✅ **1. Basic Tree Node Class**

java

CopyEdit

```
class TreeNode {
    int data;
    TreeNode left;
    TreeNode right;

    // Constructor
    TreeNode(int data) {
        this.data = data;
        this.left = null;
        this.right = null;
    }
}
```

---

✅ **2. Binary Tree with Insert and Print**

java

CopyEdit

```
public class BinaryTree {
    TreeNode root;

    // Insert data into the tree (for example: binary search tree logic)
    public TreeNode insert(TreeNode node, int data) {
        if (node == null) {
            return new TreeNode(data);
        }

        if (data < node.data) {
            node.left = insert(node.left, data);
        } else {
            node.right = insert(node.right, data);
        }

        return node;
    }

    // Inorder traversal (Left, Root, Right)
    public void inorder(TreeNode node) {
        if (node != null) {
            inorder(node.left);
            System.out.print(node.data + " ");
            inorder(node.right);
        }
    }

    public static void main(String[] args) {
        BinaryTree tree = new BinaryTree();
        tree.root = tree.insert(tree.root, 50);
        tree.insert(tree.root, 30);
        tree.insert(tree.root, 70);
        tree.insert(tree.root, 20);
        tree.insert(tree.root, 40);
        tree.insert(tree.root, 60);
        tree.insert(tree.root, 80);

        System.out.println("Inorder traversal:");
        tree.inorder(tree.root);
    }
```

```
import java.util.*;

public class Graph {
    private int vertices;
    private List<List<Integer>> adjList;

    // Constructor
    Graph(int vertices) {
        this.vertices = vertices;
        adjList = new ArrayList<>();

        // Initialize each adjacency list
        for (int i = 0; i < vertices; i++) {
            adjList.add(new ArrayList<>());
        }
    }

    // Add edge (undirected by default)
    public void addEdge(int u, int v) {
        adjList.get(u).add(v);
        adjList.get(v).add(u); // remove this line for directed graph
    }

    // Print the graph
    public void printGraph() {
        for (int i = 0; i < vertices; i++) {
            System.out.print(i + " -> ");
            for (int neighbor : adjList.get(i)) {
                System.out.print(neighbor + " ");
            }
            System.out.println();
        }
    }

    public static void main(String[] args) {
        Graph g = new Graph(5);
        g.addEdge(0, 1);
        g.addEdge(0, 4);
        g.addEdge(1, 2);
        g.addEdge(1, 3);
        g.addEdge(1, 4);
        g.addEdge(2, 3);
        g.addEdge(3, 4);

        g.printGraph();
    }
}
```