to store the matrix values
        int[][] result = new int[rows][columns];

        // TO generate a random integer.
        Random random = new Random();

        // adding values at each index.
        for (int i = 0; i < rows; i++) {
            for (int j = 0; j < columns; j++) {
                result[i][j] = random.nextInt(10);
            }
        }

        // returning output.
        return result;
    }

    // to print the matrix
    public static void print(int[][] matrix) {

        System.out.println();

        int rows = matrix.length;
        int columns = matrix[0].length;

        for (int i = 0; i < rows; i++) {
            for (int j = 0; j < columns; j++) {
                System.out.print(matrix[i][j] + " ");
            }
            System.out.println();
        }
    }
}
```

Untuk melakukan operasi perkalian matriks tanpa menggunakan multithreading dan mengukur waktu yang dibutuhkan untuk menyelesaikan operasi, perlu dibuat sebuah class baru sebagai berikut:

MatrixMultiplication.java (Tanpa Threading)

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to
change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this
template
 */
package prapertemuan12;

/**
 *
 * @author U53R
 */
import java.util.Date;

public class MatrixMultiplication {
    public static void main(String[] args) {
        Date start = new Date();

        int[][] m1 = MatrixGeneratorUtil.generateMatrix(100, 100);
        int[][] m2 = MatrixGeneratorUtil.generateMatrix(100, 100);
        int[][] result = multiply(m1, m2);

        System.out.println("Matrix 1 : ");
        MatrixGeneratorUtil.print(m1);

        System.out.println("\nMatrix 2 : ");
        MatrixGeneratorUtil.print(m2);

        System.out.println("\nOutput Matrix : ");
        MatrixGeneratorUtil.print(result);

        Date end = new Date();
        System.out.println("\n Tanpa Multithreading");
        System.out.println("\nTime taken in milli seconds: " + (end.getTime() -
start.getTime()));
    }

    public static int[][] multiply(int[][] matrix1, int[][] matrix2) {
        int resultRows = matrix1.length;
        int resultColumns = matrix2[0].length;

        int[][] result = new int[resultRows][resultColumns];

        int columns2 = matrix2[0].length;

        for (int i = 0; i < resultRows; i++) {
            for (int j = 0; j < columns2; j++) {
                result[i][j] = 0;
                for (int k = 0; k < resultColumns; k++) {
                    result[i][j] += matrix1[i][k] * matrix2[k][j];
                }
            }
        }
        return result;
    }

}
```

Untuk yang dengan threading, perlu dibuat sebuah class worker. Sesuai namanya, kelas ini adalah pekerja thread yang akan digunakan dalam perkalian matriks secara paralel. Implementasi kelas ini lebih awal memastikan bahwa kita memiliki mekanisme untuk melakukan tugas paralel pada tingkat baris, yang diperlukan sebelum kita bisa mengelola banyak thread

RowMultiplyWorker.java (Dengan Threading)

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to
change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this
template
 */
package prapertemuan12;

/**
 *
 * @author U53R
 */
public class RowMultiplyWorker implements Runnable {
    private final int[][] result;
    private int[][] matrix1;
    private int[][] matrix2;
    private final int row;

    public RowMultiplyWorker(int[][] result, int[][] matrix1, int[][] matrix2,
int row) {
        this.result = result;
        this.matrix1 = matrix1;
        this.matrix2 = matrix2;
        this.row = row;
    }

    @Override
    public void run() {

    for (int i = 0; i < matrix2[0].length; i++) {
        result[row][i] = 0;
        for (int j = 0; j < matrix1[row].length; j++) {
            result[row][i] += matrix1[row][j] * matrix2[j][i];
        }
    }
    }
}
```

Untuk mengelola eksekusi multithreading dengan menggunakan RowMultiplyWorker. Sebelum kita dapat mengimplementasikan atau menguji fungsi multithreading, kita memerlukan kelas pekerja (worker) yang akan dijalankan dalam thread. Dengan mengimplementasikan RowMultiplyWorker terlebih dahulu, kita dapat memastikan ParallelThreadsCreator dapat menjalankan tugasnya dengan benar.

ParallelThreadsCreator.java

```java
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to
change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this
template
 */
package prapertemuan12;

/**
 *
 * @author U53R
 */
import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;

public class ParallelThreadsCreator {
    // creating 10 threads and waiting for them to complete then again repeat
steps.
    public static void multiply(int[][] matrix1, int[][] matrix2, int[][] result)
{
        List threads = new ArrayList<>();
        int rows1 = matrix1.length;
        for (int i = 0; i < rows1; i++) {
        RowMultiplyWorker task = new RowMultiplyWorker(result, matrix1, matrix2,
i);
            Thread thread = new Thread(task);
            thread.start();
            threads.add(thread);
            if (threads.size() % 10 == 0) {
                waitForThreads(threads);
            }
        }
    }

    private static void waitForThreads(List threads) {
        for (Iterator it = threads.iterator(); it.hasNext();) {
            Thread thread = (Thread) it.next();
            try {
                thread.join();
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
        threads.clear();
    }
}
```

Selanjutnya untuk melakukan perkalian matriks dengan menggunakan multithreading. Implementasi kelas terakhir memungkinkan kita untuk menggunakan semua komponen yang telah diimplementasikan sebelumnya (MatrixGeneratorUtil, RowMultiplyWorker, dan ParallelThreadsCreator)

MatrixMultiplicationParallel.java

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to
change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this
template
 */
package prapertemuan12;

/**
 *
 * @author U53R
 */
import java.util.Date;

public class MatrixMultiplicationParallel {
    public static void main(String[] args) {

    Date start = new Date();
    int[][] m1 = MatrixGeneratorUtil.generateMatrix(100, 100);
    int[][] m2 = MatrixGeneratorUtil.generateMatrix(100, 100);

    int[][] result = new int[m1.length][m2[0].length];
    ParallelThreadsCreator.multiply(m1, m2, result);

    System.out.println("Matrix 1 : ");
    MatrixGeneratorUtil.print(m1);
```

