```
package datastructure.Tree;

public class BreadthFirstSearch {
    Node root;

    public BreadthFirstSearch() {
        root = null;
    }
    /*
          1
       /     \
      2       3
     /  \
    4   5            */

    void printLevelOrder() {
        int h = getTotalLevel(root);
        int i;
        for (i = 0; i < h; i++)
            printGivenLevel(root, i);
    }

    public int getTotalLevel(Node node) {
        if (node == null) {
            return 0;
        } else {
            return 1 +
                Math.max(getTotalLevel(node.left), getTotalLevel(node.right));
    }
}


/*
      1
   /     \
  2       3
 /  \
4   5            */

void printGivenLevel(Node root, int level) {
    if (root == null)
        return;
    if (level == 0)
        System.out.print(root.key + " ");
    else if (level > 0) {
        printGivenLevel(root.left, level - 1);
        printGivenLevel(root.right, level - 1);
    }
}

/*
      1
   /     \
  2       3
 /  \    /  \
4   5   6    7  */
```

```
void preOrder() {
    preOrderHelper(root);
}
// 1  2  4  5  3  6  7

void preOrderHelper(Node root) {
    if (root != null) {
        System.out.print(root.key + " ");
        preOrderHelper(root.left);
        preOrderHelper(root.right);
    }
}
/*
      1
   /     \
  2       3
 /  \    /  \
4   5   6    7  */

void inorder() {
    inorderHelper(root);
}

void inorderHelper(Node root) {
    if (root != null) {
        inorderHelper(root.left);
        System.out.print(root.key + " ");
        inorderHelper(root.right);
    }
}

/*
      1
   /     \
  2       3
 /  \    /  \
4   5   6    7  */

void postOrder() {
    postOrderHelper(root);
}

void postOrderHelper(Node root) {
    if (root != null) {
        postOrderHelper(root.left);
        postOrderHelper(root.right);
        System.out.print(root.key + " ");
    }
}
```

```
    public static void main(String[] args) {
        /*
              1
            /     \
           2       3
         /  \    /  \
        4   5   6    7  */
        BreadthFirstSearch tree = new BreadthFirstSearch();
        tree.root = new Node(1);
        tree.root.left = new Node(2);
        tree.root.right = new Node(3);
        tree.root.left.left = new Node(4);
        tree.root.left.right = new Node(5);
        tree.root.right.left = new Node(6);
        tree.root.right.right = new Node(7);



        System.out.println("DFS Pre-Order traversal of tree is(Root --> Left -->
Right) ");
        tree.preOrder();

        System.out.println();

        System.out.println("DFS In-Order traversal of tree is (Left --> Root -->
Right) ");
        tree.inorder();
        System.out.println();


        System.out.println("DFS Post-Order traversal of tree is (Left --> Right -->
Root) ");

        tree.postOrder();

    }

}
```