

Train Fare System by VDM++

Shin Sahara

Hosei University

Abstract

This is an example of requirement specification for calculation of train fare. This example uses functions, and set of records. This includes invariants, post-conditions, pre-conditions, and a simple regression test too. There is another test mechanism called "combinatorial test".

Contents

1	Fare	2
1.1	Calculate_fare	2
2	Test	3
3	Reference, Index	4

1 Fare

I'm a train fare.

```

class
Fare
types
  public Station = seq of char
  inv s == s <> "";
public
  FareRecord::fDeparture : Station
                fArrival : Station
                fFare : nat
  inv fr == fr.fDeparture <> fr.fArrival

```

1.1 Calculate_fare

Calculate train fare between 2 stations.

```

functions
public static
  Calculate_fare : set of FareRecord * Station * Station -> nat
  Calculate_fare (aSetOfFare, aDeparture, anArrival) ==
    let wFareRecord in set aSetOfFare be st
      {aDeparture, anArrival} = {wFareRecord.fDeparture, wFareRecord.fArrival} in
      wFareRecord.fFare
pre exists! wFareRecord in set aSetOfFare &
  {aDeparture, anArrival} = {wFareRecord.fDeparture, wFareRecord.fArrival}

post exists! wFareRecord in set aSetOfFare &
  {aDeparture, anArrival} = {wFareRecord.fDeparture, wFareRecord.fArrival} and
  RESULT = wFareRecord.fFare
end
Fare

```

Test Suite : vdm.tc
Class : Fare

Name	#Calls	Coverage
Fare'Calculate_fare	3	✓
Total Coverage		100%

2 Test

I'm a simple regression test.

I don't take care of error test case.

```
.....
class
Test is subclass of Fare
values
```

```
    vFareSet = {
        mk.FareRecord ("Tokyo", "Shinagawa", 220),
        mk.FareRecord ("Tokyo", "Shinjuku", 180),
        mk.FareRecord ("Shinjuku", "Shinagawa", 190)}
```

```
functions
```

```
public static
```

```
    makeOrderMap : seq of bool +> map nat to bool
```

```
    makeOrderMap (s) ==
```

```
        {i |-> s (i) | i in set inds s};
```

```
public
```

```
    run : () -> seq of char * bool * map nat to bool
```

```
    run () ==
```

```
        let testcases = [t1 (), t2 (), t3 ()],
```

```
            testResults = makeOrderMap (testcases) in
```

```
        mk_ ("The result of regression test = ", forall i in set inds testcases & testcases (i), te
```

```
public
```

```
    t1 : () -> bool
```

```
    t1 () ==
```

```
        Calculate.fare (vFareSet, "Tokyo", "Shinagawa") = 220;
```

```
public
```

```
    t2 : () -> bool
```

```
    t2 () ==
```

```
        Calculate.fare (vFareSet, "Tokyo", "Shinjuku") = 180;
```

```
public
```

```
    t3 : () -> bool
```

```
    t3 () ==
```

```
        Calculate.fare (vFareSet, "Shinjuku", "Shinagawa") = 190
```

```
end
```

```
Test
```

```
.....
    Test Suite :      vdm.tc
    Class :          Test
```

Name	#Calls	Coverage
Test't1	1	✓
Test't2	1	✓
Test't3	1	✓
Test'run	1	✓
Test'makeOrderMap	1	✓
Total Coverage		100%

3 Reference, Index

The Vienna Development Method[1] is one of the longest-established Formal Methods for the development of computer-based systems.

References

- [1] John Fitzgerald; Peter Gorm Larsen; Paul Mukherjee; Nico Plat; Marcel Verhoef. *Validated Designs For Object-oriented Systems*. Springer Verlag, 02 2005.

Index

Calculate_fare, [2](#)

Fare, [2](#)

Test, [3](#)