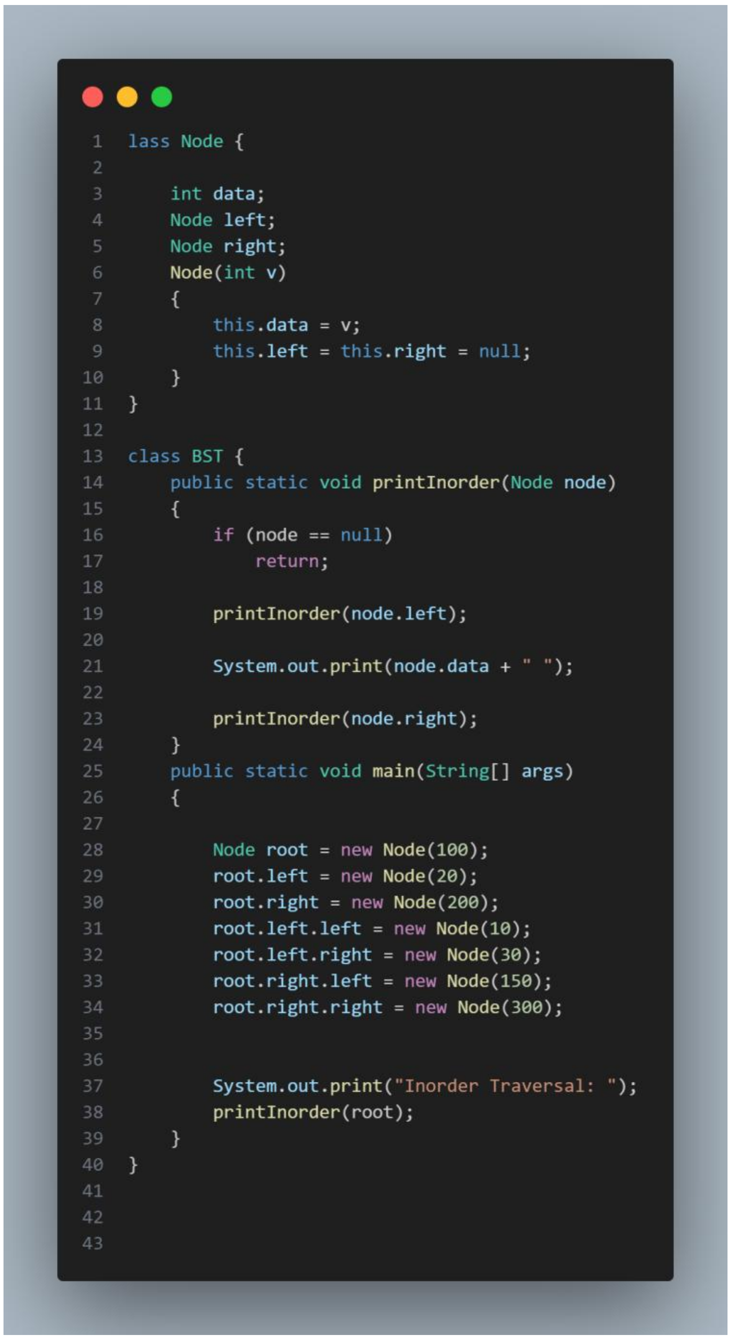
# DSA Progarms

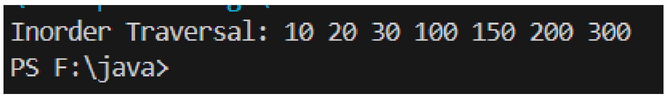
## 25/11/2024

## 1. Valid BST:



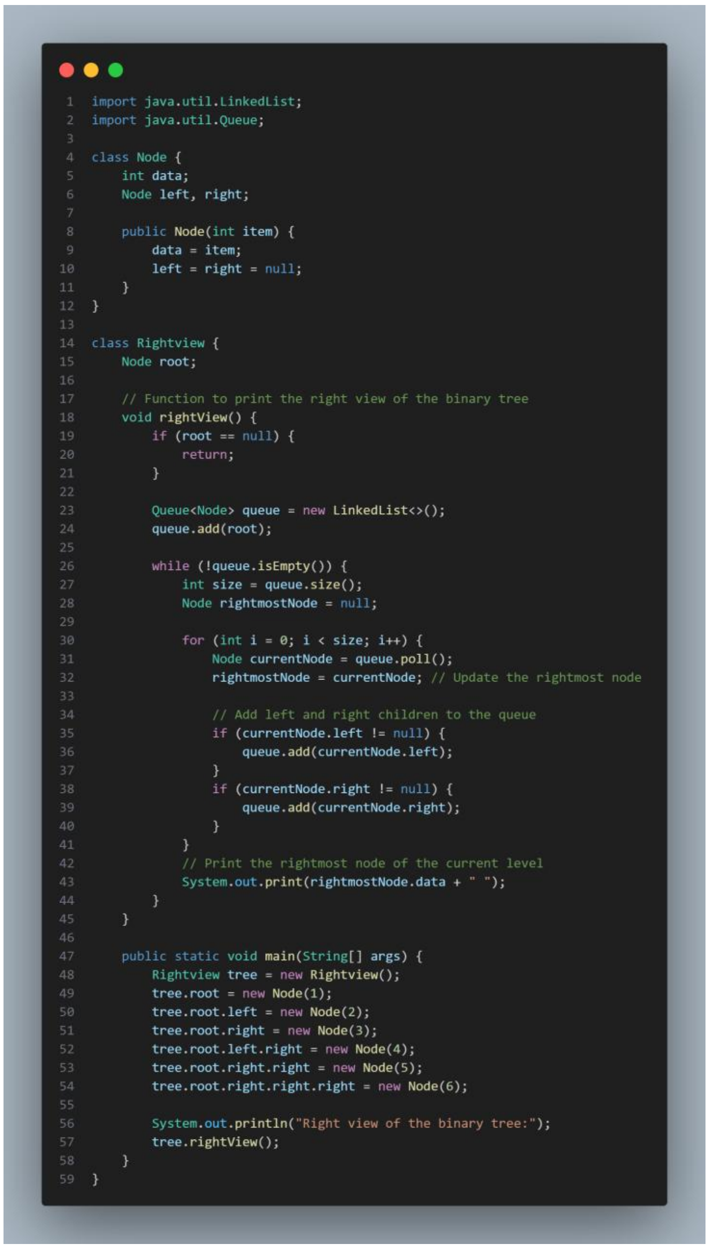
```java
lass Node {

    int data;
    Node left;
    Node right;
    Node(int v)
    {
        this.data = v;
        this.left = this.right = null;
    }
}

class BST {
    public static void printInorder(Node node)
    {
        if (node == null)
            return;

        printInorder(node.left);

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

        printInorder(node.right);
    }
    public static void main(String[] args)
    {

        Node root = new Node(100);
        root.left = new Node(20);
        root.right = new Node(200);
        root.left.left = new Node(10);
        root.left.right = new Node(30);
        root.right.left = new Node(150);
        root.right.right = new Node(300);


        System.out.print("Inorder Traversal: ");
        printInorder(root);
    }
}
```

## Output:



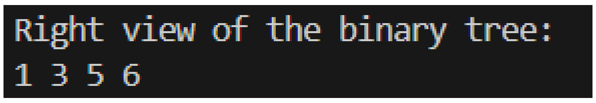
```
Inorder Traversal: 10 20 30 100 150 200 300
PS F:\java>
```

