```
1  public class CompositeFastPathLock extends CompositeLock {
2    private static final int FASTPATH = 1 << 30;
3    private boolean fastPathLock() {
4      int oldStamp, newStamp;
5      int stamp[] = {0};
6      QNode qnode;
7      qnode = tail.get(stamp);
8      oldStamp = stamp[0];
9      if (qnode != null) {
10       return false;
11     }
12     if ((oldStamp & FASTPATH) != 0) {
13       return false;
14     }
15     newStamp = (oldStamp + 1) | FASTPATH;
16     return tail.compareAndSet(qnode, null, oldStamp, newStamp);
17   }
18   public boolean tryLock(long time, TimeUnit unit) throws InterruptedException {
19     if (fastPathLock()) {
20       return true;
21     }
22     if (super.tryLock(time, unit)) {
23       while ((tail.getStamp() & FASTPATH ) != 0){};
24       return true;
25     }
26     return false;
27   }
```

**FIGURE 7.31** CompositeFastPathLock class: The private fastPathLock() method returns true if it succeeds in acquiring the lock through the fast path.