```
29        { root = remove( x, root ); }
30      public void printTree( )
31        { /* Figure 4.56 */ }
32
33      private boolean contains( AnyType x, BinaryNode<AnyType> t )
34        { /* Figure 4.18 */ }
35      private BinaryNode<AnyType> findMin( BinaryNode<AnyType> t )
36        { /* Figure 4.20 */ }
37      private BinaryNode<AnyType> findMax( BinaryNode<AnyType> t )
38        { /* Figure 4.20 */ }
39
40      private BinaryNode<AnyType> insert( AnyType x, BinaryNode<AnyType> t )
41        { /* Figure 4.22 */ }
42      private BinaryNode<AnyType> remove( AnyType x, BinaryNode<AnyType> t )
43        { /* Figure 4.25 */ }
44      private void printTree( BinaryNode<AnyType> t )
45        { /* Figure 4.56 */ }
46  }
```

图 4-17 （续）

4.3.1 contains 方法

如果在树 T 中存在含有项 X 的节点，那么这个操作需要返回 true，如果这样的节点不存在则返回 false。树的结构使得这种操作很简单。如果 T 是空集，那么可以就返回 false。否则，如果存储在 T 处的项是 X，那么可以返回 true。否则，我们对树 T 的左子树或右子树进行一次递归调用，这依赖于 X 与存储在 T 中的项的关系。图 4-18 中的代码就是对这种方法的一种实现。

```
1   /**
2    * Internal method to find an item in a subtree.
3    * @param x is item to search for.
4    * @param t the node that roots the subtree.
5    * @return node containing the matched item.
6    */
7   private boolean contains( AnyType x, BinaryNode<AnyType> t )
8   {
9       if( t == null )
10          return false;
11
12      int compareResult = x.compareTo( t.element );
13
14      if( compareResult < 0 )
15          return contains( x, t.left );
16      else if( compareResult > 0 )
17          return contains( x, t.right );
18      else
19          return true;    // Match
20  }
```

图 4-18 二叉查找树的 contains 操作