Fashion Shows

December 30, 2017

Contents

1 App

```
class App
types
-- TODO Define types here
values
-- TODO Define values here
instance variables
-- TODO Define instance variables here
operations
-- TODO Define operations here
functions
-- TODO Define functiones here
traces
-- TODO Define Combinatorial Test Traces here
```

```
end App
```

Function or operation	Line	Coverage	Calls
App.vdmpp		0.0%	0

2 Designer_Test

```
class Designer_Test is subclass of MyTestCase
types
-- TODO Define types here
values
-- TODO Define values here
instance variables
-- TODO Define instance variables here
operations
-- TODO Define operations here


   public TestDesigner :() ==> ()
   TestDesigner() ==
   (
    -- constructor
    dcl designer : Fashion_Designer := new Fashion_Designer("Andre Correia",54);

    -- gets
    assertEqual(designer.getName(),"Andre Correia");
    assertEqual(designer.getAge(),54);


    return;
   );


   public static main_test: () ==> ()
   main_test() ==
   (
    IO`print("TestDesigner -> ");
    new Designer_Test().TestDesigner();
    IO`println("Passed");
   );

functions
-- TODO Define functiones here
traces
-- TODO Define Combinatorial Test Traces here
end Designer_Test
```

Function or operation	Line	Coverage	Calls
TestDesigner	11	100.0%	3
main_test	25	100.0%	3
Designer_Test.vdmpp		100.0%	6

3 Fashion_Designer

```
class Fashion_Designer
types
-- TODO Define types here
   public String = seq of char;
values
-- TODO Define values here

instance variables
-- TODO Define instance variables here
   private name : String;
   private age : nat1;
operations
-- TODO Define operations here

   --Construtor

   public Fashion_Designer: String * nat1 ==> Fashion_Designer
   Fashion_Designer(name1,age1) == (
    name := name1;
    age := age1;
    return self;
   );

   -- Retorna o nome

   public pure getName : () ==> String
   getName() ==
   (
    return name;
   );

   -- Retorna a idade

   public pure getAge : () ==> nat1
   getAge() ==
   (
    return age;
   );

functions
-- TODO Define functiones here

traces
-- TODO Define Combinatorial Test Traces here
end Fashion_Designer
```

Function or operation	Line	Coverage	Calls
Fashion_Designer	16	100.0%	9
getAge	31	100.0%	3
getName	24	100.0%	3
Fashion_Designer.vdmpp		100.0%	15

