Eric Wilson
Java Project #9

Part 1

```
/**
 * Created by EW043872 on 10/23/2015.
 */
public class ThreadingProject extends Thread {
    public void run(){
        for (int i = 0; i < 200000000; i++){
            f();
        }
    }
    private static int x,y,z;

    public static void f(){
        x = x + 1;
        y = y + 1;
        z = z + x - y;
    }

    public static void printValues() {
        System.out.print("x = " + x + "\n");
        System.out.print("y = " + y + "\n");
        System.out.print("z = " + z + "\n");
    }

    public static void main(String args[]){
        long startTime = System.nanoTime();
        ThreadingProject tp = new ThreadingProject();
        try {
            tp.start();
            tp.join();
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
        long endTime = System.nanoTime();
        tp.printValues();
        System.out.println("Computation took " + ((endTime - startTime) / 1000000)
        + " milliseconds");
    }
}
```

```
"C:\Program ...
x = 200000000
y = 200000000
z = 0
Computation took 172 milliseconds

Process finished with exit code 0
```

Part 2

```
/**
 * Created by EW043872 on 10/23/2015.
 */
public class ThreadingProject extends Thread {
    public void run(){
        for (int i = 0; i < 100000000; i++){
            f();
        }
    }
    private static int x,y,z;

    public static void f(){
        x = x + 1;
        y = y + 1;
```

```
        z = z + x - y;
    }

    public static void printValues() {
        System.out.print("x = " + x + "\n");
        System.out.print("y = " + y + "\n");
        System.out.print("z = " + z + "\n");
    }

    public static void main(String args[]){
        long startTime = System.nanoTime();
        ThreadingProject tp = new ThreadingProject();
        tp.start();
        for (int i = 0; i < 100000000; i++){
            tp.f();
        }
        long endTime = System.nanoTime();
        tp.printValues();
        System.out.println("Computation took " + ((endTime - startTime) / 1000000)
        + " milliseconds");
    }
}
```

```
"C:\Program ...
x = 97749110
y = 97949631
z = 1442473447
Computation took 92 milliseconds

Process finished with exit code 0
```

```
"C:\Program ...
x = 103999306
y = 104284557
z = -1981584768
Computation took 102 milliseconds

Process finished with exit code 0
```

```
"C:\Program ...
x = 100000000
y = 100000000
z = 0
Computation took 111 milliseconds

Process finished with exit code 0
```

```
"C:\Program ...
x = 109522682
y = 109829146
z = 0
Computation took 135 milliseconds

Process finished with exit code 0
```

Part 3

```
/**
 * Created by EW043872 on 10/23/2015.
 */
public class ThreadingProject extends Thread {
    public void run(){
        for (int i = 0; i < 100000000; i++){
            f();
        }
    }
    private int x,y,z;

    public synchronized void f(){
        x = x + 1;
        y = y + 1;
        z = z + x - y;
    }

    public void printValues() {
        System.out.print("x = " + x + "\n");
        System.out.print("y = " + y + "\n");
        System.out.print("z = " + z + "\n");
    }

    public static void main(String args[]){
        long startTime = System.nanoTime();
        ThreadingProject tp = new ThreadingProject();
        tp.start();
```

```
        for (int i = 0; i < 100000000; i++){
            tp.f();
        }
        long endTime = System.nanoTime();
        tp.printValues();
        System.out.println("Computation took " + ((endTime - startTime) / 1000000)
        + " milliseconds");
    }
}
```

```
"C:\Program ...
x = 200000000
y = 200000000
z = 0
Computation took 8278 milliseconds

Process finished with exit code 0
```

```
"C:\Program ...
x = 200000000
y = 200000000
z = 0
Computation took 8688 milliseconds

Process finished with exit code 0
```

```
"C:\Program ...
x = 200000000
y = 200000000
z = 0
Computation took 8911 milliseconds

Process finished with exit code 0
```

```
"C:\Program ...
x = 171053134
y = 171076445
z = 0
Computation took 7529 milliseconds

Process finished with exit code 0
```

It appears that synchronized threads performing half of the same task each, at least on my machine, takes about

