Fashion Shows

January 1, 2018

Contents

1 App

```
class App
types
-- TODO Define types here

   public Users = set of User;
   public Shows = set of Fashion_Show;

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

   public users : Users := {};
   public shows : Shows := {};

operations
-- TODO Define operations here

   --Construtor

   public App: () ==> App
   App() == (
    return self;
   );

   --Retorna os utilizadores da aplica o

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

   --Retorna os shows da aplica o

   public pure getShows : () ==> Shows
   getShows() ==
   (
    return shows;
   );

   --Adiciona um utilizador   aplica o

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

   --Adiciona um show   aplica o

   public addShowToApp : (Fashion_Show) ==> ()
   addShowToApp(Fashion_Show) ==
   (
    shows := shows union {Fashion_Show};
    return;
   )
   pre Fashion_Show not in set shows;

functions
-- TODO Define functiones here
traces
```

```
-- TODO Define Combinatorial Test Traces here
end App
```

Function or operation	Line	Coverage	Calls
App	20	100.0%	3
addShowToApp	49	100.0%	3
addUserToApp	40	100.0%	3
getShows	33	100.0%	6
getUsers	26	100.0%	6
App.vdmpp		100.0%	21

2 App_Test

```
class App_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 TestApp :() ==> ()
   TestApp() ==
   (
    -- constructor
    dcl app : App := new App();
    dcl user : User := new User("Diolinda");
    dcl show : Fashion_Show := new Fashion_Show("Porto","Primavera",2017, 12, 31, 23, 59);

    -- users
    assertEqual(app.getUsers(),{});
    app.addUserToApp(user);
    assertEqual(app.getUsers(),{user});

    -- shows
    assertEqual(app.getShows(),{});
    app.addShowToApp(show);
    assertEqual(app.getShows(),{show});


    return;
   );


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

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

Function or operation	Line	Coverage	Calls
TestApp	11	100.0%	3
main_test	33	100.0%	3
App_Test.vdmpp		100.0%	6

3 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

4 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

5 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;
   public listOfCritics = map Reviewer to String;

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 := {};
   private critics : listOfCritics := {|->};

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);
);

-- Retorna os workshops do show

public pure getWorkShops : () ==> listOfWorkshops
getWorkShops() ==
(
 return workshops;
);

-- Retorna os workshops do show

public pure getCritics : () ==> listOfCritics
getCritics() ==
(
 return critics;
);

-- Adiciona um designer ao desfile

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

    -- Reservar um workshop

    public workShopBooking : WorkShop * User ==> ()
    workShopBooking(WorkShop, User) ==
    (
     WorkShop.addUserToWorkshop(User);
    )
    pre card WorkShop.getUsers() < WorkShop.getLotation();

    -- Editor adiciona a sua crtica ao show

    public addCritic : Reviewer * String ==> ()
    addCritic(Reviewer, String) ==
    (
     critics := critics ++ {Reviewer|->String};
    )
    pre Reviewer not in set dom critics;


functions
-- TODO Define functiones here

    -- Retorna o nmero de dias do ms 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	141	100.0%	49
Fashion_Show	29	100.0%	9
addCritic	129	100.0%	3
addDesignerToShow	94	100.0%	6
addModelToShow	103	100.0%	24

addWorkShopToShow	112	100.0%	6
getCritics	87	100.0%	6
getDate	52	100.0%	3
getDesigners	59	100.0%	9
getLocation	38	100.0%	3
getModels	66	100.0%	6
getModelsOfDesigner	73	100.0%	12
getTheme	45	100.0%	3
getWorkShops	80	100.0%	9
workShopBooking	121	100.0%	6
Fashion_Show.vdmpp		100.0%	154

6 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();
   private static app_test : App_Test := new App_Test();
   private static reviewer_test : Reviewer_Test := new Reviewer_Test();
   private static model_look_test : Model_Look_Test := new Model_Look_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();
    app_test.main_test();
    reviewer_test.main_test();
    model_look_test.main_test();
   );

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

Function or operation	Line	Coverage	Calls

main	18	100.0%	1
Main.vdmpp		100.0%	1

7 Model

