MFES—MagicStay

January 3, 2018

Contents

1 Accomodation

```
class Accomodation
types
 public TypeOf = <House> | <Apartment> | <ApartaHotel> | <Hotel>;
instance variables
 area: seq of char;
 numBedrooms: nat;
 numBeds: nat;
 numStars: nat;
 numOfBathrooms: nat;
 host: Landlord;
 type: TypeOf;
 price: real;

 reservations: set of Reservation := {};
 reviews: map User to Review := {|->};
operations
 public Accomodation: seq of char * nat * nat * nat * Landlord * TypeOf * real ==> Accomodation
 Accomodation(a, nBedrooms, nBeds, nBathrooms, user, typeOf, p) == (
  area := a;
  numBedrooms := nBedrooms;
  numBeds := nBeds;

  numStars := 0;
  numOfBathrooms := nBathrooms;
  host := user;
  type := typeOf;
  price := p;
  host.addHouse(self);
 );

 public getArea: () ==> seq of char
 getArea() == return area;

 public getNumBedrooms: () ==> nat
 getNumBedrooms() == return numBedrooms;


 public getNumStars: () ==> nat
 getNumStars() == return numStars;


 public getNumBeds: () ==> nat
 getNumBeds()== return numBeds;


 public getNumOfBathrooms: () ==> nat
 getNumOfBathrooms() == return numOfBathrooms;


 public getHost: () ==> Landlord
 getHost() == return host;


 public getType: () ==> TypeOf
 getType() == return type;


 public getPrice: () ==> real
 getPrice() == return price;
```

```
pure public getReservations: () ==> set of Reservation
getReservations() == return reservations;



private updateNumStars: () ==> ()
updateNumStars() == (

 dcl sum: nat := 0;
 dcl n: nat := card dom reviews;


 for all r in set rng reviews do (
  sum := sum + r.getRating()
 );
 if n = 0 then n := 1;
 numStars := sum / n;

);

public addReview: Review ==> ()
addReview(rev) == (
 reviews := reviews ++ {rev.getUser() |-> rev};

 updateNumStars()
)
pre rev.getUser() not in set dom reviews

post rev in set rng reviews;

public removeReview: Review ==> ()
removeReview(rev) == (
 reviews := {rev.getUser()} <-: reviews;
 updateNumStars()
)
pre rev in set rng reviews
post (rev not in set rng reviews) and (rev.getUser() not in set dom reviews);

public getReviews: () ==> map User to Review
getReviews() == return reviews;


public addReservation: Reservation ==> ()
addReservation(res) == (
 res.setPrice(price);
 res.getUser().transaction(-price);
 host.transaction(price);
 reservations := reservations union {res};
)
pre res.getUser().getWallet() > price and card {reservation | reservation in set reservations &
    overlaps(res, reservation)} = 0
post res in set reservations;


public cancelReservation: Reservation ==> ()
cancelReservation(res) == (
 dcl p: real := res.getPrice();
 res.getUser().transaction(p);
 host.transaction(-p);
 reservations := reservations \ {res};
)
pre res in set reservations;

pure private overlaps: Reservation * Reservation ==> bool
overlaps(res1, res2) == (
  if Types`compare(res1.getCheckIn().date, res2.getCheckOut().date) <= 0 or Types`compare(res2.
      getCheckIn().date, res1.getCheckOut().date) <= 0 then (
   return false)
  else
   return true;
 );
end Accomodation
```

Function or operation	Line	Coverage	Calls
Accomodation	22	100.0%	17
addReservation	75	96.7%	8
addReview	62	100.0%	2
cancelReservation	98	100.0%	1
getArea	35	100.0%	12
getHost	50	100.0%	4
getNumBedrooms	38	100.0%	7
getNumBeds	44	100.0%	6
getNumOfBathrooms	47	100.0%	6
getNumStars	41	100.0%	5
getPrice	56	100.0%	7
getReservations	59	100.0%	7
getReviews	72	100.0%	2
getType	53	100.0%	6
overlaps	88	92.3%	0
removeReview	67	100.0%	1
updateNumStars	57	100.0%	3
Accomodation.vdmpp		98.3%	94

