

```

1  class LockedQueue<T> {
2      final Lock lock = new ReentrantLock();
3      final Condition notFull = lock.newCondition();
4      final Condition notEmpty = lock.newCondition();
5      final T[] items;
6      int tail, head, count;
7      public LockedQueue(int capacity) {
8          items = (T[])new Object[capacity];
9      }
10     public void enq(T x) {
11         lock.lock();
12         try {
13             while (count == items.length)
14                 notFull.await();
15             items[tail] = x;
16             if (++tail == items.length)
17                 tail = 0;
18             ++count;
19             notEmpty.signal();
20         } finally {
21             lock.unlock();
22         }
23     }
24     public T deq() {
25         lock.lock();
26         try {
27             while (count == 0)
28                 notEmpty.await();
29             T x = items[head];
30             if (++head == items.length)
31                 head = 0;
32             --count;
33             notFull.signal();
34             return x;
35         } finally {
36             lock.unlock();
37         }
38     }
39 }

```

**FIGURE 8.5** The LockedQueue class: a FIFO queue using locks and conditions. There are two condition fields, one to detect when the queue becomes nonempty, and one to detect when it becomes nonfull.