CSE2001 (Data Structures & Algorithms) Lab-5

KHAN MOHD OWAIS RAZA
20BCD7138

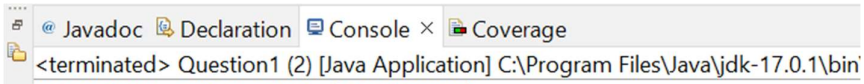
Q.1] Write a program to implement representation of binary tree with linked list.

Java code for implementing binary tree with linked list (with tree traversals) :

```
/**
 Name: KHAN MOHD OWAIS RAZA
 ID : 20BCD7138
 Course: Data Structures & Algorithm
 Code: CSE2001
 Slot: L19+L20
**/
/* Lab-5 (08-10-2022)*/
/* Java code to implement binary tree using linked list */
package CSE2001_Lab5_20BCD7138;
import java.util.Scanner;
class Binary_Tree_Node{
Binary_Tree_Node left, right;
int data;
public Binary_Tree_Node(){
left = null;
right = null;
data = 0;
}
public Binary_Tree_Node(int n){
left = null;
right = null;
data = n;
}
public void setLeft(Binary_Tree_Node n){
left = n;
}
public void setRight(Binary_Tree_Node n){
right = n;
}
public Binary_Tree_Node getLeft(){
return left;
}
public Binary_Tree_Node getRight(){
return right;
}
public void setData(int d){
data = d;
}
public int getData(){
return data;
}}
class BT{
private Binary_Tree_Node root;
public BT(){
```

```
root = null;
}
public boolean isEmpty(){
return root == null;
}
public void insert(int data){
root = insert(root, data);
}
private Binary_Tree_Node insert(Binary_Tree_Node node, int data){
if (node == null) node = new Binary_Tree_Node(data);
else{
if (node.getRight() == null) node.right = insert(node.right, data);
else node.left = insert(node.left, data);
}
return node;
}
public int countNodes(){
return countNodes(root);
}
private int countNodes(Binary_Tree_Node r){
if (r == null) return 0;
else{
int l = 1;
l += countNodes(r.getLeft());
l += countNodes(r.getRight());
return l;
}}
public boolean search(int val){
return search(root, val);
}
private boolean search(Binary_Tree_Node r, int val){
if (r.getData() == val) return true;
if (r.getLeft() != null)
if (search(r.getLeft(), val)) return true;
if (r.getRight() != null)
if (search(r.getRight(), val)) return true;
return false;
}
public void inorder(){
inorder(root);
}
private void inorder(Binary_Tree_Node r){
if (r != null){
inorder(r.getLeft());
System.out.print(r.getData() +" ");
inorder(r.getRight());
}}
public void preorder(){
preorder(root);
}
private void preorder(Binary_Tree_Node r){
if (r != null){
System.out.print(r.getData() +" ");
preorder(r.getLeft());
```

```
preorder(r.getRight());
}}
public void postorder(){
postorder(root);
}
private void postorder(Binary_Tree_Node r){
if (r != null){
postorder(r.getLeft());
postorder(r.getRight());
System.out.print(r.getData() +" ");
}}}
public class Question1{
public static void main(String[] args){
Scanner scan = new Scanner(System.in);
BT bt = new BT();
System.out.println("KHAN MOHD OWAIS RAZA (20BCD7138)\n");
System.out.println("CSE2001 (DSA) Lab-5\n");
System.out.println("\n");
System.out.println("-------------------------------------------------------
");
System.out.println("Binary tree using linked list (with tree
traversals)\n");
char ch;
do{
System.out.println("Select an operation\n");
System.out.println("[1] Insert node ");
System.out.println("[2] Search node");
System.out.println("[3] Count nodes");
System.out.println("[4] Check if empty");
int choice = scan.nextInt();
switch (choice){
case 1 : System.out.println("Enter element to be inserted: ");
bt.insert( scan.nextInt() );
break;
case 2 : System.out.println("Enter element to be searched:");
System.out.println("Search result : "+ bt.search( scan.nextInt() ));
break;
case 3 : System.out.println("No. of nodes "+ bt.countNodes());
break;
case 4 : System.out.println("Empty status: "+ bt.isEmpty());
break;
default : System.out.println("Incorrect entry \n ");
break;
}
System.out.print("\nPost-order: ");
bt.postorder();
System.out.print("\nPre-order: ");
bt.preorder();
System.out.print("\nIn-order: ");
bt.inorder();
System.out.println("\nPress 'C' to continue or 'S' to stop \n");
ch = scan.next().charAt(0);
} while (ch == 'C');
}}
```




```
<terminated> Question1 (2) [Java Application] C:\Program Files\Java\jdk-17.0.1\bin
KHAN MOHD OWAIS RAZA (20BCD7138)

CSE2001 (DSA) Lab-5

-----------------------------------------------------
Binary tree using linked list (with tree traversals)

Select an operation:
[1] Insert node
[2] Search node
[3] Count nodes
[4] Check if empty
1
Enter element to be inserted:
10

Post-order: 10
Pre-order: 10
In-order: 10

Press 'C' to continue or 'S' to stop:
C
Select an operation:
[1] Insert node
[2] Search node
[3] Count nodes
[4] Check if empty
1
Enter element to be inserted:
20

Post-order: 20 10
Pre-order: 10 20
In-order: 10 20

Press 'C' to continue or 'S' to stop:
C
Select an operation:
[1] Insert node
[2] Search node
[3] Count nodes
[4] Check if empty
1
Enter element to be inserted:
30

Post-order: 30 20 10
Pre-order: 10 30 20
In-order: 30 10 20

Press 'C' to continue or 'S' to stop:
C
Select an operation:
[1] Insert node
[2] Search node
[3] Count nodes
[4] Check if empty
1
Enter element to be inserted:
40

Post-order: 40 30 20 10
Pre-order: 10 30 40 20
In-order: 30 40 10 20
```

