

```
1 package org.example;
2
3 // Press Shift twice to open the Search Everywhere dialog and type `show whitespaces`,
4 // then press Enter. You can now see whitespace characters in your code.
5
6 class Tree<T> extends Comparable<T> {
7
8     2 usages new *
9     static class TreeNode<T> {
10
11         5 usages
12         private TreeNode<T> root;
13         13 usages new *
14         9 usages
15         T data;
16         11 usages
17         TreeNode<T> left;
18         12 usages
19         TreeNode<T> right;
20
21         3 usages new *
22         public TreeNode(T data) { this.data = data; }
23     }
24 }
```

```
19 public void insertElement(T data) {
20     if (root == null) {
21         root = new TreeNode<>(data);
22         return;
23     }
24     insertNode(root, data);
25 }
26
27 3 usages new *
28 private void insertNode(TreeNode<T> root, T data) {
29     if (root.data.compareTo(data) > 0) {
30         if (root.left == null) {
31             root.left = new TreeNode<>(data);
32             return;
33         }
34         insertNode(root.left, data);
35     } else if (root.data.compareTo(data) < 0) {
36         if (root.right == null) {
37             root.right = new TreeNode<>(data);
38             return;
39         }
40         insertNode(root.right, data);
41     }
42 }
```

```
41 public void printTree() {
42     printTree(root);
43 }
44
45 3 usages new *
46 private void printTree(TreeNode<T> root){
47     if(root != null) {
48         printTree(root.left);
49         System.out.println(root.data);
50         printTree(root.right);
51     }
52 }
53
54 1 usage new *
55 public void deleteNode(T data) {
56     deleteNode(root, data);
57 }
58
59 4 usages new *
60 private void deleteNode(TreeNode<T> root, T data) {
61     if(root == null) {
62         return null;
63     }
64     if(root.data.compareTo(data) > 0) {
65         root.left = deleteNode(root.left, data);
66     } else if (root.data.compareTo(data) < 0) {
67         root.right = deleteNode(root.right, data);
68     } else {
69         if (root.left != null && root.right != null) {
70             TreeNode<T> minElemFromRight = getMinElemFromRight(root.right);
71             root.data=minElemFromRight.data;
72             root.right=deleteNode(root.right, minElemFromRight.data);
73         } else if (root.right != null) {
74             root = root.right;
75         } else if (root.left != null) {
76             root= root.left;
77         } else {
78             root = null;
79         }
80     }
81     return root;
82 }
```

```
54 4 usages new *
55 private void deleteNode(TreeNode<T> root, T data) {
56     if(root == null) {return null;}
57     if(root.data.compareTo(data) > 0) {
58         root.left = deleteNode(root.left, data);
59     } else if (root.data.compareTo(data) < 0) {
60         root.right = deleteNode(root.right, data);
61     } else {
62         if (root.left != null && root.right != null) {
63             TreeNode<T> minElemFromRight = getMinElemFromRight(root.right);
64             root.data=minElemFromRight.data;
65             root.right=deleteNode(root.right, minElemFromRight.data);
66         } else if (root.right != null) {
67             root = root.right;
68         } else if (root.left != null) {
69             root= root.left;
70         } else {
71             root = null;
72         }
73     }
74     return root;
75 }
```

```
76 @ private TreeNode<T> getMinElemFromRight(TreeNode<T> root) {
77     if(root.left == null) {
78         return root;
79     }
80     return getMinElemFromRight(root.left);
81 }
82
83 }
84 new *
85 public class Main {
86     new *
87     public static void main(String[] args) {
88         Tree<Integer> tree = new Tree<>();
89         tree.insertElement( data: 10);
90         tree.insertElement( data: 4);
91         tree.insertElement( data: 16);
92         tree.insertElement( data: 8);
93         tree.insertElement( data: 6);
94         tree.insertElement( data: 1);
95         tree.insertElement( data: 3);
96         tree.insertElement( data: 2);
97         tree.insertElement( data: 12);
98         tree.insertElement( data: 8);
99     }
100 }
```

```
85 public static void main(String[] args) {
86     Tree<Integer> tree = new Tree<>();
87     tree.insertElement( data: 10);
88     tree.insertElement( data: 4);
89     tree.insertElement( data: 16);
90     tree.insertElement( data: 8);
91     tree.insertElement( data: 6);
92     tree.insertElement( data: 1);
93     tree.insertElement( data: 3);
94     tree.insertElement( data: 2);
95     tree.insertElement( data: 12);
96     tree.insertElement( data: 8);
97     tree.insertElement( data: 20);
98     tree.insertElement( data: 30);
99     tree.insertElement( data: 11);
100    tree.insertElement( data: 13);
101    //tree.printTree();
102    tree.deleteNode( data: 10);
103    tree.printTree();
104 }
105 }
```