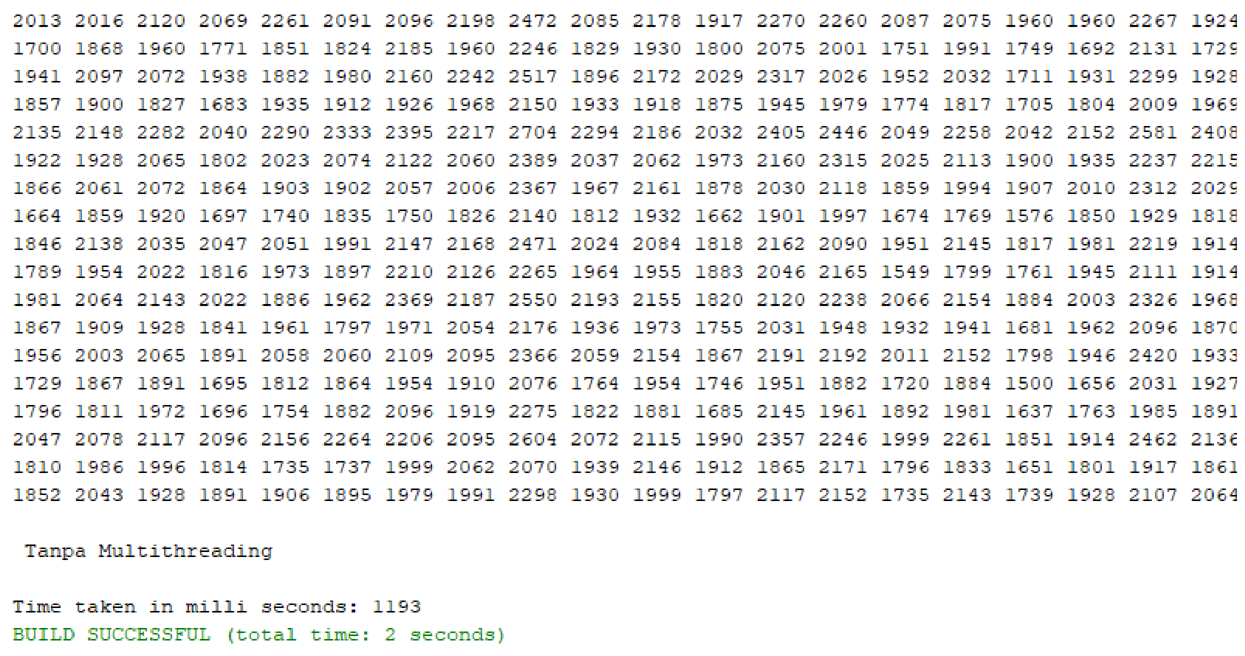
```
        System.out.println("\nMatrix 2 : ");
        MatrixGeneratorUtil.print(m2);

        System.out.println("\nOutput Matrix : ");
        MatrixGeneratorUtil.print(result);

        Date end = new Date();
        System.out.println("\n Dengan Multithreading");
        System.out.println("\nTime taken in milli seconds: " + (end.getTime() -
start.getTime()));

    }
}
```