2 Account

```
class Account
instance variables
 protected firstName: seq of char;
 protected fullName: seq of char;
 protected email: seq of char;
 protected password: seq of char;
 protected phoneNumber: seq of char;

 protected wallet: real;
 protected inbox: set of Message := {};
operations

 public getFirstName: () ==> seq of char
 getFirstName() == return firstName;


 public getFullName: () ==> seq of char
 getFullName() == return fullName;


 public getEmail: () ==> seq of char
 getEmail() == return email;
 public getPassword: () ==> seq of char
 getPassword() == return password;


 public getPhoneNumber: () ==> seq of char
 getPhoneNumber() == return phoneNumber;


 pure public getWallet: () ==> real
 getWallet() == return wallet;


 public getMessages: () ==> set of Message
 getMessages() == return inbox;


 public transaction: real ==> ()
 transaction(value) == wallet := wallet + value;


 public addMessage: Message ==> ()
 addMessage(message) == inbox := inbox union {message};


 public getUserType: () ==> seq of char
 getUserType() == is subclass responsibility;
end Account
```

Function or operation	Line	Coverage	Calls
addMessage	36	100.0%	6
getEmail	18	100.0%	4
getFirstName	12	100.0%	5
getFullName	15	100.0%	4
getMessages	30	100.0%	4
getPassword	21	100.0%	4
getPhoneNumber	24	100.0%	4
getUserType	39	100.0%	2
getWallet	27	100.0%	14
transaction	33	100.0%	18
Account.vdmpp		100.0%	65

3 Landlord

```
class Landlord is subclass of Account
types
 public TypeOf = <Owner> | <RentalAgency> | <ApartHotel>;
 public LanguageCommunication = <English> | <French> | <Spanish> | <German> | <Italian>;
instance variables
 gender: Types`Gender;
 language: LanguageCommunication;
 address: seq of char;
 city: seq of char;
 postalCode: seq of char;
 country: seq of char;
 website: seq of char;
 type: TypeOf;

 houses: set of Accomodation := {};
operations
 public Landlord: seq of char * seq of char * Types`Gender * seq of char * seq of char * seq of
     char * LanguageCommunication * seq of char * seq of char * seq of char * seq of char * seq
     of char * TypeOf * real ==> Landlord
 Landlord(fName, name, gend, phoneNum, em, pw, lang, adr, ct, pc, ctr, web, typ, initWallet) == (
  firstName := fName;
  fullName := name;
  gender := gend;
  phoneNumber := phoneNum;
  email := em;
  password := pw;
  language := lang;
  address := adr;
  city := ct;
  postalCode := pc;
  country := ctr;
  website := web;

  type := typ;
  wallet := initWallet;
 );

 public getGender: () ==> Types`Gender
 getGender() == return gender;

 public getLanguage: () ==> LanguageCommunication
 getLanguage() == return language;

 public getAddress: () ==> seq of char
 getAddress() == return address;

 public getCity: () ==> seq of char
 getCity() == return city;

 public getPostalCode: () ==> seq of char
 getPostalCode() == return postalCode;


 public getCountry: () ==> seq of char
 getCountry() == return country;


 public getWebsite: () ==> seq of char
 getWebsite() == return website;


 public getType: () ==> TypeOf
 getType() == return type;


 pure public getHouses: () ==> set of Accomodation
 getHouses() == return houses;


 public addHouse: Accomodation ==> ()
 addHouse(house) == houses := houses union {house};
 public getUserType: () ==> seq of char
 getUserType() == return "Landlord";

end Landlord
```

