Requirement Specification of the Library System

Shin Sahara

目次

```
class
Library
```

0.1 Library

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

```
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	0	0%

Name	**#Calls**	**Coverage**
Library‘add_user	0	0%
Library‘borrow_book	0	0%
Library‘remove_book	0	0%
Library‘remove_user	0	0%
Library‘return_book	0	0%
Library‘borrowedCopy	0	0%
Library‘borrowedUser	0	0%
Library‘getAttributes	0	0%
Total Coverage		**0%**

```
class
UseLibrary
```

0.2 UseLibrary

UseLibrary is a Combinatorial test class.

```
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
```

....