**package** tree;  
  
**import** java.security.PublicKey;  
**import** java.util.LinkedList;  
**import** java.util.Queue;  
**import** java.util.Stack;  
  
**class** BTNode {  
 **private int val**;  
 **private** BTNode **left**;  
 **private** BTNode **right**;  
  
 **public** BTNode(**int** val) {  
 **this**.**val** = val;  
 }  
  
 **public int** getVal() {  
 **return val**;  
 }  
  
 **public void** setVal(**int** val) {  
 **this**.**val** = val;  
 }  
  
 **public** BTNode getLeft() {  
 **return left**;  
 }  
  
 **public void** setLeft(BTNode left) {  
 **this**.**left** = left;  
 }  
  
 **public** BTNode getRight() {  
 **return right**;  
 }  
  
 **public void** setRight(BTNode right) {  
 **this**.**right** = right;  
 }  
  
 *//总结点个数* **public int** getCountNode(BTNode node) {  
 **if** (node == **null**) {  
 **return** 0;  
 }  
 **return** getCountNode(node.**left**) + getCountNode(node.**right**) + 1;  
 }  
  
 *//叶子结点个数* **public int** getLeafNodeCount(BTNode node) {  
 **if** (node == **null**) {  
 **return** 0;  
 }  
 **if** (node.**left** == **null** && node.**right** == **null**) {  
 **return** 1;  
 }  
 **return** getLeafNodeCount(node.**left**) + getLeafNodeCount(node.**right**);  
 }  
  
 *//第k层节点个数* **public int** getNodeCountFromK(BTNode node, **int** k) {  
 **if** (node == **null** || k < 1) {  
 **return** 0;  
 }  
 **if** (k == 1) {  
 **return** 1;  
 }  
 **return** getNodeCountFromK(node.**left**, k - 1) + getNodeCountFromK(node.**right**, k - 1);  
 }  
  
 *//前序遍历* **public void** preOrder(BTNode node) {  
 **if** (node == **null**) {  
 **return**;  
 }  
 System.***out***.println(node.**val** + **" "**);  
 **if** (node.**left** != **null**) {  
 node.**left**.preOrder(node.**left**);  
 }  
 **if** (node.**right** != **null**) {  
 node.**right**.preOrder(node.**right**);  
 }  
 }  
  
 *//层序遍历* **public void** cengOrder() {  
 System.***out***.println(**this**.**val**);  
 **if** (**this**.**left** != **null**) {  
 System.***out***.println(**this**.**left**.**val**);  
 }  
 **if** (**this**.**right** != **null**) {  
 System.***out***.println(**this**.**right**.**val**);  
 }  
 **if** (**this**.**left** != **null**) {  
 **if** (**this**.**left**.**left** != **null**) {  
 **this**.**left**.**left**.cengOrder();  
 }  
 **if** (**this**.**left**.**right** != **null**) {  
 **this**.**left**.**right**.cengOrder();  
 }  
 }  
 **if** (**this**.**right** != **null**) {  
 **if** (**this**.**right**.**left** != **null**) {  
 **this**.**right**.**left**.cengOrder();  
 }  
 **if** (**this**.**right**.**right** != **null**) {  
 **this**.**right**.**right**.cengOrder();  
 }  
 }  
 }  
 *//借助栈的层序遍历* **public void** preOrderNor(){  
 Stack<BTNode> stack=**new** Stack<>();  
 stack.push(**this**);  
 **while** (!stack.isEmpty()){  
 BTNode cur=stack.peek();  
 System.***out***.println(cur.**val**+**" "**);  
 stack.pop();  
 **if** (cur.**right**!=**null**){  
 stack.push(cur.**right**);  
 }  
 **if** (cur.**left**!=**null**){  
 stack.push(cur.**left**);  
 }  
 }  
 }  
 *//判断是否为完全二叉树* **public boolean** isCompleteTree(){  
 *//1.按照层序遍历的方式找到第一个不饱和的节点（叶子，只有一个孩子节点）* Queue<BTNode> queue=**new** LinkedList<>();  
 queue.offer(**this**);  
 **boolean** isLeafOrLeft=**false**;  
 **while** (!queue.isEmpty()){  
 BTNode cur=queue.poll();  
 **if** (isLeafOrLeft){  
 *//从第一个不饱和节点之后，所有的节点不能有孩子节点* **if** (cur.**left**!=**null**||cur.**right**!=**null**){  
 **return false**;  
 }  
 }  
 **else** {  
 *//cur节点的左右节点都存在* **if** (cur.**left**!=**null**&&cur.**right**!=**null**){  
 queue.offer(cur.**left**);  
 queue.offer(cur.**right**);  
 }  
 **else if** (cur.**left**!=**null**){  
 *//cur只有左孩子* queue.offer(cur.**left**);  
 isLeafOrLeft=**true**;  
 }  
 **else if** (cur.**right**!=**null**){  
 *//cur只有右孩子* **return false**;  
 }  
 **else** {  
 *//cur是叶子节点* isLeafOrLeft=**true**;  
 }  
 }  
 }  
 **return true**;  
 }  
  
 *//寻找目标节点* **public** BTNode findNode(**int** keyVal) {  
 **if** (**this**.**val** == keyVal) {  
 **return this**;  
 }  
 BTNode node = **null**;  
 **if** (**this**.**left** != **null**) {  
 node = **this**.**left**.findNode(keyVal);  
 }  
 **if** (node != **null**) {  
 **return** node;  
 }  
 **if** (**this**.**right** != **null**) {  
 node = **this**.**right**.findNode(keyVal);  
 }  
 **return** node;  
 }  
  
