C:\Users\Angel.Sahagun\Documents\NetBeansProjects\ADA\src\ada\session1\Homework1.java

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package ada.session1;

import com.utils.Utils;
import com.utils.sort.Selection;

/**
 *
 * @author Angel.Sahagun
 */
public class Homework1 {

    public static boolean isStringEquals(String objToCompare1, String objToCompare2) { // Temp = 3N + 5 , Espacial = 2N + 1
        char[] objToCompareChar1 = objToCompare1.toCharArray(); // 1
        char[] objToCompareChar2 = objToCompare2.toCharArray(); // 1
        for (int i = 0; i < objToCompareChar1.length; i++) { // 1 + (N+1) + N
            if (objToCompareChar1[i] != objToCompareChar2[i]) {// N(3)
                return false;
            }
        }
        return true;
    }

    public static int medianCalculation(int[] array) {
        if (!Utils.isSorted(array)) { // 5N + 3
            Selection.sort(array); //
            int medianIndex = (array.length) / 2;
            return array[medianIndex];
        }
        return 0;
    }

    public static void main(String[] args) {
        if (isStringEquals("Angel", "Angel")) {
            System.out.println("Strings are identical");
        } else {
            System.out.println("Strings are no equals ");
        }
        int[] array = Utils.createArray(15, 3, 30); ///{9,5,4,8,3,1,6,9,7,5,2};
        Utils.printArray(array);
        System.out.println("Median = " + medianCalculation(array));
        Utils.printArray(array);
        int[][] matrixA = new int[3][3];
        int[][] matrixB = new int[1][3];
        // filling MatrixA
        matrixA[0][0] = 1;
        matrixA[1][0] = 2;
        matrixA[2][0] = 3;
        matrixA[0][1] = 4;
        matrixA[1][1] = 5;
        matrixA[2][1] = 6;
        matrixA[0][2] = 7;
        matrixA[1][2] = 8;
        matrixA[2][2] = 9;
```

```
        // Filling MatrixB
        matrixB[0][0] = 1;
        matrixB[0][1] = 3;
        matrixB[0][2] = 9;

        // multiplicating
        int[][] matrixReutlt = matrixMultiplication(matrixA, matrixB);

        for (int i = 0; i < matrixReutlt.length; i++) {
            for (int j = 0; j < matrixReutlt[0].length; j++) {
                System.out.println("Matrix [" + i + "][" + j + "] =" + matrixReutlt[i][j]);
            }
        }
        System.out.println(" Count of prime numbers between (" + 1 + "," + 1000 + "): " + countPrimeNumbersBetween(1, 1000));

        System.out.println(" Count 3: " + countNumberDivided(243, 3));

        System.out.println(" Greatest Common Divisor: " + greatestCommonDivisor(5, 8));

    }

    public static int[][] matrixMultiplication(int[][] matrixA, int[][] matrixB) { // 6N^3 + 6N + 6

        int[][] matrixReutlt = new int[matrixA.length][matrixA[0].length];
        for (int i = 0; i < matrixA.length; i++) // 2N + 2
        {
            for (int j = 0; j < matrixB.length; j++) // 2N  + 2
            {
                for (int k = 0; k < matrixA.length; k++) // 2N + 2
                {
                    matrixReutlt[i][k] = matrixA[i][k] * matrixB[j][k] + matrixReutlt[i][k]; // 6N^3

                }
            }
        }
        return matrixReutlt;
    }

    public static int countPrimeNumbersBetween(int start, int end) { // N^2 + 5N + 7
        int counter = 0; // 1
        for (int i = start; i < end; i++) { // 1 + (N + 1) + N
            if (i != 1 && isPrime(i)) { // N^2 + 3N + 4
                counter++; //
            }
        }
        return counter;
    }

    public static boolean isPrime(int i) { // N^2 + 2N + 4
        boolean isPrime = true; // 1
        for (int j = 2; j < i; j++) { // 1 + N + 1 + N
            if (i % j == 0) { // N^2
                isPrime = false; // 1
                break;
            }
        }
        return isPrime;
    }
```

```
    public static int countNumberDivided(int num, int divNum) { // Temporal: 4N + 5, Espacial: 4
        int counter = 0; // 1
        if (num % divNum == 0) { // 1
            int aux = num / divNum;  // 1
            counter++; // 1
            while (aux > 1) { // N
                if (aux % divNum == 0) { // N
                    aux = aux / divNum; //N
                    counter++; // N
                }else{
                    counter = 0; // 1
                    break;
                }
            }
        }
        return counter;
    }

    /**
     * Algoritmo de Euclides
     *
     * @param a
     * @param b
     * @return
     */
    public static int greatestCommonDivisor(int a, int b) {
        int r = 0, div = 0;
        r = a % b;
        div++;
        while (r != 0) {
            a = b;
            b = r;
            r = a % b;
            div++;
        }
        System.out.println("el maximo comun divisor es:" + b);
        System.out.println("el numero de divisiones fue:" + div);
        return div;

    }

}
```