```
1  public class LockFreeExchanger<T> {
2    static final int EMPTY = ..., WAITING = ..., BUSY = ...;
3    AtomicStampedReference<T> slot = new AtomicStampedReference<T>(null, 0);
4    public T exchange(T myItem, long timeout, TimeUnit unit)
5      throws TimeoutException {
6      long nanos = unit.toNanos(timeout);
7      long timeBound = System.nanoTime() + nanos;
8      int[] stampHolder = {EMPTY};
9      while (true) {
10       if (System.nanoTime() > timeBound)
11         throw new TimeoutException();
12       T yrItem = slot.get(stampHolder);
13       int stamp = stampHolder[0];
14       switch(stamp) {
15       case EMPTY:
16         if (slot.compareAndSet(yrItem, myItem, EMPTY, WAITING)) {
17           while (System.nanoTime() < timeBound) {
18             yrItem = slot.get(stampHolder);
19             if (stampHolder[0] == BUSY) {
20               slot.set(null, EMPTY);
21               return yrItem;
22             }
23           }
24           if (slot.compareAndSet(myItem, null, WAITING, EMPTY)) {
25             throw new TimeoutException();
26           } else {
27             yrItem = slot.get(stampHolder);
28             slot.set(null, EMPTY);
29             return yrItem;
30           }
31         }
32         break;
33       case WAITING:
34         if (slot.compareAndSet(yrItem, myItem, WAITING, BUSY))
35           return yrItem;
36         break;
37       case BUSY:
38         break;
39       default: // impossible
40         ...
41       }
42     }
43   }
44 }
```

FIGURE 11.6 The LockFreeExchanger<T> class.