SymbolTableEntry.java

/*
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.

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, TreeNode 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,
   "\n" + type.toString(level + 30));
 }

 public String toStringDeep(int level) {
  return doToStringDeep(level, String type) {
   return indentString(level) + "Symbol: " + name + ", Type: " + type;
  }

 public String indentString(int level) {
  String indent = "\n" + " ";
  for (int i = 0; i < level; i++) {
   indent += "\n" + " ";
  }
  return indent;
 }

 public String doToStringDeep(int level, String type) {
  return doToStringDeep(level,
   type != null ? TreeNode.symPrint(type.id) : "null");
 }

 /* root ID.
 */
 public String toStringDeep(int level) {
  return doToStringDeep(level, "\n" + type.toStringDeep(level + 30));
 }

 /**
 * Common work doer for other toStrings.
 */
 protected String doToStringDeep(int level, String type) {
  return indentStringDeep(level) + 
   type.toStringDeep(level + 30);