

CS 635 Advanced Object-Oriented Design & Programming
Spring Semester, 2007
Comments on Assignment 2
Mar 8, 2007

Issues?

```
public class BSTree
```

```
    private int size() {  
        int bstsize = 0;  
        bstsize = size(root, bstsize);  
        return bstsize  
    }
```

```
    private int size(Node current, int bsize) {  
        if (current.isNull() ) {  
            return bsize;  
        }  
        bsize = size(current.left, bsize);  
        bsize = bsize + 1;  
        bsize = size(current.right, bsize);  
        return bsize;  
    }
```

How About

```
public class BSTree
{
    private int size() {
        return root.size();
    }

    blah
}
```

```
class BinaryNode extends AbstractNode {
    public int size() {
        return left.size() + right.size() + 1;
    }
}
```

```
class NullNode extends AbstractNode {
    public int size() {
        return 0;
    }
}
```

Issues?

```
BinaryNode(String key)
{
    this.data = key;
    NullNode leftchild = new NullNode();
    left = leftchild;
    leftchild.parent = this;
    NullNode rightchild = new NullNode();
    right = rightchild;
    rightchild.parent = this;
}
```

Issues

```
public int interate(Object inData, Node inNode) {
    try {
        BinaryNode bNode = (BinaryNode) this;
        int strCmp = bNode.data.compreTo(inData.toString());
        if (strCmp > 0) {
            Node nNode = (Node) bNode.left;
            nNode.iterate(inData, this);
        }
        if (strCmp < 0) {
            Node nNode = (Node) bNode.right;
            nNode.iterate(inData, this);
        }
        if (strCmp == 0)
            return 0;
        return 1;
    } catch (ClassCastException Exp) {
        throw Exp;
    }
}
```

Issues?

```
public class BSTreeNode {  
    private String value;  
    private BSTreeNode parentTree;  
    private BSTreeNode leftTree;  
    private BSTreeNode righthTree;  
  
    public BSTreeNode(String value ) {blah}  
    public void setParentTree(BSTreeNode node) {blah}  
    public BSTreeNode getParentTree() {blah}  
    public void setLeftTree(BSTreeNode node) {blah}  
    public BSTreeNode getLeftTree() {blah}  
    public void setRightTree(BSTreeNode node) {blah}  
    public BSTreeNode getRightTree() {blah}  
    public String getValue() { blah}
```

Issues?

```
public abstract class Node {  
    protected Node left;  
    protected Node right;  
    protected String strData;  
    public abstract Node add( Node current, Node node);  
    public abstract boolean isNull();  
}
```

```
public class NullNode extends Node {  
    blah  
}
```

```
public BinaryNode extends Node {  
    blah  
}
```

Issues?

```
public class EndsInIterator implements Iterator {
    private Node node;
    private Node next;
    boolean flag = false;
    private BinaryTreeIterator btltr = null;
    private char chrEndings[];

    public boolean hasNext() {
        if (!flag) {
            return false;
        }
        else {
            return true;
        }
    }
}
```

Issues?

```
public class Node extends TreeSet {
```

Issues

```
public class BSTree extends AbstractionCollection {  
    blah  
  
    public String toString() {  
        return super.toString();  
    }  
  
    blah
```

Issues?

```
public class NullNode extends Node {
    NullNode() { }
    public String getData() {
        return null;
    }
    public void setData(String newData ) {

    }
    public Node getLeftChild() {
        return null;
    }
    public void setLeftChild(Node newNode) {

    }
    public Node getRightChild() {
        return null;
    }
    public void setRightChild(Node newNode) {

    }
    public addNode(Node newNode) {
        return newNode;
    }
}
```

```
public void traverseTree(ArrayList returnList) {

}

public boolean gotoLeftNode(Stack nodeStack) {
    return false;
}
```

Issues?

```
public boolean hasNext() {  
    return valid;    //return true if element is present  
}
```

What Type of Cohesion?

```
public abstract class Node {  
    public abstract boolean add(String inpString);  
    public abstract boolean hasLeft();  
}
```

Issues?

```
while (!(leftMost instanceof NullNode)) {  
    stack.push(leftMost);  
    leftMost = leftMost.getLeft();  
}
```

Issues?

```
public class BinaryNodeNull extends Node {  
  
    blah  
  
    public boolean add(String value) {  
        if (this.parent.value.compareTo(value) < 0 ) {  
            this.parent.left = new BinaryNode(value);  
            return true;  
        }  
        else if(this.parent.value.compareTo(value) >= 0 ) {  
            this.parent.right = new BinaryNode(value);  
            return true;  
        }  
        return false;  
    }  
}
```