```
class Model
types
-- TODO Define types here
   public String = seq of char;
   public Gender = <Masculino> | <Feminino>;

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

instance variables
-- TODO Define instance variables here
   private name : String;
   private age : nat1;
   private gender : Gender;
   private height : real;
   private weight : real;
   inv age >= minAge;

operations
-- TODO Define operations here

  --Construtor

   public Model: String * nat1 * Gender * real * real ==> Model
   Model(name1,age1,gender1,height1,weight1) == (
    name := name1;
    age := age1;
    gender := gender1;
    height := height1;
    weight := weight1;
    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;
   );

   -- Retorna o genero

   public pure getGender : () ==> Gender
   getGender() ==
```

```
    (
     return gender;
    );

    -- Retorna a altura

    public pure getHeight : () ==> real
    getHeight() ==
    (
     return height;
    );

    -- Retorna o peso

    public pure getWeight : () ==> real
    getWeight() ==
    (
     return weight;
    );

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

Function or operation	Line	Coverage	Calls
Model	24	100.0%	18
getAge	43	100.0%	3
getGender	50	100.0%	3
getHeight	57	100.0%	3
getName	36	100.0%	3
getWeight	64	100.0%	3
Model.vdmpp		100.0%	33

8 Model_Look

```
class Model_Look
types
 public String = seq of char;

values
-- TODO Define values here

instance variables

  private model : Model;
  private fashion_show : Fashion_Show;
  private date : Fashion_Show`Date;
  private description : String;

operations

  --Construtor
```

```
  public Model_Look: Model * Fashion_Show * Fashion_Show`Date * String ==> Model_Look
  Model_Look(model1,fashion_show1,date1,description1) == (
   model := model1;
   fashion_show := fashion_show1;
   date := date1;
   description := description1;
   return self;
  );

  --Retorna o modelo do look

  public pure getModel : () ==> Model
  getModel() ==
  (
   return model;
  );

  --Retorna o Fashion Show

  public pure getFashionShow : () ==> Fashion_Show
  getFashionShow() ==
  (
   return fashion_show;
  );

  --Retorna o momento em que o modelo passou na passerela com este look

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

  --Retorna a descricao do look

  public pure getDescription : () ==> String
  getDescription() ==
  (
   return description;
  );

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

Function or operation	Line	Coverage	Calls
Model_Look	18	100.0%	3
getDate	42	100.0%	1
getDescription	49	100.0%	1
getFashionShow	35	100.0%	1
getModel	28	100.0%	1
Model_Look.vdmpp		100.0%	7

9 Model_Look_Test

```
class Model_Look_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 TestModelLook :() ==> ()
   TestModelLook() ==
   (
    -- constructor
    dcl model : Model := new Model("Pedro Faria",67,<Masculino>,1.78,74.32);
    dcl show : Fashion_Show := new Fashion_Show("Porto","Primavera",2017, 12, 31, 23, 59);
    dcl model_look : Model_Look := new Model_Look(model,show, mk_Fashion_Show`Date(2017, 12, 31,
        23, 00),"vestido azul e cor de rosa");

    -- gets
    assertEqual(model_look.getModel(),model);
    assertEqual(model_look.getFashionShow(),show);
    assertEqual(model_look.getDate(),mk_Fashion_Show`Date(2017, 12, 31, 23, 00));
    assertEqual(model_look.getDescription(),"vestido azul e cor de rosa");


    return;
   );


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

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

Function or operation	Line	Coverage	Calls
TestModel	11	100.0%	1
TestModelLook	11	100.0%	1
main_test	28	100.0%	1
Model_Look_Test.vdmpp		100.0%	3

10 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,<Masculino>,1.78,74.32);

