

CS 635 Advanced Object-Oriented Design &  
Programming  
Spring Semester, 2007  
Assignment 1 Comments

# Java Names

```
class bNode
```

```
class Binary_Search_Tree
```

```
public boolean addNode(String value)
```

```
Node left_child;
```

```
Node leftPointer;
```

```
String dataStr;
```

```
public List recursiveMethodToReturnValuesInOrder()
```

```
public String getReverseOrderTree()
```

```
public boolean isAcceptableNodeContent(...)
```

# Names

```
public void insert(String data) {  
    Node temp = root;           //temp is the current location in the search  
  
    while (temp != null)  
    {  
        blah  
        blah  
        blah  
        blah  
        blah  
        blah  
    } (29 lines after temp declared
```

# Comments

```
class Node {
    // Constructor
    public Node() {

// Add string to the tree
public boolean addNodeToTree(String data)

/**
 * Returns the size of the tree
 * @return the size of the tree
 */
public int size() {
    return size
}
}
```

# Comments

```
// Check to see if string ends with A or E  
if (data.ToUpper().EndsWith("A") ||  
    data.ToUpper().EndsWith("E")) {  
    blah
```

```
if (endsWithAOrE(data) )  
    blah
```

# Comments

```
/**  
 * @return All the elements in the tree in reverse alphabetic order.  
 */  
public String getReverseOrderTree() {  
    if (null == root) return null;  
    else return root.getReverseOrder();  
}
```

# Static

```
class Tree {  
    private static Node root;  
  
    public static boolean add(String element) {  
        blah  
    }  
}
```

# Inheritance

```
class Tree extends Node {  
    blah
```

# Inheritance Test

A has a B

A is a kind (type) of B

```
class A {  
    private B foo;
```

```
class A extends B
```

# Information Hiding - not

```
class BinaryTree {  
    public Node root;  
  
    blah
```

```
class BinaryTree {  
    Node root;  
  
    blah
```

# Information Hiding - not

```
class BinarySearchTree {  
    private Node root;  
  
    public Node find(String key) {  
        blah  
        blah  
        return foundNode;  
    }  
}
```

# Information Hiding - not

```
public class Tree {  
    private Node root;  
    private int size;  
  
    public Tree(Node aNode) {  
        root = aNode;  
        size = 1;  
    }  
}
```

```
Node x = new Node("cat");  
Tree y = new Tree(x);  
x.left(new Node("dog"));
```

# Programmer verses Class Builder

No methods in any collection class in Java, C#, Ruby or Smalltalk  
print to system out

Why does yours?

# Programmer verses Class Builder

```
Tree names = new Tree();  
names.add("sam");  
names.add("jose");  
names.add("nguyen");  
names.add("patil");
```

```
for (String element : tree.elements() ) {  
    System.out.println(element);  
}
```

```
for (String element : tree.elements().reverse() ) {  
    if ( element.endsWith('a') || element.endsWith('e') )  
        System.out.println(element);  
}
```

Coupling

Cohesion

# Abstraction

Text

# Struct

```
class Node {  
    private Node left;  
    private Node right;  
    private String value;  
  
    public Node getLeft() { return left; }  
    public Node getRight() { return right; }  
    public void setLeft(Node newLeft) { left = newLeft;}  
    public void setRight( blah ) { blah }  
    public String getValue() { blah }  
    public void setValue(blah) { blah}
```

# Helper Method

```
class Tree {  
    public reversePrint(Node aNode) {  
        if (null != aNode) {  
            reversePrint(aNode.right);  
            System.out.println(aNode.value);  
            reversePrint(aNode.right);  
        }  
    }  
}
```

# State verses Parameter

```
public class Tree {
    private Node root;
    ArrayList arrLstString = new ArrayList();

    public ArrayList reverseOrder() {
        reverseOrder(root);
        return arrLstString;
    }

    private ArrayList reverseOrder(Node aNode) {
        if (null != aNode) {
            reverseOrder(aNode.right);
            System.out.println(aNode.value);
            arrLstString.add(aNode.value);
            reverseOrder(aNode.right);
        }
    }
}
```

# Using Parameter

```
public class Tree {
    private Node root;

    public ArrayList reverseOrder() {
        return reverseOrder(root, new ArrayList());
    }

    private ArrayList reverseOrder(Node aNode, ArrayList elements) {
        if (null != aNode) {
            reverseOrder(aNode.right);
            elements.add(aNode.value);
            reverseOrder(aNode.left);
        }
        return elements;
    }
}
```