```
/****
 *
 * A SymbolTableEntry holds semantic information for a symbol declared in a
 * program.  The information is the string name of the symbol and its declared
 * data type.
 *                                                                        <p>
 * SymbolTableEntry is an abstract class, with extensions for variable data
 * entries and functional entries.  These are defined, respectively, in the <a
 * href= VariableEntry.html> VariableEntry </a> and <a href=
 * FunctionEntry.html> FunctionEntry </a> classes.
 *
 */
public abstract class SymbolTableEntry {

    /**
     * Construct this with null data fields.
     */
    public SymbolTableEntry() {
    }

    /**
     * Construct this with the given data field values.
     */
    public SymbolTableEntry(String name, TypeNode type) {
        this.name = name;
        this.type = type;
    }

    /**
     * Return toString(0).
     */
    public String toString() {
        return toString(0);
    }

    /**
     * Return toStringDeep(0).
     */
    public String toStringDeep() {
        return toStringDeep(0);
    }

    /**
     * Return the string representation of this' two fields, with other fields
     * added by this' extensions.  The fields are output on a single line,
     * indented level * 2 blanks.  Only the shallowest type string is output,
     * namely just its root tree ID.  The toStringDeep method outputs the full
     * type field.
     */
    public String toString(int level) {
        return doToString(level,
            type != null ? TreeNode.symPrint(type.id) : "null");
    }

    /**
     * Version of toString that dumps out the full type structure, not just its
     * root ID.
     */
    public String toStringDeep(int level) {
        return doToString(level, "\n" + type.toString(level + 30));
    }

    /**
     * Common work doer for other toStrings.
     */
    protected String doToString(int level, String type) {
        return indentString(level) + "Symbol: " + name + ", Type: " + type;
    }

    /**
     * Convenience method for creating an indent string.
     */
    protected String indentString(int level) {
        String indent = "";

        for (int i = 0; i < level; i++) {
            indent += "  ";
        }

        return indent;
    }


    /** This symbol's name */
    public String name;

    /** Data type, represented as a parse tree. */
    public TypeNode type;

}
```