view sourceprint?

```
public class BinaryTree {

    Node root;

    public void addNode(int key, String name) {

        // Create a new Node and initialize it

        Node newNode = new Node(key, name);

        // If there is no root this becomes root

        if (root == null) {

            root = newNode;

        } else {

            // Set root as the Node we will start
            // with as we traverse the tree

            Node focusNode = root;

            // Future parent for our new Node

            Node parent;

            while (true) {

                // root is the top parent so we start
                // there

                parent = focusNode;

                // Check if the new node should go on
                // the left side of the parent node

                if (key < focusNode.key) {

                    // Switch focus to the left child

                    focusNode = focusNode.leftChild;

                    // If the left child has no children

                    if (focusNode == null) {

                        // then place the new node on the left of it

                        parent.leftChild = newNode;
                        return; // All Done

                    }

                } else { // If we get here put the node on the right

                    focusNode = focusNode.rightChild;

                    // If the right child has no children

                    if (focusNode == null) {

                        // then place the new node on the right of it

                        parent.rightChild = newNode;
                        return; // All Done

                    }

                }

            }
        }

    }

    // All nodes are visited in ascending order
    // Recursion is used to go to one node and
    // then go to its child nodes and so forth

    public void inOrderTraverseTree(Node focusNode) {

        if (focusNode != null) {

            // Traverse the left node

            inOrderTraverseTree(focusNode.leftChild);

            // Visit the currently focused on node

            System.out.println(focusNode);

            // Traverse the right node

            inOrderTraverseTree(focusNode.rightChild);

        }

    }

    public void preorderTraverseTree(Node focusNode) {

        if (focusNode != null) {

            System.out.println(focusNode);

            preorderTraverseTree(focusNode.leftChild);
            preorderTraverseTree(focusNode.rightChild);

        }

    }

    public void postOrderTraverseTree(Node focusNode) {

        if (focusNode != null) {

            postOrderTraverseTree(focusNode.leftChild);
            postOrderTraverseTree(focusNode.rightChild);

            System.out.println(focusNode);

        }

    }

    public Node findNode(int key) {

        // Start at the top of the tree

        Node focusNode = root;

        // While we haven't found the Node
        // keep looking

        while (focusNode.key != key) {
            // If we should search to the left

            if (key < focusNode.key) {

                // Shift the focus Node to the left child

                focusNode = focusNode.leftChild;

            } else {

                // Shift the focus Node to the right child

                focusNode = focusNode.rightChild;

            }

            // The node wasn't found

            if (focusNode == null)
                return null;

        }

        return focusNode;

    }

public static void main(String[] args) {

        BinaryTree theTree = new BinaryTree();

        theTree.addNode(50, "Boss");

        theTree.addNode(25, "Vice President");

        theTree.addNode(15, "Office Manager");

        theTree.addNode(30, "Secretary");

        theTree.addNode(75, "Sales Manager");

        theTree.addNode(85, "Salesman 1");

        // Different ways to traverse binary trees

        // theTree.inOrderTraverseTree(theTree.root);

        // theTree.preorderTraverseTree(theTree.root);

        // theTree.postOrderTraverseTree(theTree.root);

        // Find the node with key 75

        System.out.println("\nNode with the key 75");

        System.out.println(theTree.findNode(75));

}
}

class Node {

    int key;
    String name;

    Node leftChild;
    Node rightChild;

    Node(int key, String name) {

        this.key = key;
        this.name = name;

    }

    public String toString() {

        return name + " has the key " + key;

        /*
         * return name + " has the key " + key + "\nLeft Child: " + leftChild +
         * "\nRight Child: " + rightChild + "\n";
         */

    }

}
```