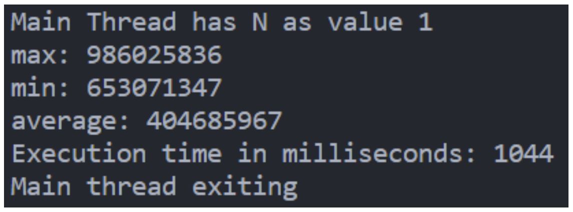
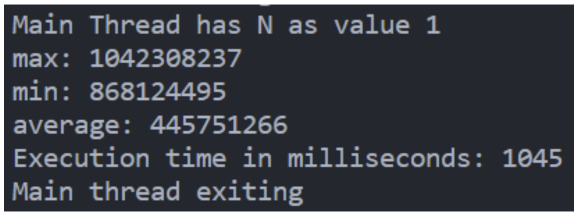
Joey Troyer
11/10/22

**Problem description:**
The goal of this program was to preform multithreaded operations on a matrix. We needed to create a matrix of size N and fill it with random values. Then each thread will preform calculation on one row of the matrix. We then use the calculations from each row to find the min, max and average of the entire matrix.

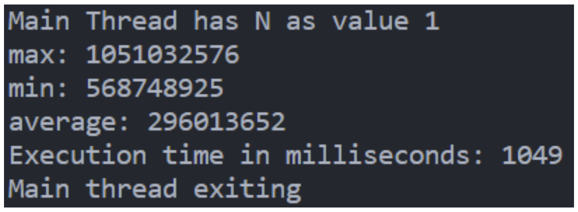
**Output for size 2:**



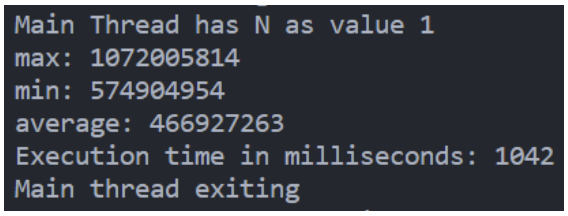
```
Main Thread has N as value 1
max: 986025836
min: 653071347
average: 404685967
Execution time in milliseconds: 1044
Main thread exiting
```



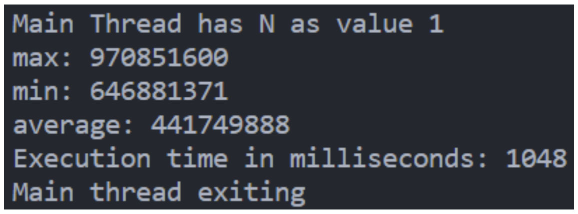
```
Main Thread has N as value 1
max: 1042308237
min: 868124495
average: 445751266
Execution time in milliseconds: 1045
Main thread exiting
```



```
Main Thread has N as value 1
max: 1051032576
min: 568748925
average: 296013652
Execution time in milliseconds: 1049
Main thread exiting
```

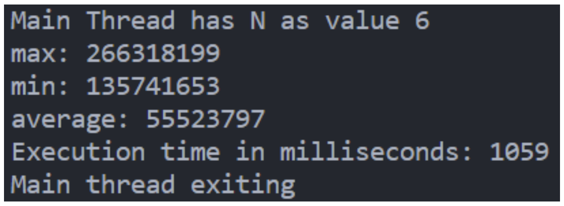


```
Main Thread has N as value 1
max: 1072005814
min: 574904954
average: 466927263
Execution time in milliseconds: 1042
Main thread exiting
```

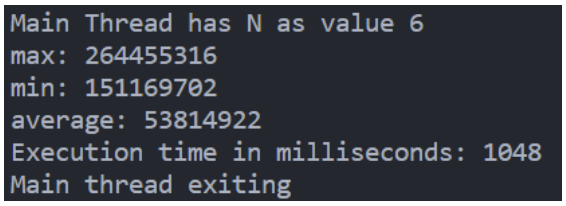


```
Main Thread has N as value 1
max: 970851600
min: 646881371
average: 441749888
Execution time in milliseconds: 1048
Main thread exiting
```