    -- gets
    assertEqual(model.getName(),"Pedro Faria");
    assertEqual(model.getAge(),67);
    assertEqual(model.getGender(),<Masculino>);
    assertEqual(model.getHeight(),1.78);
    assertEqual(model.getWeight(),74.32);


    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

11 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

12 Reviewer

```
class Reviewer
types
-- TODO Define types here
   public String = seq of char;
   public Gender = <Masculino> | <Feminino>;
values
-- TODO Define values here
instance variables
-- TODO Define instance variables here
   private name : String;
   private age : nat1;
   private gender : Gender;

operations
-- TODO Define operations here

   --Construtor

   public Reviewer: String * nat1 * Gender ==> Reviewer
   Reviewer(name1,age1,gender1) == (
    name := name1;
    age := age1;
```

```
    gender := gender1;
    return self;
   );

   -- Retorna o nome

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

   -- Retorna a idade

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

   -- Retorna o genero

   public pure getGender : () ==> Gender
   getGender() ==
   (
    return gender;
   );

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

Function or operation	Line	Coverage	Calls
Reviewer	18	100.0%	6
getAge	34	100.0%	3
getGender	41	100.0%	3
getName	27	100.0%	3
Reviewer.vdmpp		100.0%	15

13 Reviewer_Test

```
class Reviewer_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 TestReviewer :() ==> ()
   TestReviewer() ==
```

```
    (
     -- constructor
     dcl reviewer : Reviewer := new Reviewer("Ana Bacalhau",39,<Feminino>);

     -- gets
     assertEqual(reviewer.getName(),"Ana Bacalhau");
     assertEqual(reviewer.getAge(),39);
     assertEqual(reviewer.getGender(),<Feminino>);

     return;
    );


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

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

Function or operation	Line	Coverage	Calls
TestReviewer	11	100.0%	3
main_test	25	100.0%	3
Reviewer_Test.vdmpp		100.0%	6

14 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 Loua",64);
    dcl model1 : Model := new Model("Pedro Faria",67,<Masculino>,1.78,74.32);
    dcl model2 : Model := new Model("Sara Sampaio",24,<Feminino>,1.82,53.24);
    dcl model3 : Model := new Model("Daniela Hanganu",26,<Feminino>,1.79,52.78);
```

```
 dcl model4 : Model := new Model("Dariia",23,<Feminino>,1.85,56.91);
 dcl workshop : WorkShop := new WorkShop("Como costurar um boto?", mk_Fashion_Show`Date(2017,
      12, 31, 20, 00), mk_Fashion_Show`Date(2017, 12, 31, 21, 00), 20, "Joo Botes Correia",
     "A7");
 dcl workshop2 : WorkShop := new WorkShop("Como se maquilhar?", mk_Fashion_Show`Date(2017, 12,
      31, 19, 00), mk_Fashion_Show`Date(2017, 12, 31, 20, 00), 20, "Joo Botes Correia", "A9"
     );
 dcl user1 : User := new User("Diolinda");
 dcl user2: User := new User("Diofeia");
 dcl reviewer: Reviewer := new Reviewer("Ana Bacalhau",39,<Feminino>);

 -- 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});

 -- workshops
 assertEqual(show.getWorkShops(),{});
 show.addWorkShopToShow(workshop);
 assertEqual(show.getWorkShops(),{workshop});
 show.addWorkShopToShow(workshop2);
 assertEqual(show.getWorkShops(),{workshop,workshop2});
 assertEqual(workshop.getUsers(),{});
 show.workShopBooking(workshop, user1);
 assertEqual(workshop.getUsers(),{user1});
 show.workShopBooking(workshop, user2);
 assertEqual(workshop.getUsers(),{user1,user2});

 -- critics
 assertEqual(show.getCritics(),{|->});
 show.addCritic(reviewer, "Melhor festival de moda que participei!");
 assertEqual(show.getCritics(),{reviewer|->"Melhor festival de moda que participei!"});

