将算法的描述直接翻译成代码。除了增加 npl(零路径长)域外，节点类(图 6-25)与二叉树是相同的。左式堆把对根的引用作为其数据成员存储。我们在第 4 章已经看到，当一个元素被插入到一棵空的二叉树时，由根引用的节点将需要改变。我们使用通常的实现 private 递归方法的技巧进行合并。该类的架构也如图 6-25 所示。

```
1   public class LeftistHeap<AnyType extends Comparable<? super AnyType>>
2   {
3       public LeftistHeap( )
4         { root = null; }
5
6       public void merge( LeftistHeap<AnyType> rhs )
7         { /* Figure 6.26 */ }
8       public void insert( AnyType x )
9         { /* Figure 6.29 */ }
10      public AnyType findMin( )
11        { /* See online code */ }
12      public AnyType deleteMin( )
13        { /* Figure 6.30 */ }
14
15      public boolean isEmpty( )
16        { return root == null; }
17      public void makeEmpty( )
18        { root = null; }
19
20      private static class Node<AnyType>
21      {
22              // Constructors
23          Node( AnyType theElement )
24            { this( theElement, null, null ); }
25
26          Node( AnyType theElement, Node<AnyType> lt, Node<AnyType> rt )
27            { element = theElement; left = lt; right = rt; npl = 0; }
28
29          AnyType       element;      // The data in the node
30          Node<AnyType> left;         // Left child
31          Node<AnyType> right;        // Right child
32          int           npl;          // null path length
33      }
34
35      private Node<AnyType> root;    // root
36
37      private Node<AnyType> merge( Node<AnyType> h1, Node<AnyType> h2 )
38        { /* Figure 6.26 */ }
39      private Node<AnyType> merge1( Node<AnyType> h1, Node<AnyType> h2 )
40        { /* Figure 6.27 */ }
41      private void swapChildren( Node<AnyType> t )
42        { /* See online code */ }
43  }
```

图 6-25　左式堆类型声明