 *//求二叉树最大深度* **public int** getTreeMaxHeight() {  
 **int** leftHeight = 0;  
 **int** rightHeight = 0;  
 **if** (**this**.**left** != **null**) {  
 leftHeight = **this**.**left**.getTreeMaxHeight();  
 }  
 **if** (**this**.**right** != **null**) {  
 rightHeight = **this**.**right**.getTreeMaxHeight();  
 }  
 **return** leftHeight > rightHeight ? leftHeight + 1 : rightHeight + 1;  
 }  
  
 *//判断是否为平衡二叉树* **public boolean** isBalanceTree() {  
 **int** leftHeight = 0;  
 **int** rightHeight = 0;  
 **if** (**this**.**left** != **null**) {  
 leftHeight = **this**.**left**.getTreeMaxHeight();  
 }  
 **if** (**this**.**right** != **null**) {  
 rightHeight = **this**.**right**.getTreeMaxHeight();  
 }  
 **if** (Math.*abs*(leftHeight - rightHeight) > 1) {  
 **return false**;  
 }  
 **return true**;  
 }  
}  
  
**class** BTTree {  
 **private** BTNode **root**;  
  
 **public void** setRoot(BTNode root) {  
 **this**.**root** = root;  
 }  
  
 *//调用总结点个数方法* **public int** getCountNode() {  
 **if** (**root** == **null**) {  
 **return** 0;  
 } **else** {  
 **return this**.**root**.getCountNode(**root**);  
 }  
 }  
  
 *//调用叶子结点个数* **public int** getLeafNodeCount() {  
 **if** (**root** == **null**) {  
 **return** 0;  
 } **else** {  
 **return this**.**root**.getLeafNodeCount(**root**);  
 }  
 }  
  
 *//调用前序遍历* **public void** preOrder() {  
 **if** (**root** == **null**) {  
 System.***out***.println(**"二叉树为空，无法遍历"**);  
 } **else** {  
 **root**.preOrder(**root**);  
 }  
 }  
  
 *//调用层序遍历* **public void** cengOrder() {  
 **if** (**this**.**root** == **null**) {  
 System.***out***.println(**"二叉树为空，无法进行层序遍历"**);  
 } **else** {  
 **this**.**root**.cengOrder();  
 }  
 }  
  
 *//调用第k层节点个数* **public int** getNodeCountFromK() {  
 **if** (**root** == **null**) {  
 **return** 0;  
 } **else** {  
 **return this**.**root**.getNodeCountFromK(**root**, 3);  
 }  
 }  
  
 *//调用寻找目标节点* **public** BTNode findNode(**int** keyVal) {  
 **if** (**this**.**root** == **null**) {  
 **return null**;  
 }  
 **return this**.**root**.findNode(keyVal);  
 }  
  
 *//调用求深度* **public int** getTreeMaxHeight() {  
 **if** (**this**.**root** == **null**) {  
 **return** 0;  
 }  
 **return this**.**root**.getTreeMaxHeight();  
 }  
  
 *//调用判断是否为平衡二叉树* **public boolean** isBalanceTree() {  
 **if** (**this**.**root** == **null**) {  
 **return true**;  
 } **else** {  
 **return this**.**root**.isBalanceTree();  
 }  
 }  
 *//调用借助栈的层序遍历* **public void** preOrderNor(){  
 **if** (**root**==**null**){  
 System.***out***.println(**"二叉树为空"**);  
 }**else** {  
 **root**.preOrderNor();  
 }  
 }  
 *//调用判断是否为完全二叉树* **public boolean** isCompleteTree(){  
 **if** (**root**==**null**){  
 **return true**;  
 }  
 **return root**.isCompleteTree();  
 }  
}  
  
**public class** BTTreeDemo {  
 **public static void** main(String[] args) {  
 BTTree btTree = **new** BTTree();  
 BTNode node1 = **new** BTNode(1);  
 BTNode node2 = **new** BTNode(2);  
 BTNode node3 = **new** BTNode(3);  
 BTNode node4 = **new** BTNode(4);  
 BTNode node5 = **new** BTNode(5);  
 BTNode node6 = **new** BTNode(6);  
 BTNode node7 = **new** BTNode(7);  
 BTNode node8 = **new** BTNode(8);  
 BTNode node9 = **new** BTNode(9);  
 BTNode node10 = **new** BTNode(10);  
 btTree.setRoot(node1);  
 node1.setLeft(node2);  
 node1.setRight(node3);  
*// node2.setLeft(node4);* node3.setLeft(node5);  
 node3.setRight(node6);  
 node4.setLeft(node7);  
 node4.setRight(node8);  
 node6.setLeft(node9);  
 node6.setRight(node10);  
 **int** count = btTree.getCountNode();  
 System.***out***.println(count);  
 **int** count2 = btTree.getLeafNodeCount();  
 System.***out***.println(count2);  
 System.***out***.print(**"前序遍历结果为"**);  
 btTree.preOrder();  
 System.***out***.println(**"第k层节点个数"**);  
 **int** count3 = btTree.getNodeCountFromK();  
 System.***out***.println(count3);  
 System.***out***.println(**"查询目标节点"**);  
 BTNode node = btTree.findNode(6);  
 **if** (node == **null**) {  
 System.***out***.println(**"没有此节点"**);  
 } **else** {  
 System.***out***.println(node.getVal());  
 }  
 System.***out***.println(**"层序遍历结果为："**);  
 btTree.cengOrder();  
 **int** count4 = btTree.getTreeMaxHeight();  
 System.***out***.println(**"树的高度："** + count4);  
 System.***out***.println(**"是否为平衡二叉树"** + btTree.isBalanceTree());  
 System.***out***.println(**"借助栈"**);  
 btTree.preOrderNor();  
 **boolean** flag=btTree.isCompleteTree();  
 System.***out***.println(**"是否为完全二叉树："**+flag);  
 }  
}