6.001 SICP Spring 2004 4/5/2004

Tutorial 8

Upcoming Deadlines

- Problem set 7 due Tuesday, 4/6 at midnight
- Lecture 16 due Wednesday, 4/7 at 9am
- Lecture 17 LIVE on Thursday, 4/8 at 10am
- Project 4 computer exercise 8 due Wednesday, 4/7 via e-mail
- Project 4 due Friday, 4/9 at 6pm
- Office hours next week: Tuesday 4/13, 3-5pm in 34-501
- Quiz 2 Tuesday, 4/13 7:30pm-9:30pm

Sonnet 6001

Double double-oh-one trouble
Lambda turn to double bubble
Params and body annotated
Pointing where evaluated
If combination be in question
First eval each subexpression
Applying first to rest yields trouble
When thy first be double bubble
To apply beget a frame
Where bound each value be to name
Point thy frame where bubbles point
Frame and bubble be now joint
Body's value in this frame
Is what thy combo's val became

–William Shakespeare

Fun with OOP Queue

```
(define (create-queue) QUESTION-1)
```

```
(define (make-queue QUESTION-2)
  (let (QUESTION-3)
    (lambda (message)
      (case message
        ((TYPE) (lambda () (type-extend 'queue QUESTION-4)))
        ((ENQUEUE) QUESTION-5
        ((DEQUEUE) QUESTION-6
        ((EMPTY?) QUESTION-7
        (else (get-method message QUESTION-8))))))
```

I'm Hungry!

```
(define (create-tummy name) QUESTION-9)

(define (make-tummy self name)
  (let (QUESTION-10)
    (lambda (msg)
      (case msg
        ((TYPE) (lambda () (type-extend 'tummy QUESTION-11)))
        ((EMPTY?) (lambda ()
                    (ask queue 'EMPTY?)))
        ((ADD-THING)
         (lambda (new-thing)
           QUESTION-12))
        ((EJECT)
         (lambda ()
           (let ((thing QUESTION-13))
             (cond ((null? thing) nil)
                   ((ask self 'HAVE-THING? thing)
                    (ask place-part 'DEL-THING thing)
                    thing)
                   (else (ask self 'EJECT))))))
        (else (get-method msg QUESTION-14))))))

(define (create-hungry-dude name birthplace activity miserly)
  QUESTION-15)

(define (make-hungry-dude self name birthplace activity miserly)
  (let (QUESTION-16)
    (lambda (msg)
      (case msg
        ((TYPE) (lambda () (type-extend 'hungry-dude QUESTION-17)))
        ((INSTALL)
         (lambda ()
           QUESTION-18))
```

```
((EAT)
 (lambda ()
   (let ((people (ask self 'PEOPLE-AROUND)))
     (map (lambda (p)
            (ask p 'CHANGE-LOCATION tummy)
            (ask self 'SAY `(,(ask p 'NAME) " is tasty!")))
          people))))
((EJECT)
 (lambda ()
   (if (and (not (ask self 'EMPTY?)) (= (random 2) 0))
       (let ((p QUESTION-19))
         (ask p 'CHANGE-LOCATION (ask self 'LOCATION))
         (ask self 'SAY `("I don't need " ,(ask p 'NAME)))))))
((DIE)
 (lambda (perp)
   (let ((people (ask self 'THINGS))
         (location (ask self 'LOCATION)))
     (map (lambda (p)
            (ask p 'CHANGE-LOCATION location)
            (ask p 'SAY '("It was cramped in there")))
          people))
   (ask clock 'REMOVE-CALLBACK self 'eat)
   (ask clock 'REMOVE-CALLBACK self 'eject)
   (ask self 'DIE perp)))
(else (get-method msg QUESTION-20))))))
```