Function or operation	Line	Coverage	Calls
Landlord	31	100.0%	11
addHouse	100	100.0%	17
addMessage	103	0.0%	0
getAddress	70	100.0%	2
getCity	73	100.0%	2
getCountry	79	100.0%	2
getEmail	61	100.0%	17
getFirstName	49	100.0%	2
getFullName	52	100.0%	2
getGender	55	100.0%	2
getHouses	88	100.0%	1
getLanguage	67	100.0%	2
getMessages	94	0.0%	0
getPassword	64	0.0%	0
getPhoneNumber	58	100.0%	1
getPostalCode	76	100.0%	2
getType	85	100.0%	2
getUserType	65	0.0%	0
getWallet	91	0.0%	0
getWebsite	82	100.0%	2
transaction	97	0.0%	0
Landlord.vdmpp		96.3%	67

4 Message

```
class Message
instance variables
 sender: Account;
 receiver: Account;
 content: seq of char;
 inReplyTo: [Message];
operations
 public Message: Account * Account * seq of char * [Message] ==> Message
 Message(s, r, c, m) == (
  sender := s;
  receiver := r;

  content := c;
  inReplyTo := m;
  receiver.addMessage(self);
 )
 pre (m = nil) or (m.getSender() = r and m.getReceiver() = s);


 public getConversation: () ==> seq of Message
 getConversation() == (
  if inReplyTo <> nil then
   return inReplyTo.getConversation()^[self]
  else
   return [self];
 );


 pure public getSender: () ==> Account
 getSender() == return sender;


 pure public getReceiver: () ==> Account
 getReceiver() == return receiver;


 public getContent: () ==> seq of char
 getContent() == return content;

end Message
```

Function or operation	Line	Coverage	Calls
Message	12	100.0%	6
getContent	32	100.0%	1
getConversation	18	0.0%	0
getReceiver	29	100.0%	1
getSender	26	100.0%	2
Message.vdmpp		68.1%	10

5 Reservation

```
class Reservation
instance variables
 user: User;
 house: Accomodation;
 checkIn: Types`DateTime;
 checkOut: Types`DateTime;
 price: real;
operations
 public Reservation: User * Accomodation * Types`DateTime * Types`DateTime ==> Reservation
 Reservation(u, h, cI, cO) == (
  user := u;
  house := h;
  checkIn := cI;

  checkOut := cO;
  house.addReservation(self);
 );
 -- TODO
  -- Add postconditions
 pure public getUser: () ==> User
 getUser() == return user;

 public getAccomodation: () ==> Accomodation
 getAccomodation() == return house;
 pure public getCheckIn: () ==> Types`DateTime
 getCheckIn() == return checkIn;


 pure public getCheckOut: () ==> Types`DateTime
 getCheckOut() == return checkOut;


 public getPrice: () ==> nat
 getPrice() == return price;


 public setPrice: real ==> ()
 setPrice(p) == price := p;

end Reservation
```

Function or operation	Line	Coverage	Calls
Reservation	14	100.0%	8
getAccomodation	27	100.0%	3
getCheckIn	30	100.0%	23
getCheckOut	33	100.0%	23
getPrice	36	100.0%	3
getUser	24	100.0%	19
setPrice	39	100.0%	8
Reservation.vdmpp		100.0%	87

6 Review

```
class Review
instance variables
 user: User;
 house: Accomodation;
 content: seq of char;
 rating: nat;

 inv rating >= 0 and rating <= 5;
operations
 public Review: User * Accomodation * seq of char * nat ==> Review
 Review(u, h, c, star) == (

  user := u;
  house := h;
  content := c;
  rating := star;
  house.addReview(self);
 );


 pure public getUser: () ==> User
 getUser() == return user;
 public getContent: () ==> seq of char
 getContent() == return content;


 public getRating: () ==> nat
 getRating() == return rating;
end Review
```

Function or operation	Line	Coverage	Calls
Review	12	100.0%	2
getContent	22	100.0%	2
getRating	25	100.0%	2
getReview	22	100.0%	2
getUser	19	100.0%	8
Review.vdmpp		100.0%	16

