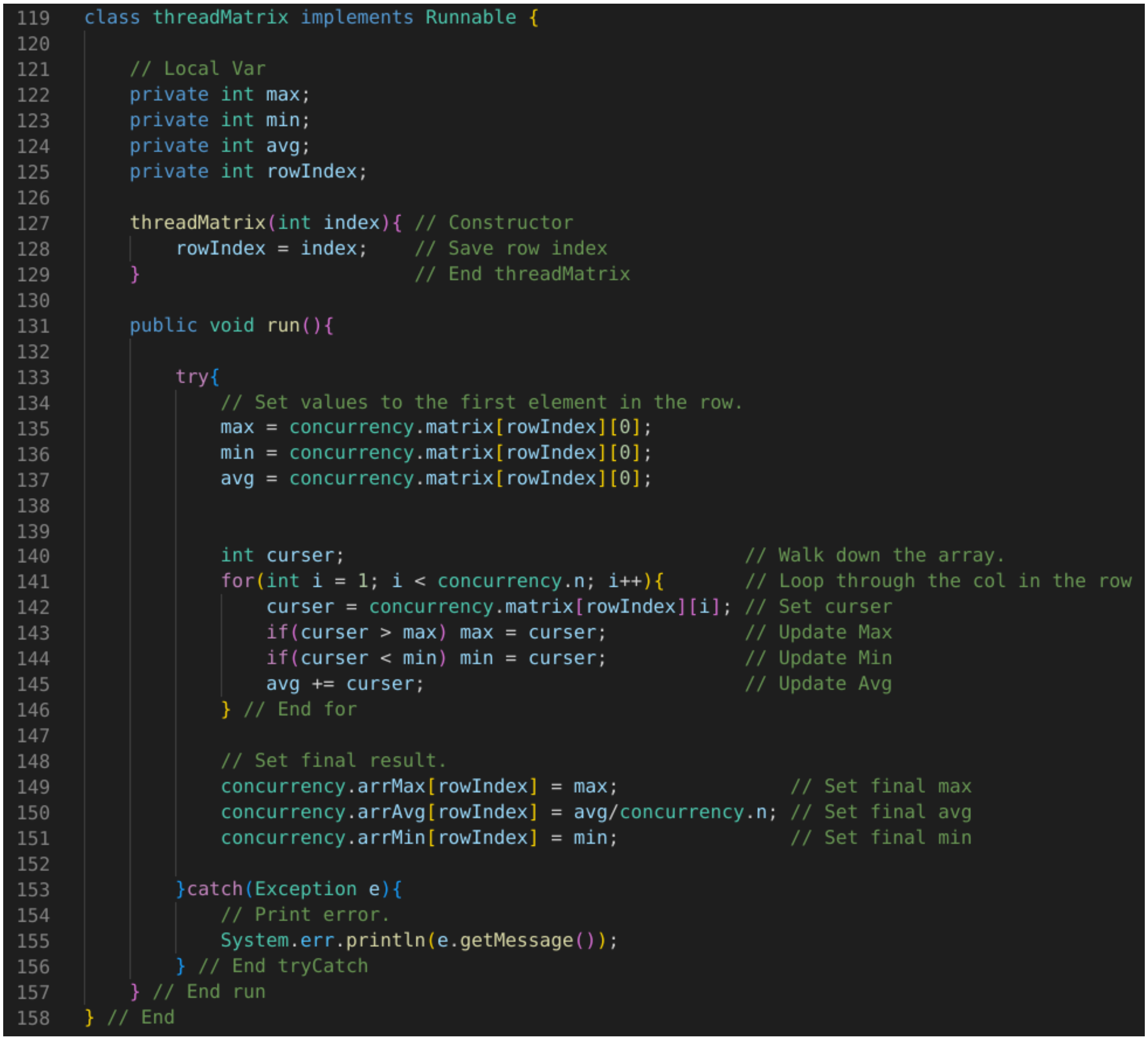
Program #8: Concurrency

Keller Sedillo-Garrido

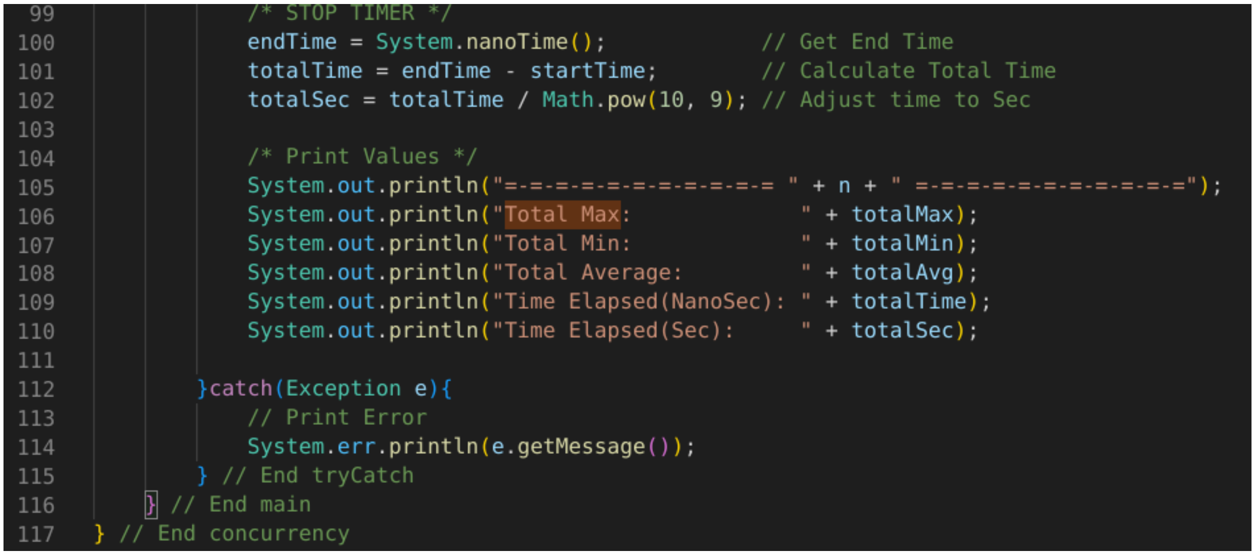
Problem Description:

For this program, we are tasked to create a program that will take one input from the user that will set the size of the array, known as N. The program will then create, randomized, and find the max, the min, and the average of the NxN matrix. We will be timing the program and find the average and the standard deviation of the time.