4 Fashion_Show

```
class Fashion_Show
types
-- TODO Define types here
   public String = seq of char;
   public Date :: year : nat month: nat1 day : nat1 hour : nat minute : nat
    inv mk_Date(y,m,d,h,min) == m <= 12 and d <= DaysOfMonth(m,y) and h < 24 and min < 60;
   public Models_to_Designers = map Fashion_Designer to set of Model;
   public listOfModels = set of Model;
   public listOfDesigners = set of Fashion_Designer;
   public listOfWorkshops = set of WorkShop;

values
-- TODO Define values here
instance variables
-- TODO Define instance variables here

   private location : String;
   private date : Date;
   private theme : String;
   private models : Models_to_Designers := {|->};
   private workshops : listOfWorkshops := {};


operations
-- TODO Define operations here

   --Construtor

   public Fashion_Show: String * String * nat * nat1 * nat1 * nat * nat ==> Fashion_Show
   Fashion_Show(location1,theme1,year, month, day, hour, minute) == (
    location := location1;
    theme := theme1;
    date := mk_Date(year, month, day, hour, minute);
    return self;
   );

   -- Retorna a localidade

   public pure getLocation : () ==> String
   getLocation() ==
   (
    return location;
   );

   -- Retorna o tema

   public pure getTheme : () ==> String
   getTheme() ==
   (
    return theme;
   );

   -- Retorna a data

   public pure getDate : () ==> Date
   getDate() ==
   (
    return date;
   );

   -- Retorna os designers

   public pure getDesigners : () ==> listOfDesigners
   getDesigners() ==
   (
    return dom models;
   );

   -- Retorna os modelos por designer

   public pure getModels : () ==> Models_to_Designers
   getModels() ==
   (
    return models;
   );

   -- Retorna os modelos de um dado designer

   public pure getModelsOfDesigner : (Fashion_Designer) ==> listOfModels
   getModelsOfDesigner(Fashion_Designer) ==
   (
    return models(Fashion_Designer);
   );

   -- Adiciona um designer ao desfile

   public addDesignerToShow : (Fashion_Designer) ==> ()
   addDesignerToShow(Fashion_Designer)==
   (
    models := models ++ {Fashion_Designer|->{}};
   )
   pre Fashion_Designer not in set dom models;

   -- Adiciona um modelo ao designer

   public addModelToShow : Fashion_Designer * Model ==> ()
   addModelToShow(Fashion_Designer, Model)==
   (
    models(Fashion_Designer) := models(Fashion_Designer) union {Model};
   )
   pre Model not in set models(Fashion_Designer);

   -- Adiciona um workshop ao show

   public addWorkShopToShow : WorkShop ==> ()
   addWorkShopToShow(WorkShop)==
   (
    workshops := workshops union {WorkShop};
   )
   pre WorkShop not in set workshops;

   -- Reservar um workshop

   public workShopBooking : WorkShop * User ==> ()
   workShopBooking(WorkShop, User) ==
   (

    WorkShop.addUserToWorkshop(User);
   )
   pre card WorkShop.getUsers() < WorkShop.getLotation();

functions
-- TODO Define functiones here

   -- Retorna o nmero de dias do m s num dado ano
   public static DaysOfMonth(month,year : nat1) r : nat1 == (
    if month = 1 or month = 3 or month = 5 or month = 7 or month = 8 or month = 10 or month = 12
       then
     31
    else if month = 2 and ((year mod 4 = 0 and year mod 100 <> 0) or year mod 400 = 0) then
     29
    else if month = 2 then
     28
    else
     30
   )

traces
-- TODO Define Combinatorial Test Traces here
end Fashion_Show
```

Function or operation	Line	Coverage	Calls
DaysOfMonth	106	100.0%	22
Fashion_Show	28	100.0%	3
addDesignerToShow	79	100.0%	6
addModelToShow	87	100.0%	12
addWorkShopToShow	95	0.0%	0
getDate	51	100.0%	3
getDesigners	58	100.0%	9
getLocation	37	100.0%	3
getModels	65	100.0%	6
getModelsOfDesigner	72	100.0%	12
getTheme	44	100.0%	3
workShopBooking	103	0.0%	0
Fashion_Show.vdmpp		88.1%	79

5 Main

```
class Main
types
-- TODO Define types here
values
-- TODO Define values here
instance variables
-- TODO Define instance variables here
   private static model_test: Model_Test := new Model_Test();
   private static designer_test : Designer_Test := new Designer_Test();
   private static show_test : Show_Test := new Show_Test();
   private static user_test : User_Test := new User_Test();
   private static workshop_test : WorkShop_Test := new WorkShop_Test();
operations

-- TODO Define operations here

   public static main: () ==> ()
   main() ==
   (
    model_test.main_test();
    designer_test.main_test();
    show_test.main_test();
    user_test.main_test();
    workshop_test.main_test();
   );

functions
-- TODO Define functiones here
traces
-- TODO Define Combinatorial Test Traces here
end Main
```

