|  |  |
| --- | --- |
| 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**

/\*

 \* 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()));

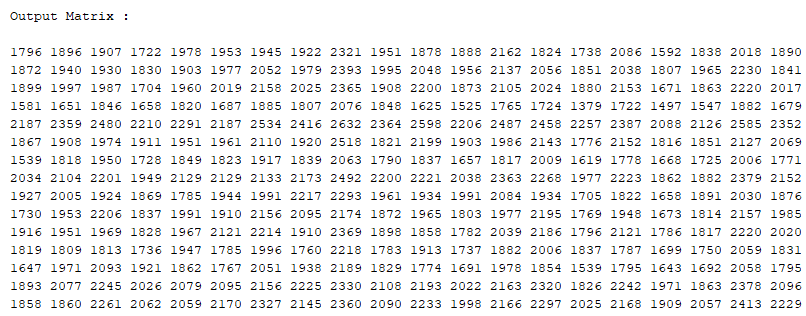
    }

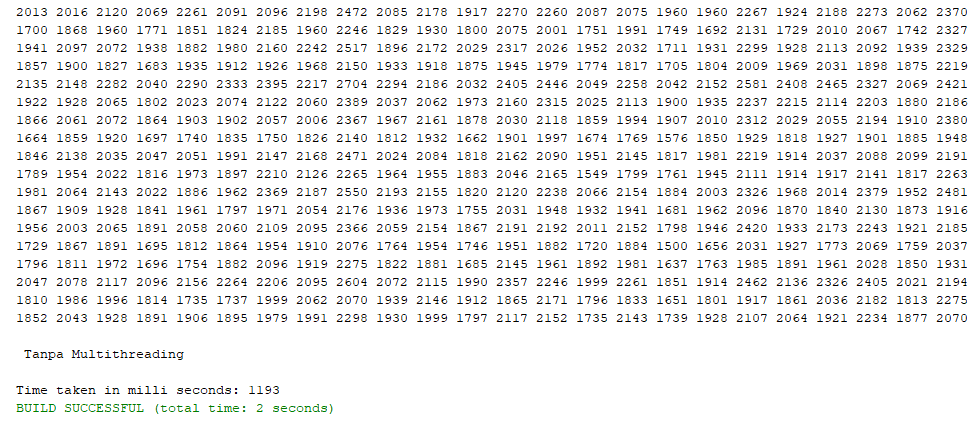
}

**Contoh output tanpa threding**

****

****

****

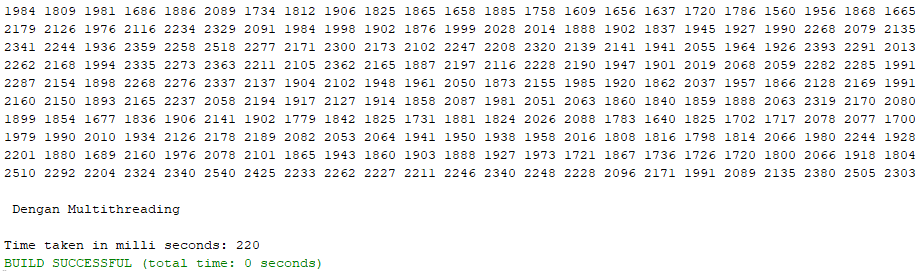
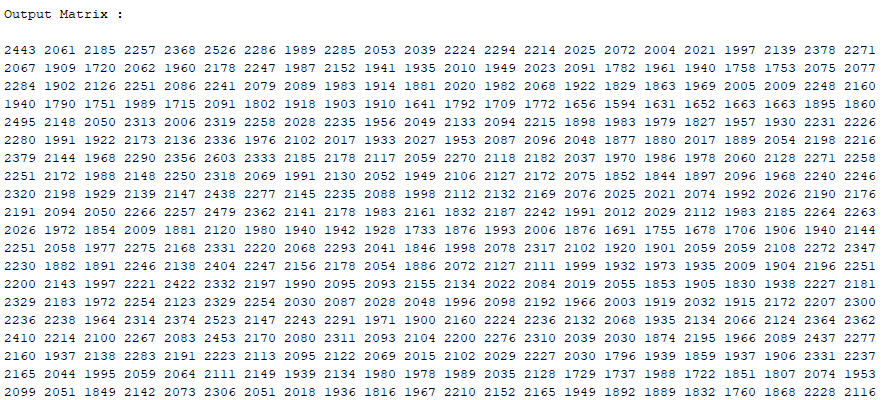
****

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

**Contoh output dengan Multithreding**

****

****

****

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