Quiz 7a

1. (2 points) We want to create an object-oriented system for representing trees in a particular region (the living ones, not the CS ones). What should each of the following be? Choose out of the following list:

Class variable, Instance variable, Method, Object, Subclass.

* 1. General Sherman (biggest tree in the world)
  2. Number of trees in the Amazon
  3. Photosynthesis (making food)
  4. Coniferous (cone-shaped) trees

1. (3 points) Andrew doesn’t like Piazza. He thinks that we should write our own, better version of Piazza. However, he’s really busy and so has hired you to help. He wants you to create objects that represent the things he needs to work with in Piazza.

Write a question-thread class that represents an entire question thread on Piazza. It accepts a question as an instantiation argument. It should store the question, the student answer, the instructor’s answer, and a list of follow-ups. A follow-up is an object which has one method: add-comment, which adds a comment (taken as an argument) to that follow-up. Your question-thread class should support the following methods:

* unanswered? returns #t if there is no answer (student or instructor), else #f.
* set-student-answer accepts 1 argument, and sets the student answer to that argument.
* set-instructor-answer is the corresponding method for the instructor answer.
* new-follow-up accepts a follow-up as an argument, and stores it.
* add-comment accepts an index j and a comment c, and adds c to the jth follow-up. (The first follow-up to be added using new-follow-up would be the 0th follow-up.)
* There should be methods to access the answers and the follow-ups.

1. (1+1+3 points) Define a class ideal-fruit-shop that has a fruit and a corresponding price. An ideal-fruit-shop accepts the message buy, which takes as argument a quantity. buy should return the total price of the purchase.

> (define orange-shop (instantiate ideal-fruit-shop ‘orange 3))

> (ask orange-shop ‘buy 4)

12

Louis Reasoner looked at the code you wrote above, and was confused. “You take in a fruit as an argument when creating an ideal-fruit-shop, but you never use it in any of your methods! Why bother having it at all?”

You answer that there actually is a purpose for having the fruit as an instantiation variable. What is this purpose?

As the name suggest, an ideal-fruit-shop is ideal. Let’s make it more realistic. A fruit-shop inherits from ideal-fruit-shop, but it also has an amount of the fruit that the fruit shop has. When we send a buy message to a fruit-shop, it must check if the fruit shop has enough of the fruit to sell. If it doesn’t, it executes (error “Not enough fruit”). If it does, it must reduce the amount of the fruit by the appropriate amount (and then return the price of the purchase). Write the fruit-shop class. It **must** take advantage of inheritance wherever possible.

> (define orange-shop (instantiate fruit-shop ‘orange 3 10))

> (ask orange-shop ‘buy 6)

18

> (ask orange-shop ‘buy 6)

Error: Not enough fruit