C:\Users\Rich\Documents\NetBeansProjects\Lab08\src\TreeClient.java

```
1 import java.util.Iterator;
2 /**
3  *
4  * @author Richelin Metellus
5  * @version 03/10/2017
6  * Printing of different ways to visit a tree.
7  */
8 public class TreeClient {
9
10    public static void main(String[] args) {
11       LinkedBinaryTree<String>[] lbt = new LinkedBinaryTree[19];
12       for(int i = 0; i<19; i++)
13          lbt[i] = new LinkedBinaryTree();
14       Position[] positions = new Position[19];
15
16       positions[0] = lbt[0].addRoot("*");
17       positions[1] = lbt[1].addRoot("/");
18       positions[2] = lbt[2].addRoot("9");
19       positions[3] = lbt[3].addRoot("*");
20       positions[4] = lbt[4].addRoot("+");
21       positions[5] = lbt[5].addRoot("+");
22       positions[6] = lbt[6].addRoot("-");
23       positions[7] = lbt[7].addRoot("+");
24       positions[8] = lbt[8].addRoot("-");
25       positions[9] = lbt[9].addRoot("7");
26       positions[10] = lbt[10].addRoot("5");
27       positions[11] = lbt[11].addRoot("9");
28       positions[12] = lbt[12].addRoot("3");
29       positions[13] = lbt[13].addRoot("15");
30       positions[14] = lbt[14].addRoot("24");
31       positions[15] = lbt[15].addRoot("-");
32       positions[16] = lbt[16].addRoot("5");
33       positions[17] = lbt[17].addRoot("6");
34       positions[18] = lbt[18].addRoot("1");
35
36       // attaching the subtree in bottom up fashion
37       lbt[15].attach(positions[15], lbt[17], lbt[18]);
38
39       lbt[5].attach(positions[5], lbt[9], lbt[10]);
40       lbt[6].attach(positions[6],lbt[11],lbt[12]);
41       lbt[7].attach(positions[7], lbt[13],lbt[14]);
42       lbt[8].attach(positions[8],lbt[15], lbt[16]);
43
44       lbt[3].attach(positions[3],lbt[5],lbt[6]);
45       lbt[4].attach(positions[4], lbt[7],lbt[8]);
46
47       lbt[1].attach(positions[1],lbt[3],lbt[4]);
48
49       lbt[0].attach(positions[0],lbt[1],lbt[2]);
50
51       System.out.println(" The experession: (((7+5)*(9-3))/((15+24)+((6-1)-5))*9)");
52       // Inorder
53       Iterator<Position<String>> inOrderIterator = lbt[0].inorder().iterator();
54       while(inOrderIterator.hasNext())
55          System.out.print(inOrderIterator.next().getElement());
56       System.out.println(" Inorder Traversal of the Tree\n");
```

```
57
58      Iterator<Position<String>> preOrderIterator = lbt[0].preorder().iterator();
59      while(preOrderIterator.hasNext())
60          System.out.print(preOrderIterator.next().getElement()+ " ");
61      System.out.println(" preOrder traversal of the Tree \n");
62
63      //Postorder traversal
64      Iterator<Position<String>> postOrderIterator = lbt[0].postorder().iterator();
65      while(postOrderIterator.hasNext())
66          System.out.print(postOrderIterator.next().getElement()+ " ");
67      System.out.println(" postOrder traversal of the Tree \n");
68
69      Iterator<Position<String>> breathFirstIterator = lbt[0].breathFirst().iterator();
70      while(breathFirstIterator.hasNext())
71          System.out.print(breathFirstIterator.next().getElement()+ " ");
72      System.out.println(" breathFirst traversal of the Tree \n");
73
74      System.out.println("preOrderIndent traversal of the tree");
75      AbstractBinaryTree.printPreorderIndent(lbt[0], lbt[0].root, 0);
76
77      System.out.println("Parenthesize representation of the tree");
78      AbstractBinaryTree.parenthesize(lbt[0], positions[0]);
79      System.out.println("");
80  }
81
82 }
83
```