Traversals Of Binary Search Tree Competency #8

Trees	Mastered 1 point	Progressing 0.001 points	Novice 0 points	Criterion Score
Can design and implement a binary search tree				/1

Traversals	Mastered 1 point	Progressing 0.001 points	Novice 0 points	Criterion Score
Can implement an in-order traversal of a binary tree				/ 1
Can implement a pre-order traversal of a binary tree				/ 1
Can implement a post-order traversal of a binary tree				/1

```
🞵 Order.java 🗙 🚺 Menultem.java
                                    BSTNode.java
   1 package restaurant;
   3⊝ import java.text.DecimalFormat;
      import java.util.LinkedList;
      import java.util.Queue;
     import java.util.Stack;
   8
      public class Order {
   9
          private BSTNode root;
  10
           private String tableNumber;
  11
           static String name = "Chipotle";
  12
  13
  140
           public Order(String tableNumber) {
  15
               this.tableNumber = tableNumber;
  16
               root = null;
  17
          }
  18
  19⊝
           public void insert(MenuItem menuItem) {
  20
               if (root == null) {
21
                   root = new BSTNode(menuItem, null, null);
  22
               } else {
  23
                   insert(root, menuItem);
  24
  25
          }
  26
<u>∿</u> 27⊝
           public void insert(BSTNode current, MenuItem menuItem) {
  28
               if (menuItem.compareTo((MenuItem) current.getData()) <= 0) {</pre>
  29
                   if (current.getLeft() == null) {
N 30
                       current.setLeft(new BSTNode(menuItem, null, null));
  31
                   } else {
  32
                       insert(current.getLeft(), menuItem);
                   }
  33
  34
               } else {
                   if (current.getRight() == null) {
  35
36
                       current.setRight(new BSTNode(menuItem, null, null));
                   } else {
  37
  38
                       insert(current.getRight(), menuItem);
  39
  40
               }
  41
          }
  42
           public void preOrder() {
  43⊝
               preOrder(root);
  44
  45
  46
<u></u> 47⊝
           public void preOrder(BSTNode node) {
  48
              if (node == null)
  49
                   return;
               System.out.print(node.data + "->");
  50
  51
               preOrder(node.left);
  52
               preOrder(node.right);
  53
  54
  55⊝
           public void inOrder() {
  56
               inOrder(root);
  57
  58
<u></u> 59⊝
          public void inOrder(BSTNode current) {
  60
               if (current != null) {
                   inOrder(current.getLeft());
  61
  62
                   System.out.println(current);
  63
                   inOrder(current.getRight());
               }
  64
          }
  65
```

```
    ■ BSTNode.java ×
```

```
1 package restaurant;
  3 public class BSTNode<E> {
        E data;
  5
        BSTNode<E> left;
  6
        BSTNode<E> right;
  7
  8⊝
        public BSTNode(E data, BSTNode<E> left, BSTNode<E> right) {
  9
            this.data = data;
 10
            this.left = left;
 11
            this.right = right;
 12
 13
 14⊝
        public E getData() {
 15
             return data;
 16
 17
 18⊝
        public void setData(E data) {
 19
            this.data = data;
 20
 21
 22⊝
        public BSTNode<E> getLeft() {
 23
             return left;
 24
 25
        public void setLeft(BSTNode<E> left) {
 26⊝
 27
            this.left = left;
 28
 29
 30⊝
        public BSTNode<E> getRight() {
 31
            return right;
        }
 32
 33
 34⊝
        public void setRight(BSTNode<E> right) {
 35
            this.right = right;
 36
 37
 38
 39 }
 40
```

```
    Menultem.java 

    ✓
1 package restaurant;
   import java.text.DecimalFormat;
     public class MenuItem implements Comparable<MenuItem>{
          private String name;
private double price;
private int quantity;
          public MenuItem(String name, double price, int quantity) {
   this.name = name;
   this.price = price;
   this.quantity = quantity;
}
 10⊖
11
12
13
14
15
16⊝
          public String getName() {
    return name;
}
17
188
19 = 20
21
22
23
24 = 25
26
27
28 = 26
27
33
33
33
34
44
45
46
47
48
49
49