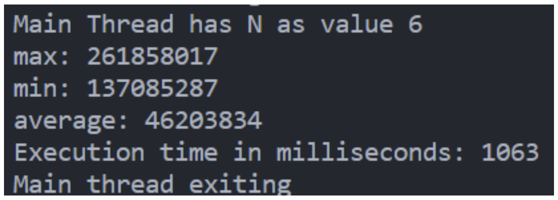
**Output for size 4:**



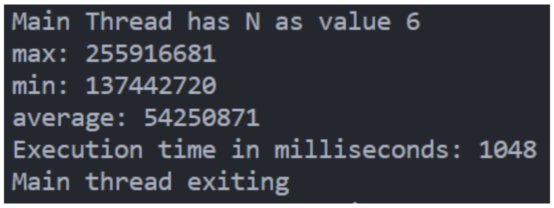
```
Main Thread has N as value 6
max: 266318199
min: 135741653
average: 55523797
Execution time in milliseconds: 1059
Main thread exiting
```



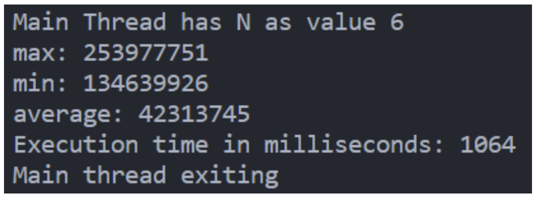
```
Main Thread has N as value 6
max: 264455316
min: 151169702
average: 53814922
Execution time in milliseconds: 1048
Main thread exiting
```



```
Main Thread has N as value 6
max: 261858017
min: 137085287
average: 46203834
Execution time in milliseconds: 1063
Main thread exiting
```



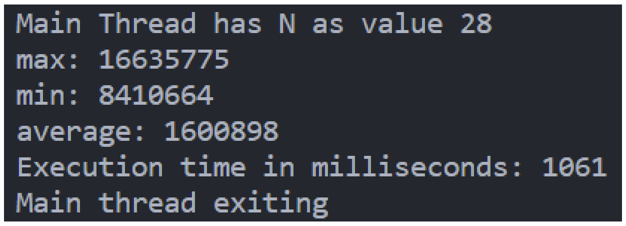
```
Main Thread has N as value 6
max: 255916681
min: 137442720
average: 54250871
Execution time in milliseconds: 1048
Main thread exiting
```



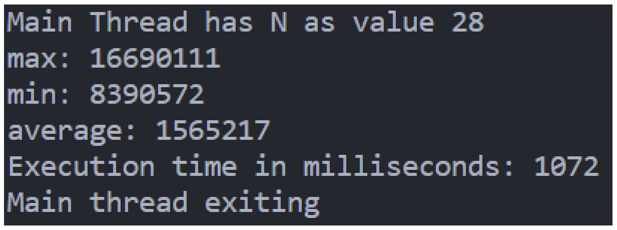
```
Main Thread has N as value 6
max: 253977751
min: 134639926
average: 42313745
Execution time in milliseconds: 1064
Main thread exiting
```

Joey Troyer
11/10/22

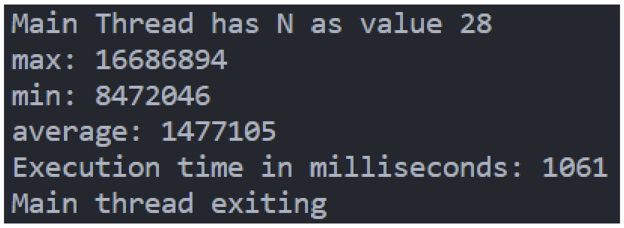
**Output for size 8:**



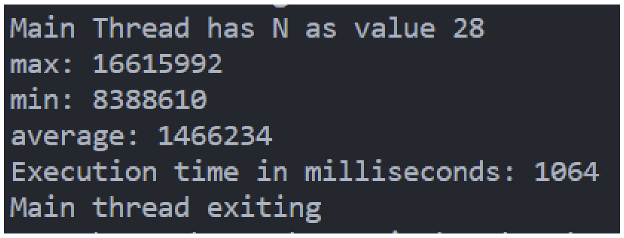
```
Main Thread has N as value 28
max: 16635775
min: 8410664
average: 1600898
Execution time in milliseconds: 1061
Main thread exiting
```