Contoh output tanpa threding

```
run:
Matrix 1 :

6 4 8 8 4 0 2 5 3 0 2 5 8 3 5 1 8 8 9 8 0 5 1 7 4 3 4 1 8 3 2 3 8 5 5 3 1 4 6 5 7 2 1 0 2 4 1 5 6 3 5 2 3 0 7 9 4 5
2 8 8 9 7 2 3 8 6 3 9 9 0 7 8 1 2 4 8 1 3 8 4 5 6 8 1 3 4 3 6 6 7 4 2 0 1 1 2 5 0 2 4 4 8 4 1 0 4 9 2 8 9 0 9 6 9 1
7 5 0 3 5 1 0 2 4 3 0 9 2 3 5 5 7 5 3 8 3 8 7 8 8 1 5 4 7 7 9 0 1 2 3 5 6 2 6 2 9 3 4 6 5 7 1 4 3 0 6 4 5 8 1 4 6 7
6 5 2 7 7 0 2 9 5 4 6 8 5 5 6 6 1 9 6 7 4 0 3 6 1 8 5 0 5 9 3 2 5 0 3 5 1 1 7 0 8 1 5 1 1 0 7 5 2 1 8 1 5 5 5 1 2 9
9 8 3 5 0 9 1 5 9 8 8 9 9 5 4 5 7 4 6 4 9 1 6 5 8 6 7 8 2 8 7 2 6 9 0 7 2 9 8 8 6 9 5 9 4 1 7 4 3 7 4 5 4 7 4 2 6 0
6 5 4 1 8 3 6 3 8 5 3 8 0 7 7 7 1 4 0 2 9 7 4 1 7 1 5 6 4 4 8 0 8 9 3 9 4 1 7 4 1 4 8 6 7 9 6 0 8 7 8 2 7 7 6 6 8 3
5 5 7 8 0 5 2 5 3 7 1 2 0 0 3 7 9 2 7 1 6 7 0 0 8 9 7 2 4 5 0 2 1 5 2 2 9 5 1 9 3 2 9 4 6 7 8 7 7 1 2 5 8 6 0 3 6 0
7 3 2 9 5 8 9 4 9 5 4 7 4 3 5 4 4 4 4 7 1 8 4 6 6 1 8 4 6 0 3 4 5 0 9 4 8 5 2 1 6 7 5 3 6 8 9 8 0 1 7 3 0 6 7 8 9 6
1 8 0 7 0 1 3 9 8 8 1 9 3 4 4 9 6 4 6 6 6 4 7 9 4 5 1 4 2 3 2 8 9 4 4 2 3 1 4 2 0 7 8 1 6 1 2 0 6 2 9 0 9 3 8 2 4 7
8 7 1 0 7 4 5 8 7 1 9 1 1 2 2 9 1 1 1 2 8 8 9 2 0 7 5 5 8 4 4 9 2 4 5 8 3 5 6 6 6 4 1 6 2 5 6 9 1 3 1 2 2 7 4 2 6 1
3 8 1 5 5 8 5 6 0 4 3 1 2 4 2 9 4 5 2 4 5 9 0 4 2 6 4 4 5 3 4 5 1 5 0 1 8 7 8 6 7 8 8 1 3 8 2 3 7 4 7 3 0 4 7 8 1 6
8 0 5 1 8 4 4 4 1 9 8 8 2 2 9 2 2 6 3 7 5 9 6 3 0 6 1 4 0 1 7 6 0 2 7 5 0 3 6 2 9 0 8 2 1 0 0 6 3 7 4 9 5 1 4 5 2 5
0 7 6 2 2 8 9 2 3 9 7 5 9 7 8 9 0 6 1 4 6 1 9 9 0 2 9 2 4 1 4 7 6 7 3 7 0 8 4 3 2 7 2 0 3 8 8 5 1 0 9 5 0 3 2 9 2 0
5 7 9 7 0 1 9 3 6 9 7 4 7 9 2 8 9 6 9 9 5 3 4 1 6 2 9 3 0 7 6 3 1 6 8 9 5 8 0 7 2 8 8 7 6 4 8 6 2 3 3 8 2 7 4 0 8 2
9 7 5 4 9 9 4 6 2 2 3 7 7 7 1 3 3 2 1 3 7 5 3 6 9 1 0 0 4 6 5 2 4 2 5 7 9 8 9 8 6 1 5 9 7 3 9 4 5 8 4 5 9 5 7 4 7 0
2 7 1 8 5 4 2 2 5 1 0 7 3 6 3 4 9 1 6 9 8 4 8 7 5 9 0 4 2 2 6 4 9 2 5 5 8 7 8 5 0 3 7 4 8 2 3 8 4 9 3 9 9 5 5 7 2 4
2 6 2 2 2 2 6 8 3 0 9 4 2 1 3 1 8 7 3 0 9 3 7 0 0 1 4 6 6 6 0 6 2 4 8 4 6 3 3 4 2 7 6 5 1 7 4 4 4 3 9 6 9 1 4 8 7 4
0 6 7 1 6 0 3 1 1 9 3 5 1 3 2 1 5 2 2 5 8 7 8 5 2 9 4 2 4 2 2 0 0 0 4 7 1 4 8 3 2 0 7 8 4 8 3 5 0 1 3 9 1 4 6 8 3 5
1 1 4 8 2 6 6 8 3 4 4 5 9 3 4 3 6 7 6 6 0 6 3 0 4 3 3 4 4 9 8 2 0 6 5 8 9 1 1 5 5 2 2 9 8 8 5 6 8 6 1 5 5 0 6 6 3 6
1 3 1 6 0 8 2 2 7 2 2 7 7 0 7 9 6 3 9 7 6 3 8 0 1 3 1 7 3 7 3 5 8 5 8 4 8 1 9 8 9 6 6 0 8 4 6 0 4 7 6 3 9 1 5 6 2 0
9 3 5 2 5 7 8 6 8 2 1 6 6 0 2 4 6 4 0 6 9 1 2 5 4 8 3 1 3 9 3 0 6 2 5 3 3 2 8 4 7 5 6 3 4 1 6 3 4 8 7 0 0 7 1 0 6 5
2 6 6 3 5 6 7 8 3 7 4 5 7 2 6 2 4 2 1 0 4 8 2 7 4 0 8 6 9 0 3 5 9 6 5 5 9 6 0 8 2 0 6 4 4 8 3 4 8 3 8 8 0 1 0 6 7 3
9 4 7 6 0 2 5 5 8 9 0 8 7 5 7 1 0 9 3 7 1 3 8 5 7 0 0 9 5 2 5 6 5 3 1 4 1 0 2 9 1 6 0 4 5 9 7 7 2 2 5 8 7 1 5 8 9 4
