```
1  public class MCSLock implements Lock {
2    AtomicReference<QNode> tail;
3    ThreadLocal<QNode> myNode;
4    public MCSLock() {
5      tail = new AtomicReference<QNode>(null);
6      myNode = new ThreadLocal<QNode>() {
7        protected QNode initialValue() {
8          return new QNode();
9        }
10     };
11   }
12   public void lock() {
13     QNode qnode = myNode.get();
14     QNode pred = tail.getAndSet(qnode);
15     if (pred != null) {
16       qnode.locked = true;
17       pred.next = qnode;
18       // wait until predecessor gives up the lock
19       while (qnode.locked) {}
20     }
21   }
22   public void unlock() {
23     QNode qnode = myNode.get();
24     if (qnode.next == null) {
25       if (tail.compareAndSet(qnode, null))
26         return;
27       // wait until successor fills in its next field
28       while (qnode.next == null) {}
29     }
30     qnode.next.locked = false;
31     qnode.next = null;
32   }
33   class QNode {
34     volatile boolean locked = false;
35     volatile QNode next = null;
36   }
37 }
```

FIGURE 7.11 The MCSLock class.