```
1  public class BitonicSort {
2    static final int[][][] bitonicTable = ...;
3    static final int width = ...; // counting network width
4    static final int depth = ...; // counting network depth
5    static final int p = ...;  // number of threads
6    static final int s = ...;  // a power of 2
7    Barrier barrier;
8    ...
9      public <T> void sort(Item<T>[] items) {
10       int i = ThreadID.get();
11       for (int d = 0; d < depth; d++) {
12         barrier.await();
13         for (int j = 0; j < s; j++) {
14           int north = bitonicTable[(i*s)+j][d][0];
15           int south = bitonicTable[(i*s)+j][d][1];
16           if (items[north].key < items[south].key) {
17             Item<T> temp = items[north];
18             items[north] = items[south];
19             items[south] = temp;
20           }
21         }
22       }
23     }
```

FIGURE 12.29 The BitonicSort class.