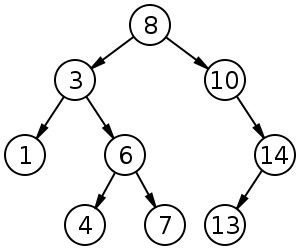
Insertion Tree

*/\*\*  
 \* Insertion Node Algorithm  
 \* Create a Tree with Empty Node  
 \* Check if the data you want to insert is lt; or gt;  
 \* if lt;  
 \* check if Node left child is null  
 \* if null  
 \* create a new code  
 \* if Not null  
 \* iterate recursively to insert Node  
 \*/*public class Tree {  
  
 // STEP 1: Create a Tree with Empty Node  
 static class Node {  
 int data;  
 Node left;  
 Node right;  
  
 Node(int data) {  
 this.data = data;  
 this.left = null;  
 this.right = null;  
 }  
 }  
  
 public void insertNode(Node node, int data) {  
 // If Data is less than Node  
 if (data < node.data) {  
 // If Left Child Node is not null, Iterate and add it at the end  
 if (node.left != null) {  
 insertNode(node.left, data);  
 // If left Child Node is null crate a New Node  
 } else {  
 node.left = new Node(data);  
 }  
 }  
 if (data > node.data) {  
 if (node.right != null) {  
 insertNode(node.right, data);  
 } else {  
 node.right = new Node(data);  
 }  
 }  
 }  
  
 public static void main(String[] args) {  
 // Create a Tree  
 Tree tree = new Tree();  
  
 // create a node with 5 as a value  
 Node node = new Node(5);  
  
 tree.insertNode(node, 4);  
 tree.insertNode(node, 6);  
 System.*out*.println(node);  
 }  
}

Delete Node



***Case 1: Node to be deleted has is a leaf node (no children).***

1. This is very simple implementation. First find the node reference with given value.
2. Set corresponding link of the parent node to null. With this the node to be deleted lost its connectivity and eligible for garbage collection.

***Case 2: Node to be deleted has one child (eight left or right child node).***

1. First find the node reference with given value.
2. Take the reference of the child node and assign its reference to the corresponding link of the parent node. With this the node to be deleted lost its connectivity and eligible for garbage collection.

***Case 3: Node to be deleted has two nodes.***

1. It is little complicated process.
2. First find the node reference with given value.
3. Find the minimum/maximum value of the right/left sub tree.
4. Replace the node value with the minimum/maximum value.
5. Now delete the minimum/maximum value from the nodes right/left sub tree.

*/\*\*  
 \* Insertion Node Algorithm  
 \* Create a Tree with Empty Node  
 \* Check if the data you want to insert is lt; or gt;  
 \* if lt;  
 \* check if Node left child is null  
 \* if null  
 \* create a new code  
 \* if Not null  
 \* iterate recursively to insert Node  
 \*/*public class Tree {  
 // STEP 1: Create a Tree with Empty Node  
 static class Node {  
 int data;  
 Node left;  
 Node right;  
  
 Node(int data) {  
 this.data = data; this.left = null; this.right = null;  
 }  
 }  
 public Node deleteNode(Node node, int data) {  
 if (node == null)  
 return node;  
 /\*  
 8  
 / \  
 3 10  
 / \ \  
 1 6 14  
 / \ /  
 4 7 13  
 \*/  
 // traverse left recursively to find the passed Node i.e, data  
 if (data < node.data) {  
 node.left = deleteNode(node.left, data);  
 // traverse Right recursively to find the passed Node i.e, data  
 } else if (data > node.data) {  
 node.right = deleteNode(node.right, data);  
 // finally when Node is found i.e, Node data matches the above data  
 } else {  
 // case 1: if leaf Node  
 if (node.left == null && node.right == null) {  
 return null;  
 // case 2: if left Node is present  
 } else if (node.left != null && node.right == null) {  
 return node.left;  
 // case 3: if right node is present  
 } else if (node.left == null && node.right != null) {  
 return node.right;  
 // case 4: If both child nodes are present  
 } else {  
 Integer minValue = minValue(node.right);  
 node.data = minValue;  
 node.right = deleteNode(node.right, minValue);  
 System.*out*.println("deleting "+data);  
 }  
 }  
 return node;  
 }  
  
 private Integer minValue(Node node) {  
 if(node.left != null) {  
 return minValue(node.left);  
 }  
 return node.data;  
 }  
 public static void main(String[] args) {  
 // Create a Tree  
 Tree tree = new Tree();  
  
 // create a node with 5 as a value  
 Node node = new Node(8);  
  
 tree.insertNode(node, 10);  
 tree.insertNode(node, 14);  
 tree.insertNode(node, 3);  
 tree.insertNode(node, 6);  
 tree.insertNode(node, 7);  
 tree.insertNode(node, 1);  
 tree.insertNode(node, 4);  
 tree.insertNode(node, 13);  
 System.*out*.println(node);  
  
 tree.deleteNode(node, 8);  
 System.*out*.println(node);  
 }  
}