Fashion Shows

January 2, 2018

Contents

1 App

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

   public Users = set of Regular_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 : (Regular_User) ==> ()
   addUserToApp(User) ==
   (
    users := users union {User};
    return;
   )
   pre User not in set users
   post User in set users;

   --Remove um utilizador   aplica o

   public rmvUserToApp : (Regular_User) ==> ()
   rmvUserToApp(User) ==
   (
    users := users \ {User};
    return;
   )
   pre User in set users
   post 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
   post Fashion_Show in set shows;

   --Remove um show   aplica o

   public rmvShowToApp : (Fashion_Show) ==> ()
   rmvShowToApp(Fashion_Show) ==
   (
    shows := shows \ {Fashion_Show};
    return;
   )
   pre Fashion_Show in set shows
   post 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%	7
addShowToApp	60	100.0%	7
addUserToApp	40	100.0%	7
getShows	33	100.0%	21
getUsers	26	100.0%	21
rmvShowToApp	70	100.0%	7
rmvUserToApp	50	100.0%	7
App.vdmpp		100.0%	77

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 : Regular_User := new Regular_User("Diolinda",<Feminino>,"diolinda@gmail.com","
        password_diolinda",false);
    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});
    app.rmvUserToApp(user);
    assertEqual(app.getUsers(),{});

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


    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%	7
main_test	37	100.0%	7
App_Test.vdmpp		100.0%	14

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%	7
main_test	25	100.0%	7
Designer_Test.vdmpp		100.0%	14

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%	35
getAge	31	100.0%	7
getName	24	100.0%	7
Fashion_Designer.vdmpp		100.0%	49

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 seq of Model_In_Runway;
   public listOfModels = seq of Model_In_Runway;
   public listOfDesigners = set of Fashion_Designer;
   public listOfWorkshops = set of WorkShop;
   public listOfCritics = map Reviewer to Critic;
   public programShow = map Date to Fashion_Designer;
   public listOfDates = set of Date;
   public Critic :: description : String rate: nat
    inv v == v.rate <= 5;

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 := {|->};
   private program : programShow := {|->};

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 o programa do evento

   public pure getProgramShow : () ==> programShow
```

```
getProgramShow() ==
(
 return program;
);

-- Retorna os modelos de um dado designer

public pure getModelsOfDesigner : (Fashion_Designer) ==> listOfModels
getModelsOfDesigner(Fashion_Designer) ==
(
 return models(Fashion_Designer);
)
pre Fashion_Designer in set dom models;

-- 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;

-- Remove um designer ao desfile

public rmvDesignerToShow : (Fashion_Designer) ==> ()
rmvDesignerToShow(Fashion_Designer)==
(
 models := {Fashion_Designer} <-: models ;
)
pre Fashion_Designer in set dom models
post Fashion_Designer not in set dom models;

-- Adiciona um modelo ao designer

public addModelToShow : Fashion_Designer * Model_In_Runway ==> ()
addModelToShow(Fashion_Designer, Model)==
(
 models(Fashion_Designer) := models(Fashion_Designer)^[Model];
)
pre Model not in set elems models(Fashion_Designer) and Fashion_Designer in set dom models
post Model in set elems models(Fashion_Designer);

-- Remove um modelo do designer

public removeModelToShow : Fashion_Designer * Model_In_Runway ==> ()
removeModelToShow(Fashion_Designer, Model)==
(
 dcl seqAux : listOfModels := [];

 for all i in set inds models(Fashion_Designer) do
 if models(Fashion_Designer)(i) <> Model
 then seqAux := seqAux^[models(Fashion_Designer)(i)];

 models(Fashion_Designer) := seqAux;
)
pre Model in set elems models(Fashion_Designer) and Fashion_Designer in set dom models
post Model not in set elems models(Fashion_Designer);

-- Adiciona um designer ao programa

public addDesignerToProgramShow : Date * Fashion_Designer ==> ()
addDesignerToProgramShow(dateShow,designer)==
(
 program := program ++ {dateShow |-> designer};
)
pre dateShow.year = date.year and dateShow.month = date.month and dateShow.day = date.day and
    dateShow.hour >= date.hour and dateShow.minute >= date.minute and dateShow not in set dom
    program
post dateShow in set dom program;

-- Remove um designer do programa

public removeDesignerFromProgramShow : Date ==> ()
removeDesignerFromProgramShow(dateShow)==
(
 program := {dateShow} <-: program;
)
pre dateShow in set dom program
post dateShow not in set dom program;

