Assignment16.java

```
public class Assignment16 {
    public static void main(String[] args) {
        if (args.length != 1) {
            System.out.println("One command line parameter only: must be an interger
(or long) number");
        } else {
            long iterationsPerThread = Long.parseLong(args[0]);

            // Call to student's class:
            // takes number of iterations per thread as input
            // returns result of two threads each incrementing a shared
            // variable "in" (initialised with 0) "iterationsPerThread" times.
            long result = MyAssignment16.main(iterationsPerThread);

            System.out.println(result);
        }
    }
}
```

MonitorSemaphore.java

```
public class MonitorSemaphore {
    private volatile int count;

    public MonitorSemaphore(int init) {
        count = init;
    }

    public synchronized void op1MyWait() {
        count = count - 1;
        try {
            if (count < 0) {
                this.wait();
            }
        } catch (Exception e) {
            //Error
        }
    }

    public synchronized void op2MySignal() {
        count = count + 1;
        try {
            if (count <= 0) {
                this.notify();
            }
        } catch (Exception e) {
```

```
            //Error
        }
    }
}
```

MyAssignment16.java

```
public class MyAssignment16 extends Thread {

    private long theNumberOfIterations;

    private Counter theCounter;

    public MyAssignment16(Counter counter, long iterationsPerThread) {
        theCounter = counter;
        theNumberOfIterations = iterationsPerThread;
    }

    public void run() {
        for (long i = 0; i < theNumberOfIterations; ++i) {
            System.err.print(this.getName());
            theCounter.increment();
        }
    }

    public static long main(long iterationsPerThread) { // Do not modify this
                                                        // line!
        Counter counter = new Counter();

        MyAssignment16 raceDemo0 = new MyAssignment16(counter, iterationsPerThread);
        MyAssignment16 raceDemo1 = new MyAssignment16(counter, iterationsPerThread);
        raceDemo0.setName("0");
        raceDemo1.setName("1");

        raceDemo0.start();
        raceDemo1.start();
        try {
            raceDemo0.join();
            raceDemo1.join();
        } catch (InterruptedException e) {
            e.printStackTrace();
        }

        return counter.getIn();
    }

}
```

Counter.java

```java
public class Counter {
    private volatile long in = 0;

    private MonitorSemaphore myMonSem = new MonitorSemaphore(1);

    public void increment() {
        long next_free_slot;

        myMonSem.op1MyWait();

        next_free_slot = in;
        next_free_slot = next_free_slot + 1;
        in = next_free_slot;

        myMonSem.op2MySignal();
    }

    public long getIn() {
      return in;
    }
}
```