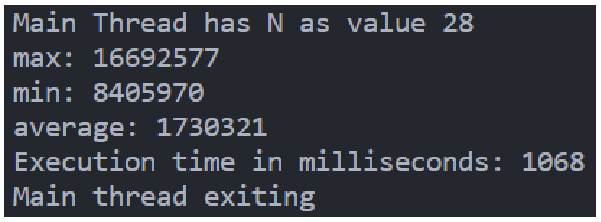
```
Main Thread has N as value 28
max: 16690111
min: 8390572
average: 1565217
Execution time in milliseconds: 1072
Main thread exiting
```



```
Main Thread has N as value 28
max: 16686894
min: 8472046
average: 1477105
Execution time in milliseconds: 1061
Main thread exiting
```

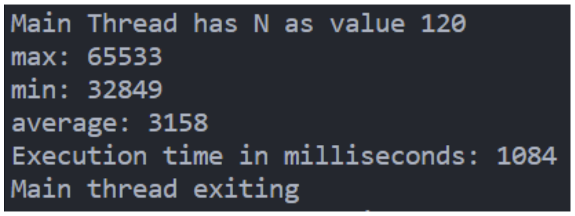


```
Main Thread has N as value 28
max: 16615992
min: 8388610
average: 1466234
Execution time in milliseconds: 1064
Main thread exiting
```

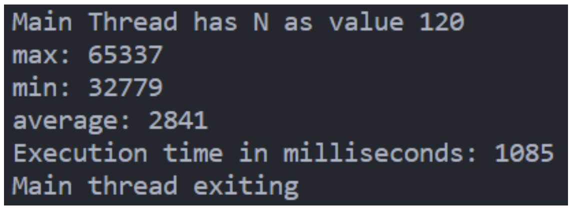


```
Main Thread has N as value 28
max: 16692577
min: 8405970
average: 1730321
Execution time in milliseconds: 1068
Main thread exiting
```

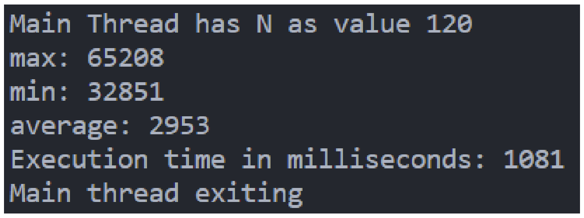
**Output for size 16:**



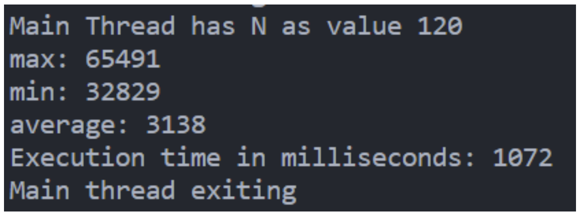
```
Main Thread has N as value 120
max: 65533
min: 32849
average: 3158
Execution time in milliseconds: 1084
Main thread exiting
```



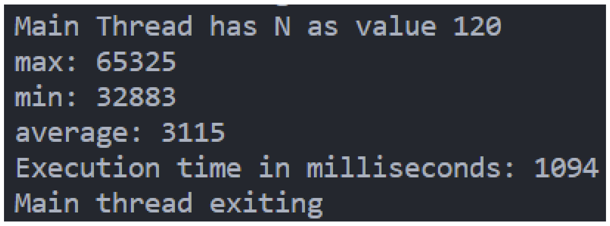
```
Main Thread has N as value 120
max: 65337
min: 32779
average: 2841
Execution time in milliseconds: 1085
Main thread exiting
```



```
Main Thread has N as value 120
max: 65208
min: 32851
average: 2953
Execution time in milliseconds: 1081
Main thread exiting
```