-- Retorna o designer para uma determinada data do programa

public pure getDesignerByDate : (Date) ==> Fashion_Designer
getDesignerByDate(dateShow) ==
(
 return program(dateShow);
)
pre dateShow in set dom program;

-- Retorna as datas do programa para um determinado designer

public pure getListOfDatesByDesigner : (Fashion_Designer) ==> listOfDates
getListOfDatesByDesigner(designer) ==
(
 dcl m : map Date to Fashion_Designer := program :> {designer};
 dcl l : listOfDates := {};

 for all d in set dom m do l := l union {d};

 return l;
)
pre designer in set rng program;

-- 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;
```

```
   -- Remove um workshop ao show

   public rmvWorkShopToShow : WorkShop ==> ()
   rmvWorkShopToShow(WorkShop)==
   (
    workshops := workshops \ {WorkShop};
   )
   pre WorkShop in set workshops
   post WorkShop not in set workshops;

   -- Reservar um workshop

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

   -- Editor adiciona a sua crtica ao show

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

   -- Editor remove a sua crtica ao show

   public rmvCritic : Reviewer ==> ()
   rmvCritic(Reviewer) ==
   (
    critics := {Reviewer} <-: critics;
   )
   pre Reviewer in set dom critics
   post Reviewer not in set dom critics;

   -- Mdia crtica ao show

   public getAvgReview : () ==> (real)
   getAvgReview() ==
   (
    dcl sum : real :=0;

    for all reviewer in set dom critics do
     sum := sum + critics(reviewer).rate;
    if (card dom critics > 0) then
     return sum / card dom critics
    else
     return sum;
   );

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	249	100.0%	26
Fashion_Show	34	100.0%	28
addCritic	214	100.0%	12
addDesignerToProgramShow	149	100.0%	42
addDesignerToShow	107	100.0%	28
addModelToShow	125	100.0%	28
addWorkShopToShow	188	100.0%	12
getAvgReview	232	100.0%	30
getCritics	100	100.0%	35
getDate	57	100.0%	7
getDesignerByDate	167	100.0%	24
getDesigners	64	100.0%	35
getListOfDatesByDesigner	175	100.0%	12
getLocation	43	100.0%	7
getModels	71	100.0%	14
getModelsOfDesigner	85	100.0%	30
getProgramShow	78	100.0%	196
getTheme	50	100.0%	7
getWorkShops	93	100.0%	35
removeDesignerFromProgramShow	158	100.0%	42
removeModelToShow	134	100.0%	2
rmvCritic	223	100.0%	12
rmvDesignerToShow	116	100.0%	14
rmvWorkShopToShow	197	100.0%	12
workShopBooking	206	100.0%	36
Fashion_Show.vdmpp		100.0%	726

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 : Regular_User_Test := new Regular_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();
   private static model_in_runway_test : Model_In_Runway_Test := new Model_In_Runway_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();
    model_in_runway_test.main_test();
   );

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

Function or operation	Line	Coverage	Calls
main	20	100.0%	5
Main.vdmpp		100.0%	5

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%	54
getAge	43	100.0%	7
getGender	50	100.0%	7
getHeight	57	100.0%	7
getName	36	100.0%	7
getWeight	64	100.0%	7
Model.vdmpp		100.0%	89

8 Model_In_Runway

```
class Model_In_Runway
types
  public Image = seq of char;

values
-- TODO Define values here

instance variables
  private model : Model;
  private photo : [Image] := nil;
  private date : Fashion_Show`Date;

operations

  --Construtor

  public Model_In_Runway: Model * Fashion_Show`Date ==> Model_In_Runway
  Model_In_Runway(model1,date1) == (
   model := model1;
   date := date1;
   return self;
  );

  -- Retorna o modelo

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

  -- Retorna a fotografia

  public pure getPhoto : () ==> Image
  getPhoto() ==
  (
   return photo;
  )
  pre photo <> nil;

  -- Retorna a data

  public pure getDate : () ==> Fashion_Show`Date
  getDate() ==
  (
   return date;
  );
  -- Define a data

  public setDate : (Fashion_Show`Date) ==> ()
  setDate(date1) ==
  (
   date := date1;
  )
  post date = date1;

  -- Guarda fotografia

  public setPhoto : (Image) ==> ()
  setPhoto(photo1) ==
  (
   photo := photo1;
  )
  post photo = photo1;

  -- Remove fotografia

  public removePhoto : () ==> ()
  removePhoto() ==
  (
   photo := nil;
  )
  pre photo <> nil
  post photo = nil;

functions
-- TODO Define functiones here

traces
-- TODO Define Combinatorial Test Traces here

end Model_In_Runway
```

