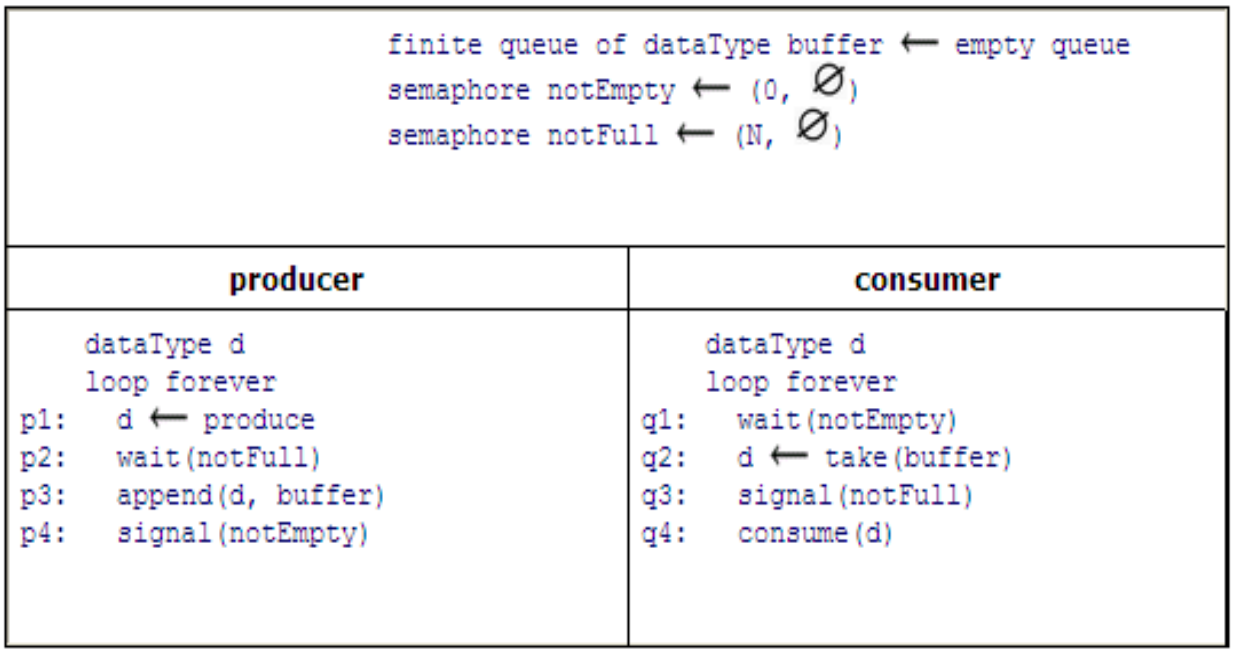
/* Copyright (C) 2006 M. Ben-Ari. See copyright.txt */



/* Programmed by Panu Pitkämäki */

```
import java.util.concurrent.Semaphore;

import java.util.Queue;

import java.util.LinkedList;

/* Producer-consumer using split semaphore */

class ProducerConsumer {

   /* Size of the finite queue */

   static final int N = 10;

   /* Split semaphore. At the start N empty slots which equals 0 used slots. */

   static Semaphore empty = new Semaphore(N);

   static Semaphore full = new Semaphore(0);

   /* Finite queue of produced items */

   static Queue<Integer> queue = new LinkedList<Integer>();

   class Producer extends Thread {
```

```
    public void run() {
        int c = 1;
        while (true) {
            try {
                empty.acquire();
            } catch (InterruptedException e) {
            }
            System.out.println("Producing item " + c);
            queue.add(c++);
            full.release();
        }
    }
}

class Consumer extends Thread {
    public void run() {
        while (true) {
            try {
                full.acquire();
            } catch (InterruptedException e) {
            }
            System.out.println("Consuming item " + queue.remove());
            empty.release();
        }
    }
```

```
    }

    ProducerConsumer() {
        new Producer().start();
        new Consumer().start();
    }
    public static void main(String[] args) {
        new ProducerConsumer();
    }
}
```