```
boolean add(T x) {
  int topLevel = randomLevel();
  Node<T>[] preds = (Node<T>[]) new Node[MAX_LEVEL + 1];
  Node<T>[] succs = (Node<T>[]) new Node[MAX_LEVEL + 1];
  while (true) {
    int lFound = find(x, preds, succs);
    if (lFound != -1) {
      Node<T> nodeFound = succs[lFound];
      if (!nodeFound.marked) {
        while (!nodeFound.fullyLinked) {}
        return false;
      }
      continue;
    }
    int highestLocked = -1;
    try {
      Node<T> pred, succ;
      boolean valid = true;
      for (int level = 0; valid && (level <= topLevel); level++) {
        pred = preds[level];
        succ = succs[level];
        pred.lock.lock();
        highestLocked = level;
        valid = !pred.marked && !succ.marked && pred.next[level]==succ;
      }
      if (!valid) continue;
      Node<T> newNode = new Node(x, topLevel);
      for (int level = 0; level <= topLevel; level++)
        newNode.next[level] = succs[level];
      for (int level = 0; level <= topLevel; level++)
        preds[level].next[level] = newNode;
      newNode.fullyLinked = true; // successful add linearization point
      return true;
    } finally {
      for (int level = 0; level <= highestLocked; level++)
        preds[level].unlock();
    }
  }
}
```

FIGURE 14.6 The LazySkipList class: the add() method.