Function or operation	Line	Coverage	Calls
Model_In_Runway	16	100.0%	29
getDate	39	100.0%	9
getModel	24	100.0%	5
getPhoto	31	100.0%	5
removePhoto	62	100.0%	5
setDate	46	100.0%	4
setPhoto	54	100.0%	5
Model_In_Runway.vdmpp		100.0%	62

9 Model_In_Runway_Test

```
class Model_In_Runway_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 TestModelInRunaway :() ==> ()
   TestModelInRunaway() ==
   (
    -- constructor
    dcl model1 : Model := new Model("Pedro Faria",67,<Masculino>,1.78,74.32);
    dcl model1_rw : Model_In_Runway := new Model_In_Runway(model1, mk_Fashion_Show`Date(2017, 12,
        31, 20, 00));


    -- gets
    assertEqual(model1_rw.getModel(),model1);
    assertEqual(model1_rw.getDate(),mk_Fashion_Show`Date(2017, 12, 31, 20, 00));
    model1_rw.setPhoto("picture.png");
    assertEqual(model1_rw.getPhoto(),"picture.png");
    model1_rw.removePhoto();


    -- n o pode devolver foto se n o existir - pre-condi o
    -- assertEqual(model1_rw.getPhoto(),nil);

    model1_rw.setDate(mk_Fashion_Show`Date(2017, 11,10, 15, 00));
    assertEqual(model1_rw.getDate(),mk_Fashion_Show`Date(2017, 11,10, 15, 00));

    return;
   );

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

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

Function or operation	Line	Coverage	Calls
TestDesigner	11	100.0%	5
TestModelInRunaway	11	100.0%	5
main_test	25	100.0%	2
Model_In_Runway_Test.vdmpp		100.0%	12

10 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%	14
getDate	42	100.0%	7
getDescription	49	100.0%	7
getFashionShow	35	100.0%	7
getModel	28	100.0%	7
Model_Look.vdmpp		100.0%	42

11 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
TestModelLook	11	100.0%	7
main_test	29	100.0%	7
Model_Look_Test.vdmpp		100.0%	14

12 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%	7
main_test	28	100.0%	7
Model_Test.vdmpp		100.0%	14

13 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

14 Regular_User

```
class Regular_User is subclass of 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 favorite_looks : Looks := {};
  private premium_user : bool := false;
```

```
operations
-- TODO Define operations here

  --Construtor

  public Regular_User: String * Gender * String * String * bool ==> Regular_User
  Regular_User(name1, gender1, email1, password1, premium_user1) == (
   name := name1;
   gender := gender1;
   email := email1;
   password := password1;
   premium_user := premium_user1;
   return self;
  );

  --Retorna se   um user premium ou nao

  public pure getPremium : () ==> bool
  getPremium() ==
  (
   return premium_user;
  );

  --Retorna os looks favoritos

  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 Regular_User
```

Function or operation	Line	Coverage	Calls
Regular_User	17	100.0%	42
addLookToFavoriteLooks	42	100.0%	6

getFavoriteLooks	35	100.0%	18
getPremium	28	100.0%	12
removeLookFromFavoriteLooks	51	100.0%	12
Regular_User.vdmpp		100.0%	90

15 Regular_User_Test

```
class Regular_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 : Regular_User := new Regular_User("Diolinda", <Feminino>, "diolinda@gmail.com", "
        password_diolinda", true);
    dcl user2 : Regular_User := new Regular_User("Diofeia", <Feminino>, "diofeia@gmail.com", "
        password_diofeia", false);
    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");
    assertEqual(user2.getName(),"Diofeia");
    assertEqual(user.getPremium(),true);
    assertEqual(user2.getPremium(),false);
    assertEqual(user.getGender(),<Feminino>);
    assertEqual(user2.getGender(),<Feminino>);
    assertEqual(user.getEmail(),"diolinda@gmail.com");
    assertEqual(user2.getEmail(),"diofeia@gmail.com");
    assertEqual(user.getPassword(),"password_diolinda");
    assertEqual(user2.getPassword(),"password_diofeia");

    -- 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 Regular_User_Test().TestUser();
    IO`println("Passed");
   );