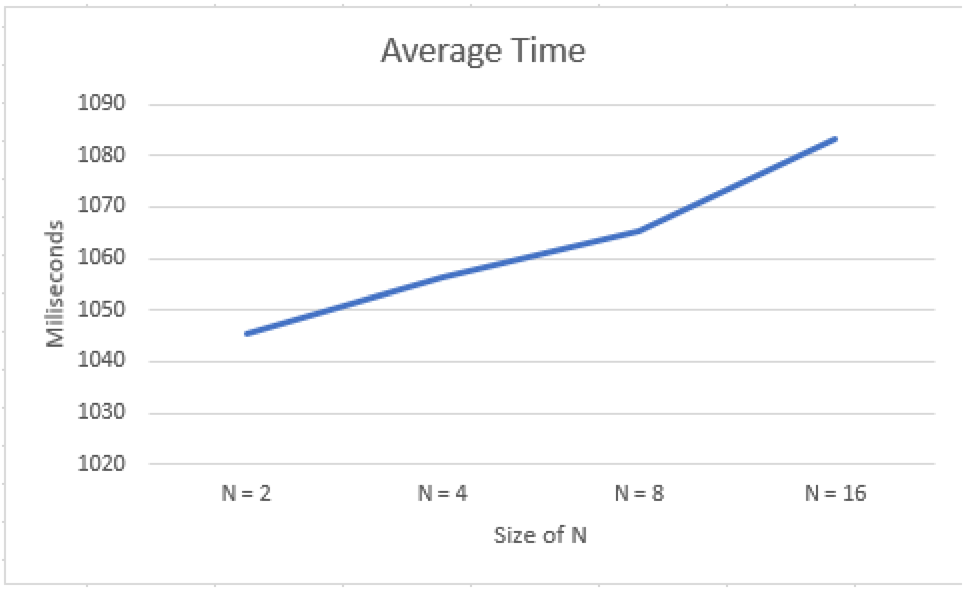
```
Main Thread has N as value 120
max: 65491
min: 32829
average: 3138
Execution time in milliseconds: 1072
Main thread exiting
```

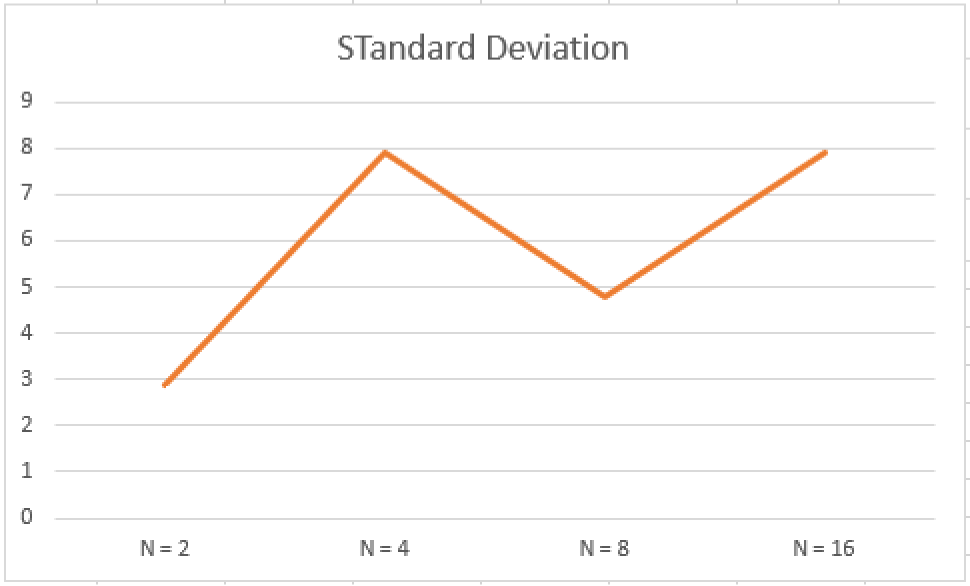


```
Main Thread has N as value 120
max: 65325
min: 32883
average: 3115
Execution time in milliseconds: 1094
Main thread exiting
```