PART 4

```
import java.util.ArrayList;
import java.util.Random;

/**
 * Created by EW043872 on 10/23/2015.
 */
public class ThreadingProject extends Thread {
    int[][] matrix = new int[3][10000000];
    int x; //used to determine which thing will be thinging
    int total; //holds the total for each thread

    //This would probably be better if I only passed individual rows.
    public ThreadingProject(int[][] passedMatrix, int passedX){
        matrix = passedMatrix;
        //x is the row that this instance of the object will be working on
        x = passedX;
    }

    //actually does the calculations
    @Override
    public void run() {
        for (int y = 0; y < matrix[x].length; y++) {
            total = total + (int)Math.log(matrix[x][y]);
        }
    }

    //for adding individual totals to the megaSum
    public int getTotal(){
        return total;
```

```
    }

    //calculates all of the rows
    public int logSum(){
        int sum = 0;
        for (int x = 0; x < matrix.length ; x++) {
            for (int y = 0; y < matrix[x].length; y++) {
                sum = sum + (int)Math.log(matrix[x][y]);
            }
        }
        return sum;
    }



//MAINMAINMAIN
    public static void main(String args[]){
        int[][] mainMatrix = new int[3][10000000];
        Random rand = new Random();
        // Initialize matrix with random numbers
        for (int x = 0; x < mainMatrix.length; x++) {
            for (int y = 0; y < mainMatrix[x].length; y++) {
                int randomNum = rand.nextInt(200); // 0 - 199
                mainMatrix[x][y] = randomNum;
            }
        }
        //Single threaded portion
        ThreadingProject tp = new ThreadingProject(mainMatrix, -1);
        System.out.print("matrix.length " + tp.matrix.length + "\n");
        long startTime = System.nanoTime();
        System.out.print("Logsum single thread = " + tp.logSum() + "\n");
        long endTime = System.nanoTime();
        System.out.println("Computation took " + ((endTime - startTime) / 1000000)
                + " milliseconds\n");

        //multithreaded portion
        int megaSum = 0;
        //Declare an array of thready objects and
        //give them their row assignments
       ArrayList<ThreadingProject> arr = new ArrayList<ThreadingProject>();

        for (int t = 0; t < mainMatrix.length; t++){
            arr.add(t, new ThreadingProject(mainMatrix, t));
        }

        //GO!
        startTime = System.nanoTime();
        //start all threads
        for(int i =0; i < mainMatrix.length; i++){
            arr.get(i).start();
        }
        //wait for threads to die and then harvest their sweet data
        for(int i =0; i < mainMatrix.length; i++){
            try {
                arr.get(i).join();
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
            megaSum += arr.get(i).getTotal();
        }
        //aaaaaand... TIME!
        endTime = System.nanoTime();
        System.out.print("Logsum multi thread = " + megaSum + "\n");
        System.out.println("Computation took " + ((endTime - startTime) / 1000000)
                + " milliseconds");



    }
}
```

```
"C:\Program ...
matrix.length 3
Logsum single thread = 114597945
Computation took 835 milliseconds

Logsum multi thread = 114597945
Computation took 290 milliseconds

Process finished with exit code 0
```

Time Ratio(single/multi): 2.88

```
"C:\Program ...
matrix.length 5
Logsum single thread = 114607105
Computation took 950 milliseconds

Logsum multi thread = 114607105
Computation took 243 milliseconds

Process finished with exit code 0
```

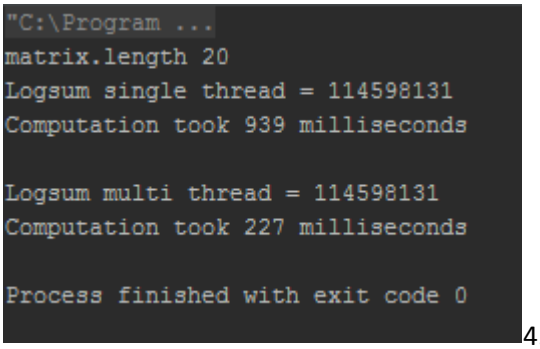
3.91

```
"C:\Program ...
matrix.length 7
Logsum single thread = 114592057
Computation took 866 milliseconds

Logsum multi thread = 114592057
Computation took 239 milliseconds

Process finished with exit code 0
```

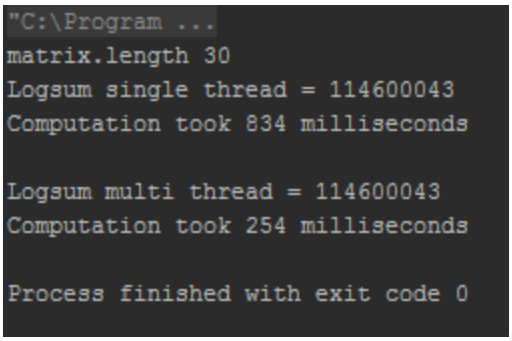
3.62

```
"C:\Program ...
matrix.length 10
Logsum single thread = -2032886181
Computation took 980 milliseconds

Logsum multi thread = -2032886181
Computation took 241 milliseconds

Process finished with exit code 0
```

4.07



4.13



3.28

I do not notice a significant speedup after 3 threads. If anything it becomes more sluggish.