TÖL304G Forritunarmál

Hópverkefni 13

Hjörvar Sigurðsson

1.

Kóði:

```
/**
 * A class containing nested classes for
 *  group assignement 13 in TÖL304G.
 *  The nested classes are a container,
 *  a producer, and a consumer.
 *  The producer is a thread which places
 *  integers in range 1^2, 2^2, ... 10000^2
 *  to the container; the consumer is a thread
 *  which removes the same numbers from said
 *  container, sums the removed numbers
 *  together, and prints out the sum.
 */
public class H13 {

    // Hvert tilvik af þessum klasa er ílát fyrir
    // eina heiltölu (int). Fleiri en einn þráður
    // getur notað ílátið án þess að spilla
    // því að ílátið sé í réttu ástandi.
    static class Container
    {
        boolean isEmpty = true;
        int theValue;

        // Fastayrðing gagna:
        // Ilátið er tómt þþaa isEmpty sé true.
        // Ef ílátið er ekki tómt þá inniheldur
        // theValue gildið í ílátinu.

        // Notkun: Container c = new Container();
        // Fyrir: Ekkert.
        // Eftir: c vísar á nýjan tóman Container.
        // Ath.: Ekki þarf að forrita þennan smið,
        // hann verður sjálfkrafa til.

        // Notkun: c.put(x);
        // Eftir: Búið er að setja x í ílátið c.
        // Ef til vill þurfti að bíða eftir
        // að ílátið tæmdist áður en það
        // tókst.
        public synchronized void put( int x )
```

```
        throws InterruptedException
        {
            while( !isEmpty )
            // Fastayrðing gagna er sönn.
            // Þessi þráður hefur núll sinnum eða
            // oftar kallað á wait() til að reyna
            // að fá pláss til að setja gildið x
            // í hlutinn. A undan sérhverju slíku
            // kalli var ekki pláss, þ.e. isEmpty
            // var ósatt.
            wait();
            isEmpty = false;
            theValue = x;
            notifyAll();
        }
        // Notkun: x = c.get();
        // Eftir: Búið er að sækja x úr ílátinu c.
        // Ef til vill þurfti að bíða eftir
        // að ílátið fylltist áður en það
        // tókst.
        public synchronized int get()
        throws InterruptedException
        {
            while( isEmpty )
            // Fastayrðing gagna er sönn.
            // Þessi þráður hefur núll sinnum eða
            // oftar kallað á wait() til að reyna
            // að gildi í hlutinn til að fjarlægja.
            // A undan sérhverju slíku kalli var
            // ekki gildi í hlutnum, þ.e. isEmpty
            // var satt.
            wait();
            int x = theValue;
            isEmpty = true;
            notifyAll();
            return x;
        }
    }

    /**
     * Each instance of this class is a thread
     *  which produces the integers 1^2, 2^2, ... 10000^2
     *  and puts them into a container of the
     *  class Container.
     */
    public static class Producer extends Thread{

        // Instance variables.
```

```
        private Container container;    // The container into which the
producer puts the numbers.

        /**
         * Data invariant:
         *  container is a Container object which
         *  may either be empty or contain a single
         *  integer.
         */

        /**
         * Constructor for a Producer.
         * @param c - Container object into which the
         *  producer will place an integer.
         *
         * Use: Producer p = new Producer(c);
         * Pre: c is a Container object.
         * Post: p is an uninitiated thread with
         *       Container c as an instance variable.
         */
        Producer(Container c) {
            this.container = c;
        }

        /**
         * Use: p.run();
         * Pre: p is an uninitiated Producer.
         * Post: p has put the integers
         *        1^2, 2^2, ... , 10000^2
         *        into its container.
         */
        @Override
        public void run() {
            /**
             * Loop invariant:
             * i is an int in the range
             *  1 <= i <= 10001.
             * container is either empty
             *  or contains an integer in
             *  the range 1^2, 2^2, ..., 10000^2.
             */
            for (int i = 1; i < 10001; i++) {
                try {
                    container.put(i*i);
                } catch (InterruptedException e) {
                }
            }
        }
    }
```

```
    /**
     * Each instance of this class is a thread
     *  which removes 10000 integers in the range
     *  1^2, 2^2, ... 10000^2 from a container of
     *  the class Container, adds them all together,
     *  and stores the sum in a Long variable.
     */
    public static class Consumer extends Thread{
        // Instance variables.
        private Container container;    // The container from which the
consumer removes the integers.
        private Long sum = (long) 0;    // The sum of the removed numbers so
far.

        /**
         * Constructor for the consumer.
         * @param c - Container object into which the
         *  producer will place an integer.
         *
         * Use: Consumer consumer = new Consumer(c);
         * Pre: c is a Container object.
         * Post: consumer is an uninitiated Consumer with
         *        Container c as an instance variable.
         */
        public Consumer(Container c) {
            this.container = c;
        }

        /**
         * Use: consumer.run();
         * Pre: consumer is an uninitiated Consumer.
         * Post: consumer has removed 10000 integers
         *        in the range 1^2, 2^2, ... , 10000^2
         *        from its container, added them all
         *        together and stored it as the
         *        Long variable sum.
         *        The contents of sum has been printed
         *        in the standard output.
         */
        @Override
        public void run() {
            /**
             * Loop invariant:
             * i is an int in the range
             *  1 <= i <= 10001.
             * container is either empty
             *  or contains an integer in
             *  the range 1^2, 2^2, ..., 10000^2.
```

```
             */
            for (int i = 1; i < 10001; i++) {
                try {
                    int g = container.get();
                    sum += g;
                } catch (InterruptedException e) {
                }
            }
            System.out.println("Sum = " + sum);

        }
    }

    /**
     * Creates a Container, and two producers and two consumers with
     *  said container as a paramater. The consumers and producers
     *  run seperate threads and compete for access to the container;
     *  the producer repeatedly places an int into the container, and the
     *  consumer repeatedly removes an integer from the container.
     *  The integers are in the range 1^2, 2^2, ..., 10000^2.
     */
    public static void main(String[] args) {
        Container container = new Container();
        Producer producer1 = new Producer(container);
        Producer producer2 = new Producer(container);
        Consumer consumer1 = new Consumer(container);
        Consumer consumer2 = new Consumer(container);
        consumer1.start();
        consumer2.start();
        producer1.start();
        producer2.start();
    }
}
```

Keyrsla:

```
Sum = 304534285574
Sum = 362232384426
```

```
Sum = 303771037600
Sum = 362995632400
```

```
Sum = 360884449347
Sum = 305882220653
```