```
1  public final class LockFreeSkipList<T> {
2    static final int MAX_LEVEL = ...;
3    final Node<T> head = new Node<T>(Integer.MIN_VALUE);
4    final Node<T> tail = new Node<T>(Integer.MAX_VALUE);
5    public LockFreeSkipList() {
6      for (int i = 0; i < head.next.length; i++) {
7        head.next[i]
8          = new AtomicMarkableReference<LockFreeSkipList.Node<T>>(tail, false);
9      }
10   }
11   public static final class Node<T> {
12     final T value; final int key;
13     final AtomicMarkableReference<Node<T>>[] next;
14     private int topLevel;
15     // constructor for sentinel nodes
16     public Node(int key) {
17       value = null; key = key;
18       next = (AtomicMarkableReference<Node<T>>[])
19              new AtomicMarkableReference[MAX_LEVEL + 1];
20       for (int i = 0; i < next.length; i++) {
21         next[i] = new AtomicMarkableReference<Node<T>>(null,false);
22       }
23       topLevel = MAX_LEVEL;
24     }
25     // constructor for ordinary nodes
26     public Node(T x, int height) {
27       value = x;
28       key = x.hashCode();
29       next = (AtomicMarkableReference<Node<T>>[])
30              new AtomicMarkableReference[height + 1];
31       for (int i = 0; i < next.length; i++) {
32         next[i] = new AtomicMarkableReference<Node<T>>(null,false);
33       }
34       topLevel = height;
35     }
36   }
```

FIGURE 14.10 The LockFreeSkipList class: fields and constructor.