2. Write a Java Program to traverse a binary tree using PreOrder traversal without recursion

import java.util.Stack;

public class BinaryTreePreOrder {

public static class TreeNode

{

int data;

TreeNode left;

TreeNode right;

TreeNode(int data)

{

this.data=data;

}

}

public void preorder(TreeNode root) {

if(root != null) {

System.out.printf("%d ",root.data);

preorder(root.left);

preorder(root.right);

}

}

public void preorderIter(TreeNode root) {

if(root == null)

return;

Stack<TreeNode> stack = new Stack<TreeNode>();

stack.push(root);

while(!stack.empty()){

TreeNode n = stack.pop();

System.out.printf("%d ",n.data);

if(n.right != null){

stack.push(n.right);

}

if(n.left != null){

stack.push(n.left);

}

}

}

public static void main(String[] args)

{

BinaryTreePreOrder bi=new BinaryTreePreOrder();

// Creating a binary tree

TreeNode rootNode=createBinaryTree();

System.out.println("Using Recursive solution:");

bi.preorder(rootNode);

System.out.println();

}

public static TreeNode createBinaryTree()

{

TreeNode rootNode =new TreeNode(40);

TreeNode node20=new TreeNode(20);

TreeNode node10=new TreeNode(10);

TreeNode node30=new TreeNode(30);

TreeNode node60=new TreeNode(60);

TreeNode node50=new TreeNode(50);

TreeNode node70=new TreeNode(70);

rootNode.left=node20;

rootNode.right=node60;

node20.left=node10;

node20.right=node30;

node60.left=node50;

node60.right=node70;

return rootNode;

}

}

**Output:**