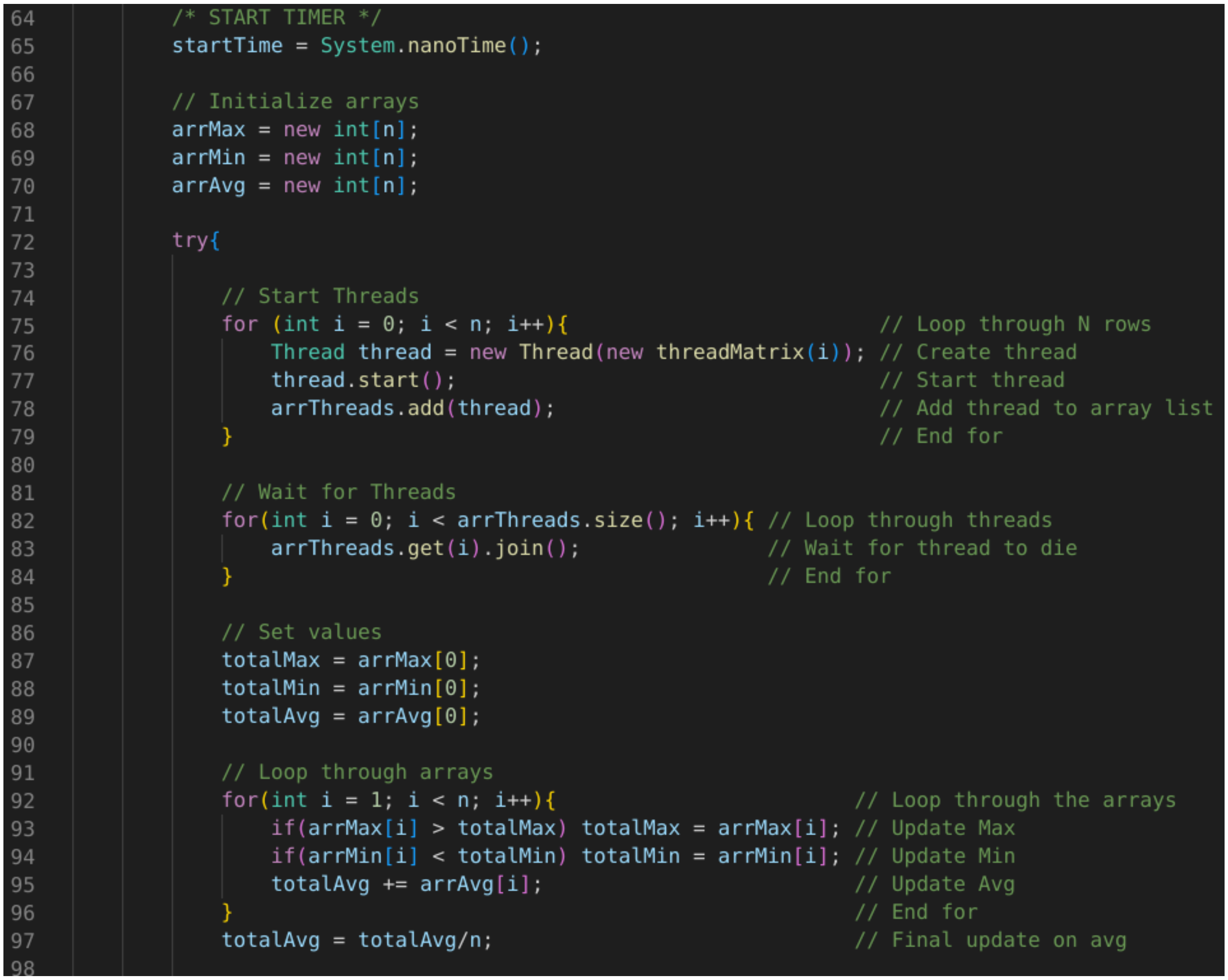
Code:



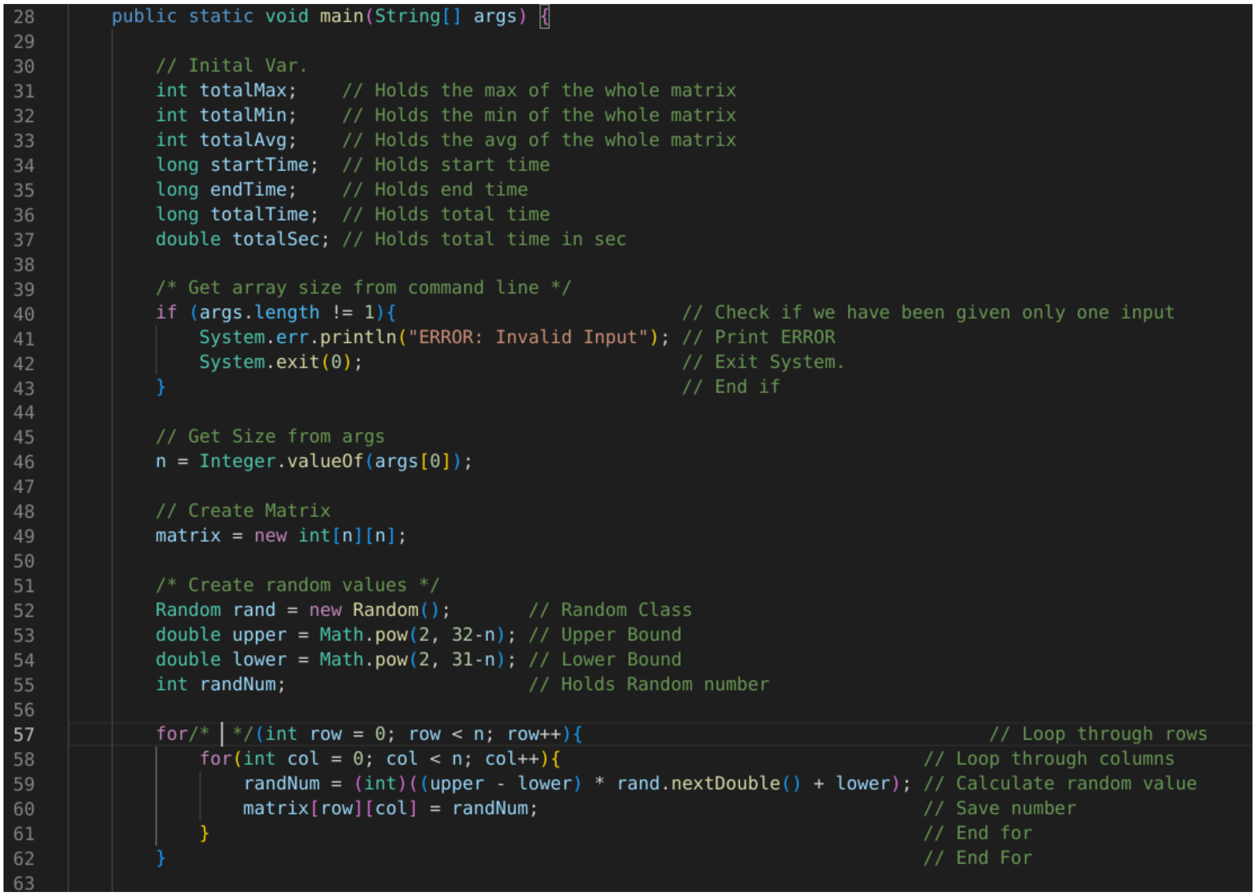
```java
class threadMatrix implements Runnable {

    // Local Var
    private int max;
    private int min;
    private int avg;
    private int rowIndex;

    threadMatrix(int index){ // Constructor
        rowIndex = index;    // Save row index
    }                        // End threadMatrix

    public void run(){

        try{
            // Set values to the first element in the row.
            max = concurrency.matrix[rowIndex][0];
            min = concurrency.matrix[rowIndex][0];
            avg = concurrency.matrix[rowIndex][0];


            int curser;                                   // Walk down the array.
            for(int i = 1; i < concurrency.n; i++){       // Loop through the col in the row
                curser = concurrency.matrix[rowIndex][i]; // Set curser
                if(curser > max) max = curser;            // Update Max
                if(curser < min) min = curser;            // Update Min
                avg += curser;                            // Update Avg
            } // End for

            // Set final result.
            concurrency.arrMax[rowIndex] = max;               // Set final max
            concurrency.arrAvg[rowIndex] = avg/concurrency.n; // Set final avg
            concurrency.arrMin[rowIndex] = min;               // Set final min

        }catch(Exception e){
            // Print error.
