CS 340 Homework 3

“Due” Wednesday October 21
Homework 3

• Implement the class shown in the following slides. The class implements a binary tree. You must write your own test driver. Your class implementation must not depend on the test driver. I will use my own test driver to test your class.
public class BinaryTree {
    //Implements a Binary Tree of Strings
    private class Node {
        private Node left;
        private String data;
        private Node right;
        private Node parent; //reference to the parent node
        // the parent is null for the root node

        private Node(Node L, String d, Node r, Node p) {
            left = L;
            data = d;
            right = r;
            parent = p;
        }
    }

    private Node root;
}
Binary Trees
public BinaryTree() {
    //create an empty tree
}

public BinaryTree(String d) {
    //create a tree with a single node
}

public BinaryTree(BinaryTree b1, String d, BinaryTree b2) {
    //merge the trees b1 AND b2 with a common root with data d
}
public BinaryTree(String t, String open, String close, String empty) {

    /*create a binary tree from the post order format shown on the next slide. Assume t is a syntactically correct string representation of the tree. Open and close are the strings which represent the beginning and end markers of a tree. Empty represents an empty tree. The example in class used ( ) and ! for open, close and empty respectively. The data in the tree will not include strings matching open, close or empty. All tokens (data, open, close and empty) will be separated by white space. Most of the work should be done in a private recursive method. */
}

Binary Tree String Representation
Postorder Format

- Empty Tree: !
- Non-Empty Tree: ( left right data )

Examples

- !
- ( ! ! Hume )
- ( ( ( ! ! Hegel ) ! Nietzsche ) ! Kant )
- ( ! ( ! ( ! ! Frege ) Russell ) Wittgenstein )
- ( ( ! ! Satre ) ( ! ! deBeauvoir ) Camus )
Binary Tree String Representation
Postorder Format

- Empty Tree: @#&

- Non-Empty Tree: $% left right data %$

Examples

- @#&

- $% $% @#& @## Satre $% $% @#& @## deBeauvior $% Camus $%$
public class PostorderIterator implements Iterator<String> {
    //An iterator that returns data in the tree in an post order pattern
    //the implementation must use the parent pointer and must not use an
    //additional data structure such as a list or stack

    public PostorderIterator() {
    }

    public boolean hasNext() {
    }

    public String next() {
    }

    public void remove() {
        //optional method not implemented
    }
}
public Iterator<String> postorder() {
    //return a new post order iterator object
}

public String toString() {
    //returns the string representation of the tree using the post order format
    //shown above. If the tree was created from a string, use the
    //the values of open, close and empty given to the constructor otherwise
    //use (, ) and ! for open, close and empty respectively
    //most of the work should be done in a recursive private method.
}

Homework 3 Submission

- At the top of the file include a comment that lists your name.

- Add a comment for each private method or private instance variable you add.

- Upload one zip file called h3YOURNAME.zip to Canvas. The zip file must contain only your implementation of BinaryTree.java. If you use System.out.println to debug your program those lines should be put in a comment (or removed). Your debug output should not print when I test your program.