Joey Troyer
11/10/22

**Graphs:**



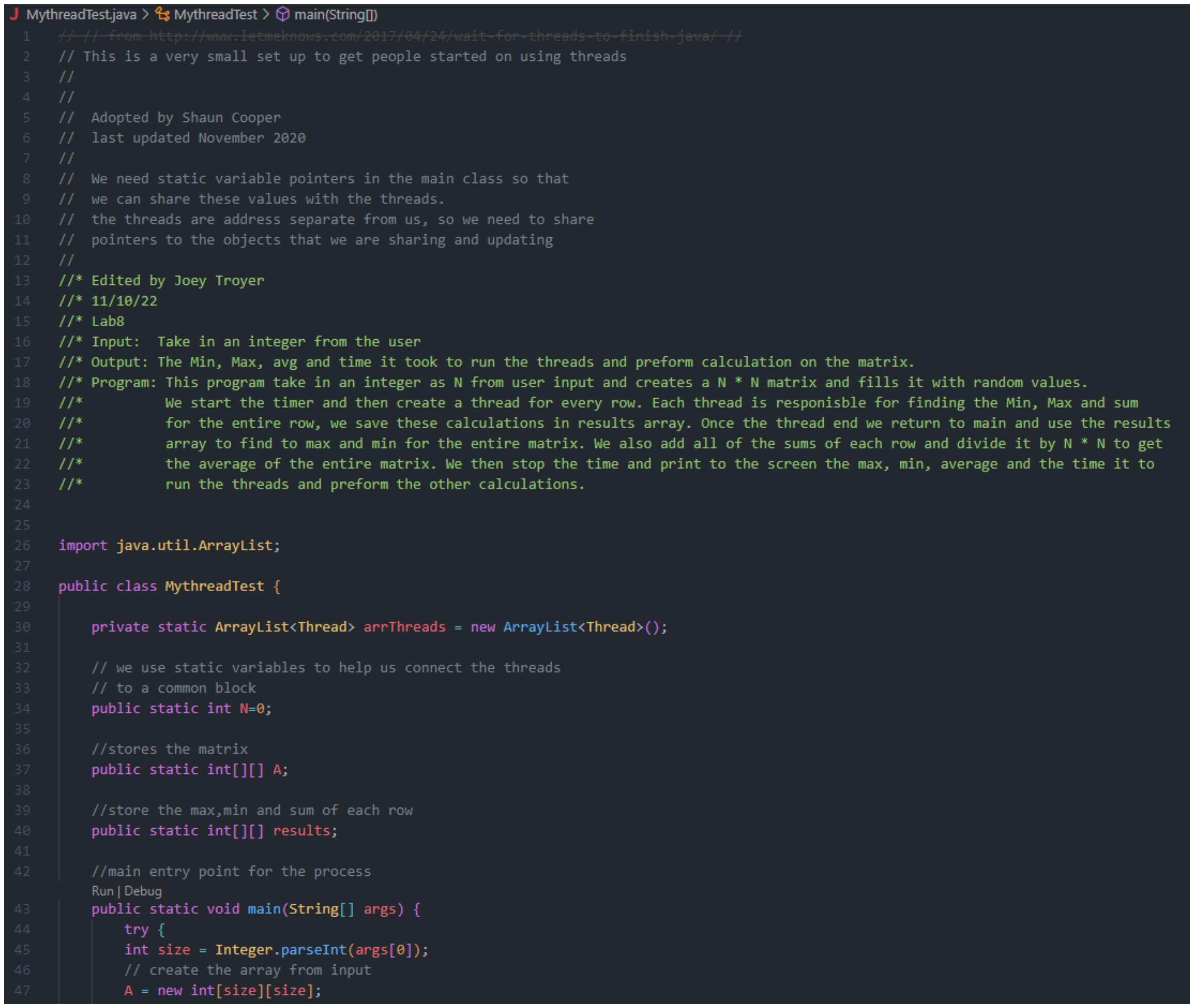


**Analysis:**

The average time it what I would we expect, the larger the size of the matrix the longer the computation takes to complete. The time difference it from the smallest size to the largest size is overall still very short, only about 40ms. I think this is due to the fact the larger matrix we have the smaller the numbers in the matrix become. The standard deviation was lowest for size 2 meaning that we had the most consistent data for size of 2. Sizes 4 and 16 were both significantly greater than size 2 and 4. I am unsure why standard deviation is much greater for those two sizes and smaller for the other two.