## 2. Right view :



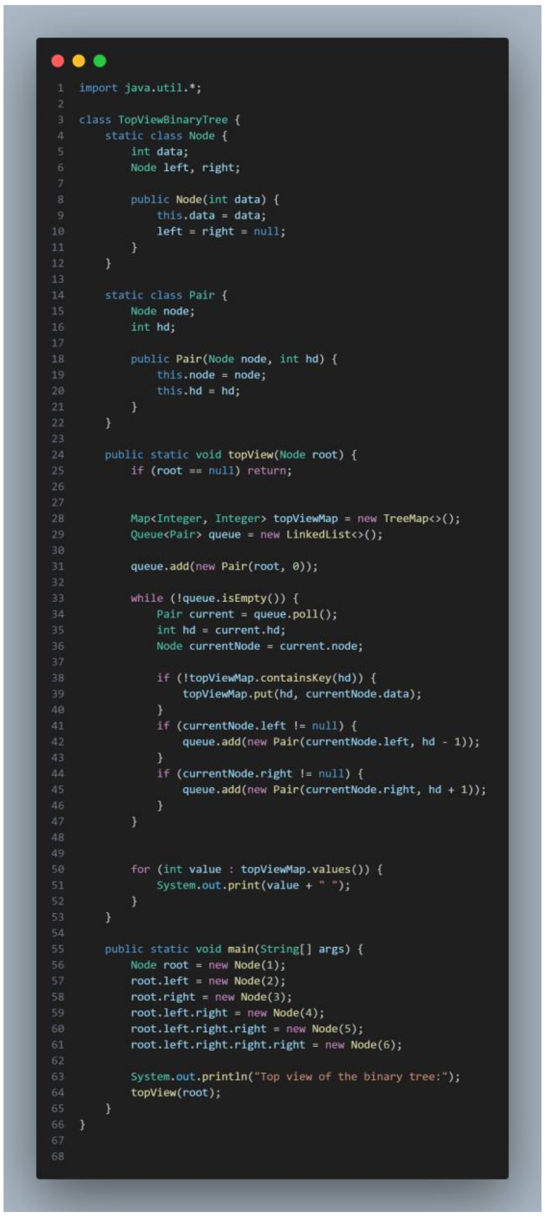
```java
import java.util.LinkedList;
import java.util.Queue;

class Node {
    int data;
    Node left, right;

    public Node(int item) {
        data = item;
        left = right = null;
    }
}

class Rightview {
    Node root;

    // Function to print the right view of the binary tree
    void rightView() {
        if (root == null) {
            return;
        }

        Queue<Node> queue = new LinkedList<>();
        queue.add(root);

        while (!queue.isEmpty()) {
            int size = queue.size();
            Node rightmostNode = null;

            for (int i = 0; i < size; i++) {
                Node currentNode = queue.poll();
                rightmostNode = currentNode; // Update the rightmost node

                // Add left and right children to the queue
                if (currentNode.left != null) {
                    queue.add(currentNode.left);
                }
                if (currentNode.right != null) {
                    queue.add(currentNode.right);
                }
            }
            // Print the rightmost node of the current level
            System.out.print(rightmostNode.data + " ");
        }
    }

    public static void main(String[] args) {
        Rightview tree = new Rightview();
        tree.root = new Node(1);
        tree.root.left = new Node(2);
        tree.root.right = new Node(3);
        tree.root.left.right = new Node(4);
        tree.root.right.right = new Node(5);
        tree.root.right.right.right = new Node(6);

        System.out.println("Right view of the binary tree:");
        tree.rightView();
    }
}
```

## output:



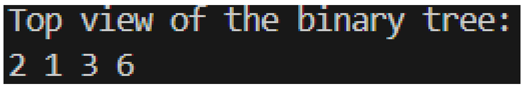
```
Right view of the binary tree:
1 3 5 6
```

## 3. Top view:



```java
import java.util.*;

class TopViewBinaryTree {
    static class Node {
        int data;
        Node left, right;

        public Node(int data) {
            this.data = data;
            left = right = null;
        }
    }

    static class Pair {
        Node node;
        int hd;

        public Pair(Node node, int hd) {
            this.node = node;
            this.hd = hd;
        }
    }

    public static void topView(Node root) {
        if (root == null) return;


        Map<Integer, Integer> topViewMap = new TreeMap<>();
        Queue<Pair> queue = new LinkedList<>();

        queue.add(new Pair(root, 0));

        while (!queue.isEmpty()) {
            Pair current = queue.poll();
            int hd = current.hd;
            Node currentNode = current.node;

            if (!topViewMap.containsKey(hd)) {
                topViewMap.put(hd, currentNode.data);
            }
            if (currentNode.left != null) {
                queue.add(new Pair(currentNode.left, hd - 1));
            }
            if (currentNode.right != null) {
                queue.add(new Pair(currentNode.right, hd + 1));
            }
        }


        for (int value : topViewMap.values()) {
            System.out.print(value + " ");
        }
    }

    public static void main(String[] args) {
        Node root = new Node(1);
        root.left = new Node(2);
        root.right = new Node(3);
        root.left.right = new Node(4);
        root.left.right.right = new Node(5);
        root.left.right.right.right = new Node(6);

        System.out.println("Top view of the binary tree:");
        topView(root);
    }
}
```

## Output:



```
Top view of the binary tree:
2 1 3 6
```