5 5 4 9 8 3 6 0 3 6 9 0 9 4 8 2 2 0 5 9 1 6 1 4 4 2 2 5 4 9 8 7 5 2 2 3 6 4 5 2 3 9 7 7 6 9 3 2 7 3 7 9 4 5 8 4 2 4
1 3 1 5 0 2 1 3 5 2 8 4 1 1 0 5 2 3 3 0 1 5 6 7 7 0 9 5 6 8 8 9 2 8 9 5 0 4 6 2 3 7 9 3 7 1 3 3 8 3 2 6 0 4 0 2 5 5
2 6 3 9 4 1 6 5 0 4 1 0 0 4 0 4 6 8 5 3 1 3 3 2 0 8 2 6 8 9 9 1 3 6 1 0 6 0 3 5 8 6 6 9 3 4 6 8 7 3 8 4 1 1 8 7 2 8
9 4 7 5 8 0 8 7 4 1 9 6 5 0 7 8 4 1 2 0 6 5 8 8 8 4 6 4 3 1 0 0 1 4 0 7 1 2 6 8 4 5 7 4 9 8 0 8 8 1 3 4 8 2 7 6 1 9
9 2 9 5 1 9 6 0 2 9 7 7 5 1 9 1 3 3 3 4 0 2 1 7 0 4 8 4 5 8 8 5 8 2 0 9 8 1 2 8 6 6 6 2 1 6 8 3 1 4 4 6 8 6 6 4 1 1
3 7 0 3 8 2 5 6 8 3 4 4 1 2 1 7 4 2 7 9 8 3 9 9 9 1 8 6 5 6 3 1 3 8 9 6 3 6 4 3 1 3 7 0 4 7 0 6 5 6 9 9 8 4 0 1 7 2
3 8 4 4 0 6 0 7 0 8 7 4 8 2 6 4 2 0 5 4 1 0 6 9 5 8 5 6 2 9 2 5 8 0 4 7 2 3 5 2 7 1 3 9 2 9 0 8 6 2 4 7 1 2 6 4 1 1
2 9 6 6 1 2 3 6 8 3 3 3 4 9 4 1 5 2 5 0 4 3 7 9 6 6 6 3 8 6 5 0 8 6 8 6 2 0 1 6 3 1 8 1 3 2 5 8 8 2 2 8 6 0 6 6 7 5
2 0 4 3 9 1 0 2 1 1 8 9 8 7 2 5 7 9 0 7 3 2 0 8 3 4 0 5 5 3 5 9 6 8 1 7 9 8 8 4 3 6 1 6 0 7 5 2 7 3 6 3 9 9 0 6 4 1
3 7 0 5 1 4 1 2 3 6 2 9 0 8 2 3 3 4 1 0 9 5 1 1 7 1 0 0 0 8 6 5 1 2 9 7 5 1 5 1 7 5 3 1 3 4 8 9 3 4 0 7 9 0 9 3 4 6
0 6 3 2 2 6 7 0 5 0 1 0 3 1 1 9 9 2 5 8 2 8 4 9 2 5 0 9 0 7 0 8 6 3 7 2 9 1 0 9 8 8 6 4 2 8 4 8 6 0 3 9 9 7 3 9 5 8
2 5 9 2 1 2 1 8 8 6 9 2 0 6 2 7 1 9 6 3 6 0 2 2 4 3 6 0 4 2 1 0 9 4 8 4 1 8 4 7 6 3 3 3 4 7 0 2 8 7 1 0 5 8 0 3 6 0
```

```
Matrix 2 :

0 6 2 5 4 4 3 5 3 1 8 7 2 6 8 4 8 0 2 9 4 5 8 7 4 7 8 0 2 7 8 7 0 4 4 8 7 7 0 5 0 2 6 3 6 8 1 2 0 8 4 3 9 2 3
1 7 4 9 4 4 3 8 7 0 5 6 2 0 1 7 3 3 1 1 2 9 4 4 4 5 1 3 8 1 3 7 8 8 5 9 8 0 3 4 7 9 8 2 8 8 9 2 9 5 9 1 9 6 3
0 4 0 8 8 6 1 2 8 0 7 4 8 1 0 4 5 3 9 1 1 3 4 0 3 9 5 5 1 4 3 9 1 3 9 6 0 7 8 0 8 1 6 4 0 6 6 0 5 2 7 1 3 2 4
2 5 8 9 6 5 8 0 8 9 4 4 7 7 3 7 5 2 1 0 0 5 0 8 0 1 6 6 3 6 1 4 1 8 0 2 9 8 4 5 4 9 2 3 4 3 1 9 4 9 8 0 8 9 4
5 1 7 9 9 7 9 0 5 5 3 1 7 4 5 9 3 7 4 4 6 0 9 5 7 1 4 2 8 9 0 5 7 8 3 3 2 0 4 8 9 6 0 6 4 7 5 6 1 5 0 9 3 6 1
8 7 2 8 0 2 9 6 7 2 3 3 5 9 2 3 7 2 8 9 4 7 2 8 9 6 5 0 8 6 6 6 9 0 2 2 3 2 7 6 5 0 8 9 0 1 9 6 0 7 1 3 3 5 6
2 3 0 3 1 7 2 1 4 3 2 0 1 8 5 4 8 3 4 5 9 5 5 1 0 3 9 1 1 5 4 1 4 8 1 5 7 5 7 1 0 4 1 7 1 2 2 8 8 8 4 0 8 8 1
8 5 1 2 4 4 8 9 9 2 0 7 7 3 6 1 2 0 8 3 5 3 1 0 0 4 4 7 9 4 4 0 3 8 3 8 8 1 5 6 3 9 0 4 0 1 0 8 9 0 6 9 8 4 6
5 6 5 6 8 5 3 1 2 8 6 1 6 6 5 7 1 2 3 3 9 1 9 0 4 7 7 7 9 2 2 8 9 1 4 3 8 2 9 5 4 2 0 7 5 7 7 9 4 9 9 1 6 7 2
2 8 2 7 6 1 9 2 1 1 0 0 1 1 2 2 6 0 1 7 6 2 0 2 2 1 6 0 8 6 4 6 1 2 3 7 9 2 8 1 6 7 7 1 2 8 0 0 2 8 8 1 6 0 6
5 6 5 4 3 3 2 7 2 5 5 3 5 8 3 6 4 3 2 9 8 9 1 8 3 3 6 0 4 8 0 4 8 6 7 8 8 9 7 8 3 8 7 5 8 7 7 2 3 0 3 9 4 1 9
6 5 0 1 7 3 1 7 3 5 9 8 6 0 1 6 7 5 3 1 8 9 4 3 1 8 4 5 3 1 1 9 0 4 9 3 0 3 7 4 2 8 3 1 6 9 6 4 1 4 2 6 6 0 7