Function or operation	Line	Coverage	Calls
main	14	100.0%	2
Main.vdmpp		100.0%	2

6 Model

```
class Model
types
-- TODO Define types here
   public String = seq of char;

values
-- TODO Define values here
   public minAge = 18;

instance variables
-- TODO Define instance variables here
   private name : String;
   private age : nat1;

operations
-- TODO Define operations here

  --Construtor

   public Model: String * nat1 ==> Model
   Model(name1,age1) == (
    name := name1;
    age := age1;
    return self;
   )
   pre age1 >= minAge;

   -- Retorna o nome

   public pure getName : () ==> String
   getName() ==
   (
    return name;
   );

   -- Retorna a idade

   public pure getAge : () ==> nat1
   getAge() ==
   (
    return age;
   );
```

```
functions
-- TODO Define functiones here
traces
-- TODO Define Combinatorial Test Traces here
end Model
```

Function or operation	Line	Coverage	Calls
Model	19	100.0%	15
getAge	35	100.0%	3
getName	28	100.0%	3
Model.vdmpp		100.0%	21

7 Model_Test

```
class Model_Test is subclass of MyTestCase
types
-- TODO Define types here
values
-- TODO Define values here
instance variables
-- TODO Define instance variables here
operations
-- TODO Define operations here


   private TestModel :() ==> ()
   TestModel() ==
   (
    -- constructor
    dcl model : Model := new Model("Pedro Faria",67);

    -- bad constructor
    -- dcl model1 : Model := new Model("Filipe Cordeiro",15);

    -- gets
    assertEqual(model.getName(),"Pedro Faria");
    assertEqual(model.getAge(),67);


    return;
   );


   public static main_test: () ==> ()
   main_test() ==
   (
    IO`print("TestModel -> ");
    new Model_Test().TestModel();
    IO`println("Passed");
   );

functions
-- TODO Define functiones here
traces
-- TODO Define Combinatorial Test Traces here
```

```
end Model_Test
```

Function or operation	Line	Coverage	Calls
TestModel	11	100.0%	3
main_test	28	100.0%	3
Model_Test.vdmpp		100.0%	6

8 MyTestCase

```
class MyTestCase
/*
  Superclass for test classes, simpler but more practical than VDMUnit`TestCase.
  For proper use, you have to do: New -> Add VDM Library -> IO.
  JPF, FEUP, MFES, 2014/15.
*/

operations

 -- Simulates assertion checking by reducing it to pre-condition checking.
 -- If 'arg' does not hold, a pre-condition violation will be signaled.

 protected assertTrue: bool ==> ()
 assertTrue(arg) ==
  return
 pre arg;

 -- Simulates assertion checking by reducing it to post-condition checking.
 -- If values are not equal, prints a message in the console and generates
 -- a post-conditions violation.

 protected assertEqual: ? * ? ==> ()
 assertEqual(expected, actual) ==
  if expected <> actual then (
     IO`print("Actual value (");
     IO`print(actual);
     IO`print(") different from expected (");
     IO`print(expected);
     IO`println(")\n")
  )
 post expected = actual

end MyTestCase
```

Function or operation	Line	Coverage	Calls
assertEqual	20	38.8%	0
assertTrue	12	0.0%	0
MyTestCase.vdmpp		35.0%	0

