Assignment 7

Instructions

- Max score is 10.
- Deadline is 11:59PM, March 21, Saturday.
- Try to attempt every question and keep practicing from other online sites.
- Contact us if any assistance is needed.

1. When a program fails due to an uncaught exception, the system automatically prints out the exception's stack trace. If the failure is not easily reproducible, it may be difficult or impossible to get any more information. Therefore, it is critically important that the exception's toString() method return, as much information as possible concerning the cause of the failure. In other words, the detail message of an exception should capture the failure for subsequent analysis. To capture the failure, the detail message of an exception should contain the values of all parameters and fields that "contributed to the exception." (Score 2).

- Create your own **MyIndexOutOfBoundException** Class. It should contain these parameters
    - **lowerBound** - the lowest legal index value.
    - **upperBound** - the highest legal index value.
    - **index** - the current index value.

- Test your code in main method, by creating an indexOutOfBoundException. Output error message should be like this:

  `"Error Message: Index: 10, but Lower bound: 0, Upper bound: 9"`

2. Write a program that calculate the sum value in an array of ints using 4 threads. You should construct an 4,000,000 long array of random numbers and return the sum of the array. Please finish those two method: generateRandomArray and sum.

```
public class SumValue {
```

```
    /*generate array of random numbers*/
    static void generateRandomArray(int[] arr){

    }

    /*get sum of an array using 4 threads*/
    static long sum(int[] arr){
        return 0;
    }

    public static void main(String[] args){
        int[] arr = new int[4000000];
        generateArray(arr);
        long sum = sum(arr);
        System.out.println("Sum: " + sum);
    }
}
```

3. Write a program called **ReverseHello.java** that creates a thread (let's call it Thread 1). Thread 1 creates another thread (Thread 2); Thread 2 creates Thread 3; and so on, up to Thread 50. Each thread should print

Hello from Thread num!

but you should structure your program such that the threads print their greetings in reverse order.