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

Function or operation	Line	Coverage	Calls
TestUser	11	100.0%	6
main_test	43	100.0%	12
Regular_User_Test.vdmpp		100.0%	18

16 Reviewer

```
class Reviewer is subclass of User
types
-- TODO Define types here
   public String = seq of char;
values
-- TODO Define values here
instance variables
-- TODO Define instance variables here
   private age : nat1;

operations
-- TODO Define operations here

   --Construtor

   public Reviewer: String * nat1 * Gender * String * String ==> Reviewer
   Reviewer(name1,age1,gender1,email1,password1) == (
    name := name1;
    age := age1;
    gender := gender1;
    email := email1;
    password := password1;
    return self;
   );

   -- Retorna a idade

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

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

Function or operation	Line	Coverage	Calls
Reviewer	15	100.0%	18
getAge	26	100.0%	7
Reviewer.vdmpp		100.0%	25

17 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>,"anabacalhau@gmail.com",
        "password_anabacalhau");

    -- gets
    assertEqual(reviewer.getName(),"Ana Bacalhau");
    assertEqual(reviewer.getAge(),39);
    assertEqual(reviewer.getGender(),<Feminino>);
    assertEqual(reviewer.getEmail(),"anabacalhau@gmail.com");
    assertEqual(reviewer.getPassword(),"password_anabacalhau");

    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%	7
main_test	27	100.0%	6
Reviewer_Test.vdmpp		100.0%	13

18 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, 9, 00);
    dcl designer1 : Fashion_Designer := new Fashion_Designer("Andre Correia",54);
    dcl designer2 : Fashion_Designer := new Fashion_Designer("Francisco Loua",64);
    dcl designer3 : Fashion_Designer := new Fashion_Designer("Afonso Martins",58);
    dcl designer4 : Fashion_Designer := new Fashion_Designer("Carlos Silva",62);
    --dcl designer5 : Fashion_Designer := new Fashion_Designer("Mario Andrade",59);
    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 model1_rw : Model_In_Runway := new Model_In_Runway(model1, mk_Fashion_Show`Date(2017, 12,
         31, 20, 00));
    dcl model2_rw : Model_In_Runway := new Model_In_Runway(model2, mk_Fashion_Show`Date(2017, 12,
         31, 20, 00));
    dcl model3_rw : Model_In_Runway := new Model_In_Runway(model3, mk_Fashion_Show`Date(2017, 12,
         31, 20, 00));
    dcl model4_rw : Model_In_Runway := new Model_In_Runway(model4, mk_Fashion_Show`Date(2017, 12,
         31, 20, 00));
    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 : Regular_User := new Regular_User("Diolinda",<Feminino>, "diolinda@gmail.com", "
        password_diolinda",true);
    dcl user2: Regular_User := new Regular_User("Diofeia",<Feminino>, "diofeia@gmail.com", "
        password_diofeia",false);
    dcl reviewer: Reviewer := new Reviewer("Ana Bacalhau",39,<Feminino>,"anabacalhau@gmail.com","
        password_anabacalhau");
    dcl reviewer2: Reviewer := new Reviewer("Ana Moura",45,<Feminino>,"anamoura@gmail.com","
        password_anamoura");
    dcl data1 : Fashion_Show`Date := mk_Fashion_Show`Date(2017, 12, 31, 10, 30);
    dcl data2 : Fashion_Show`Date := mk_Fashion_Show`Date(2017, 12, 31, 11, 00);
    dcl data3 : Fashion_Show`Date := mk_Fashion_Show`Date(2017, 12, 31, 12, 30);
    dcl data4 : Fashion_Show`Date := mk_Fashion_Show`Date(2017, 12, 31, 14, 00);
    dcl data5 : Fashion_Show`Date := mk_Fashion_Show`Date(2017, 12, 31, 15, 30);
    dcl data6 : Fashion_Show`Date := mk_Fashion_Show`Date(2017, 12, 31, 17, 00);
    dcl data7 : Fashion_Show`Date := mk_Fashion_Show`Date(2017, 12, 31, 18, 30);
    dcl critic : Fashion_Show`Critic := mk_Fashion_Show`Critic("Melhor festival de moda que
        participei!",5);
    dcl critic2 : Fashion_Show`Critic := mk_Fashion_Show`Critic("Evento aqum das expectativas."
        ,2);

    -- gets
```

```
assertEqual(show.getTheme(),"Primavera");
assertEqual(show.getLocation(),"Porto");
assertEqual(show.getDate(),mk_Fashion_Show`Date(2017, 12, 31, 9, 00));
assertEqual(show.getModels(),{|->});
assertEqual(show.getProgramShow(),{|->});

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

--get models
show.addDesignerToShow(designer1);
show.addDesignerToShow(designer2);
assertEqual(show.getModelsOfDesigner(designer1),[]);
assertEqual(show.getModelsOfDesigner(designer2),[]);
assertEqual(show.getModels(),{designer1|->[],designer2|->[]});
show.addModelToShow(designer1,model1_rw);
show.addModelToShow(designer1,model2_rw);
show.addModelToShow(designer1,model3_rw);
show.addModelToShow(designer2,model4_rw);
assertEqual(show.getModelsOfDesigner(designer1),[model1_rw,model2_rw,model3_rw]);
assertEqual(show.getModelsOfDesigner(designer2),[model4_rw]);
show.removeModelToShow(designer1,model2_rw);
assertEqual(show.getModelsOfDesigner(designer1),[model1_rw,model3_rw]);