2 0 5 4 5 3 2 4 8 6 2 8 6 4 6 4 7 0 3 4 3 9 7 4 0 4 5 0 5 5 0 6 1 6 2 8 2 4 5 7 0 8 0 1 9 6 1 5 2 1 0 8 7 9 5
9 2 8 9 3 1 6 7 8 9 2 2 7 0 2 8 7 2 9 5 3 3 5 9 9 0 6 8 6 0 6 7 1 1 9 7 9 5 7 5 2 9 1 4 8 9 5 7 5 9 1 1 3 1 8
4 1 2 5 9 7 6 0 9 0 4 4 7 8 0 7 8 0 7 8 6 3 4 8 0 9 0 7 5 1 6 6 0 4 6 6 5 0 5 1 8 3 3 8 5 1 3 5 4 0 2 9 7 9 4
5 9 6 0 4 0 6 9 7 1 2 3 9 8 4 5 7 0 4 3 0 4 5 3 9 0 2 6 2 6 7 2 2 1 5 7 3 6 9 6 2 5 5 2 0 6 3 0 1 7 2 0 3 6 1
3 0 2 9 2 3 1 9 3 9 8 8 6 5 7 8 0 6 0 1 9 7 4 6 7 4 9 4 7 4 5 9 0 2 6 3 5 1 3 4 5 1 8 3 0 1 7 6 1 8 3 4 8 7 4
0 2 8 3 6 1 7 5 5 3 7 3 3 7 4 4 9 2 2 9 7 6 2 8 5 6 4 2 3 0 1 4 8 8 5 2 0 5 1 4 1 3 7 1 0 5 4 2 2 8 3 9 0 9 3
9 0 4 1 7 7 0 3 9 6 2 1 8 7 2 5 7 3 9 5 7 7 7 1 2 3 3 5 3 4 5 7 3 8 5 0 4 6 9 3 9 5 5 8 4 3 3 4 8 1 3 5 4 0 3
9 3 8 3 6 8 9 3 1 9 6 5 4 0 3 7 3 1 8 5 0 6 1 6 4 6 5 2 8 2 4 1 7 6 1 8 6 7 3 0 4 8 2 4 0 5 1 7 8 5 0 9 5 2 4
4 1 9 7 8 2 8 5 0 0 5 2 1 5 5 0 9 6 0 2 7 6 5 2 5 5 5 3 7 3 2 4 4 2 5 4 2 9 1 2 8 3 5 5 5 9 8 6 7 1 9 0 3 3 4
9 4 6 0 4 8 2 0 9 3 6 7 4 5 7 9 2 2 7 4 8 1 1 3 6 6 1 3 3 7 0 3 6 7 5 8 4 5 6 3 1 8 5 2 7 6 4 0 7 1 8 7 3 9 1
1 4 7 6 4 5 4 5 6 5 7 4 3 8 6 1 3 6 3 0 4 7 2 6 3 5 7 1 8 2 6 0 0 6 8 0 9 9 5 0 2 7 2 7 4 5 1 1 5 3 6 1 5 2 1
5 4 1 2 1 7 9 7 5 4 0 3 5 0 6 2 0 4 2 7 6 0 3 2 3 8 5 4 2 6 3 0 2 8 2 2 7 4 6 7 0 9 9 9 8 3 5 9 5 8 1 7 0 3 2
8 6 3 0 8 5 2 9 3 6 9 0 5 4 4 3 0 9 5 6 4 6 5 6 4 7 2 4 2 5 0 5 0 9 5 2 6 4 2 6 1 8 1 3 2 2 9 4 1 3 3 4 4 7 0
2 8 6 4 5 8 7 8 9 9 0 3 7 9 2 5 2 8 8 1 4 5 2 5 9 9 5 3 9 9 8 5 8 6 0 5 2 5 4 3 0 2 4 8 9 3 9 8 6 2 2 9 4 4 0
3 4 5 1 6 5 0 1 6 0 7 6 8 8 2 9 3 0 8 2 6 6 7 5 3 7 0 4 6 3 4 2 6 2 1 3 8 3 2 8 5 0 5 1 6 1 7 5 4 3 1 1 5 2 6
9 5 2 6 8 0 2 5 8 3 1 1 5 4 9 5 1 2 7 9 5 0 4 0 4 1 9 1 4 8 7 0 3 5 9 8 8 8 2 8 0 2 6 5 7 4 4 1 4 4 6 2 1 9 9
3 2 6 4 4 1 5 2 7 7 2 3 9 2 1 3 3 8 1 6 6 2 3 3 2 1 5 2 2 2 1 5 6 0 9 5 1 6 9 1 6 2 5 0 7 3 7 2 5 5 6 1 7 5 9
7 6 2 0 0 5 4 4 6 4 6 4 1 0 0 0 0 3 9 8 6 2 6 0 8 5 8 8 0 3 7 2 4 9 6 4 6 8 7 1 6 8 8 6 8 5 9 9 7 4 8 9 5 8 9
8 8 2 7 0 2 7 0 6 9 7 8 4 7 9 6 3 2 3 3 9 8 0 8 6 6 6 5 4 9 3 8 7 9 6 0 1 1 9 1 3 1 7 3 6 7 3 6 6 9 3 5 4 1 8
4 6 6 5 1 7 9 7 2 4 1 6 4 7 1 0 7 9 3 0 7 5 2 8 4 8 0 4 0 8 2 4 6 6 7 0 4 3 9 3 5 6 6 1 9 3 0 4 3 1 5 5 4 9 1
5 5 6 9 0 8 6 8 9 5 1 2 6 0 3 6 2 6 8 6 3 6 1 3 0 1 1 9 8 1 1 4 5 5 5 2 8 5 7 3 5 5 3 3 8 7 0 5 9 2 5 6 5 8 0
6 9 0 7 1 2 4 7 6 2 0 1 3 5 0 7 1 2 8 0 7 2 5 0 4 0 9 4 5 2 2 6 3 6 7 2 8 3 8 0 7 7 9 5 8 0 6 0 4 1 9 4 7 7 6
8 6 2 8 6 1 0 7 7 5 1 9 1 9 3 0 1 8 3 8 4 4 2 5 1 2 2 5 3 4 2 9 8 8 7 3 0 6 4 4 8 8 9 9 7 5 1 6 4 9 0 6 5 0 4
4 3 9 0 7 2 9 4 3 0 6 1 7 0 1 7 2 3 4 6 1 5 0 1 1 8 3 3 6 1 8 2 2 2 1 9 6 7 9 2 3 3 5 8 7 8 3 4 9 0 1 1 4 5 7
5 6 7 7 3 7 9 8 8 5 7 3 7 9 3 9 8 4 6 7 7 8 0 2 5 6 9 7 7 1 2 9 8 3 9 3 7 2 2 5 7 4 6 8 3 1 0 4 1 6 3 5 7 9 4
```