9 Show_Test

```
class Show_Test is subclass of MyTestCase
types
-- TODO Define types here
values
-- TODO Define values here
instance variables
-- TODO Define instance variables here
operations
-- TODO Define operations here


   public TestShow :() ==> ()
   TestShow() ==
   (
    -- constructor

    dcl show : Fashion_Show := new Fashion_Show("Porto","Primavera",2017, 12, 31, 23, 59);
    dcl designer1 : Fashion_Designer := new Fashion_Designer("Andre Correia",54);
    dcl designer2 : Fashion_Designer := new Fashion_Designer("Francisco Lou a",64);
    dcl model1 : Model := new Model("Pedro Faria",67);
    dcl model2 : Model := new Model("Sara Sampaio",24);
    dcl model3 : Model := new Model("Daniela Hanganu",26);
    dcl model4 : Model := new Model("Dariia",23);

    -- gets
    assertEqual(show.getTheme(),"Primavera");
    assertEqual(show.getLocation(),"Porto");
    assertEqual(show.getDate(),mk_Fashion_Show`Date(2017, 12, 31, 23, 59));
    assertEqual(show.getModels(),{|->});

    -- get designers
    assertEqual(show.getDesigners(),{});
    show.addDesignerToShow(designer1);
    assertEqual(show.getDesigners(),{designer1});
    show.addDesignerToShow(designer2);
    assertEqual(show.getDesigners(),{designer1,designer2});

    --get models
    assertEqual(show.getModelsOfDesigner(designer1),{});
    assertEqual(show.getModelsOfDesigner(designer2),{});
    assertEqual(show.getModels(),{designer1|->{},designer2|->{}});
    show.addModelToShow(designer1,model1);
    show.addModelToShow(designer1,model2);
    show.addModelToShow(designer1,model3);
    show.addModelToShow(designer2,model4);
    assertEqual(show.getModelsOfDesigner(designer1),{model1,model2,model3});
    assertEqual(show.getModelsOfDesigner(designer2),{model4});

    --test functions
    assertEqual(show.DaysOfMonth(1,2000),31);
    assertEqual(show.DaysOfMonth(4,2000),30);
    assertEqual(show.DaysOfMonth(2,2000),29);
    assertEqual(show.DaysOfMonth(2,1900),28);

    return;
   );


   public static main_test: () ==> ()
   main_test() ==
   (
    IO`print("TestShow -> ");
    new Show_Test().TestShow();
    IO`println("Passed");
   );

functions
-- TODO Define functiones here
traces
-- TODO Define Combinatorial Test Traces here
end Show_Test
```

Function or operation	Line	Coverage	Calls
TestShow	11	100.0%	3
main_test	57	100.0%	3
Show_Test.vdmpp		100.0%	6

10 User

```
class User
types
-- TODO Define types here
   public String = seq of char;

values
-- TODO Define values here
instance variables
-- TODO Define instance variables here
   private name : String;

operations
-- TODO Define operations here

   --Construtor

   public User: String ==> User
   User(name1) == (
    name := name1;
    return self;
   );

   --gets

   public pure getName : () ==> String
   getName() ==
   (
    return name;
   );

functions
-- TODO Define functiones here
traces
-- TODO Define Combinatorial Test Traces here
end User
```

Function or operation	Line	Coverage	Calls

User	16	100.0%	5
getName	23	100.0%	3
User.vdmpp		100.0%	8

11 User_Test

```
class User_Test is subclass of MyTestCase
types
-- TODO Define types here
values
-- TODO Define values here
instance variables
-- TODO Define instance variables here
operations
-- TODO Define operations here



   private TestUser :() ==> ()

   TestUser() ==
   (
    -- constructor
    dcl user : User := new User("Diolinda");

    -- gets
    assertEqual(user.getName(),"Diolinda");


    return;
   );

   public static main_test: () ==> ()
   main_test() ==
   (
    IO`print("TestUser -> ");
    new User_Test().TestUser();
    IO`println("Passed");
   );

functions
-- TODO Define functiones here
traces
-- TODO Define Combinatorial Test Traces here
end User_Test
```

Function or operation	Line	Coverage	Calls
TestModel	11	100.0%	3
TestUser	11	100.0%	3
main_test	12	100.0%	3
User_Test.vdmpp		100.0%	9