-- 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});
show.rmvWorkShopToShow(workshop);
assertEqual(show.getWorkShops(),{workshop2});
show.rmvWorkShopToShow(workshop2);
assertEqual(show.getWorkShops(),{});

-- program
show.addDesignerToProgramShow(data1,designer1);
assertEqual(show.getProgramShow(),{data1|->designer1});
assertEqual(card dom show.getProgramShow(),1);
show.addDesignerToProgramShow(data2,designer2);
assertEqual(show.getProgramShow(),{data1|->designer1,data2|->designer2});
assertEqual(card dom show.getProgramShow(),2);
show.addDesignerToProgramShow(data3,designer3);
assertEqual(show.getProgramShow(),{data1|->designer1,data2|->designer2,data3|->designer3});
assertEqual(card dom show.getProgramShow(),3);
show.addDesignerToProgramShow(data4,designer4);
assertEqual(show.getProgramShow(),{data1|->designer1,data2|->designer2,data3|->designer3,
    data4|->designer4});
assertEqual(card dom show.getProgramShow(),4);

--invalid entry (try to break pre-condition by adding another event with already existing
    same date)
```

```
--show.addDesignerToProgramShow(data1,designer2);

--get designers by date
assertEqual(show.getDesignerByDate(data1),designer1);
assertEqual(show.getDesignerByDate(data2),designer2);
assertEqual(show.getDesignerByDate(data3),designer3);
assertEqual(show.getDesignerByDate(data4),designer4);

--add same designer to another date
show.addDesignerToProgramShow(data5,designer1);
assertEqual(show.getProgramShow(),{data1|->designer1,data2|->designer2,data3|->designer3,
    data4|->designer4,data5|->designer1});
assertEqual(card dom show.getProgramShow(),5);
show.addDesignerToProgramShow(data6,designer2);
assertEqual(show.getProgramShow(),{data1|->designer1,data2|->designer2,data3|->designer3,
    data4|->designer4,data5|->designer1,data6|->designer2});
assertEqual(card dom show.getProgramShow(),6);
show.addDesignerToProgramShow(data7,designer1);
assertEqual(show.getProgramShow(),{data1|->designer1,data2|->designer2,data3|->designer3,
    data4|->designer4,data5|->designer1,data6|->designer2,data7|->designer1});
assertEqual(card dom show.getProgramShow(),7);


--get list of dates by designer
assertEqual(show.getListOfDatesByDesigner(designer1),{data1,data5,data7});
assertEqual(show.getListOfDatesByDesigner(designer2),{data2,data6});

--invalid entry (try to break pre-condition by retriving list of dates by a designer that not
     exists on program)
--assertEqual(show.getListOfDatesByDesigner(designer5),{});

--remove date from program show
show.removeDesignerFromProgramShow(data7);
assertEqual(show.getProgramShow(),{data1|->designer1,data2|->designer2,data3|->designer3,
    data4|->designer4,data5|->designer1,data6|->designer2});
assertEqual(card dom show.getProgramShow(),6);
show.removeDesignerFromProgramShow(data6);
assertEqual(show.getProgramShow(),{data1|->designer1,data2|->designer2,data3|->designer3,
    data4|->designer4,data5|->designer1});
assertEqual(card dom show.getProgramShow(),5);
show.removeDesignerFromProgramShow(data5);
assertEqual(show.getProgramShow(),{data1|->designer1,data2|->designer2,data3|->designer3,
    data4|->designer4});
assertEqual(card dom show.getProgramShow(),4);

--invalid entry (try to break pre-condition by removing a date that not exists on program)
--show.removeDesignerFromProgramShow(data5);

show.removeDesignerFromProgramShow(data4);
assertEqual(show.getProgramShow(),{data1|->designer1,data2|->designer2,data3|->designer3});
assertEqual(card dom show.getProgramShow(),3);

--invalid entry (try to break pre-condition by removing a date that not exists on program)
--show.removeDesignerFromProgramShow(data4);

show.removeDesignerFromProgramShow(data3);
assertEqual(show.getProgramShow(),{data1|->designer1,data2|->designer2});
assertEqual(card dom show.getProgramShow(),2);

show.removeDesignerFromProgramShow(data2);
assertEqual(show.getProgramShow(),{data1|->designer1});
assertEqual(card dom show.getProgramShow(),1);

show.removeDesignerFromProgramShow(data1);
assertEqual(show.getProgramShow(),{|->});
```

```
    -- critics
    assertEqual(show.getCritics(),{|->});
    assertEqual(show.getAvgReview(),0);
    show.addCritic(reviewer, critic);
    assertEqual(show.getCritics(),{reviewer|->critic});
    assertEqual(show.getAvgReview(),5);
    show.addCritic(reviewer2, critic2);
    assertEqual(show.getCritics(),{reviewer|->critic,reviewer2|->critic2});
    assertEqual(show.getAvgReview(),3.5);
    show.rmvCritic(reviewer);
    assertEqual(show.getCritics(),{reviewer2|->critic2});
    assertEqual(show.getAvgReview(),2);
    show.rmvCritic(reviewer2);
    assertEqual(show.getCritics(),{|->});
    assertEqual(show.getAvgReview(),0);

    --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	192	100.0%	2
Show_Test.vdmpp		100.0%	5

19 User

