**Write a Java program to implement Binary Tree using the Linked List.**

import java.util.LinkedList;

import java.util.Queue;

public class BinaryTree {

public static class Node{

int data;

Node left;

Node right;

public Node(int data){

this.data = data;

this.left = null;

this.right = null;

}

}

public Node root;

public BinaryTree(){

root = null;

}

public void insertNode(int data) {

Node newNode = new Node(data);

if(root == null){

root = newNode;

return;

}

else {

Queue<Node> queue = new LinkedList<Node>();

queue.add(root);

while(true) {

Node node = queue.remove();

if(node.left != null && node.right != null) {

queue.add(node.left);

queue.add(node.right);

}

else {

if(node.left == null) {

node.left = newNode;

queue.add(node.left);

}

else {

node.right = newNode;

queue.add(node.right);

}

break;

}

}

}

}

public void inorderTraversal(Node node) {

if(root == null){

System.out.println("Tree is empty");

return;

}

else {

if(node.left!= null)

inorderTraversal(node.left);

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

if(node.right!= null)

inorderTraversal(node.right);

}

}

public static void main(String[] args) {

BinaryTree bt = new BinaryTree();

bt.insertNode(1);

System.out.println("Binary tree after insertion");

bt.inorderTraversal(bt.root);

bt.insertNode(2);

bt.insertNode(3);

System.out.println("\nBinary tree after insertion");

bt.inorderTraversal(bt.root);

bt.insertNode(4);

bt.insertNode(5);

System.out.println("\nBinary tree after insertion");

bt.inorderTraversal(bt.root);

bt.insertNode(6);

bt.insertNode(7);

System.out.println("\nBinary tree after insertion");

bt.inorderTraversal(bt.root);

}

}

**Output:**

