```
1  /****
2   *
3   * TreeNodeList extends TreeNode by adding a node component and a siblings
4   * component.  The node is a reference to a single TreeNode.  The siblings
5   * component is a reference to a TreeNodeList, which may contain zero or more
6   * additional TreeNodes.  Hence, TreeNodeList is used to represent list
7   * constructs in a parse tree.  It can equivalently be viewed as a way to
8   * represent n-ary constructs.
9   *
10  */
11 public class TreeNodeList extends TreeNode {
12
13     /**
14      * Construct this with the given id and child TreeNode references.
15      */
16     public TreeNodeList(TreeNode node, TreeNodeList siblings) {
17         this.node = node;
18         this.siblings = siblings;
19     }
20
21     /**
22      * Return the String representation of this subtree, which is the recursive
23      * toString of each of its nodes, separated by a ';' on a new line plus
24      * another blank line.  See the documentation for <a href=
25      * "TreeNode.html#toString()"> TreeNode.toString() </a> for a general
26      * description the way trees are represented as strings.
27      */
28     public String toString(int level) {
29         String indent = "";
30         for (int i = 0; i < level; i++) {
31             indent += "  ";
32         }
33         if (siblings == null) {
34             return node == null ? " " : node.toString(level);
35         }
36         else {
37             return (node == null ? "" : node.toString(level)) + "\n" +
38                 indent + "  ;\n" + indent + siblings.toString(level);
39         }
40     }
41
42     /** Reference to the first node of this (sub)list. */
43     public TreeNode node;
44
45     /** Reference to the rest of this (sub)list */
46     public TreeNodeList siblings;
47
48 }
49
```