            System.err.println(e.getMessage());
        } // End tryCatch
    } // End run
} // End
```



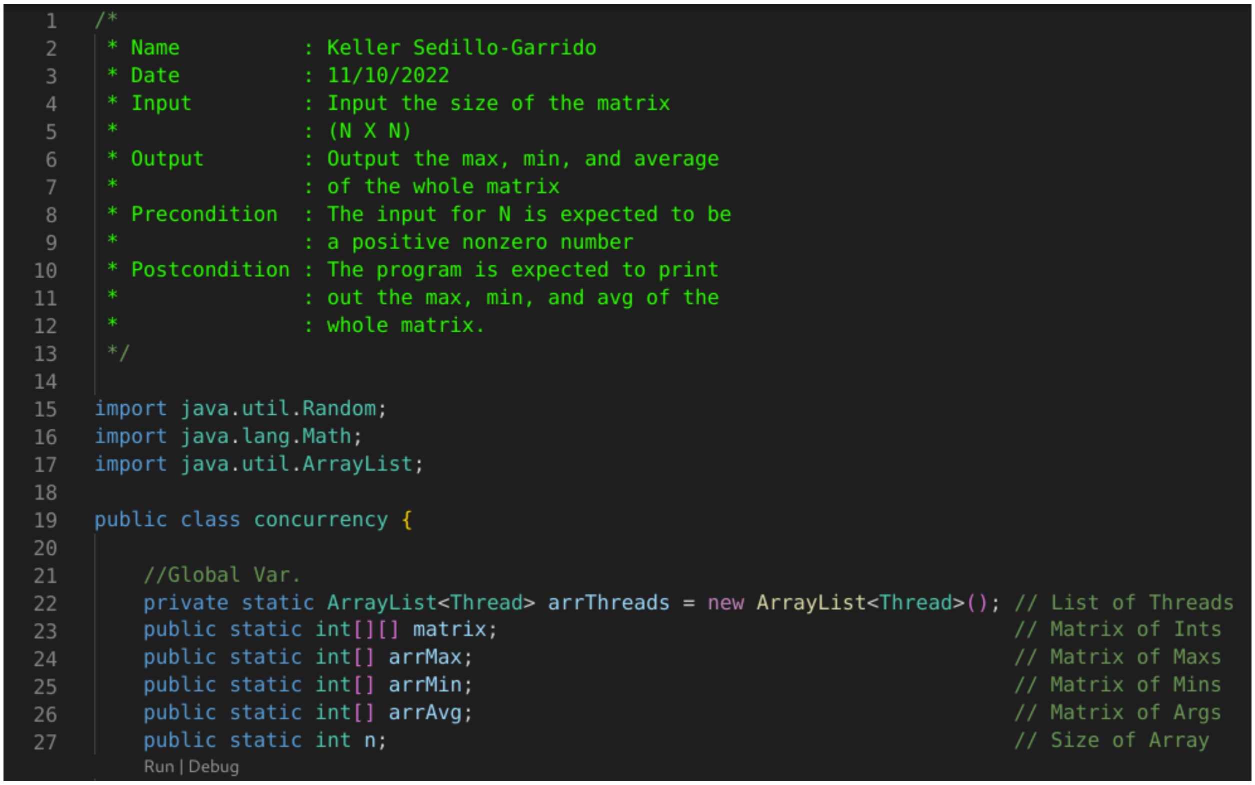
```java
            /* STOP TIMER */
            endTime = System.nanoTime();            // Get End Time
            totalTime = endTime - startTime;        // Calculate Total Time
            totalSec = totalTime / Math.pow(10, 9); // Adjust time to Sec

            /* Print Values */
            System.out.println("=-=-=-=-=-=-=-=-=-=-= " + n + " =-=-=-=-=-=-=-=-=-=-=");
            System.out.println("Total Max:             " + totalMax);
            System.out.println("Total Min:             " + totalMin);
            System.out.println("Total Average:         " + totalAvg);
            System.out.println("Time Elapsed(NanoSec): " + totalTime);
            System.out.println("Time Elapsed(Sec):     " + totalSec);

        }catch(Exception e){
            // Print Error
            System.err.println(e.getMessage());
        } // End tryCatch
    } // End main
} // End concurrency
```



```java
            /* START TIMER */
            startTime = System.nanoTime();

            // Initialize arrays
            arrMax = new int[n];
            arrMin = new int[n];
            arrAvg = new int[n];

            try{

                // Start Threads
                for (int i = 0; i < n; i++){                         // Loop through N rows