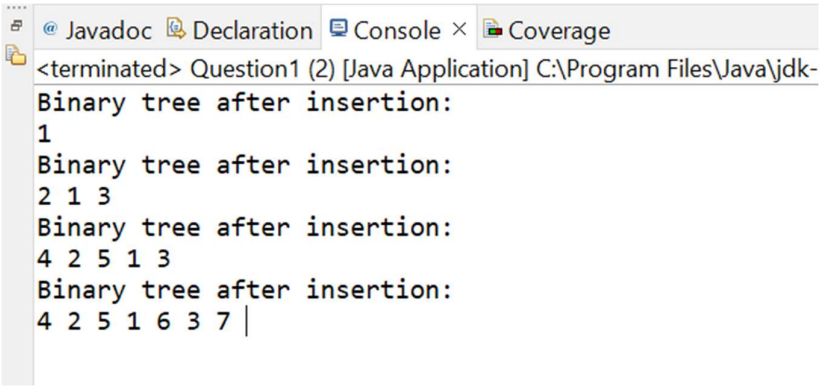
```
Press 'C' to continue or 'S' to stop:
C
Select an operation:
[1] Insert node
[2] Search node
[3] Count nodes
[4] Check if empty
1
Enter element to be inserted:
50

Post-order: 50 40 30 20 10
Pre-order: 10 30 50 40 20
In-order: 50 30 40 10 20

Press 'C' to continue or 'S' to stop:
C
Select an operation:
[1] Insert node
[2] Search node
[3] Count nodes
[4] Check if empty
2
Enter element to be searched:
20
Search result: true

Post-order: 50 40 30 20 10
Pre-order: 10 30 50 40 20
In-order: 50 30 40 10 20

Press 'C' to continue or 'S' to stop:
C
Select an operation:
[1] Insert node
[2] Search node
[3] Count nodes
[4] Check if empty
2
Enter element to be searched:
25
Search result: false

Post-order: 50 40 30 20 10
Pre-order: 10 30 50 40 20
In-order: 50 30 40 10 20

Press 'C' to continue or 'S' to stop:
C
Select an operation:
[1] Insert node
[2] Search node
[3] Count nodes
[4] Check if empty
3
No. of nodes 5

Post-order: 50 40 30 20 10
Pre-order: 10 30 50 40 20
In-order: 50 30 40 10 20
```

```
Press 'C' to continue or 'S' to stop:
C
Select an operation:
[1] Insert node
[2] Search node
[3] Count nodes
[4] Check if empty
4
Empty status: false

Post-order: 50 40 30 20 10
Pre-order: 10 30 50 40 20
In-order: 50 30 40 10 20

Press 'C' to continue or 'S' to stop:
S
```

Java code to implement binary tree using linked list :

```
/**
 Name: KHAN MOHD OWAIS RAZA
 ID : 20BCD7138
 Course: Data Structures & Algorithm
 Code: CSE2001
 Slot: L19+L20
**/
/* Lab-5 (08-10-2022)*/
/* Java code to implement binary tree using linked list */
package CSE2001_Lab5_20BCD7138;
import java.util.LinkedList;
import java.util.Queue;
public class Question1 {
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 Question1(){
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) {
Question1 bt = new Question1();
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);
}}
```



Q.2] Write a program to implement binary tree traversals in-order, pre-order, post-order using recursion.

```
/**
 Name: KHAN MOHD OWAIS RAZA
 ID : 20BCD7138
 Course: Data Structures & Algorithm
 Code: CSE2001
 Slot: L19+L20
**/
/* Lab-5 (08-10-2022)*/
/* Java code to implement binary tree traversals using recursion */
package CSE2001_Lab5_20BCD7138;
class Node {
int item;
Node left, right;
public Node(int key) {
item = key;
left = right = null;
}}
public class Question2 {
Node root;
Question2() {
root = null;
}
void postorder(Node node) {
if (node == null) return;
postorder(node.left);
postorder(node.right);
System.out.print(node.item + "  ");
}
void inorder(Node node) {
if (node == null) return;
inorder(node.left);
System.out.print(node.item + "  ");
inorder(node.right);
}
void preorder(Node node) {
if (node == null) return;
System.out.print(node.item + "  ");
preorder(node.left);
preorder(node.right);
}
public static void main(String[] args) {
Question2 tree = new Question2();
tree.root = new Node(1);
tree.root.left = new Node(12);
tree.root.right = new Node(9);
tree.root.left.left = new Node(5);
tree.root.left.right = new Node(6);
System.out.println("KHAN MOHD OWAIS RAZA (20BCD7138)");
System.out.println("CSE2001 (DSA) Lab-5");
System.out.println("");
```

```
System.out.println("Java code implement binary tree traversals using
recursion");
System.out.println("-----------------------------------------------------
----");
System.out.println("");
System.out.println("Inorder traversal:");
tree.inorder(tree.root);
System.out.println("\n\nPreorder traversal:");
tree.preorder(tree.root);
System.out.println("\n\nPostorder traversal:");
tree.postorder(tree.root);
}}
```