CS 60A Course summary

You aren't expected to understand this yet, but keep it for reference during the semester and see if it starts to make sense!

ABSTRACTION:
voluntary submission to a discipline in order to gain expressive power

----

1. FUNCTIONAL PROGRAMMING

   focus: repeatable input-output behavior
   composition of functions to layer complexity
   hidden: side effect mechanisms (assignment)
   internal control structure of procedures

2. DATA ABSTRACTION

   focus: semantic view of data aggregates
   hidden: actual representation in memory

3. OBJECT ORIENTED PROGRAMMING

   focus: time-varying local state
   metaphor of many autonomous actors
   hidden: scheduling of interactions within the one computer
   procedural methods within an object

4. STREAMS

   focus: metaphor of parallel operations on data aggregates
   signal processing model of computation
   hidden: actual sequence of events in the computation

5. PROGRAMMING LANGUAGES

   focus: provide a metaphor for computation
   embody common elements of large groups of problems
   hidden: technology-specific implementation medium
   storage allocation, etc.

6. LOGIC PROGRAMMING

   focus: declarative representation of knowledge
   inference rules
   hidden: inference algorithm

Note: each of these abstractions can be approached "from above," focusing on the view of computing that the abstraction provides, or "from below," focusing on the techniques by which the abstraction is implemented. In the metacircular evaluator we emphasize the view from below, since we've been working all along with the view from above. In the query evaluator we emphasize the view from above, barely mentioning the implementation techniques. In our discussion of object programming both views are used.

Topic: Review

Reading: No new reading; study for the final.

• Go over first-day handout about abstraction; show how each topic involves an abstraction barrier and say what's above and what's below the line.

• Go over the big ideas within each programming paradigm:

Functional Programming:
composition of functions
first-class functions (function as object)
higher-order functions
recursion
delayed (lazy) evaluation
(vocabulary: parameter, argument, scope, iterative process)

Object-Oriented Programming:
actors
message passing
local state
inheritance
identity vs. equal value
(vocabulary: dispatch procedure, delegation, mutation)

Client/Server Programming:
event-driven process (idle if nothing to do)
callback from operating system for events
cooperation among separate computers
(vocabulary: client, server, IP address, port, socket, thread)

Logic Programming:
focus on ends, not means
multiple solutions
running a program backwards
(vocabulary: pattern matching, unification)

• Review where 61A fits into the curriculum. (See the CS abstraction hierarchy in week 1.)

Please, please, don't forget the ideas of 61A just because you're not programming in Scheme!