```java
/****
 *
 * TreeNode is the abstract parent class for a parse tree node.  It contains an
 * integer ID data field that is common to all types of node.  The ID defines
 * what type of tree node this is, e.g., an IF node, a PLUS, etc.  The ID
 * values are those defined for symbols in <a href="sym.html">sym.java</a>.
 *                                                                          <p>
 * Extensions of TreeNode add additional data fields to hold information
 * necessary for a particular node type.  The TreeNode extensions are the
 * following:
 *                                                                     <ul><li>
 *     <a href="TreeNode1.html">TreeNode1</a> -- a node with one subtree
 *              reference, used to define unary expressions, or other unary
 *              constructs, such as a single declaration
 *                                                                      <p><li>
 *     <a href="TreeNode2.html">TreeNode2</a> -- a node with two subtree
 *              references, used to define binary expressions, or other binary
 *              constructs, such an assignment statement
 *                                                                      <p><li>
 *     <a href="TreeNode3.html">TreeNode3</a> -- a node with three subtree
 *              references, used to define trinary expressions, or other
 *              trinary constructs, such as an if-then-else statement
 *                                                                      <p><li>
 *     <a href="TreeNode4.html">TreeNode4</a> -- a node with four subtree
 *              references, used to define quartinary constructs
 *                                                                      <p><li>
 *     <a href="TreeNodeList.html">TreeNodeList</a> -- a node with an
 *              indefinite number of subtree references, used to define node
 *              lists of any form, or equivalently, n-ary constructs
 *                                                                      <p><li>
 *     <a href="LeafNode.html">LeafNode</a> -- a leaf node with value
 *              information, but no subtree references
 *                                                                     </ul><p>
 * See the documentation for each of these extending classes for further
 * detail.
 *
 */
public abstract class TreeNode {

    /**
     * Construct a tree node with id = 0.  This is used, e.g., for nodes in a
     * list, that don't need individual id's.
     */
    public TreeNode() {
        this.id = 0;
    }

    /**
     * Construct a tree node with the given id.
     */
    public TreeNode(int id) {
        this.id = id;
    }

    /**
     * Output the String representation of a pre-order tree traversal.  The
     * value of each node is written on a separate line, with subtree nodes
     * indented two spaces per each level of depth, starting at depth 0 for the
     * root.
     *                                                                      <p>
     * For example, the following tree
     *                                                                      <p>
     *     <img src= "images/expr-tree.gif">
     *                                                                      <p>
     * looks like this from TreeNode.toString
     *                                                                    <pre>
     * +
     *   a
     *   *
     *     b
     *     c
     *                                                                   </pre>
     * The implementation of toString() uses an int-valued overload to perform
     * recursive traversal, passing an incrementing level value to successive
     * recursive invocations.  See the definitions of toString(int) in each
     * TreeNode extension for further details.
     */
    public String toString() {
        return toString(0);
    }

    /**
     * This is the recursive work-doer for toString.  See its definition in
     * extending classes for details.
     */
    public abstract String toString(int level);


    /**
     * Print a readable string value for a numeric-valued tree ID.  This method
     * uses the mapping defined in the symNames class.
     */
    public static String symPrint(int id) {
        return symNames.map[id];
    }

    /** The ID of this node.  Yea, it's public.  Take that, you pain-in-the-xxx
     * software engineers. */
    public int id;

}
```