CS 784 Programming Languages

HW4: On EOPL3 CLASSES

Fall 2011

Recall that CLASSES is the name of the toy language defined in EOPL3 OOPL chapter. This HW consists of the following exercises.

EOPL3 Exercise 9.2: Inheritance can be dangerous, because a child class can arbitrarily change the behavior of a method by overriding it. Define a class `bogus-oddeven` that inherits from `oddeven` and overrides the method `even` so that `let o1 = new bogus-oddeven() in send o1 odd (13)` gives the wrong answer.

EOPL3 Exercise 9.10: Some object-oriented languages include facilities for named-class method invocation and field references. In a named-class method invocation, one might write `named-send c1 o m1()`. This would invoke c1's m1 method on o, so long as o was an instance of c1 or of one of its subclasses, even if m1 were overridden in o's actual class. This is a form of static method dispatch. Named-class field reference provides a similar facility for field reference. Add named-class method invocation, field reference, and field setting to the CLASSES language.

EOPL3 Exercise 9.11: Add to CLASSES the ability to specify that each method is either `private` and only accessible from within the host class, `protected` and only accessible from the host class and its descendants, or `public` and accessible from anywhere. Many object-oriented languages include some version of this feature.

EOPL3 Exercise 9.12: Add to CLASSES the ability to specify that each field is either `private`, `protected`, or `public` as in exercise 9.11.

EOPL3 Exercise 9.13: To defend against malicious subclasses like `bogus-oddeven` in exercise 9.2, many object-oriented languages have a facility for `final` methods, which may not be overridden. Add such a facility to CLASSES, so that we could write

```
class oddeven extends object
  method initialize() 1
  final method even(n)
    if zero?(n) then 1 else send self odd(-(n,1))
  final method odd(n)
    if zero?(n) then 0 else send self even(-(n,1))
```

What to hand-in?

Answer Exercise 9.2 in a file named `exercise902.scm`, using your answer as the argument string to the run method. Include one or two test results.

Use the code of EOPL3, `chapter9/classes`, as the base for HW4. Modify these files as needed. Provide the combined code result for all exercises 9.10 to 9.13 together. Do not submit the resulting files at the end of each exercise; just submit the cumulative results. Submit the revised files, keeping the file names as they were in `chapter9/classes`, namely `classes.scm drscheme-init.scm interp.scm store.scm top.scm data-structures.scm environments.scm lang.scm tests.scm`. Include a `ReadMe.txt`.

Also, in order for me to better grasp the changes you made, comment appropriately. Comment out the code you are modifying, rather than deleting it. Include test calls and outcomes in your turned in solution.

Submit on `gandalf.cs.wright.edu` as follows:

```
% ~pmateti/CS784/turnin HW4 ReadMe.txt exercise902.scm\
    classes.scm drscheme-init.scm interp.scm store.scm\
    top.scm data-structures.scm environments.scm lang.scm tests.scm
```

Should the turnin fail for some reason, make a tar-ball or zip of all these files, and email to `pmateti@wright.edu` with a subject line of "CS784/HW4".

Prabhaker Mateti • (937) 775-5114