```
32
33          return minItem;
34      }
```

图 6-56 （续）

6.9 标准库中的优先队列

在 Java 1.5 之前，Java 类库中不存在对优先队列的支持。然而在 Java 1.5 中出现了泛型类 PriorityQueue，在该类中 insert、findMin 和 deleteMin 通过调用 add、element 和 remove 而被表示。PriorityQueue 对象可以通过无参数、一个比较器、或另一个兼容的集合构造出来。

由于优先队列有许多有效的实现方法，因此该类库的设计者们没有选择让 PriorityQueue 成为一个接口。虽然如此，PriorityQueue 在 Java 1.5 中的实现对大多数优先队列的应用还是足够的。

小结

本章介绍了优先队列 ADT 的各种实现方法和用途。标准的二叉堆实现具有简单和快速的优点。它不需要链，只需要常量的附加空间，且有效地支持优先队列的操作。

我们考虑了附加的 merge 操作，开发了三种实现方法，每种都有其独到之处。左式堆是递归威力的完美实例。斜堆则代表缺少平衡原则的一种重要的数据结构。它的分析是有趣的，我们将在第 11 章进行。二项队列指出一个简单的想法如何能够用来达到好的时间界。

我们还看到优先队列的几个用途，从操作系统的工作调度到事件模拟。我们将在第 7、9 和 10 章再次看到它们的应用。

练习

6.1 操作 insert 和 findMin 都能以常数时间实现吗？

6.2 a. 写出一次一个地将 10、12、1、14、6、5、8、15、3、9、7、4、11、13 和 2 插入到一个初始为空的二叉堆中的结果。

b. 写出使用上述相同的输入通过线性时间算法建立一个二叉堆的结果。

6.3 写出对上面练习中的堆执行 3 次 deleteMin 操作的结果。

6.4 N 个元素的完全二叉树用到数组位置 1 到 N。设试图使用数组表示法表示非完全的二叉树。对于下列的情况确定数组必须要多大：

a. 一棵有两个附加层(即它是非常轻微地不平衡)的二叉树

b. 在深度 2 log N 处有一个最深的节点的二叉树

c. 在深度 4.1 log N 处有一个最深的节点的二叉树

d. 最坏情形的二叉树

6.5 通过把被插入项的引用放在位置 0 处重写 BinaryHeap 的 inset 方法。

6.6 在图 6-13 的大的堆中有多少节点？

6.7 a. 证明对于二叉堆，buildHeap 至多在元素间进行 $2N-2$ 次比较。

b. 证明 8 个元素的堆可以通过堆元素间的 8 次比较构成。

** c. 给出一个算法，用 $\frac{13}{8}N + O(\log N)$ 次元素比较构建一个二叉堆。

6.8 证明下列关于堆中的最大项的结论：

a. 它必然在一片树叶上。