Requirement Specification of the Library System

Shin Sahara

目次

1 Library

I am a simple library system to manage borrowing copies of books.

```
class
Library
types
  public Copy = token;
  public User = token;
   Borrowing = map Copy to User
instance variables
  sCopies : set of Copy := {};
  sUsers : set of User := {};
  sBorrowing : Borrowing := { |-> };
  inv dom sBorrowing subset sCopies and
      rng sBorrowing subset sUsers

operations
public
  add_user : User ==> ()
  add_user (aUser) ==
    sUsers := sUsers union {aUser}
  pre  aUser not in set sUsers
  post sUsers = sUsers~ union {aUser} ;
public
  remove_user : User ==> ()
  remove_user (aUser) ==
    sUsers := sUsers \ {aUser}
  pre  aUser in set sUsers and
       not borrowedUser (sBorrowing, aUser)
  post sUsers = sUsers~ \ {aUser} ;
public
  add_book : Copy ==> ()
  add_book (aCopy) ==
    sCopies := sCopies union {aCopy}
  pre  aCopy not in set sCopies
  post sCopies = sCopies~ union {aCopy} ;
public
```

```
  remove_book : Copy ==> ()
  remove_book (aCopy) ==
    sCopies := sCopies \ {aCopy}
  pre  aCopy in set sCopies and
       not borrowedCopy (sBorrowing, aCopy)
  post sCopies = sCopies~ \ {aCopy} ;
public
  borrow_book : User * Copy ==> ()
  borrow_book (aUser, aCopy) ==
    sBorrowing := sBorrowing munion {aCopy |-> aUser}
  pre  aUser in set sUsers and
       aCopy in set sCopies and
       not borrowedCopy (sBorrowing, aCopy)
  post sBorrowing = sBorrowing~ munion {aCopy |-> aUser} ;
public
  return_book : Copy ==> ()
  return_book (aCopy) ==
    sBorrowing := {aCopy} <-:  sBorrowing
  pre  borrowedCopy (sBorrowing, aCopy)
  post sBorrowing = {aCopy} <-:  sBorrowing~ ;
public
  getAttributes : () ==> set of Copy * set of User * map Copy to User
  getAttributes () ==
    return mk_ (sCopies, sUsers, sBorrowing)
functions
  borrowedCopy : Borrowing * Copy +> bool
  borrowedCopy (aBorrowing, aCopy) ==
    aCopy in set dom aBorrowing;
  borrowedUser : Borrowing * User +> bool
  borrowedUser (aBorrowing, aUser) ==
    aUser in set rng aBorrowing
end
Library
```

Test Suite : vdm.tc
Class : Library

Name	#Calls	Coverage
Library‘add_book	32	√
Library‘add_user	32	√

Name	**#Calls**	**Coverage**
Library‘borrow_book	32	√
Library‘remove_book	8	√
Library‘remove_user	16	√
Library‘return_book	24	√
Library‘borrowedCopy	64	√
Library‘borrowedUser	16	√
Library‘getAttributes	24	√
Total Coverage		**100%**

2 TestApp

I am a simple regression test.

```
class
TestApp
operations
public static
  run : () ==> seq of char * bool * map nat1 to bool
  run () ==
    let testcases = [
                      t1 (), t2 (), t3 (), t4 ()],
        testResults = makeOrderMap (testcases) in
    return mk_ ("The result of regression test  =  ", forall i in set inds testcases & testcases (i), testR
functions
public static
  makeOrderMap : seq of bool +> map nat1 to bool
  makeOrderMap (s) ==
    {i |-> s (i) | i in set inds s}
  pre  s <> []
operations
public static
  print : seq of char ==> bool
  print (s) ==
    let - = new IO ().echo (s) in
    return true;

  t1 : () ==> bool
  t1 () ==
    let l = new Library (),
        p = mk_token ("Sakoh"),
        c = mk_token ("OO Construction_1") in
    (   l.add_user (p) ;
        l.add_book (c) ;
        l.borrow_book (p, c) ;
        l.return_book (c) ;
        l.remove_user (p) ;
        return l.getAttributes () = mk_ ({mk_token ("OO Construction_1")}, {}, { |-> })
    ) ;
```

```
 t2 : () ==> bool
 t2 () ==
   let l = new Library (),
       p = mk_token ("Sakoh"),
       c = mk_token ("OO Construction_1") in
   (   l.add_user(p) ;
       l.add_book(c) ;
       l.borrow_book(p, c) ;
       l.return_book(c) ;
       l.remove_book(c) ;
       return l.getAttributes () = mk_ ({}, {mk_token ("Sakoh")}, { |-> })
   ) ;

 t3 : () ==> bool
 t3 () ==
   let l = new Library (),
       p = mk_token ("Sakoh"),
       c = mk_token ("OO Construction_1") in
   trap <RuntimeError>
   with  print("\tt3 Can't remove_user as planned.\n") in
   (   l.add_user(p) ;
       l.add_book(c) ;
       l.borrow_book(p, c) ;
       l.remove_user(p) ;
       return l.getAttributes () = mk_ ({mk_token ("OO Construction_1")}, {}, { |-> })
   ) ;

 t4 : () ==> bool
 t4 () ==
   let l = new Library (),
       p = mk_token ("Sakoh"),
       c = mk_token ("OO Construction_1") in
   (   l.add_user(p) ;
       l.add_book(c) ;
       l.borrow_book(p, c) ;
       l.return_book(c) ;
       return l.getAttributes () = mk_ ({mk_token ("OO Construction_1")}, {mk_token ("Sakoh")}, { |-> })
   )
end
```

TestApp

....

3 UseLibrary

I am a combinatorial test.

```
class
UseLibrary
instance variables
  sL : Library := new Library();

traces
T1 :
  let p in set {mk_token("Sakoh"),mk_token("Larsen")} in
  let c in set {mk_token("OO Construction_1"),mk_token("VDM_1")} in
  (sL.add_user(p);
   sL.add_book(c);
   sL.borrow_book(p, c);
   sL.return_book(c);
   sL.remove_user(p);
   sL.getAttributes())
  ;
; T2 :
  let p1, p2 in set {mk_token("Sakoh"),mk_token("Larsen")} in
  let c1, c2 in set {mk_token("OO Construction_1"),mk_token("VDM_1")} in
  (sL.add_user(p1);
   sL.add_user(p2);
   sL.add_book(c1);
   sL.add_book(c2);
   sL.borrow_book(p1, c1);
   sL.borrow_book(p1, c2);
   sL.getAttributes())
  ;
```

```
; T3 :
  let p1,p2 in set {mk_token("Sakoh"),mk_token("Larsen")} in
  let c1,c2 in set {mk_token("OO Construction_1"),mk_token("VDM_1")} in
  (sL.add_user(p1);
   sL.add_user(p2);
   sL.add_book(c1);
   sL.add_book(c2);
   sL.borrow_book(p2,c1);
   sL.borrow_book(p2,c2);
   sL.remove_user(p1)sL.remove_user(p2);
   sL.getAttributes())
  ;
end
UseLibrary
```

....