```
public final class LazySkipList<T> {
  static final int MAX_LEVEL = ...;
  final Node<T> head = new Node<T>(Integer.MIN_VALUE);
  final Node<T> tail = new Node<T>(Integer.MAX_VALUE);
  public LazySkipList() {
    for (int i = 0; i < head.next.length; i++) {
      head.next[i] = tail;
    }
  }
  ...
  private static final class Node<T> {
    final Lock lock = new ReentrantLock();
    final T item;
    final int key;
    final Node<T>[] next;
    volatile boolean marked = false;
    volatile boolean fullyLinked = false;
    private int topLevel;
    public Node(int key) { // sentinel node constructor
      this.item = null;
      this.key = key;
      next = new Node[MAX_LEVEL + 1];
      topLevel = MAX_LEVEL;
    }
    public Node(T x, int height) {
      item = x;
      key = x.hashCode();
      next = new Node[height + 1];
      topLevel = height;
    }
    public void lock() {
      lock.lock();
    }
    public void unlock() {
      lock.unlock();
    }
  }
}
```

FIGURE 14.4 The LazySkipList class: constructor, fields, and Node class.