 --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	79	100.0%	3
Show_Test.vdmpp		100.0%	6

15 User

```
class User
types
-- TODO Define types here
  public String = seq of char;
  public Looks = set of Model_Look;
values
-- TODO Define values here
instance variables
-- TODO Define instance variables here
  private name : String;
  private favorite_looks : Looks := {};

operations
-- TODO Define operations here

  --Construtor

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

  --gets

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


  public pure getFavoriteLooks : () ==> Looks
  getFavoriteLooks() ==
  (
   return favorite_looks;
  );
```

```
  -- Adiciona um look aos looks favoritos

  public addLookToFavoriteLooks : Model_Look ==> ()
  addLookToFavoriteLooks(look)==
  (
   favorite_looks := favorite_looks union {look};
  )
  pre look not in set favorite_looks
  post look in set favorite_looks;

  -- Remove um look dos looks favoritos

  public removeLookFromFavoriteLooks : Model_Look ==> ()
  removeLookFromFavoriteLooks(look)==
  (
   favorite_looks := favorite_looks \ {look};
  )
  pre look in set favorite_looks
  post look not in set favorite_looks;

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

Function or operation	Line	Coverage	Calls
User	17	100.0%	18
addLookToFavoriteLooks	37	100.0%	2
getFavoriteLooks	30	100.0%	6
getName	24	100.0%	3
removeLookFromFavoriteLooks	46	100.0%	2
User.vdmpp		100.0%	31

16 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");
    dcl model : Model := new Model("Pedro Faria",67,<Masculino>,1.78,74.32);
    dcl show : Fashion_Show := new Fashion_Show("Porto","Primavera",2017, 12, 31, 23, 59);
```

```
    dcl model_look : Model_Look := new Model_Look(model,show, mk_Fashion_Show`Date(2017, 12, 31,
        23, 00),"vestido azul e cor de rosa");

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

    -- looks

    assertEqual(user.getFavoriteLooks(),{});
    user.addLookToFavoriteLooks(model_look);
    assertEqual(user.getFavoriteLooks(),{model_look});
    user.removeLookFromFavoriteLooks(model_look);
    assertEqual(user.getFavoriteLooks(),{});



    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
TestUser	11	100.0%	2
main_test	24	100.0%	2
User_Test.vdmpp		100.0%	4

17 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;
   inv card registered_users <= lotation;

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;
   );

   --Retorna o tema do workshop

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

   --Retorna a data de incio

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

   --Retorna a data de fim

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

   --Retorna a lota o

   public pure getLotation : () ==> nat1
   getLotation() ==
   (
    return lotation;
   );

   --Retorna o orador

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

   --Retorna a sala do workshop
```

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

   --Retorna os utilizadores que participam

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

   --Adiciona um utilizador   workshop

   public addUserToWorkshop : (User) ==> ()
   addUserToWorkshop(User) ==
   (
    registered_users := registered_users union {User};
    return;
   )
   pre User not in set registered_users and card registered_users < lotation
   post User 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	25	100.0%	9
addUserToWorkshop	88	100.0%	12
getBeginDate	46	100.0%	3
getEndDate	53	100.0%	3
getLotation	60	100.0%	9
getOrator	67	100.0%	3
getRoom	74	100.0%	3
getTheme	39	100.0%	3
getUsers	81	100.0%	24
WorkShop.vdmpp		100.0%	69

18 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 botо?", mk_Fashion_Show`Date(2017,
        12, 31, 20, 00), mk_Fashion_Show`Date(2017, 12, 31, 21, 00), 20, "Joo Botes Correia",
       "A7");
   dcl user1 : User := new User("Diolinda");
   dcl user2: User := new User("Diofeia");

   -- gets
   assertEqual(workshop.getTheme(),"Como costurar um botо?");
   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(),"Joo Botes 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
TestWorkShop	11	100.0%	9
main_test	38	100.0%	6
WorkShop_Test.vdmpp		100.0%	15