Joey Troyer
11/10/22

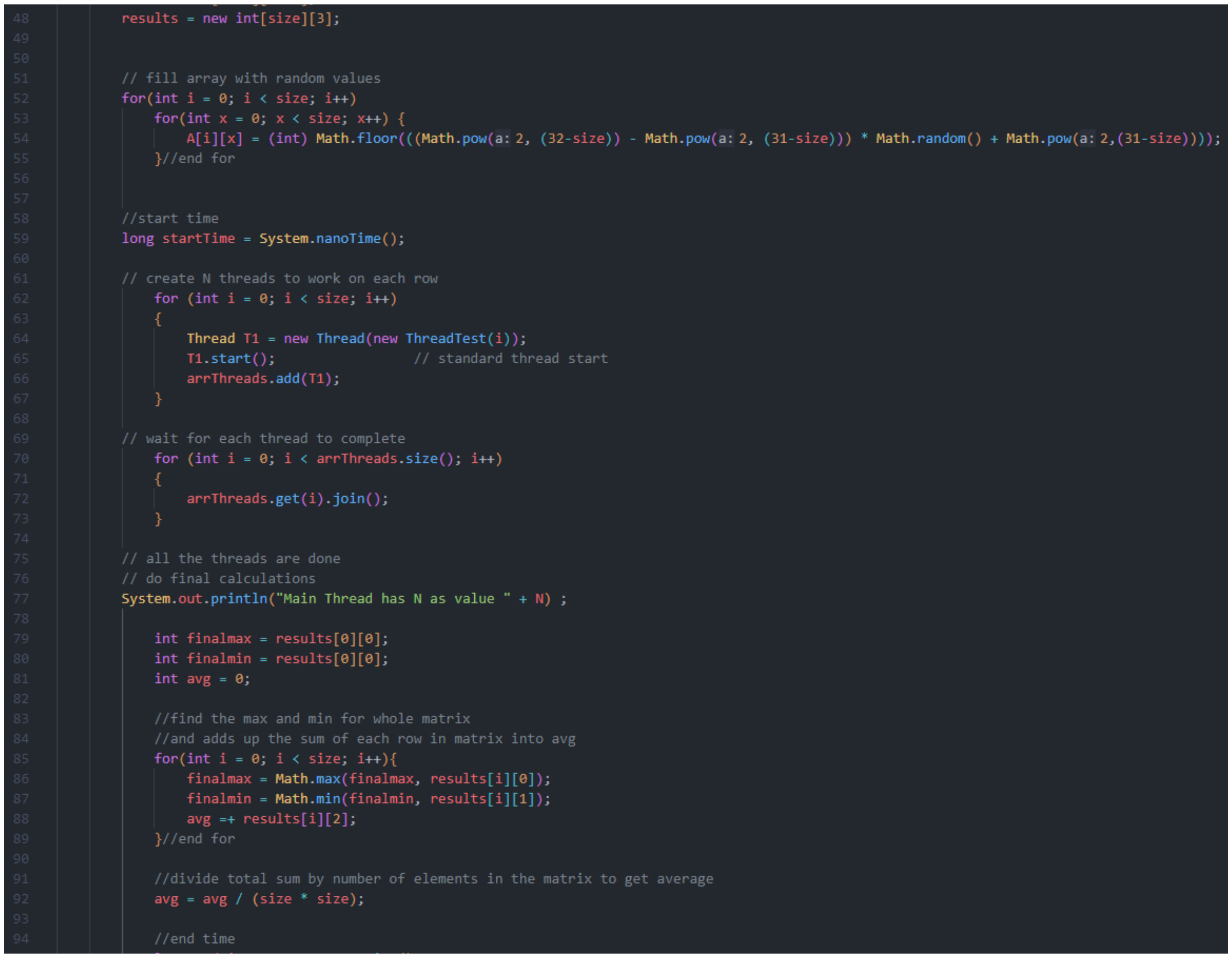
**Code:**



MythreadTest.java > MythreadTest > main(String[])

