「プログラミング言語」課題

1029-24-9540 山崎啓太郎

April 24, 2013

1 Ex. 3.1

1.1 考え方

make-accumulator で返される関数内で、make-acumulator に与えられた引数に破壊的代入を使うことにより、それを更新していく。
3.1.scm では make-accumulator 関数の引数 x に対して、`(set! x (+ x add))` とすることで x を更新している。

1.2 実行例

test/3.1.scm

```
1  (use gauche.test)
2  (add-load-path ".")
3
4  (test-start "3.1")
5
6  (load "3.1")
7  (define A (make-accumulator 5))
8
9  (test* "(A 10)" 15 (A 10))
10 (test* "(A 10)" 25 (A 10))
```

結果

```
Testing 3.1 ...
test (A 10), expects 15 ==> ok
test (A 10), expects 25 ==> ok
```

2 Ex. 3.3

2.1 考え方

make-account 関数で与えられたパスワードは、口座の操作ごとに入力する仕組みになっている。
3.3.scm では口座の操作の際に dispatch 関数が呼ばれているため、dispatch 関数内で口座の操作をする前にパスワードが正しいかどうかを判断し、パスワードが正しければ操作をし、正しくなければ”Incorrect password”を返せばよい。

2.2 実行例

test/3.3.scm

```
1  (use gauche.test)
2  (add-load-path ".")
3
4  (test-start "3.3")
5
6  (load "3.3")
7
8  (test-section "Account test")
9  (define acc (make-account 100 'secret-password))
10 (test* 引出"30" 70 ((acc 'secret-password 'withdraw) 30))
11 (test* 振込"40" 110 ((acc 'secret-password 'deposit) 40))
12 (test* 引出"300" "Insufficient funds" ((acc 'secret-password 'withdraw) 300)
13 (test* 存在しない操作をする"" *test-error* ((acc 'secret-password 'mes) 10))
14
15 (test-section "Password test")
16 (define acc (make-account 100 'secret-password))
17 (test* 正しいパスワードで引出"40" 60 ((acc 'secret-password 'withdraw) 40))
18 (test* 間違ったパスワードで振込"50" "Incorrect password" ((acc 'some-other-passwo
```

結果

```
Testing 3.3 ...
<Account test>----------------------------------------------------------------
test 30引出, expects 70 ==> ok
test 40振込, expects 110 ==> ok
test 300引出, expects "Insufficient funds" ==> ok
```

```
test 存在しない操作をする, expects #<error> ==> ok
<Password test>----------------------------------------------------------------
test 正しいパスワードで40引出, expects 60 ==> ok
test 間違ったパスワードで50振込, expects "Incorrect password" ==> ok
```

3 Ex. 3.7

3.1 考え方

make-joint は第一引数に make-account で生成された口座（account）、第二引数にその account のパスワード（old-password）、第三引数に新しく作る account 用のパスワード（old-password）を取る。
まず、make-joint では old-password が正しいものか判定しなくてはいけないが、account にはそのための機能が備わっていないため、make-account を改良する必要がある。
3.7.scm では（`account old-password 'check-password`）を評価することで、old-password が正しい場合 true が、間違っていた場合 false が帰ってくるようになっている。
old-password が正しかった場合、新規 account で操作をするための lambda 式を返す。
その lambda 式は make-account で生成された口座と同様に、パスワードと操作のためのメッセージを引数としてとり、与えられたパスワードが new-password と一致していれば account にメッセージを渡す。
以上のようにすれば、make-joint で必要な機能を満たすことができる。

3.2 実行例

test/3.3.scm

```
1  (use gauche.test)
2  (add-load-path ".")
3
4  (test-start "3.7")
5
6  (load "3.7")
7
8  (define peter-acc (make-account 100 'open-sesame))
9  (define paul-acc (make-joint peter-acc 'open-sesame 'rosebud))
10 (test* "peter-を使って引出acc40" 60 ((peter-acc 'open-sesame 'withdraw) 40))
11 (test* さらに"paul-を使って引出acc40" 20 ((paul-acc 'rosebud 'withdraw) 40))
```

```
12 (test* さらに"paul-を使って振込acc30" 50 ((paul-acc 'rosebud 'deposit) 30))
13 (test* "peter-のパスワードを間違えるacc" "Incorrect password" ((peter-acc 'some-
14 (test* "paul-のパスワードを間違えるacc" "Incorrect password" ((paul-acc 'some-oth
```

結果

```
Testing 3.7 ...
test peter-acc を使って 40 引出, expects 60 ==> ok
test さらに paul-acc を使って 40 引出, expects 20 ==> ok
test さらに paul-acc を使って 30 振込, expects 50 ==> ok
test peter-acc のパスワードを間違える, expects "Incorrect password" ==> ok
test paul-acc のパスワードを間違える, expects "Incorrect password" ==> ok
```