12 WorkShop

```
class WorkShop
types
-- TODO Define types here
   public String = seq of char;
   public Users = set of User;


values
-- TODO Define values here
instance variables
-- TODO Define instance variables here

   private theme : String;
   private begin_date : Fashion_Show`Date;
   private end_date : Fashion_Show`Date;
   private lotation : nat1;
   private orator : String;
   private registered_users : Users := {};
   private room : String;

operations
-- TODO Define operations here


   public WorkShop: String * Fashion_Show`Date * Fashion_Show`Date * nat1 * String * String ==>
       WorkShop
   WorkShop(theme1, begin_date1, end_date1, lotation1, orator1, room1) == (

    theme := theme1;
    begin_date := begin_date1;
    end_date := end_date1;
    lotation := lotation1;
    orator := orator1;
    room := room1;

    return self;
   );

   --gets

   public pure getTheme : () ==> String
   getTheme() ==
   (
    return theme;
   );


   public pure getBeginDate : () ==> Fashion_Show`Date
   getBeginDate() ==
   (
    return begin_date;
   );


   public pure getEndDate : () ==> Fashion_Show`Date
   getEndDate() ==
   (
    return end_date;
   );
   public pure getLotation : () ==> nat1
   getLotation() ==
   (
    return lotation;
   );


   public pure getOrator : () ==> String
   getOrator() ==
   (
    return orator;
   );


   public pure getRoom : () ==> String
   getRoom() ==
   (
    return room;
   );


   public pure getUsers : () ==> Users
   getUsers() ==
   (
    return registered_users;
   );

   -- add User to workshop

   public addUserToWorkshop : (User) ==> ()
   addUserToWorkshop(User) ==
   (
    registered_users := registered_users union {User};
    return;
   )
   pre User not in set registered_users;

functions
-- TODO Define functiones here
traces
-- TODO Define Combinatorial Test Traces here
end WorkShop
```

Function or operation	Line	Coverage	Calls
WorkShop	24	100.0%	1
addUserToWorkshop	81	100.0%	2
getBeginDate	44	100.0%	1
getEndDate	50	100.0%	1
getLotation	56	100.0%	1
getOrator	62	100.0%	1
getRoom	68	100.0%	1
getTheme	38	100.0%	1
getUsers	74	100.0%	3
WorkShop.vdmpp		100.0%	12

13 WorkShop_Test

```
class WorkShop_Test is subclass of MyTestCase
types
-- TODO Define types here
values
-- TODO Define values here
instance variables
-- TODO Define instance variables here
operations
-- TODO Define operations here



   private TestWorkShop :() ==> ()
   TestWorkShop() ==
   (
    -- constructor
    dcl workshop : WorkShop := new WorkShop("Como costurar um bot o?", mk_Fashion_Show`Date(2017,
        12, 31, 20, 00), mk_Fashion_Show`Date(2017, 12, 31, 21, 00), 20, "Jo o Bot es Correia",
        "A7");
    dcl user1 : User := new User("Diolinda");
    dcl user2: User := new User("Diofeia");

    -- gets
    assertEqual(workshop.getTheme(),"Como costurar um bot o?");
    assertEqual(workshop.getBeginDate(),mk_Fashion_Show`Date(2017, 12, 31, 20, 00));
    assertEqual(workshop.getEndDate(),mk_Fashion_Show`Date(2017, 12, 31, 21, 00));
    assertEqual(workshop.getLotation(),20);

    assertEqual(workshop.getOrator(),"Jo o Bot es Correia");
    assertEqual(workshop.getRoom(),"A7");
    assertEqual(workshop.getUsers(),{});

    -- Adicionar utilizadores ao workshop
    workshop.addUserToWorkshop(user1);
    assertEqual(workshop.getUsers(),{user1});
    workshop.addUserToWorkshop(user2);
    assertEqual(workshop.getUsers(),{user1,user2});


    return;
   );

   public static main_test: () ==> ()
   main_test() ==
   (
    IO`print("TestWorkShop -> ");
    new WorkShop_Test().TestWorkShop();
    IO`println("Passed");
   );

functions
-- TODO Define functiones here
traces
-- TODO Define Combinatorial Test Traces here
end WorkShop_Test
```

Function or operation	Line	Coverage	Calls

TestUser	11	100.0%	3
TestWorkShop	11	100.0%	3
main_test	24	100.0%	1
WorkShop_Test.vdmpp		100.0%	7