                    Thread thread = new Thread(new threadMatrix(i)); // Create thread
                    thread.start();                                  // Start thread
                    arrThreads.add(thread);                          // Add thread to array list
                }                                                    // End for

                // Wait for Threads
                for(int i = 0; i < arrThreads.size(); i++){ // Loop through threads
                    arrThreads.get(i).join();               // Wait for thread to die
                }                                           // End for

                // Set values
                totalMax = arrMax[0];
                totalMin = arrMin[0];
                totalAvg = arrAvg[0];

                // Loop through arrays
                for(int i = 1; i < n; i++){                        // Loop through the arrays
                    if(arrMax[i] > totalMax) totalMax = arrMax[i]; // Update Max
                    if(arrMin[i] < totalMin) totalMin = arrMin[i]; // Update Min
                    totalAvg += arrAvg[i];                         // Update Avg
                }                                                  // End for
                totalAvg = totalAvg/n;                             // Final update on avg
```

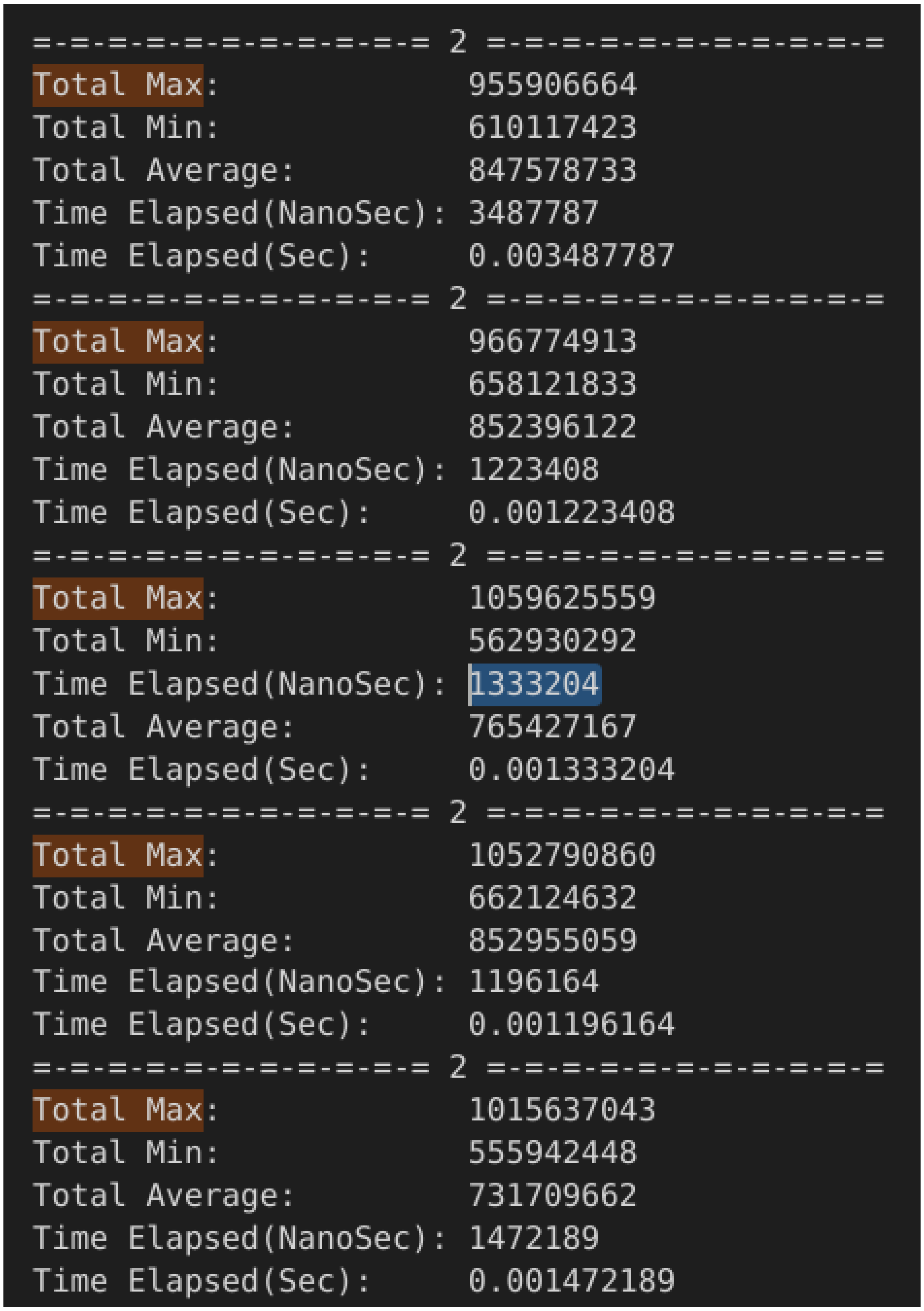


```java
    public static void main(String[] args) {

        // Inital Var.
