Lab: Object orientation: Extending the CLASSES language
CSC 312P, “Programming language implementation: the good parts”
Department of Computer Science · Grinnell College

Exercises

A. Create a subdirectory called CLASSES-LAB in the directory you’re using for labs in this course and copy my implementation of the CLASSES language into it. The seven language-specific modules are in the /home/stone/courses/languages/code/CLASSES directory.

- tokens.scm
- syntax-trees.scm
- expvals-and-environments.scm
- stores.scm
- scanner.scm
- parser.scm
- interpreter.scm

There is one new helper module this time, which you should copy into the parent directory of CLASSES-LAB from /home/stone/courses/languages/code/maybe.scm.

B. Today’s exercise will be to add a new expression type to the CLASSES language, with the syntax

\[
\text{<expression>} ::= \text{instanceof } \text{<expression>} \text{<identifier>}
\]

The value of an instanceof-expression is always a Boolean value. It is the true Boolean value if the value of the subexpression that follows the new keyword instanceof is an object that instantiates the class denoted by the identifier or any of its subclasses. It is the false Boolean value if the value of the subexpression is not an object or is an object that does not instantiate the class denoted by the identifier or any of its subclasses. It is an error if the identifier does not denote a class.

Here are the steps in the process of getting instanceof-expressions to work.

- Revise the tokens module, adding a token for the new keyword.
- Revise the scanner module so that get-token will return the new token when it is encountered in the source code.
- Revise the grammar in the syntax-trees module, adding instanceof-expressions to the formal syntax.
- Revise the datatype definition for expressions to include a variant for instanceof-expressions.
- Revise the parser module so that it can construct syntax trees for instanceof-expressions when they are encountered in the input. (You’ll also need to add a new variant in cases-statements in some of the error-reporting procedures to deal with the new kind of token that the scanner can produce.)
- Revise the interpreter module so that it carries out the instantiation test as described and constructs and returns the appropriate boolean value.
- Devise and run tests for the code that you have written.