41
48
49
50
51
51
52
          public void setName(String name) {
                this.name = name;
          public double getPrice() {
   return price;
          public void setPrice(double price) {
   this.price = price;
          public int getQuantity() {
    return quantity;
          public void setQuantity(int quantity) {
    this.quantity = quantity;
          public String toString() {
    DecimalFormat df = new DecimalFormat("0.00");
    return this.getName() + "\t$" + df.format(this.getPrice()) + "\t" + this.getQuantity() + "\t$" + df.format(this.quantity*this.price);
          public boolean equals(MenuItem item) {
   if(this.getName().compareToIgnoreCase(item.getName()) == 0) {
               return true;
} else {
   return false;
          }
    52
                       @Override
    53⊜
 △54
                       public int compareTo(MenuItem item) {
    55
                                 if(this.name.compareTo(item.name) == 0) {
     56
                                            return 0;
                                 } else if(this.name.compareTo(this.name) > 0) {
    57
    58
                                           return 1;
    59
                                 } else {
    60
                                           return -1;
    61
                       }
    62
    63
    64
    65
             }
    66
```

```
Order.java X
   1 package restaurant;
   3⊕ import java.text.DecimalFormat;
   8 public class Order {
  9
          private BSTNode root;
  10
          private String tableNumber;
  11
          static String name = "Chipotle";
  12
  13
  14⊖
          public Order(String tableNumber) {
  15
              this.tableNumber = tableNumber;
  16
              root = null;
  17
  18
  19⊖
          public void insert(MenuItem menuItem) {
              if (root == null) {
  20
N 21
                  root = new BSTNode(menuItem, null, null);
  22
              } else {
  23
                  insert(root, menuItem);
  24
  25
          }
  26

27⊖

          public void insert(BSTNode current, MenuItem menuItem) {
  28
              if (menuItem.compareTo((MenuItem) current.getData()) <= 0) {</pre>
  29
                  if (current.getLeft() == null) {
  30
                      current.setLeft(new BSTNode(menuItem, null, null));
  31
                  } else {
  32
                       insert(current.getLeft(), menuItem);
                  }
  33
  34
              } else {
  35
                  if (current.getRight() == null) {
36
                      current.setRight(new BSTNode(menuItem, null, null));
  37
                  } else {
  38
                       insert(current.getRight(), menuItem);
  39
                  }
  40
              }
  41
          }
  42
  43⊖
          public void preOrder() {
  44
              pre0rder(root);
  45
  46

47⊝

          public void preOrder(BSTNode node) {
  48
              if (node == null)
  49
                  return;
  50
              System.out.print(node.data + "->");
  51
              preOrder(node.left);
  52
              pre0rder(node.right);
  53
          }
  54
  55⊖
          public void inOrder() {
  56
              inOrder(root);
  57
          }
  58
```

```
🚺 Order.java 🗙
  117⊖
             public MenuItem search(String target) {
  118
                  while (root != null) {
  119
                       if (root.getData().toString().compareTo(target) < 1)</pre>
  120
                             root = root.getRight();
  121
                       else if (root.getData().toString().compareTo(target) > 1)
  122
                            root = root.getLeft();
  123
  124
                             return (MenuItem) root.getData();
  125
  126
                  return null;
  127
  128
  129⊖
             public double getTotalBeforeTax() {
  130
                  if (root == null)
  131
                        return 0;
  132
                  Queue<BSTNode<MenuItem>> queue = new LinkedList<BSTNode<MenuItem>>();
133
                  queue.add(root);
  134
                  int total = 0;
  135
                  while (!queue.isEmpty()) {
  136
                       BSTNode<MenuItem> temp = queue.poll();
