TreeNode4.java

1  /**
2  * TreeNode4 extends TreeNode by adding four child components, which are
3  * references other TreeNodes. Hence, TreeNode4 is used to represent
4  * quaternary syntactic constructs in a parse tree.
5  */
6  public class TreeNode4 extends TreeNode {
7     public TreeNode4(int id, TreeNode child1, TreeNode child2, TreeNode child3, TreeNode child4) {
8         super(id);
9         this.child1 = child1;
10        this.child2 = child2;
11        this.child3 = child3;
12        this.child4 = child4;
13     }
14
15     /**
16     * A la the other constructor, but with line and column numbers.
17     */
18     public TreeNode4(int id, TreeNode child1, TreeNode child2, TreeNode child3, TreeNode child4, int line, int column) {
19         super(id, line, column);
20         this.child1 = child1;
21        this.child2 = child2;
22        this.child3 = child3;
23        this.child4 = child4;
24     }
25
26     /**
27     * Return the String representation of this subtree, which is the String
28     * value of its ID, followed on the next four indented lines by the
29     * recursive toString of its four children. See the documentation for <a
30     * href="#toString()"> TreeNode.toString() </a> for a general
31     * description the way trees are represented as strings.
32     */
33     public String toString(int level) {
34         String indent = "";
35         for (int i = 0; i < level; i++) {
36             indent += " ";
37         }
38         return symPrint(id) + toStringLineAndColumn(" ") + "\n" +
39                 indent + * * + (child1 == null ? "null" : child1.toString(level+1)) + "\n" +
40                 indent + * * + (child2 == null ? "null" : child2.toString(level+1)) + "\n" +
41                 indent + * * + (child3 == null ? "null" : child3.toString(level+1)) + "\n" +
42                 indent + * * + (child4 == null ? "null" : child4.toString(level+1));
43     }
44
45     /**
46     * Reference to the left child of this node. */
47     public TreeNode child1;
48
49     /**
50     * Reference to the second middle (or third, or next-to-the-last) child of
51     * this node. */
52     public TreeNode child2;
53
54     /**
55     * Reference to the right (or fourth, or last, or rightmost, or
56     * whatever-the-heck-you-want-to-call-it) child of this node. */
57     public TreeNode child3;
58
59     /**
60     * Reference to the third (or second, or second-from-the-last) child
61     * of this node. */
62     public TreeNode child4;