Objects combine data and procedural abstractions. What does that mean? Before, when talking about Abstract Data Types, we would create a model for the types, and separately write procedures for creating and manipulating the abstract type. Object-oriented programming allows us to combine the procedures with the data it manipulates. In effect, this is one way to enforce that the abstractions are followed.

Here’s a definition for a (not very interesting) animal class:

```
(define (create-animal name)
  (create-instance animal name))

(define (animal self name)
  (let ((named-part (named-object self name)))
    (make-handler
     'ANIMAL
     (make-methods
      'DRINK (lambda () (display-message (list "slurp")))
      'EAT (lambda () (display-message (list "crunch crunch")))
     )
    named-part)))

(define fluffy (create-animal 'fluffy))
(ask fluffy 'NAME)
```

1. What are all the messages that fluffy can respond to?

2. Draw a class diagram for animals
Now here's a cat class that derives from the animal class, and specific instance of a cat, garfield.

(define (create-cat name)
  (create-instance cat name))

(define (cat self name)
  (let ((animal-part (animal self name))
         (mood 0))
    (make-handler
      'CAT
      (make-methods
        'FETCH
        (lambda ()
          (display "What did you throw your ball for?\n"
          (set! mood (- mood 3)))
        'MOOD
        (lambda () (if (>= mood 4) 'content 'angry))
        'INSTALL
        (lambda ()
          (ask animal-part 'INSTALL)
          (display-message (list "I am" (ask self 'NAME) "yaawn"))
          (ask our-clock
            'ADD-CALLBACK
            (create-clock-callback 'EAT-CB self 'EAT)))
        'EAT
        (lambda ()
          (ask animal-part 'EAT)
          (set! mood (+ mood 1)))
          animal-part
        )))

(define garfield (create-cat 'garfield))

3. What will the following evaluate to:

(ask garfield 'MOOD)
(ask our-clock 'TICK)
(ask our-clock 'TICK)
(ask garfield 'MOOD)
(ask our-clock 'TICK)
(ask our-clock 'TICK)
(ask garfield 'MOOD)
4. Add a method to the cat class called 'GREET. If the cat is content, the cat should (display) “purr”, otherwise “hiss”.

5. Write a class definition for a dog, and then create an instance of a dog named odie. Dogs should respond to the same methods as cats, but with opposite effects – Odie should grow more unhappy as time progresses, rather than being happier the longer he’s left alone, and any attention (such as being asked to 'FETCH) should improve his mood.
6. Draw a complete class diagram that includes both dogs and cats.