FunctionEntry.java

1 /****
2 * FunctionEntry extends SymbolTableEntry by adding data fields to support
3 * functions, procedures, and methods. These forms of functional construct are
4 * considered equivalent for the purposes of storing data in a symbol table.
5 * <p>
6 * The public data fields of a FunctionEntry are a TreeNodeList of formal
7 * parameters, a TreeNode body, and a SymbolTable scope. The inherited type
8 * field is used to hold the return type of the function.
9 * <p>
10 * The scope field holds a reference to the function's own local scope. All of
11 * the function's formal parameters and local variables are entered in this
12 * local table. In this way, the table defines a scope that belongs to the
13 * function, which is the standard semantics in block-structured programming
14 * languages.
15 * <p>
16 * In programming languages that allow nested function definitions, a
17 * function's local scope may have further nested scopes. These are
18 * represented simply by having function entries in a parent function's scope
19 * table. Nested symbol tables are also used to represent anonymous inner
20 * scopes, such as nested declaration/statement blocks, in languages that all
21 * such constructs. See the documentation of the SymbolTable class for a
22 * large-grain picture and description of nested scope representation.
23 * <p>
24 * A function's formal parameters are stored both in the formals list as well
25 * as being entered in the local symtab scope. The list is necessary when
26 * parameters need to be accessed in left-to-right declared order. The formals
27 * are also entered in the function's local scope, so they have a storage
28 * identity that is distinct to this scope.
29 * <p>
30 * The body data field of a function is a reference to the entire parse tree
31 * for its executable body. This tree is used for back-end processing, which
32 * can include one or more of the following phases: type checking,
33 * interpretation, and/or code generation.
34 * *
35 */
36 public class FunctionEntry extends SymbolTableEntry {
37  
38  /**
39  * Construct this with null data fields.
40  */
41  public FunctionEntry() {
42  }
43  
44  /**
45  * Construct this with the given data field values.
46  */
47  public FunctionEntry(String name, TreeNode type, TreeNodeList formals,
48   TreeNode body, SymbolTable scope) {
49    super(name, type);
50    this.formals = formals;
51    this.body = body;
52    this.scope = scope;
53  }