```
class AvlTree{
    private Node root = null;
    int count = 0;
    private int i = 0;
    AvlTree(Comparable myVal){
        root = new Node(myVal);
    }
    AvlTree(){
    }
        private int height(Node t) {
        return t == null ? -1 : t.height;
    }
    public void insert(Comparable x){
        root = insert(x, root);
        count++;
    }
    private Node insert(Comparable x, Node t){
        if (t == null)
            t = new Node(x, null, null );
        else if(x.compareTo(t.data) < 0)
        {
            t.left = insert(x, t.left);
            if(height(t.left) - height(t.right) == 2 )
                if ( x.compareTo(t.left.data) < 0 )
                    t = rotateWithLeftChild( t );
                else
                    t = doubleWithLeftChild( t );

        }
        else if ( x.compareTo( t.data ) > 0 ){
            t.right = insert(x, t.right);
            if (height(t.right) - height(t.left) == 2)
                if ( x.compareTo(t.right.data ) > 0 )
                    t = rotateWithRightChild( t );
                else
                    t = doubleWithRightChild( t );


        }
        else
            ;
        t.height = max(height(t.left) , height(t.right)) + 1;
        return t;
    }
    public int max(int x, int y){
        return x > y ? x : y;
    }
    private Node rotateWithLeftChild( Node k2){
        Node k1 = k2.left;
        k2.left = k1.right;
        k1.right = k2;
        k2.height = max(height(k2.left),height(k2.right)) + 1;
        k1.height = max(height(k1.left),k2.height) + 1;
        return k1;
    }
    private Node doubleWithLeftChild(Node k3){
        k3.left = rotateWithRightChild(k3.left);
        return rotateWithLeftChild(k3);
    }
    private Node rotateWithRightChild( Node k1){
        Node k2 = k1.right;
        k1.right = k2.left;
        k2.left = k1;
        k1.height = max(height(k1.left),height(k1.right)) + 1;
        k2.height = max(k1.height,height(k2.right)) + 1;
        return k2;
    }
    private Node doubleWithRightChild(Node k3){
        k3.right = rotateWithLeftChild(k3.right);
        return rotateWithRightChild(k3);
    }
    public void printTree(){
        inPrint(root);
```

```
    }
    private void inPrint(Node t){
        if (t != null){
            inPrint(t.left);
            System.out.print (t.data + " ");
            inPrint(t.right);
        }
        if ((i%10) != 0)
            i++;
        else {
            System.out.println();
            i++;
        }
    }
    public int numberOfNodes(){
        return count;
    }
    public int getHeight(){
        return height(root);
    }

}
```