

Exercise Sheet 7

Complete the **first question** during the exercise class on 26 March, 2010, and hand it to the tutor at the end of the hour. The remaining questions are unassessed, and meant to be used during your revision for the final exam.

In-class Exercise

Exercise 1: Objects

Given below is a class for bank account objects (as a “new” procedure) and a sample piece of code using it. The class is written in applied lambda calculus with assignments, similar to Scheme, but using structs to represent structures with multiple components (instead of tuples or message dispatch functions).

```
let newaccount =
  λ(initial).
    let balance = initial
    in {deposit =
        λ(amount). balance := balance + amount;
        wdraw =
          λ(amount). if balance > amount then
                      balance := balance - amount
                      else print(`insufficient funds`);
        getBalance =
          λ(). balance
    }
in let
  Mary = newaccount(500);
  Doug = newaccount(200);
  Lisa = Doug
in
  (Mary.wdraw(200);
   Lisa.deposit(200);
   Doug.wdraw(200);
   Mary.getBalance()
  )
```

- Write equivalent code in Java with a class definition corresponding to `newaccount` and statements invoking the class in the fashion shown.
- Draw an environment diagram showing all the frames and variable values at the end of the inner `let` body.
- Add a method called `transfer` to the account class. It should take as arguments another account object to which money should be transferred and the amount to be transferred.
- Write a procedure `payInterest`, which takes a list of account objects and deposits 5% interest in all of them.

Unassessed homework exercises

Exercise 2: Higher order procedures

- Write a procedure called `mapdo` which takes as its arguments

- a one-argument procedure `p`, and
- a list of items `items`

and applies `p` to all the items in `items` in the left-to-right order.

- Use the procedure `mapdo` to rewrite the `payInterest` procedure. You should not use recursion in this definition.
- Use the procedure `mapdo` to calculate the total balance of a list of accounts. Again, you should not use recursion in this definition.
- Explain how the total balance procedure works, showing the environment frames that arise during its execution.

Exercise 3: More objects

- Write a class `newbuffer2` for two-place buffers, which can hold at most two items. The objects should have:
 - a `put` method, using which an item can be placed in the buffer,
 - a `get` method, using which an item can be retrieved from the buffer, and
 - a method `present` which tells whether there are any items in the buffer.

Items should be retrieved in the same order in which they were placed.

If `put` is called when the buffer is full or if `get` is called when the buffer is empty, print an error message and take no action.

- Next write a class `newbuffer`, for buffers that can hold an unbounded number of items. You should not use lists or any other data structure to write this class. (**Hint:** Can you implement a two-place buffer using two one-place buffers?)