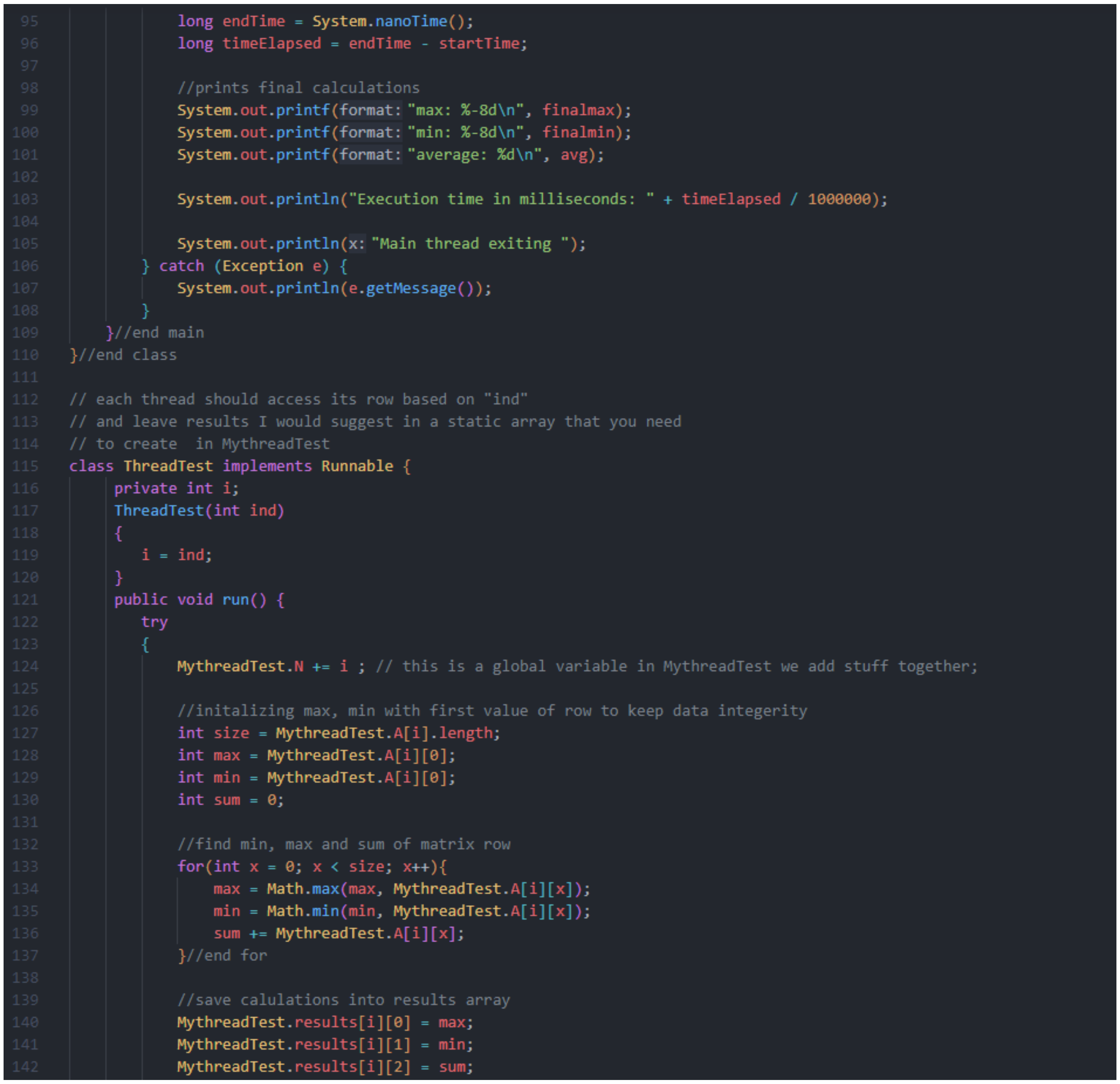
```
~~// // from http://www.letmeknows.com/2017/04/24/wait-for-threads-to-finish-java/ //~~
// This is a very small set up to get people started on using threads
//
//
//  Adopted by Shaun Cooper
//  last updated November 2020
//
//  We need static variable pointers in the main class so that
//  we can share these values with the threads.
//  the threads are address separate from us, so we need to share
//  pointers to the objects that we are sharing and updating
//
//* Edited by Joey Troyer
//* 11/10/22
//* Lab8
//* Input:  Take in an integer from the user
//* Output: The Min, Max, avg and time it took to run the threads and preform calculation on the matrix.
//* Program: This program take in an integer as N from user input and creates a N * N matrix and fills it with random values.
//*          We start the timer and then create a thread for every row. Each thread is responisble for finding the Min, Max and sum
//*          for the entire row, we save these calculations in results array. Once the thread end we return to main and use the results
//*          array to find to max and min for the entire matrix. We also add all of the sums of each row and divide it by N * N to get
//*          the average of the entire matrix. We then stop the time and print to the screen the max, min, average and the time it to
//*          run the threads and preform the other calculations.


import java.util.ArrayList;

public class MythreadTest {

    private static ArrayList<Thread> arrThreads = new ArrayList<Thread>();

    // we use static variables to help us connect the threads
    // to a common block
    public static int N=0;

    //stores the matrix
    public static int[][] A;

    //store the max,min and sum of each row
    public static int[][] results;

    //main entry point for the process
    Run | Debug
    public static void main(String[] args) {
        try {
        int size = Integer.parseInt(args[0]);
        // create the array from input
        A = new int[size][size];
```

