VDM++入門セミナー　演習回答

佐原伸

SCSK株式会社
ソリューション・機能事業部門
開発ソリューション事業本部
開発部
基盤開発課
VDM推進担当
2012年6月21日

型と値の定義

```
class 図書館0
types
public 蔵書 = map 蔵書ID to 本;
public 本 ::
    f題名 : seq of char
    f著者 : seq of char
    f分野集合 : set of 分野;
public 題名 = seq of char;
public 著者 = seq of char;
public 分野 = seq of char;
public 蔵書ID = token;
public 職員 = token ;
public 利用者 = token;
```

1.今後のモデル化で、項目が追加される可能性が強いので、本はレコード型として定義した。

2.蔵書を一意に識別するため、蔵書IDを定義する。

3.蔵書ID、職員、利用者は、他と識別するためにtokenとした。

4.著者などは、モデルを詳細化するに伴いオブジェクトとしてモデル化する可能性があるが、ここでは単に名前だけ管理することにした。

```
values
public v最大蔵書数 = 10000;
public v最大貸出数 = 3;


end 図書館0
```

蔵書を追加する、蔵書を削除する、の陰仕様

```
operations
public 蔵書を追加する : 蔵書 ==> ()
蔵書を追加する(a追加本) == is not yet specified
pre
    蔵書に存在しない(a追加本, i蔵書)
post
    蔵書が追加されている(a追加本, i蔵書, i蔵書~);

public 蔵書を削除する: 蔵書 ==> ()
蔵書を削除する(a削除本) == is not yet specified
pre
    蔵書に存在する(a削除本, i蔵書)
post
    蔵書が削除されている(a削除本, i蔵書, i蔵書~);
```

インスタンス変数と、
蔵書に存在しない、のインタフェース

```
instance variables
private i蔵書 : 蔵書 := {|->};
inv
  card dom i蔵書 <= v最大蔵書数;
```

1.インスタンス変数は蔵書IDから本への写像である蔵書型のi蔵書
  1.1. 濃度(集合の要素数)がv最大蔵書数以下であるという不変条件を持つ。
  1.2.初期値は空の写像である。

```
functions
蔵書に存在しない : 蔵書 * 蔵書 +> bool
蔵書に存在しない(a追加本, a図書館蔵書) == is not yet specified;
```

蔵書に存在しない、蔵書に存在する、の陽仕様

```
functions
蔵書に存在する : 蔵書 * 蔵書 +> bool
蔵書に存在する(a追加本, a図書館蔵書) ==
   dom a追加本 subset dom a図書館蔵書;
-- a追加本 = (dom a追加本 <: a図書館蔵書)
-- forall id in set dom a追加本 &
--    id in set dom a図書館蔵書

蔵書に存在しない : 蔵書 * 蔵書 +> bool
蔵書に存在しない(a追加本, a図書館蔵書) ==
   not 蔵書に存在する(a追加本, a図書館蔵書) ;
```

蔵書が追加されている、蔵書が削除されている、の陽仕様

```
functions
蔵書が追加されている : 蔵書 * 蔵書 * 蔵書 +> bool
蔵書が追加されている(a追加本, a図書館蔵書, a図書館蔵書旧値) ==
    a図書館蔵書 = a図書館蔵書旧値 munion a追加本;

蔵書が削除されている : 蔵書 * 蔵書 * 蔵書 +> bool
蔵書が削除されている(a削除本, a図書館蔵書, a図書館蔵書旧値) ==
    a図書館蔵書 = dom a削除本 <-: a図書館蔵書旧値;
    --a図書館蔵書旧値 = a図書館蔵書 munion a追加本;
```

本を検索する、の陰仕様

```
public 本を検索する: (題名 | 著者 | 分野) ==> 蔵書
本を検索する(a検索キー) ==  is not yet specified
pre
    検索キーが空でない(a検索キー)
post
    forall book in set rng RESULT &
        (book.f題名 = a検索キー or
        book.f著者 = a検索キー or
        a検索キー in set book.f分野集合);
```

forallは全称限量子、RESULTは返値を表す

本を貸す、の陰仕様

```
operations
public 本を貸す: 蔵書 * 利用者 * 職員 ==> ()
本を貸す(a貸出本, a利用者, -) ==  is not yet specified
pre
    貸出可能である(a利用者, a貸出本, i貸出, i蔵書, v最大貸出数)
post
    貸出に追加されている(a貸出本, a利用者, i貸出, i貸出~);
```

本を貸す、の事前条件の関数を作成せよ

```
public 貸出可能である : 利用者 * 蔵書 * 貸出 * 蔵書 * nat1 +> bool
貸出可能である(a利用者, a貸出本, a貸出, a図書館蔵書, a最大貸出数) ==
    蔵書に存在する(a貸出本, a図書館蔵書) and
    貸出に存在しない(a貸出本, a貸出) and
    最大貸出数を超えていない(a利用者, a貸出本, a貸出, a最大貸出数);

貸出に存在しない : 蔵書 * 貸出 +> bool
貸出に存在しない(a貸し出す本, a貸出) ==
    not 貸出に存在する(a貸し出す本, a貸出);

貸出に存在する : 蔵書 * 貸出 +> bool
貸出に存在する(a貸出本, a貸出) ==
    let w蔵書 = merge rng a貸出 in
    dom a貸出本 subset dom w蔵書;
```

最大貸出数を超えていない、の陽仕様

```
public 最大貸出数を超えていない : 利用者 * 蔵書 * 貸出 * nat1  +> bool
最大貸出数を超えていない(a利用者, a貸出本, a貸出, a最大貸出数) ==
    if a利用者 not in set dom a貸出 then
        card dom a貸出本 <= a最大貸出数
    else
        card dom a貸出(a利用者) + card dom a貸出本 <= a最大貸出数;
```

本を貸す、の事後条件の関数を作成せよ

```
貸出に追加されている : 蔵書 * 利用者 * 貸出 * 貸出 +> bool
貸出に追加されている(a貸出本, a利用者, a貸出, a貸出旧値) ==
    if a利用者 in set dom a貸出 then
        a貸出 = a貸出旧値 ++ {a利用者 |-> (a貸出(a利用者) munion a貸出本)}
    else
        a貸出 = a貸出旧値 munion {a利用者 |-> a貸出本};
```

本を返す、の陰仕様

```
operations
public 本を返す: 蔵書 * 利用者 * 職員 ==> ()
本を返す(a返却本, a利用者, -) ==  is not yet
specified
pre
    貸出に存在する(a返却本, i貸出)
post
    貸出から削除されている(a返却本, i貸出);
```

本を返す、の事後条件

```
貸出から削除されている : 蔵書 * 貸出 +> bool
貸出から削除されている(a削除本, a貸出) ==
    let  w蔵書 = merge rng a貸出 in
    dom a削除本 inter dom w蔵書 = {};
```

オブジェクトモデルの問題点

- 蔵書の型を変えると、他のクラスに修正が波及する。
  - 蔵書を型ではなく、クラスとすべきである？
- 書庫クラスと、貸出情報クラスと図書館2クラスの情報隠蔽が不十分である。
  - 図書館2クラスを貸出情報のサブクラスにすべきではない？

演習終了