  137
                       total += temp.getData().getPrice() * temp.getData().getQuantity();
  138
                       if (temp.getLeft() != null) {
                             queue.add(temp.getLeft());
  139
  140
  141
                       if (temp.getRight() != null) {
  142
                             queue.add(temp.getRight());
  143
  144
  145
                  return total;
  146
  147
  148⊖
             public double getTax(double percent) {
  149
                  return getTotalBeforeTax() * percent / 100;
  150
  151
  152⊖
             public double getTip(double percent) {
  153
                  return getTotalBeforeTax() * percent / 100;
  154
             }
  100
 155
 156⊖
        @Override
△157
        public String toString() {
            StringBuilder tableMenu = new StringBuilder();
if (root != null) {
 158
 159
               Stack<BSTNode<MenuItem>> s = new Stack<BSTNode<MenuItem>>();
               BSTNode<MenuItem> curr = root;
while (curr != null || s.size() > 0) {
  while (curr != null) {
<u>3</u>161
 162
                      s.push(curr);
 165
                      curr = curr.getLeft();
 166
                   curr = s.pop();
                   tableMenu.append(curr.getData().toString()).append("\n");
 168
                   curr = curr.getRight();
 169
               }
 171
            DecimalFormat dec = new DecimalFormat("#0.00"):
 172
 173
            return name + "\t" + tableNumber + "\n" +
                   + tableMenu.toString() + "-----\n" + "Total \t\t$"

+ dec.format(getTotalBeforeTax()) + "\n" + "Tax \t\t$" + dec.format(getTax(8)) + "\n" + "Tip \t\t$"

+ dec.format(getTax(8)) + "\n" + "Tip \t\t$"
 175
 176
                                                                                         --\n" + "Grand total: $"
                   + dec.format(getTip(20)) + "\n'
                   + dec.format(getTotalBeforeTax() + getTax(8) + getTip(20)) + "\n\n";
 178
        }
 179
 181 }
 182
```

```
1
     package restaurant;
  3
     public class RestaurantDriver {
  4
  5⊝
          public static void main(String[] args) {
               Order ord1 = new Order("12");
  6
  7
               Order ord2 = new Order("45");
  8
  9
               ord1.insert(new MenuItem("Chow Mein", 2.5,1));
               ord1.insert(new MenuItem("Fried Wontons", 1.5,3));
10
               ord1.insert(new MenuItem("Egg Foo Young",7.8,4));
11
12
               ord1.insert(new MenuItem("Kong Pao Chicken",18.25,2));
               ord1.insert(new MenuItem("Sushi",12.5,1));
13
               ord1.insert(new MenuItem("Chicken Fried Rice",13.5,3));
14
15
               ord1.insert(new MenuItem("Spring rolls ",5.8,4));
16
               ord1.insert(new MenuItem("Sweet and Sour Chicken", 8.25,2));
17
18
               ord1.in0rder();
 19
               System.out.println("\n\n");
               ord1.pre0rder();
20
               System.out.println("\n\n");
 21
 22
               ord1.post0rder();
 23
               System.out.println("\n\n");
 24
 25
               ord2.insert(new MenuItem("Chow Mein", 2.5,1));
               ord2.insert(new MenuItem("Fried Wontons",1.5,3));
ord2.insert(new MenuItem("Fried Wontons",1.5,3));
ord2.insert(new MenuItem("Spring rolls ",5.8,4));
ord2.insert(new MenuItem("Sweet and Sour Chicken",8.25,2));
ord2.insert(new MenuItem("Sushi",12.5,1));
ord2.insert(new MenuItem("Chicken Fried Rice",13.5,3));
 26
 27
28
29
30
               ord2.insert(new MenuItem("Egg Foo Young",7.8,4));
ord2.insert(new MenuItem("Kong Pao Chicken",18.25,2));
31
32
33
34
35
36
               System.out.println(ord1);
37
               System.out.println( ord1.size());
38
               System.out.println(ord1.depth());
39
               System.out.println("quantity" + ord1.getTotalQty());
               System.out.println(ord1.search("Chow Mein"));
40
41
               System.out.println(ord1.getTotalBeforeTax());
42
               System.out.println(ord1.getTax(8));
43
               System.out.println(ord1.getTip(20));
44
               System.out.println(ord1.toString());
45
               System.out.println(ord1.size());
46
               System.out.println(ord1.size());
47
48
49
```