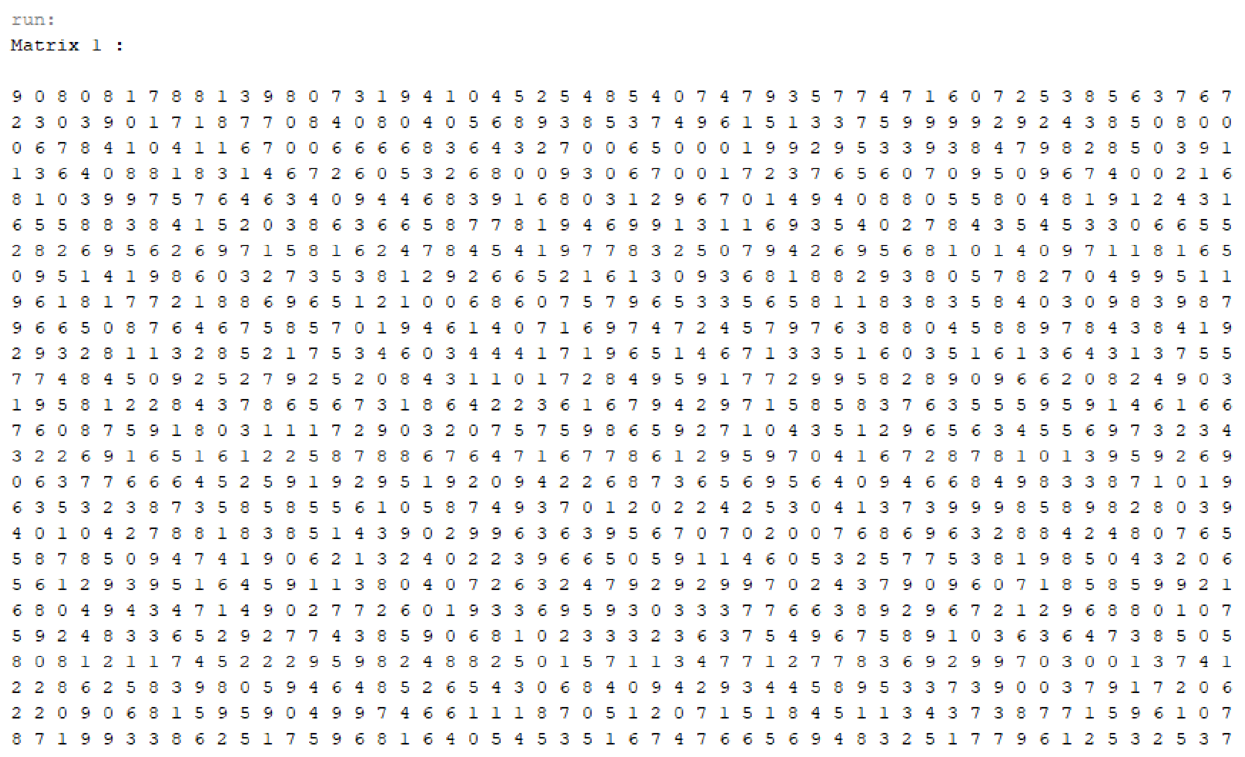
```
Output Matrix :

1796 1896 1907 1722 1978 1953 1945 1922 2321 1951 1878 1888 2162 1824 1738 2086 1592 1838 2018 1890
1872 1940 1930 1830 1903 1977 2052 1979 2393 1995 2048 1956 2137 2056 1851 2038 1807 1965 2230 1841
1899 1997 1987 1704 1960 2019 2158 2025 2365 1908 2200 1873 2105 2024 1880 2153 1671 1863 2220 2017
1581 1651 1846 1658 1820 1687 1885 1807 2076 1848 1625 1525 1765 1724 1379 1722 1497 1547 1882 1679
2187 2359 2480 2210 2291 2187 2534 2416 2632 2364 2598 2206 2487 2458 2257 2387 2088 2126 2585 2352
1867 1908 1974 1911 1951 1961 2110 1920 2518 1821 2199 1903 1986 2143 1776 2152 1816 1851 2127 2069
1539 1818 1950 1728 1849 1823 1917 1839 2063 1790 1837 1657 1817 2009 1619 1778 1668 1725 2006 1771
2034 2104 2201 1949 2129 2129 2133 2173 2492 2200 2221 2038 2363 2268 1977 2223 1862 1882 2379 2152
1927 2005 1924 1869 1785 1944 1991 2217 2293 1961 1934 1991 2084 1934 1705 1822 1658 1891 2030 1876
1730 1953 2206 1837 1991 1910 2156 2095 2174 1872 1965 1803 1977 2195 1769 1948 1673 1814 2157 1985
1916 1951 1969 1828 1967 2121 2214 1910 2369 1898 1858 1782 2039 2186 1796 2121 1786 1817 2220 2020
1819 1809 1813 1736 1947 1785 1996 1760 2218 1783 1913 1737 1882 2006 1837 1787 1699 1750 2059 1831
1647 1971 2093 1921 1862 1767 2051 1938 2189 1829 1774 1691 1978 1854 1539 1795 1643 1692 2058 1795
1893 2077 2245 2026 2079 2095 2156 2225 2330 2108 2193 2022 2163 2320 1826 2242 1971 1863 2378 2096
1858 1860 2261 2062 2059 2170 2327 2145 2360 2090 2233 1998 2166 2297 2025 2168 1909 2057 2413 2229
```



```
2013 2016 2120 2069 2261 2091 2096 2198 2472 2085 2178 1917 2270 2260 2087 2075 1960 1960 2267 1924
1700 1868 1960 1771 1851 1824 2185 1960 2246 1829 1930 1800 2075 2001 1751 1991 1749 1692 2131 1729
1941 2097 2072 1938 1882 1980 2160 2242 2517 1896 2172 2029 2317 2026 1952 2032 1711 1931 2299 1928
1857 1900 1827 1683 1935 1912 1926 1968 2150 1933 1918 1875 1945 1979 1774 1817 1705 1804 2009 1969
2135 2148 2282 2040 2290 2333 2395 2217 2704 2294 2186 2032 2405 2446 2049 2258 2042 2152 2581 2408
1922 1928 2065 1802 2023 2074 2122 2060 2389 2037 2062 1973 2160 2315 2025 2113 1900 1935 2237 2215
1866 2061 2072 1864 1903 1902 2057 2006 2367 1967 2161 1878 2030 2118 1859 1994 1907 2010 2312 2029
1664 1859 1920 1697 1740 1835 1750 1826 2140 1812 1932 1662 1901 1997 1674 1769 1576 1850 1929 1818
1846 2138 2035 2047 2051 1991 2147 2168 2471 2024 2084 1818 2162 2090 1951 2145 1817 1981 2219 1914
1789 1954 2022 1816 1973 1897 2210 2126 2265 1964 1955 1883 2046 2165 1549 1799 1761 1945 2111 1914
1981 2064 2143 2022 1886 1962 2369 2187 2550 2193 2155 1820 2120 2238 2066 2154 1884 2003 2326 1968
1867 1909 1928 1841 1961 1797 1971 2054 2176 1936 1973 1755 2031 1948 1932 1941 1681 1962 2096 1870
1956 2003 2065 1891 2058 2060 2109 2095 2366 2059 2154 1867 2191 2192 2011 2152 1798 1946 2420 1933
1729 1867 1891 1695 1812 1864 1954 1910 2076 1764 1954 1746 1951 1882 1720 1884 1500 1656 2031 1927
1796 1811 1972 1696 1754 1882 2096 1919 2275 1822 1881 1685 2145 1961 1892 1981 1637 1763 1985 1891
2047 2078 2117 2096 2156 2264 2206 2095 2604 2072 2115 1990 2357 2246 1999 2261 1851 1914 2462 2136
1810 1986 1996 1814 1735 1737 1999 2062 2070 1939 2146 1912 1865 2171 1796 1833 1651 1801 1917 1861
1852 2043 1928 1891 1906 1895 1979 1991 2298 1930 1999 1797 2117 2152 1735 2143 1739 1928 2107 2064

 Tanpa Multithreading

Time taken in milli seconds: 1193
BUILD SUCCESSFUL (total time: 2 seconds)
```

