```
class TwoThreadLockFreeQueue<T> {
  int head = 0, tail = 0;
  T[] items;
  public TwoThreadLockFreeQueue(int capacity) {
    head = 0; tail = 0;
    items = (T[]) new Object[capacity];
  }
  public void enq(T x) {
    while (tail - head == items.length) {};
    items[tail % items.length] = x;
    tail++;
  }
  public Object deq() {
    while (tail - head == 0) {};
    Object x = items[head % items.length];
    head++;
    return x;
  }
}
```

FIGURE 10.21 A lock-free FIFO queue with blocking semantics for a single enqueuer and single dequeuer. The queue is implemented in an array. Initially the head and tail fields are equal and the queue is empty. If the head and tail differ by capacity, then the queue is full. The enq() method reads the head field, and if the queue is full, it repeatedly checks the head until the queue is no longer full. It then stores the object in the array, and increments the tail field. The deq() method works in a symmetric way.