System.out.println(ord2);

50 51

52 53

54 55 } 56 }

Chow Mein \$2.50 1 \$2.50->Fried Wontons \$1.50 3 \$4.50->Egg Foo Young \$7.80 4 \$31.20->Kong Pao Chicken \$18.25 2 \$36.50->Sushi \$12.50 1 \$12.50->Chicken Fried Rice \$13.50 3

Sweet and Sour Chicken \$8.25 2 \$16.50->Spring rolls \$5.80 4 \$23.20->Chicken Fried Rice \$13.50 3 \$40.50->Sushi \$12.50 1 \$12.50->Kong Pao Chicken \$18.25 2 \$36.50->Egg Foo Young \$7.

Item	Price		Qty		Total
Sweet and Sour		\$8.25		\$16.50	
Spring rolls		4	\$23.20		
Chicken Fried Rice		\$13.50 3	3 \$40.50	\$40.50	
Sushi \$12.50	1	\$12.50			
Kong Pao Chicken		\$18.25	2	\$36.50	
Egg Foo Young	\$7.80	4	\$31.20		
Fried Wontons	\$1.50	3	\$4.50		
Chow Mein	\$2.50	1	\$2.50		
Total	\$164.00				-
Tax	\$13.12				
Tip	\$32.80				

Grand total: \$209.92

```
8
0
quantity20
null
0.0
0.0
0.0
                12
Chipotle
Item
                 Price
                                   Qty
                                                  Total
Total
                $0.00
Tax
                $0.00
Tip
                $0.00
Grand total: $0.00
0
Chipotle
                45
                                                  Total
Item
                 Price
                                   Qty
Kong Pao Chicken
                                         $36.50
                         $18.25 2
Egg Foo Young $7.80
                         4
                                 $31.20
Chicken Fried Rice
                         $13.50
                                 3
                                         $40.50
Sushi
        $12.50 1
                         $12.50
Sweet and Sour Chicken
                        $8.25
                                         $16.50
Spring rolls
                $5.80
                         4
                                 $23.20
Fried Wontons
                $1.50
                         3
                                 $4.50
Chow Mein
                $2.50
                         1
                                 $2.50
Total
                $164.00
Tax
                $13.12
Tip
                $32.80
```

Grand total: \$209.92

```
58
          public void inOrder(BSTNode current) {
  59⊝
  60
              if (current != null) {
  61
                  inOrder(current.getLeft());
                  System.out.println(current);
  62
  63
                  inOrder(current.getRight());
  64
              }
  65
          }
  66
  67⊝
          public void postOrder() {
  68
              postOrder(root);
  69
  70
          public void postOrder(BSTNode node) {
  71⊝
  72
              if (node == null)
                  return;
  73
74
  75
              postOrder(node.left);
  76
              postOrder(node.right);
  77
              System.out.print(node.data + "->");
  78
  79
<u></u> 80⊝
          public int size(BSTNode node) {
  81
              if (node == null)
  82
                  return 0;
  83
              else
                  return (size(node.left) + 1 + size(node.right));
  84
  85
          }
  86
  87⊝
          public int size() {
  88
  89
              return size(root);
  90
  91
  92⊝
          public int depth() {
  93
              int height = 0;
  94
              if (root == null)
                  height = -1;
  95
              if (root.getLeft() != null && root.getRight() != null) {
  96
  97
                  height = 0;
  98
              return height;
  99
 100
          }
 101
 102⊖
          public int getTotalQty() {
№103
              return getTotalQty(root);
 104
 105
 106
          public int getTotalQty(BSTNode<MenuItem> node) {
   if(node == null){
 107⊝
 108
 109
                return 0;
 110
              } else{
                  return
 111
                           getTotalQty(node.getLeft()) + getTotalQty(node.getRight()) + node.data.getQuantity();
 112
 113
              }
 114
          }
 115
116
```