Maka Waktu yang diperlukan untuk mengeksekusi matriks 100x100 tanpa threading adalah selama 1193 milisecond

Contoh output dengan Multithreding



```
run:
Matrix 1 :

9 0 8 0 8 1 7 8 8 1 3 9 8 0 7 3 1 9 4 1 0 4 5 2 5 4 8 5 4 0 7 4 7 9 3 5 7 7 4 7 1 6 0 7 2 5 3 8 5 6 3 7 6 7
2 3 0 3 9 0 1 7 1 8 7 7 0 8 4 0 8 0 4 0 5 6 8 9 3 8 5 3 7 4 9 6 1 5 1 3 3 7 5 9 9 9 9 2 9 2 4 3 8 5 0 8 0 0
0 6 7 8 4 1 0 4 1 1 6 7 0 0 6 6 6 6 8 3 6 4 3 2 7 0 0 6 5 0 0 0 1 9 9 2 9 5 3 3 9 3 8 4 7 9 8 2 8 5 0 3 9 1
1 3 6 4 0 8 8 1 8 3 1 4 6 7 2 6 0 5 3 2 6 8 0 0 9 3 0 6 7 0 0 1 7 2 3 7 6 5 6 0 7 0 9 5 0 9 6 7 4 0 0 2 1 6
8 1 0 3 9 9 7 5 7 6 4 6 3 4 0 9 4 4 6 8 3 9 1 6 8 0 3 1 2 9 6 7 0 1 4 9 4 0 8 8 0 5 5 8 0 4 8 1 9 1 2 4 3 1
6 5 5 8 8 3 8 4 1 5 2 0 3 8 6 3 6 6 5 8 7 7 8 1 9 4 6 9 9 1 3 1 1 6 9 3 5 4 0 2 7 8 4 3 5 4 5 3 3 0 6 6 5 5
2 8 2 6 9 5 6 2 6 9 7 1 5 8 1 6 2 4 7 8 4 5 4 1 9 7 7 8 3 2 5 0 7 9 4 2 6 9 5 6 8 1 0 1 4 0 9 7 1 1 8 1 6 5
0 9 5 1 4 1 9 8 6 0 3 2 7 3 5 3 8 1 2 9 2 6 6 5 2 1 6 1 3 0 9 3 6 8 1 8 8 2 9 3 8 0 5 7 8 2 7 0 4 9 9 5 1 1
9 6 1 8 1 7 7 2 1 8 8 6 9 6 5 1 2 1 0 0 6 8 6 0 7 5 7 9 6 5 3 3 5 6 5 8 1 1 8 3 8 3 5 8 4 0 3 0 9 8 3 9 8 7
9 6 6 5 0 8 7 6 4 6 7 5 8 5 7 0 1 9 4 6 1 4 0 7 1 6 9 7 4 7 2 4 5 7 9 7 6 3 8 8 0 4 5 8 8 9 7 8 4 3 8 4 1 9
2 9 3 2 8 1 1 3 2 8 5 2 1 7 5 3 4 6 0 3 4 4 4 1 7 1 9 6 5 1 4 6 7 1 3 3 5 1 6 0 3 5 1 6 1 3 6 4 3 1 3 7 5 5
7 7 4 8 4 5 0 9 2 5 2 7 9 2 5 2 0 8 4 3 1 1 0 1 7 2 8 4 9 5 9 1 7 7 2 9 9 5 8 2 8 9 0 9 6 6 2 0 8 2 4 9 0 3
1 9 5 8 1 2 2 8 4 3 7 8 6 5 6 7 3 1 8 6 4 2 2 3 6 1 6 7 9 4 2 9 7 1 5 8 5 8 3 7 6 3 5 5 5 9 5 9 1 4 6 1 6 6
7 6 0 8 7 5 9 1 8 0 3 1 1 1 7 2 9 0 3 2 0 7 5 7 5 9 8 6 5 9 2 7 1 0 4 3 5 1 2 9 6 5 6 3 4 5 5 6 9 7 3 2 3 4
3 2 2 6 9 1 6 5 1 6 1 2 2 5 8 7 8 8 6 7 6 4 7 1 6 7 7 8 6 1 2 9 5 9 7 0 4 1 6 7 2 8 7 8 1 0 1 3 9 5 9 2 6 9
0 6 3 7 7 6 6 6 4 5 2 5 9 1 9 2 9 5 1 9 2 0 9 4 2 2 6 8 7 3 6 5 6 9 5 6 4 0 9 4 6 6 8 4 9 8 3 3 8 7 1 0 1 9
6 3 5 3 2 3 8 7 3 5 8 5 8 5 5 6 1 0 5 8 7 4 9 3 7 0 1 2 0 2 2 4 2 5 3 0 4 1 3 7 3 9 9 9 8 5 8 9 8 2 8 0 3 9
4 0 1 0 4 2 7 8 8 1 8 3 8 5 1 4 3 9 0 2 9 9 6 3 6 3 9 5 6 7 0 7 0 2 0 0 7 6 8 6 9 6 3 2 8 8 4 2 4 8 0 7 6 5
5 8 7 8 5 0 9 4 7 4 1 9 0 6 2 1 3 2 4 0 2 2 3 9 6 6 5 0 5 9 1 1 4 6 0 5 3 2 5 7 7 5 3 8 1 9 8 5 0 4 3 2 0 6
5 6 1 2 9 3 9 5 1 6 4 5 9 1 1 3 8 0 4 0 7 2 6 3 2 4 7 9 2 9 2 9 9 7 0 2 4 3 7 9 0 9 6 0 7 1 8 5 8 5 9 9 2 1
6 8 0 4 9 4 3 4 7 1 4 9 0 2 7 7 2 6 0 1 9 3 3 6 9 5 9 3 0 3 3 3 7 7 6 6 3 8 9 2 9 6 7 2 1 2 9 6 8 8 0 1 0 7
5 9 2 4 8 3 3 6 5 2 9 2 7 7 4 3 8 5 9 0 6 8 1 0 2 3 3 3 2 3 6 3 7 5 4 9 6 7 5 8 9 1 0 3 6 3 6 4 7 3 8 5 0 5
8 0 8 1 2 1 1 7 4 5 2 2 2 9 5 9 8 2 4 8 8 2 5 0 1 5 7 1 1 3 4 7 7 1 2 7 7 8 3 6 9 2 9 9 7 0 3 0 0 1 3 7 4 1
2 2 8 6 2 5 8 3 9 8 0 5 9 4 6 4 8 5 2 6 5 4 3 0 6 8 4 0 9 4 2 9 3 4 4 5 8 9 5 3 3 7 3 9 0 0 3 7 9 1 7 2 0 6
2 2 0 9 0 6 8 1 5 9 5 9 0 4 9 9 7 4 6 6 1 1 1 8 7 0 5 1 2 0 7 1 5 1 8 4 5 1 1 3 4 3 7 3 8 7 7 1 5 9 6 1 0 7
8 7 1 9 9 3 3 8 6 2 5 1 7 5 9 6 8 1 6 4 0 5 4 5 3 5 1 6 7 4 7 6 6 5 6 9 4 8 3 2 5 1 7 7 9 6 1 2 5 3 2 5 3 7
```

```
 Matrix 2 :

 6 8 2 0 8 6 8 0 7 4 6 4 2 9 2 8 2 9 8 7 1 3 8 6 2 6 8 3 5 4 0 5 9 8 4 0 5 1 6 1 0 5 1 7 4 7 0 3 6 9 8 1 0 3
 2 8 3 0 1 3 0 9 1 5 6 1 9 7 9 4 8 3 6 9 5 6 6 7 3 5 7 9 9 4 6 0 9 6 9 8 4 1 3 3 7 8 6 0 0 8 7 1 3 6 8 5 9 0
 8 7 9 1 0 3 2 3 2 2 6 0 8 0 3 3 3 5 6 5 7 2 8 7 1 0 6 8 6 8 3 1 8 2 8 1 4 6 9 6 1 3 5 0 0 5 1 3 5 1 5 8 5 9
 5 9 2 5 9 4 6 4 9 3 0 4 0 2 7 4 3 4 2 1 7 2 5 0 4 1 2 3 6 5 3 7 4 8 3 6 8 4 2 0 1 5 2 7 0 5 6 4 9 0 4 1 3 8
 6 7 0 5 3 8 9 3 0 2 2 6 7 9 3 7 3 1 3 7 4 2 0 5 1 6 7 0 9 2 2 4 8 4 9 4 2 9 2 5 0 5 4 9 3 3 9 0 0 8 3 9 6 5
 9 9 5 6 9 9 4 9 4 7 6 7 5 7 0 5 5 1 3 9 8 1 1 8 9 0 2 4 1 4 4 2 6 3 4 8 1 2 1 1 7 7 4 6 2 0 4 2 2 1 5 9 3 9
 5 8 6 9 9 5 0 3 5 0 9 9 9 4 1 4 3 0 6 7 9 1 5 9 6 9 0 6 3 9 4 3 2 0 4 9 7 0 5 4 0 6 0 0 2 5 7 7 4 8 4 8 0 3
 9 1 7 0 9 0 5 2 4 6 0 2 8 2 9 6 1 1 1 8 7 6 5 8 9 3 1 7 6 2 0 9 9 5 9 2 7 3 7 7 0 7 1 4 4 2 4 4 5 0 1 8 7 7
 2 5 9 3 9 9 4 4 2 2 5 9 1 0 5 3 1 1 6 6 3 6 3 1 0 5 4 4 5 8 8 8 3 4 2 8 0 5 5 7 4 9 8 0 8 5 0 7 4 9 2 4 6 8
 2 9 0 6 0 5 9 9 3 0 4 0 9 9 2 7 8 4 4 5 3 5 1 9 5 3 9 8 8 1 3 2 1 7 5 5 4 8 8 5 1 7 5 8 9 3 7 7 2 1 7 4 6 8
 2 3 0 3 9 7 8 0 5 4 4 7 1 9 8 0 9 9 0 2 2 9 8 0 5 3 2 7 6 6 8 3 6 6 7 4 4 8 4 0 0 0 6 1 0 5 9 1 0 2 8 8 2 9
 3 0 8 7 5 1 5 1 9 0 1 9 7 8 0 1 7 8 9 7 7 1 6 4 9 9 7 7 7 4 0 8 4 9 9 2 2 1 8 6 7 6 5 5 0 2 5 2 2 6 8 1 0 2
 2 5 4 8 4 9 9 9 5 3 3 8 3 8 6 3 4 8 7 7 6 0 2 2 8 8 0 2 9 8 0 9 3 4 0 2 5 6 1 5 4 5 8 8 1 3 0 2 7 0 0 8 5 1
 4 4 2 5 2 6 7 7 3 9 5 5 1 1 5 3 5 5 4 7 3 8 8 0 8 3 5 2 0 6 3 5 4 6 6 8 9 6 8 2 9 4 7 4 4 7 2 9 0 5 5 1 7 7
 9 3 5 5 4 9 6 0 9 2 0 0 5 4 2 9 2 2 4 9 7 4 2 0 8 8 6 4 1 2 3 6 6 7 8 1 8 3 9 4 5 1 9 4 7 5 2 9 5 0 3 3 6 0
 6 1 9 5 0 3 0 9 5 3 0 9 2 1 1 7 0 7 4 3 7 6 9 5 8 4 8 2 9 9 7 9 5 0 8 3 4 2 9 4 3 7 0 4 3 3 8 8 3 5 9 2 1 3
 2 7 6 9 9 5 9 8 4 3 5 4 4 5 9 6 1 7 6 8 1 7 1 7 7 1 0 8 6 6 2 3 1 8 9 3 0 5 9 8 1 3 2 0 5 7 2 4 0 4 1 3 2 4
