```
1  public class WorkSharingThread {
2    Queue[] queue;
3    private static final int THRESHOLD = ...;
4    public WorkSharingThread(Queue[] queue) {
5      this.queue = queue;
6    }
7    public void run() {
8      int me = ThreadID.get();
9      while (true) {
10       RecursiveAction task = queue[me].deq();
11       if (task != null) task.compute();
12       int size = queue[me].size();
13       if (ThreadLocalRandom.current().nextInt(size+1) == size) {
14         int victim = ThreadLocalRandom.current().nextInt(queue.length);
15         int min = (victim <= me) ? victim : me;
16         int max = (victim <= me) ? me : victim;
17         synchronized (queue[min]) {
18           synchronized (queue[max]) {
19             balance(queue[min], queue[max]);
20           }
21         }
22       }
23     }
24   }
25   private void balance(Queue q0, Queue q1) {
26     Queue qMin = (q0.size() < q1.size()) ? q0 : q1;
27     Queue qMax = (q0.size() < q1.size()) ? q1 : q0;
28     int diff = qMax.size() - qMin.size();
29     if (diff > THRESHOLD)
30       while (qMax.size() > qMin.size())
31         qMin.enq(qMax.deq());
32   }
33 }
```

FIGURE 16.17 The WorkSharingThread class: a simplified work-sharing thread pool.