

[view sourceprint?](#)

```
001 public class BinaryTree {
002
003     Node root;
004
005     public void addNode(int key, String name) {
006
007         // Create a new Node and initialize it
008
009         Node newNode = new Node(key, name);
010
011         // If there is no root this becomes root
012
013         if (root == null) {
014
015             root = newNode;
016
017         } else {
018
019             // Set root as the Node we will start
020             // with as we traverse the tree
021
022             Node focusNode = root;
023
024             // Future parent for our new Node
025
026             Node parent;
027
028             while (true) {
029
030                 // root is the top parent so we start
031                 // there
032
033                 parent = focusNode;
034
035                 // Check if the new node should go on
036                 // the left side of the parent node
037
038                 if (key < focusNode.key) {
039
040                     // Switch focus to the left child
041
042                     focusNode = focusNode.leftChild;
043
044                     // If the left child has no children
```

```
045
046         if (focusNode == null) {
047
048             // then place the new node on the left of it
049
050             parent.leftChild = newNode;
051             return; // All Done
052
053         }
054
055     } else { // If we get here put the node on the right
056
057         focusNode = focusNode.rightChild;
058
059         // If the right child has no children
060
061         if (focusNode == null) {
062
063             // then place the new node on the right of it
064
065             parent.rightChild = newNode;
066             return; // All Done
067
068         }
069
070     }
071
072 }
073 }
074
075 }
076
077 // All nodes are visited in ascending order
078 // Recursion is used to go to one node and
079 // then go to its child nodes and so forth
080
081 public void inOrderTraverseTree(Node focusNode) {
082
083     if (focusNode != null) {
084
085         // Traverse the left node
086
087         inOrderTraverseTree(focusNode.leftChild);
088
089         // Visit the currently focused on node
090
```

```
091         System.out.println(focusNode);
092
093         // Traverse the right node
094
095         inOrderTraverseTree(focusNode.rightChild);
096
097     }
098
099 }
100
101 public void preorderTraverseTree(Node focusNode) {
102
103     if (focusNode != null) {
104
105         System.out.println(focusNode);
106
107         preorderTraverseTree(focusNode.leftChild);
108         preorderTraverseTree(focusNode.rightChild);
109
110     }
111
112 }
113
114 public void postOrderTraverseTree(Node focusNode) {
115
116     if (focusNode != null) {
117
118         postOrderTraverseTree(focusNode.leftChild);
119         postOrderTraverseTree(focusNode.rightChild);
120
121         System.out.println(focusNode);
122
123     }
124
125 }
126
127 public Node findNode(int key) {
128
129     // Start at the top of the tree
130
131     Node focusNode = root;
132
133     // While we haven't found the Node
134     // keep looking
135
136     while (focusNode.key != key) {
```

```
137
138         // If we should search to the left
139
140         if (key < focusNode.key) {
141
142             // Shift the focus Node to the left child
143
144             focusNode = focusNode.leftChild;
145
146         } else {
147
148             // Shift the focus Node to the right child
149
150             focusNode = focusNode.rightChild;
151
152         }
153
154         // The node wasn't found
155
156         if (focusNode == null)
157             return null;
158
159     }
160
161     return focusNode;
162
163 }
164
165 public static void main(String[] args) {
166
167     BinaryTree theTree = new BinaryTree();
168
169     theTree.addNode(50, "Boss");
170
171     theTree.addNode(25, "Vice President");
172
173     theTree.addNode(15, "Office Manager");
174
175     theTree.addNode(30, "Secretary");
176
177     theTree.addNode(75, "Sales Manager");
178
179     theTree.addNode(85, "Salesman 1");
180
181     // Different ways to traverse binary trees
```

```
182
183     // theTree.inOrderTraverseTree(theTree.root);
184
185     // theTree.preorderTraverseTree(theTree.root);
186
187     // theTree.postOrderTraverseTree(theTree.root);
188
189     // Find the node with key 75
190
191     System.out.println("\nNode with the key 75");
192
193     System.out.println(theTree.findNode(75));
194
195 }
196 }
197
198 class Node {
199
200     int key;
201     String name;
202
203     Node leftChild;
204     Node rightChild;
205
206     Node(int key, String name) {
207
208         this.key = key;
209         this.name = name;
210
211     }
212
213     public String toString() {
214
215         return name + " has the key " + key;
216
217         /*
218          * return name + " has the key " + key + "\nLeft Child: " + leftChild +
219          * "\nRight Child: " + rightChild + "\n";
220          */
221
222     }
223
224 }
```