        int totalMax;    // Holds the max of the whole matrix
        int totalMin;    // Holds the min of the whole matrix
        int totalAvg;    // Holds the avg of the whole matrix
        long startTime;  // Holds start time
        long endTime;    // Holds end time
        long totalTime;  // Holds total time
        double totalSec; // Holds total time in sec

        /* Get array size from command line */
        if (args.length != 1){                          // Check if we have been given only one input
            System.err.println("ERROR: Invalid Input"); // Print ERROR
            System.exit(0);                             // Exit System.
        }                                               // End if

        // Get Size from args
        n = Integer.valueOf(args[0]);

        // Create Matrix
        matrix = new int[n][n];

        /* Create random values */
        Random rand = new Random();       // Random Class
        double upper = Math.pow(2, 32-n); // Upper Bound
        double lower = Math.pow(2, 31-n); // Lower Bound
        int randNum;                      // Holds Random number

        for/* */(int row = 0; row < n; row++){                                      // Loop through rows
            for(int col = 0; col < n; col++){                                 // Loop through columns
                randNum = (int)((upper - lower) * rand.nextDouble() + lower); // Calculate random value
                matrix[row][col] = randNum;                                   // Save number
            }                                                                 // End for
        }                                                                     // End For

```



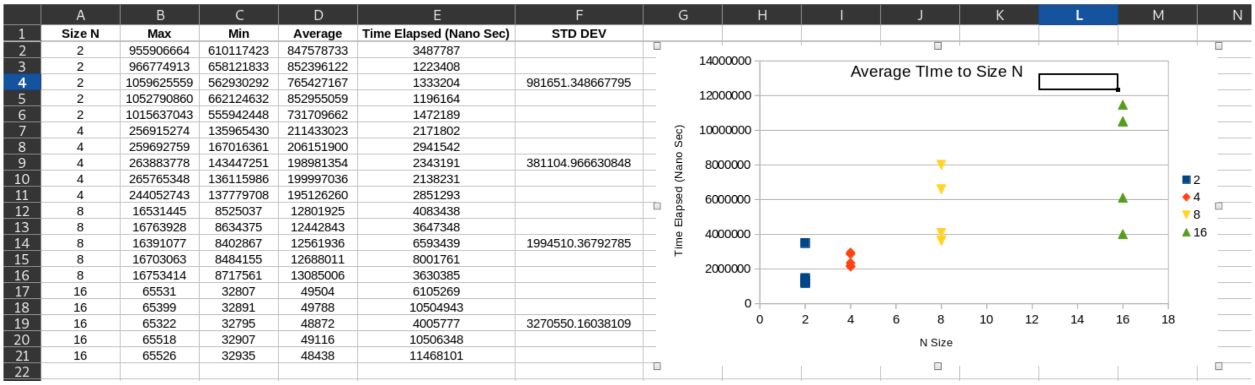
```java
/*
 * Name           : Keller Sedillo-Garrido
 * Date           : 11/10/2022
 * Input          : Input the size of the matrix
 *                : (N X N)
 * Output         : Output the max, min, and average
 *                : of the whole matrix
 * Precondition   : The input for N is expected to be
 *                : a positive nonzero number
 * Postcondition : The program is expected to print
 *                : out the max, min, and avg of the
 *                : whole matrix.
