Threads task

1. See and try the SumThread example below- be ready to explain it
2. Write a class called MaxValue that finds the maximum value in an array of ints using ThreadPool of 4 threads (use the executor to manage them). Your main should be similar as the one in the below SumThread example, though you should construct your array of random numbers instead of increasing numbers. You may assume in your threaded code that the array has at least 4 elements.
3. Write a class called ReverseHello 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*.
4. Create 3 threads. Threads must access and change one primitive variable, one Atomic variable, synchronized and unsynchronized method.
5. \*Create and run the same thread using a lambda expression and access the atomicInteger variable with updateAndGet() (for example) method also via lambdas)

\*\* You can choose – which task to do: 4 either 5;

SumThread example:

/\*\*

 \* This thread finds the sum of a subsection of an array.

 \*/

public class SumThread extends Thread {

    private int lo, hi;

    private int[] arr;

    private int ans = 0;

    public SumThread(int[] arr, int lo, int hi) {

        this.lo = lo;

        this.hi = hi;

        this.arr = arr;

    }

    @Override

    public void run() {

        for (int i = lo; i < hi; i++) {

            ans += arr[i];

        }

    }

    /\*\*

     \* Sum the elements of an array.

     \*

     \* @param arr array to sum

     \* @return sum of the array's elements

     \* @throws InterruptedException shouldn't happen

     \*/

    public static int sum(int[] arr) throws InterruptedException {

        int len = arr.length;

        int ans = 0;

        // Create and start 4 threads.

        SumThread[] ts = new SumThread[4];

        for (int i = 0; i < 4; i++) {

            ts[i] = new SumThread(arr, (i \* len) / 4, ((i + 1) \* len / 4));

            ts[i].start();

        }

        // Wait for the threads to finish and sum their results.

        for (int i = 0; i < 4; i++) {

            ts[i].join();

            ans += ts[i].ans;

        }

        return ans;

    }

    public static void main(String[] args) throws InterruptedException {

        int[] arr = new int[100];

        for (int i = 0; i < arr.length; i++) {

            arr[i] = i;

        }

        int sum = sum(arr);

        System.out.println("Sum: " + sum);

    }

}