Index of /~cs61b/lectures/03

Name	Last modified	Size	Description
Parent Directory	24-Sep-98 21:22	-	
Bank.java	28-Aug-98 07:39	2k	
bank.scm	28-Aug-98 07:39	1k	

```
                     CS61b: Lecture 3
                    Friday, August 28

ANNOUNCEMENTS:
-------------

CLASSES: IMPLEMENTING ABSTRACT DATA TYPES IN JAVA
-------------------------------------------------

What is an abstract data type?
   Some data (variables with their values)
 + A set of operations on that data

A SCHEME EXAMPLE: BANK ACCOUNT
------------------------------
(define (make-account balance)
    (define (with amount)
      (if (>= balance amount)
         (set! balance (- balance amount))
         "Insufficient funds"))
    (define (dep amount)
         (set! balance (+ balance amount))
         balance)
    (define (state)
         balance)
    (define (dispatch m)
      (cond ((eq? m 'withdraw) with)
            ((eq? m 'deposit) dep)
            ((eq? m 'statement) state)
            (else (error "Unknown request" m))))
    dispatch)

1) What does (make-account 200) return?
2) How do we withdraw money?

Design:
 1) what operations do we need?
       make, withdraw, deposit, statement
 2) what data needs to be kept?
       current balance
 3) protection
       want to prevent overdrafts.  balance > 0 is an "invariant."
       could require a passwd on withdrawals.
       do not allow direct modification of the balance.
```

General strategy (common, but not always true):
  1) only methods are visible outside
  2) most data and sometimes helper functions are not

BANK ACCOUNT IN JAVA

```
  class Account {
     some functions (called methods)
     some variables that each object will have (called fields)
  }
```

Account is the name of the class.
Account is the type of all objects that are instances of the class.

How do we access the balance value inside the object?
   1. balance
   2. this.balance (the balance value inside "this" object)

CONSTRUCTORS

A constructor is like a method, but:
  - has the same name as the class
  - has no return type declared (because it always returns some of the class)
  - Java will make up a "no-arg" default if you don't define any (not a good idea)

We can use this Account class by creating object of type Account.

```
       Account myAcct = new Account (200);
       myAcct.withdraw(50);
       x = myAcct.balance;  <<-- what's wrong with this?
```

In general, we can say: objectname.fieldname to access a data field or
                          objectname.methodname(arguments) to invoke a method

However: private things can only be accessed by the methods in the class
            public things can be accessed anywhere

FULL CODE FOR ACCOUNT CLASS

```
class Account {
    public Account(int initBalance) {
       balance = initBalance;
    }
    public void deposit(int amount) {
        balance += amount;
    }
    public void withdraw(int amount) {
       if (balance >= amount) {
           balance -= amount;
       }
       else {
         System.out.println("Insufficient funds");
       }
    }
    public void statement() {
       System.out.println("Current balance is " + balance);
    }
    public void transfer(Account other, int amount) {
      if (other.balance > amount) {
          balance += amount;
          other.balance -= amount;
      }
      else {
          System.out.println("Insufficient funds");
```

```
      }
    }
    /* invariant: balance >= 0 */
    private int balance;
}
```

```
SCHEME                                  JAVA
------------------------------------------------------------------
(let ((x 1)) ...)                       int x = 1;
(set! x 3)                              x = 3;
(+ x 1)                                 x + 1
(set! x (+ x 1))                        x = x + 1; or x+=1; or x++;
(and b1 b2)                             (b1 && b2)
(or b1 b2)                              (b1 || b2)
(not b1)                                (! b1)
(eq? b1 b2)                             (b1 == b2)
;; comment                              /* comment */ or // comment
                                        for (i = 0; i < n; i = i+1) { ... }
(if b e1 e2)                            if (b) { e1 } else { e2 }
```