 */

import java.util.Random;
import java.lang.Math;
import java.util.ArrayList;

public class concurrency {

    //Global Var.
    private static ArrayList<Thread> arrThreads = new ArrayList<Thread>(); // List of Threads
    public static int[][] matrix;                                          // Matrix of Ints
    public static int[] arrMax;                                            // Matrix of Maxs
    public static int[] arrMin;                                            // Matrix of Mins
    public static int[] arrAvg;                                            // Matrix of Args
    public static int n;                                                   // Size of Array
```

Output:



```
=-=-=-=-=-=-=-=-=-=-= 2 =-=-=-=-=-=-=-=-=-=-=
Total Max:             955906664
Total Min:             610117423
Total Average:         847578733
Time Elapsed(NanoSec): 3487787
Time Elapsed(Sec):     0.003487787
=-=-=-=-=-=-=-=-=-=-= 2 =-=-=-=-=-=-=-=-=-=-=
Total Max:             966774913
Total Min:             658121833
Total Average:         852396122
Time Elapsed(NanoSec): 1223408
Time Elapsed(Sec):     0.001223408
=-=-=-=-=-=-=-=-=-=-= 2 =-=-=-=-=-=-=-=-=-=-=
Total Max:             1059625559
Total Min:             562930292
Time Elapsed(NanoSec): 1333204
Total Average:         765427167
Time Elapsed(Sec):     0.001333204
=-=-=-=-=-=-=-=-=-=-= 2 =-=-=-=-=-=-=-=-=-=-=
Total Max:             1052790860
Total Min:             662124632
Total Average:         852955059
Time Elapsed(NanoSec): 1196164
Time Elapsed(Sec):     0.001196164
=-=-=-=-=-=-=-=-=-=-= 2 =-=-=-=-=-=-=-=-=-=-=
Total Max:             1015637043
Total Min:             555942448
Total Average:         731709662
Time Elapsed(NanoSec): 1472189
Time Elapsed(Sec):     0.001472189
```

Graph:



	A	B	C	D	E	F
1	Size N	Max	Min	Average	Time Elapsed (Nano Sec)	STD DEV
2	2	955906664	610117423	847578733	3487787	
3	2	966774913	658121833	852396122	1223408	
4	2	1059625559	562930292	765427167	1333204	981651.348667795
5	2	1052790860	662124632	852955059	1196164	
6	2	1015637043	555942448	731709662	1472189	
7	4	256915274	135965430	211433023	2171802	
8	4	259692759	167016361	206151900	2941542	
9	4	263883778	143447251	198981354	2343191	381104.966630848
10	4	265765348	136115986	199997036	2138231	
11	4	244052743	137779708	195126260	2851293	
12	8	16531445	8525037	12801925	4083438	
13	8	16763928	8634375	12442843	3647348	
14	8	16391077	8402867	12561936	6593439	1994510.36792785
15	8	16703063	8484155	12688011	8001761	
16	8	16753414	8717561	13085006	3630385	
17	16	65531	32807	49504	6105269	
18	16	65399	32891	49788	10504943	
19	16	65322	32795	48872	4005777	3270550.16038109
20	16	65518	32907	49116	10506348	
21	16	65526	32935	48438	11468101	
22						

Conclusion:

We see that our time was more varied as N increased. Interestingly enough, the 4x4 beat 2x2 in average time. I’m not sure why that is the case but maybe there is a “sweat spot” where running the number of N threads can work efficiently. Personally, I want to see if even implementation of the program effects the time of the program.