```
(define meter1
   (let ((amount 0))
      (lambda ()
         (lambda (m)
            (cond ((equal? m 'deposit)
                     (set! amount (+ amount .25)))
                  ((equal? m 'total) amount)
                  ((equal? m 'collect)
                     (let ((amt amount))
                        (set! amount 0)
                        amt))
                  (else 'Eh?))))))


(define meter2
   (lambda ()
      (let ((amount 0))
         (lambda (m)
            (cond ((equal? m 'deposit)
                     (set! amount (+ amount .25)))
                  ((equal? m 'total) amount)
                  ((equal? m 'collect)
                     (let ((amt amount))
                        (set! amount 0)
                        amt))
                  (else 'Eh?))))))


(define p1 (meter1))
(define p2 (meter1))

(define q1 (meter2))
(define q2 (meter2))
```

1. What is the difference between how meter1 and meter2 work? (Hint: Do testing with p1, p2, q1, and q2.)

With meter1, the let is before the lambda so any function called on p1 will also affect p2. On the other hand with meter2, the let is after the lambda so q1 and q2 are independent.

2. What is the technical term for the variable amount in meter1?

Class variable

3. What keyword in Java is associated with the variable amount in meter1?

??

4. What is the technical term for the variable amount in meter2?

Instance variable

5. Draw an environment diagram for meter1, meter2, p1, p2, q1, and q2. (You will probably need a whole sheet of paper to do this as there is a lot of stuff to draw.)
6. Can you tell from the diagram why it is that the different types of parking meters behave as they do?

In the diagram for meter1, p1 and p2 both point to the same amount so when it is updated, both p1 and p2 are affected. In meter2, q1 and q2 each have their own variable for amount so they are updated separately.

```
(define meter3
   (let ((total-amount 0))
      (lambda ()
        (let ((amount 0))
          (lambda (m)
            (cond ((equal? m 'deposit)
                   (set! amount (+ amount .25))
                   (set! total-amount (+ total-amount .25)))
                  ((equal? m 'amount) amount)
                  ((equal? m 'total) total-amount)
                  ((equal? m 'collect)
                   (let ((amt amount))
                     (set! total-amount (- total-amount amount))
                     (set! amount 0)
                     amt))
                  (else 'Eh?)))))))

(define a1 (meter3))
(define a2 (meter3))
```

- What will total-amount and amount be after (a1 'deposit) is performed three times?

0.75

- Suppose (a2 'deposit) were also performed one time. What are total-amount and amount now?

The amount for a1 is still 0.75 but the total-amount is now 1.0