

VDMTools

VDM++ ソートアルゴリズム



How to contact SCSK:

http://www.vdmtools.jp/	VDM information web site(in Japanese)
http://www.vdmtools.jp/en/	VDM information web site(in English)
http://www.scsk.jp/	SCSK Corporation web site(in Japanese)
http://www.scsk.jp/index_en.html	SCSK Corporation web site(in English)
vdm.sp@scsk.jp	Mail

VDM++ ソートアルゴリズム 2.0

— Revised for VDMTools v9.0.2

© COPYRIGHT 2013 by SCSK CORPORATION

The software described in this document is furnished under a license agreement.
The software may be used or copied only under the terms of the license agreement.

This document is subject to change without notice.

目 次

1	はじめに	1
2	Sort Machine クラス	2
3	Sorter クラス	4
4	MergeSort クラス	5
5	DoSort クラス	7
6	ImplSort クラス	9
7	ExplSort クラス	10

1 はじめに

本書は、ソートの例題を1つ収めている。クラス図は、図 1 で見ることができる。この例題の構造は、*strategy* パターンとして知られているものである。このパターンはアルゴリズムの集団を定義するもので、それぞれ隠蔽し合い交換可能となっている。*strategy* パターンは、それを用いるクライアントから独立したものとしてアルゴリズムを変化させる。異なるソートアルゴリズムを複数用いるクライアントとして、**SortMachine** クラスが置かれている。**Sorter** クラスは、サポートされるアルゴリズムすべてに対して共通なインターフェイスを定義している抽象クラスである。

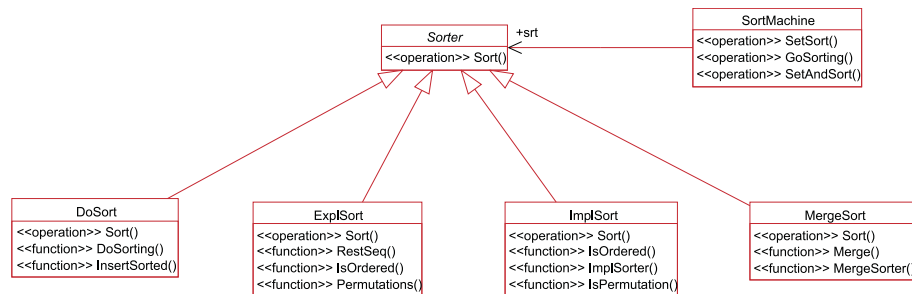


図 1: ソート例題のクラス図

2 Sort Machine クラス

```
class SortMachine
```

```
instance variables
```

```
  srt: Sorter := new MergeSort();
```

インスタンス変数 “srt” は現在使用中のソートアルゴリズムのオブジェクト参照となる。初期設定されるソートアルゴリズムは MergeSort である。

使用するソートアルゴリズムの設定／変更。

```
operations
```

```
  public SetSort: Sorter ==> ()  
  SetSort(s) ==  
    srt := s;
```

現在設定されているソートアルゴリズムによるソート。

```
  public GoSorting: seq of int ==> seq of int  
  GoSorting(arr) ==  
    return srt.Sort(arr);
```

最初にソートアルゴリズムの設定／変更を行った後のソート。

```
public SetAndSort: Sorter * seq of int ==> seq of int
SetAndSort(s, arr) ==
( srt := s;
  return srt.Sort(arr)
)

end SortMachine
```

3 Sorter クラス

```
class Sorter

operations

  public
  Sort: seq of int ==> seq of int
  Sort(arg) ==
    is subclass responsibility

end Sorter
```

4 MergeSort クラス

```
class MergeSort is subclass of Sorter
```

```
operations
```

```
public Sort: seq of int ==> seq of int
Sort(l) ==
  return MergeSorter(l)
```

```
functions
```

```
MergeSorter: seq of real -> seq of real
MergeSorter(l) ==
  cases l:
    []      -> l,
    [e]     -> l,
    others  -> let l1^l2 in set {l} be st abs (len l1 - len l2) < 2
               in
               let l_l = MergeSorter(l1),
                 l_r = MergeSorter(l2) in
                 Merge(l_l, l_r)
  end;
```

```
Merge: seq of int * seq of int -> seq of int
Merge(l1,l2) ==
  cases mk_(l1,l2):
    mk_([],l),mk_(l,[]) -> l,
    others               -> if hd l1 <= hd l2 then
                           [hd l1] ^ Merge(tl l1, l2)
                           else
                           [hd l2] ^ Merge(l1, tl l2)
    end
  pre forall i in set inds l1 & l1(i) >= 0 and
```

```
forall i in set inds l2 & l2(i) >= 0
```

```
end MergeSort
```

5 DoSort クラス

```
class DoSort is subclass of Sorter
```

```
operations
```

```
public Sort: seq of int ==> seq of int
Sort(l) ==
  return DoSorting(l)
```

```
functions
```

```
DoSorting: seq of int -> seq of int
DoSorting(l) ==
  if l = [] then
    []
  else
    let sorted = DoSorting (tl l) in
      InsertSorted (hd l, sorted);
```

```
InsertSorted: int * seq of int -> seq of int
InsertSorted(i,l) ==
  cases true :
    (l = [])    -> [i],
    (i <= hd l) -> [i] ^ l,
    others      -> [hd l] ^ InsertSorted(i,tl l)
  end
```

```
end DoSort
```

DoSort クラスに対するテストカバレッジ情報の概要は下のテーブルにリストされている。テストカバレッジ情報は *sort.arg* ファイルを引数として生成される。

Test Suite : vdm.tc

Class : DoSort

Name	#Calls	Coverage
DoSort'DoSorting	6	✓
DoSort'InsertSorted	13	✓
DoSort'Sort	1	✓
Total Coverage		100%

6 ImplSort クラス

ImplSort クラスは、陰関数で定義されたソートアルゴリズムの一例である。

```
class ImplSort is subclass of Sorter

operations

  public Sort: seq of int ==> seq of int
  Sort(l) ==
    return ImplSorter(l);

functions

  public ImplSorter(l: seq of int) r: seq of int
  post IsPermutation(r,l) and IsOrdered(r);

  IsPermutation: seq of int * seq of int -> bool
  IsPermutation(l1,l2) ==
    forall e in set (elems l1 union elems l2) &
      card {i | i in set inds l1 & l1(i) = e} =
      card {i | i in set inds l2 & l2(i) = e};

  IsOrdered: seq of int -> bool
  IsOrdered(l) ==
    forall i,j in set inds l & i > j => l(i) >= l(j)

end ImplSort
```

7 ExplSort クラス

ExplSort クラスは、*ImplSort* で記述されたアルゴリズムを洗練したものである。

```
class ExplSort is subclass of Sorter
```

```
operations
```

```
public Sort: seq of int ==> seq of int
Sort(l) ==
  let r in set Permutations(l) be st IsOrdered(r) in
  return r
```

```
functions
```

```
Permutations: seq of int -> set of seq of int
Permutations(l) ==
  cases l:
    [], [-] -> {l},
    others -> dunion {{[l(i)]^j |
                      j in set Permutations(RestSeq(l,i))} |
                      i in set inds l}
  end;
```

```
RestSeq: seq of int * nat -> seq of int
RestSeq(l,i) ==
  [l(j) | j in set (inds l \ {i})]
pre i in set inds l
post elems RESULT subset elems l and
  len RESULT = len l - 1;
```

```
IsOrdered: seq of int -> bool
IsOrdered(l) ==
```

```
forall i,j in set inds l & i > j => l(i) >= l(j)  
end ExplSort
```