 1 0 9 7 2 8 1 4 2 3 2 2 1 6 3 3 9 5 0 8 5 8 8 7 2 6 8 7 4 9 6 9 2 9 4 4 5 0 8 3 5 8 5 8 3 6 7 0 7 7 5 6 0 9
 9 5 5 4 6 7 9 9 8 9 8 6 2 0 3 8 3 0 4 2 6 5 1 0 8 9 7 5 0 0 5 1 0 9 4 2 2 8 5 6 5 8 9 8 2 1 4 5 2 1 9 0 7 1
 6 5 3 0 8 7 0 3 4 3 8 2 1 6 0 4 3 6 7 1 9 8 9 8 2 5 7 6 5 5 0 7 4 5 2 4 8 9 0 9 5 4 4 3 7 5 1 4 9 4 1 5 6 1
 0 7 6 1 1 2 1 8 6 9 5 8 2 6 4 5 2 2 3 6 4 2 0 8 4 2 5 5 8 5 5 9 1 6 5 2 1 0 5 4 4 9 1 2 2 5 9 3 8 8 1 2 6 2
 9 3 1 5 3 5 9 2 6 8 9 1 6 3 6 8 1 1 3 0 0 1 7 9 0 3 2 3 7 3 1 2 9 0 7 9 7 1 6 8 4 3 1 5 7 1 0 8 8 5 0 9 7 6
 8 0 1 6 1 5 5 4 3 1 4 8 4 0 4 4 6 7 4 2 8 0 4 6 2 9 9 5 3 8 4 9 0 1 6 3 2 0 1 9 7 9 9 2 0 5 7 1 7 8 8 6 5 5
 1 4 8 4 5 8 4 2 7 1 8 7 4 6 3 4 7 3 3 1 9 3 0 9 5 6 6 4 4 4 9 0 6 2 1 0 5 9 5 5 6 9 6 5 9 7 1 3 8 5 8 7 4 3

Output Matrix :

2443 2061 2185 2257 2368 2526 2286 1989 2285 2053 2039 2224 2294 2214 2025 2072 2004 2021 1997 2139 2378 2271
2067 1909 1720 2062 1960 2178 2247 1987 2152 1941 1935 2010 1949 2023 2091 1782 1961 1940 1758 1753 2075 2077
2284 1902 2126 2251 2086 2241 2079 2089 1983 1914 1881 2020 1982 2068 1922 1829 1863 1969 2005 2009 2248 2160
1940 1790 1751 1989 1715 2091 1802 1918 1903 1910 1641 1792 1709 1772 1656 1594 1631 1652 1663 1663 1895 1860
2495 2148 2050 2313 2006 2319 2258 2028 2235 1956 2049 2133 2094 2215 1898 1983 1979 1827 1957 1930 2231 2226
2280 1991 1922 2173 2136 2336 1976 2102 2017 1933 2027 1953 2087 2096 2048 1877 1880 2017 1889 2054 2198 2216
2379 2144 1968 2290 2356 2603 2333 2185 2178 2117 2059 2270 2118 2182 2037 1970 1986 1978 2060 2128 2271 2258
2251 2172 1988 2148 2250 2318 2069 1991 2130 2052 1949 2106 2127 2172 2075 1852 1844 1897 2096 1968 2240 2246
2320 2198 1929 2139 2147 2438 2277 2145 2235 2088 1998 2112 2132 2169 2076 2025 2021 2074 1992 2026 2190 2176
2191 2094 2050 2266 2257 2479 2362 2141 2178 1983 2161 1832 2187 2242 1991 2012 2029 2112 1983 2185 2264 2263
2026 1972 1854 2009 1881 2120 1980 1940 1942 1928 1733 1876 1993 2006 1876 1691 1755 1678 1706 1906 1940 2144
2251 2058 1977 2275 2168 2331 2220 2068 2293 2041 1846 1998 2078 2317 2102 1920 1901 2059 2059 2108 2272 2347
2230 1882 1891 2246 2138 2404 2247 2156 2178 2054 1886 2072 2127 2111 1999 1932 1973 1935 2009 1904 2196 2251
2200 2143 1997 2221 2422 2332 2197 1990 2095 2093 2155 2134 2022 2084 2019 2055 1853 1905 1830 1938 2227 2181
2329 2183 1972 2254 2123 2329 2254 2030 2087 2028 2048 1996 2098 2192 1966 2003 1919 2032 1915 2172 2207 2300
2236 2238 1964 2314 2374 2523 2147 2243 2291 1971 1900 2160 2224 2236 2132 2068 1935 2134 2066 2124 2364 2362
2410 2214 2100 2267 2083 2453 2170 2080 2311 2093 2104 2200 2276 2310 2039 2030 1874 2195 1966 2089 2437 2277
2160 1937 2138 2283 2191 2223 2113 2095 2122 2069 2015 2102 2029 2227 2030 1796 1939 1859 1937 1906 2331 2237
2165 2044 1995 2059 2064 2111 2149 1939 2134 1980 1978 1989 2035 2128 1729 1737 1988 1722 1851 1807 2074 1953
2099 2051 1849 2142 2073 2306 2051 2018 1936 1816 1967 2210 2152 2165 1949 1892 1889 1832 1760 1868 2228 2116
1984 1809 1981 1686 1886 2089 1734 1812 1906 1825 1865 1658 1885 1758 1609 1656 1637 1720 1786 1560 1956 1868 1665
2179 2126 1976 2116 2234 2329 2091 1984 1998 1902 1876 1999 2028 2014 1888 1902 1837 1945 1927 1990 2268 2079 2135
2341 2244 1936 2359 2258 2518 2277 2171 2300 2173 2102 2247 2208 2320 2139 2141 1941 2055 1964 1926 2393 2291 2013
2262 2168 1994 2335 2273 2363 2211 2105 2362 2165 1887 2197 2116 2228 2190 1947 1901 2019 2068 2059 2282 2285 1991
2287 2154 1898 2268 2276 2337 2137 1904 2102 1948 1961 2050 1873 2155 1985 1920 1862 2037 1957 1866 2128 2169 1991
2160 2150 1893 2165 2237 2058 2194 1917 2127 1914 1858 2087 1981 2051 2063 1860 1840 1859 1888 2063 2319 2170 2080
1899 1854 1677 1836 1906 2141 1902 1779 1842 1825 1731 1881 1824 2026 2088 1783 1640 1825 1702 1717 2078 2077 1700
1979 1990 2010 1934 2126 2178 2189 2082 2053 2064 1941 1950 1938 1958 2016 1808 1816 1798 1814 2066 1980 2244 1928
2201 1880 1689 2160 1976 2078 2101 1865 1943 1860 1903 1888 1927 1973 1721 1867 1736 1726 1720 1800 2066 1918 1804
2510 2292 2204 2324 2340 2540 2425 2233 2262 2227 2211 2246 2340 2248 2228 2096 2171 1991 2089 2135 2380 2505 2303

 Dengan Multithreading

Time taken in milli seconds: 220
BUILD SUCCESSFUL (total time: 0 seconds)
```

Maka Waktu yang diperlukan untuk mengeksekusi matriks 100x100 dengan Multtithreading adalah hanya selama 220 milisecond.

Sehingga dapat disimpukan bahwa terdapat perbedaan lama waktu eksekusi antara perkalian matriks tanpa multithreading dan dengan multithreading. Penggunaan thread dalam perkalian matriks memungkinkan pemrosesan paralel yang dapat meningkatkan kecepatan eksekusi dan memanfaatkan sumber daya komputasi yang ada lebih efisien. Sedangkan tanpa thread dilakukan perkalian secara serial. Namun, implementasi dengan thread juga melibatkan overhead tambahan, seperti sinkronisasi data antar thread, yang perlu diperhatikan untuk mendapatkan manfaat yang maksimal