```
class User
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
   protected name : String :="Default_Name";
   protected gender : Gender := <Masculino>;
   protected email : String := "";
   protected password : String := "";
operations
-- TODO Define operations here

   --Retorna o nome

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

   --Retorna o genero

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

   --Retorna o email

   public pure getEmail : () ==> String
   getEmail() ==
   (
    return email;
   );

   --Retorna a password

   public pure getPassword : () ==> String
   getPassword() ==
   (
    return password;
   );

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

Function or operation	Line	Coverage	Calls
getEmail	32	100.0%	18
getGender	25	100.0%	18
getName	18	100.0%	18
getPassword	39	100.0%	18
User.vdmpp		100.0%	72

20 WorkShop

```
class WorkShop
```

```
types
-- TODO Define types here
   public String = seq of char;
   public Users = set of Regular_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 in cio

   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 : (Regular_User) ==> ()
   addUserToWorkshop(Regular_User) ==
   (
    registered_users := registered_users union {Regular_User};
    return;
   )
   pre Regular_User not in set registered_users and card registered_users < lotation
   post Regular_User in set registered_users;

   --Remove um utilizador   workshop

   public rmvUserToWorkshop : (Regular_User) ==> ()
   rmvUserToWorkshop(Regular_User) ==
   (
    registered_users := registered_users \ {Regular_User};
    return;
   )
   pre Regular_User in set registered_users
   post Regular_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	25	100.0%	21
addUserToWorkshop	88	100.0%	28

getBeginDate	46	100.0%	7
getEndDate	53	100.0%	7
getLotation	60	100.0%	21
getOrator	67	100.0%	7
getRoom	74	100.0%	7
getTheme	39	100.0%	7
getUsers	81	100.0%	70
rmvUserToWorkshop	98	100.0%	14
WorkShop.vdmpp		100.0%	189

21 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 : Regular_User := new Regular_User("Diolinda",<Feminino>, "diolinda@gmail.com", "
        password_diolinda",true);
    dcl user2: Regular_User := new Regular_User("Diofeia",<Feminino>, "diofeia@gmail.com", "
        password_diofeia",false);

    -- 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});
    workshop.rmvUserToWorkshop(user1);
    assertEqual(workshop.getUsers(),{user2});
    workshop.rmvUserToWorkshop(user2);
    assertEqual(workshop.getUsers(),{});

    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%	7
main_test	41	100.0%	12
WorkShop_Test.vdmpp		100.0%	19