

This lab has **seven (7)** checkpoints.

## Learning Outcomes

Upon completing this lab, students should be able to

- **Follow** code that passes a visitor object around a tree structure.

## Introduction

1. Given the `Tree` and visitor classes given at the end of the lab, what is the output of each of the following snippets of code:

(a)

```
Tree bt = new Tree();

bt.add("50");
bt.add("25");
bt.add("75");
bt.add("35");
bt.add("55");
bt.add("05");

Visitor cv = new CommaVisitor();
bt.inOrder(cv);
System.out.println(String.format("CommaVistor.result() = %s", cv.result()));
```

✓ Show the lab instructor the output. Be prepared to explain how you figured it out.

**Solution:**

```
CommaVistor.result() = 05, 25, 35, 50, 55, 75
```

(b)

```
Tree bt = new Tree();

bt.add("50");
bt.add("25");
bt.add("75");
bt.add("35");
bt.add("55");
bt.add("05");

Visitor cv = new CommaVisitor();
bt.preOrder(cv);
System.out.println(String.format("CommaVistor.result() = %s", cv.result()));
```

✓ Show the lab instructor the output. Be prepared to explain how you figured it out.

**Solution:**

```
CommaVistor.result() = 50, 25, 05, 35, 75, 55
```

(c)

```
Tree bt = new Tree();

bt.add("50");
```

1  
2  
3

```

    bt.add("25");
    bt.add("75");
    bt.add("35");
    bt.add("55");
    bt.add("05");

    Visitor sv = new SumVisitor();
    bt.inOrder(sv);
    System.out.println(String.format("SumVistor.result() = %s", sv.result()));

```

4  
5  
6  
7  
8  
9  
10  
11  
12

Would the results change if `inOrder` were replaced with `preOrder` in line 11?

✓ Show the lab instructor the output. Be prepared to explain your answers.

**Solution:**

`SumVistor.result() = 245`

No, it would not be different: order does not matter with addition rather than with string concatenation.

Listing 1: `Tree.java`

```

public class Tree {
    class Node {
        String data;
        Node left;
        Node right;

        Node(String data, Node left, Node right) {
            this.data = data;
            this.left = left;
            this.right = right;
        }

        Node(String data) {
            this(data, null, null);
        }
    }

    Node root;

    public Tree() {
        root = null;
    }

    public void add(String newbie) {
        if (root == null)
            root = new Node(newbie);
        else
            add(root, newbie);
    }

    private void add(Node curr, String newbie) {
        if (curr.data.compareTo(newbie) < 0) { // go right

```

```
        if (curr.right == null)
            curr.right = new Node(newbie);
        else
            add(curr.right, newbie);
    } else { // go left
        if (curr.left == null)
            curr.left = new Node(newbie);
        else
            add(curr.left, newbie);
    }
}

public void inOrder(Visitor i0) {
    inOrder(root, i0);
}

private void inOrder(Node curr, Visitor i0) {
    if (curr != null) {
        inOrder(curr.left, i0);
        i0.apply(curr.data);
        inOrder(curr.right, i0);
    }
}

public void preOrder(Visitor pre0) {
    preOrder(root, pre0);
}

private void preOrder(Node curr, Visitor pre0) {
    if (curr != null) {
        pre0.apply(curr.data);
        preOrder(curr.left, pre0);
        preOrder(curr.right, pre0);
    }
}
}
```

Listing 2: Visitor.java

```
public interface Visitor {  
    public void apply(String string);  
    public String result();  
}
```

Listing 3: CommaVisitor.java

```
public class CommaVisitor implements Visitor {  
    String accumulator;  
    public CommaVisitor() {  
        accumulator = "";  
    }  
  
    public void apply(String string) {  
        if (!accumulator.isEmpty())  
            accumulator += ", ";  
        accumulator += string;  
    }  
  
    public String result() {  
        return accumulator;  
    }  
}
```

Listing 4: SumVisitor.java

```
public class SumVisitor implements Visitor {  
    int accumulator;  
  
    public SumVisitor() {  
        accumulator = 0;  
    }  
  
    public void apply(String string) {  
        accumulator += Integer.parseInt(string);}  
  
    public String result() {  
        return Integer.toString(accumulator);  
    }  
}
```