//Build a Tree from its Preorder traversal

public class BinaryTreesYT {

   static class Node {

       int data;

       Node left;

       Node right;

       Node(int data) {

           this.data = data;

           this.left = null;

           this.right = null;

       }

   }

   static class BinaryTree {

       static int idx = -1;

       public static Node buildTree(int nodes[]) {

           idx++;

           if(nodes[idx] == -1) {

               return null;

           }

           Node newNode = new Node(nodes[idx]);

           newNode.left = buildTree(nodes);

           newNode.right = buildTree(nodes);

           return newNode;

       }

   }

   public static void main(String args[]) {

       int nodes[] = {1, 2, 4, -1, -1, 5, -1, -1, 3, -1, 6, -1, -1};

       BinaryTree tree = new BinaryTree();

       Node root = tree.buildTree(nodes);

       System.out.println(root.data);

   }

}

1. **Tree Traversals**
2. Preorder

public static void preorder(Node root) {

       if(root == null) {

           System.out.print(-1+" ");

           return;

       }

       System.out.print(root.data+" ");

       preorder(root.left);

       preorder(root.right);

   }

1. Inorder

public static void inorder(Node root) {

       if(root == null) {

           System.out.print(-1+" ");

           return;

       }

       inorder(root.left);

       System.out.print(root.data+" ");

       inorder(root.right);

   }

1. Postorder

public static void postorder(Node root) {

       if(root == null) {

           System.out.print(-1+" ");

           return;

       }

       postorder(root.left);

       postorder(root.right);

       System.out.print(root.data+" ");

   }

**Bst:**