Joey Troyer
11/10/22

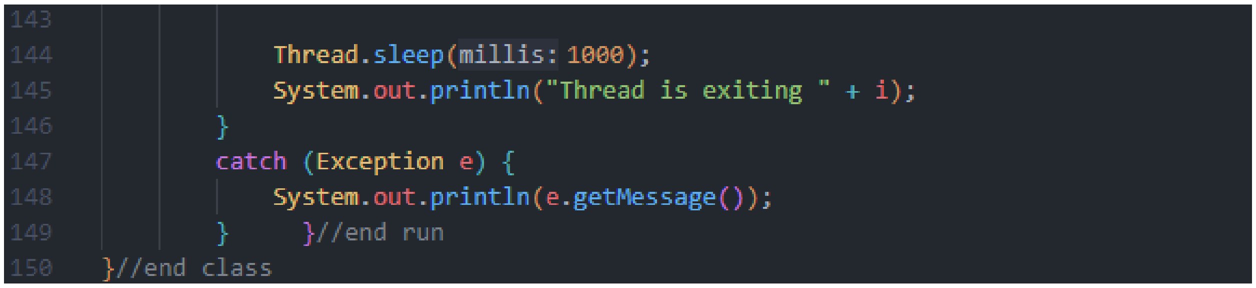


```
        results = new int[size][3];


        // fill array with random values
        for(int i = 0; i < size; i++)
            for(int x = 0; x < size; x++) {
                A[i][x] = (int) Math.floor(((Math.pow(a: 2, (32-size)) - Math.pow(a: 2, (31-size))) * Math.random() + Math.pow(a: 2,(31-size))));
            }//end for


        //start time
        long startTime = System.nanoTime();

        // create N threads to work on each row
            for (int i = 0; i < size; i++)
            {
                Thread T1 = new Thread(new ThreadTest(i));
                T1.start();                 // standard thread start
                arrThreads.add(T1);
            }

        // wait for each thread to complete
            for (int i = 0; i < arrThreads.size(); i++)
            {
                arrThreads.get(i).join();
            }

        // all the threads are done
        // do final calculations
        System.out.println("Main Thread has N as value " + N) ;

            int finalmax = results[0][0];
            int finalmin = results[0][0];
            int avg = 0;

            //find the max and min for whole matrix
            //and adds up the sum of each row in matrix into avg
            for(int i = 0; i < size; i++){
                finalmax = Math.max(finalmax, results[i][0]);
                finalmin = Math.min(finalmin, results[i][1]);
                avg =+ results[i][2];
            }//end for

            //divide total sum by number of elements in the matrix to get average
            avg = avg / (size * size);

            //end time
```

Joey Troyer
11/10/22



```java
            long endTime = System.nanoTime();
            long timeElapsed = endTime - startTime;

            //prints final calculations
            System.out.printf(format: "max: %-8d\n", finalmax);
            System.out.printf(format: "min: %-8d\n", finalmin);
            System.out.printf(format: "average: %d\n", avg);

            System.out.println("Execution time in milliseconds: " + timeElapsed / 1000000);

            System.out.println(x: "Main thread exiting ");
        } catch (Exception e) {
            System.out.println(e.getMessage());
        }
    }//end main
}//end class

// each thread should access its row based on "ind"
// and leave results I would suggest in a static array that you need
// to create  in MythreadTest
class ThreadTest implements Runnable {
     private int i;
     ThreadTest(int ind)
     {
        i = ind;
     }
     public void run() {
        try
        {
            MythreadTest.N += i ; // this is a global variable in MythreadTest we add stuff together;

            //initalizing max, min with first value of row to keep data integerity
            int size = MythreadTest.A[i].length;
            int max = MythreadTest.A[i][0];
            int min = MythreadTest.A[i][0];
            int sum = 0;

            //find min, max and sum of matrix row
            for(int x = 0; x < size; x++){
                max = Math.max(max, MythreadTest.A[i][x]);
                min = Math.min(min, MythreadTest.A[i][x]);
                sum += MythreadTest.A[i][x];
            }//end for

            //save calulations into results array
            MythreadTest.results[i][0] = max;
            MythreadTest.results[i][1] = min;
            MythreadTest.results[i][2] = sum;
```



```java

            Thread.sleep(millis: 1000);
            System.out.println("Thread is exiting " + i);
        }
        catch (Exception e) {
            System.out.println(e.getMessage());
        }     }//end run
}//end class
```