CS 635 Advanced Object-Oriented Design & Programming
Spring Semester, 2016
Doc 15 Assignment 2 Comments
April 12, 2016

```
public class BinarySearchTree<T> extends AbstractCollection<T> {

    public boolean add(T key) {
        if (root.isNull()) { blah; return true;}
        ...
        if (strategy.compare(currentNode, newNode > 0) {
            if (currentNode.leftNode.isNull()) { blah}
        } else if (strategy.compare(currentNode, newNode > 0) {
            if (currentNode.rightNode.isNull()) { blah}
```

```
public class BinarySearchTree<T> extends AbstractCollection<T> {

    public boolean add(T key) {
        if (root == null) { blah; return true;}
        ...
        if (strategy.compare(currentNode, newNode > 0) {
            if (currentNode.leftNode == null) { blah}
        } else if (strategy.compare(currentNode, newNode > 0) {
            if (currentNode.rightNode == null) { blah}
```

```
abstract class Node { ... }

class NullNode extends Node {
    public boolean add(String value, Comparison compare) {
        blah
    }
}

class InnerNode extends Node {

    public boolean isNull() {
        return false;
    }

    public boolean add(String value, Comparison compare) {
        if (this.isNull()) {
            foo
        } else {
            bar
        }
```

```
public boolean hasNext() {
    // TODO Auto-generated method stub
    Node k = (Node) inorderSuccessor(mainNode, currentNode, new NullNode());
    if (k.isNull())
    return false;
    return true;
}
```

```
public boolean hasNext() {
    Node k = (Node) inorderSuccessor(mainNode, currentNode, new NullNode());
    return !k.isNull();
}
```

```
class RootNode<T> extends Node {
    RootNode(T t) {
        super(t);
    }
}
```

```
public class Node<E> {

    public String[ ] toArray(Node<E> root) {
        String result = toString(root);
        String[ ] resultArray = result.split(“ “);
        return resultArray;
    }


Collection tree = new BinaryTree();
tree.add(“This is valid data. The tree has one element”);
String[] trouble = tree.toArray();
trouble.length == 9
```

```
public class Node<E> {

    public String[ ] toArray(Node<E> root) {
        String result = toString(root);
        String[ ] resultArray = result.split(“ “);
        return resultArray;
    }
```

```
toArray(aNode);
```

```
public class Node<E> {

    public String[ ] toArray() {
        String result = toString();
        String[ ] resultArray = result.split(“ “);
        return resultArray;
    }
```

```
aNode.toArray();
```

```
public class Node<E> {
    String data;
    Node<E> left;
    Node<E> right;
    private String result = “”
```

```
Node<String> test = new Node<>(“A”);

test.toString(test);        //”A”
test.toString(test);        //”A A”
test.toString(test);        //”A A A”
```

```
    public String toString(Node<E> root) {
        if (!root.isNull()) {
            toString(root.left);
            result += root.data + “ “;
            toString(root.right);
        }
        return result + “ “;
    }
```

```
public class Node<E> {
    String data;
    Node<E> left;
    Node<E> right;
```

```
Node<String> test = new Node<>("A");

test.toString(test);        //"A"
test.toString(test);        //"A"
test.toString(test);        //"A"
```

```
public String toString(Node<E> root) {
    if (!root.isNull()) {
        return toString(root.left) +", "+ root.data + ", "+toString(root.right);
    }
    return "";
}
```

```
public class Node<E> {
    String data;
    Node<E> left;
    Node<E> right;
```

```
Node<String> test = new Node<>("A");

test.toString();          //"A"
```

```
    public String toString() {
        String leftString = "";
        if (!left.isNull())
            leftString = left.toString();
        String rightString = "";
        if (!right.isNull())
            rightString = right.toString();
        return leftString + ", "+data + ", "+ rightString;
    }
```

```
public class Node<E> {
    String data;
    Node<E> left;
    Node<E> right;

    public String toString() {
        return left.toString() + ”, “+data + ”, “+ right.toString();
    }
```

```
public class NullNode<E> extends Node<E> {

    public String toString() {
        return “”;
    }
```

```
public class NullNode<E> extends Node<E> {

    public boolean isEqual(Node<E> node) {
        return super.isEqual(node);
    }
```