```
public class BinaryTreeNode {
  Object item;
  /* For the search tree below, replaced line above with "Entry entry;" */
  BinaryTreeNode parent;
  BinaryTreeNode left;
  BinaryTreeNode right;

  public void inorder() {
    if (left != null) {
      left.inorder();
    }
    this.visit();
    if (right != null) {
      right.inorder();
    }
  }
}


public class BinaryTree {
  BinaryTreeNode root;
  int size;

  public Entry find(Object k) {
    BinaryTreeNode node = root;                 // Start at the root.
    while (node != null) {
      int comp = ((Comparable) k).compareTo(node.entry.key());
      if (comp < 0) {                           // Repeatedly compare search
        node = node.left;                       // key k with current node; if
      } else if (comp > 0) {                    // k is smaller, go to the left
        node = node.right;                      // child; if k is larger, go to
      } else {    /* The keys are equal */     // the right child.  Stop when
        return node.entry;                      // we find a match (success;
      }                                         // return the entry) or reach
    }                                           // a null pointer (failure;
    return null;                                // return null).
  }
}
```