7 Search

```
class Search
instance variables
 area: seq of char;
 checkIn: Types`Date;
 checkOut: Types`Date;
 typeOfAccomodation: Accomodation`TypeOf;
 numBedrooms: nat;
 numOfStars: nat;
 numOfBeds: nat;
 numOfBathrooms: nat;
 lowRangePrice: real;
 highRangePrice: real;

 results: set of Accomodation := {};

 --Inv
 inv numOfStars >= 0 and numOfStars <= 5;
 inv numOfBathrooms > 0 and numOfBathrooms <6;
 inv lowRangePrice > 0;
 inv highRangePrice > 0 and highRangePrice > lowRangePrice;

operations
 public Search: seq of char * Types`Date * Types`Date * Accomodation`TypeOf * nat * nat * nat *
     nat * real * real ==> Search
 Search(a, cIn, cOut, typeOfAcco, nBedrooms, nOfStars,nOfBeds,nOfBathrooms,lowPrice,highPrice) ==
      (
  area := a;
  checkIn := cIn;
  checkOut := cOut;
  numBedrooms:= nBedrooms;
  typeOfAccomodation := typeOfAcco;

  numOfStars := nOfStars;
  numOfBeds := nOfBeds;
  numOfBathrooms := nOfBathrooms;
  lowRangePrice := lowPrice;
  highRangePrice := highPrice;
 );
 --TODO
  --Adicionar post conditions
 public searchResults: set of Accomodation ==> set of Accomodation
 searchResults(accomodations) == (
  results := {};
  for all a in set accomodations do (
   if a.getArea() = area and
      a.getNumBedrooms() = numBedrooms and
      a.getType() = typeOfAccomodation and

      a.getNumStars() = numOfStars and
      a.getNumBeds() = numOfBeds and
      a.getNumOfBathrooms() = numOfBathrooms and
      a.getPrice() >= lowRangePrice and
      a.getPrice() <= highRangePrice then (
       results := results union {a};
      );
  );
  return results;
 );

 public getArea: () ==> seq of char
 getArea() == return area;

 public getCheckIn: () ==> Types`Date
 getCheckIn() == return checkIn;

 public getCheckOut: () ==> Types`Date

 getCheckOut() == return checkOut;

 public getNumBedrooms: () ==> nat

 getNumBedrooms() == return numBedrooms;

 public getTypeOfAccomodation: () ==> Accomodation`TypeOf

 getTypeOfAccomodation() == return typeOfAccomodation;

 public getNumOfStars: () ==> nat

 getNumOfStars() == return numOfStars;

 public getNumOfBeds: () ==> nat

 getNumOfBeds() == return numOfBeds;

 public getNumOfBathrooms: () ==> nat

 getNumOfBathrooms() == return numOfBathrooms;

 public getLowRangePrice: () ==> real

 getLowRangePrice() == return lowRangePrice;

 public getHighRangePrice: () ==> real

 getHighRangePrice() == return highRangePrice;
end Search
```

Function or operation	Line	Coverage	Calls
Search	30	100.0%	3
getArea	63	100.0%	1
getCheckIn	66	100.0%	1
getCheckOut	69	100.0%	1
getHighRangePrice	90	100.0%	1
getLowRangePrice	87	100.0%	1
getNumBedrooms	72	100.0%	1
getNumOfBathrooms	84	100.0%	1
getNumOfBeds	81	100.0%	1
getNumOfStars	78	100.0%	1
getTypeOfAccomodation	75	100.0%	1
searchResults	45	100.0%	4
Search.vdmpp		100.0%	17

8 Types

```
class Types
types
 public Gender = <Male> | <Female>;
 public Date :: year : int
         month : int
         day : int
     inv date == date.year >=0 and date.month >0 and date.month <=12 and date.day >0 and date.day
          <=31;
 public Time :: hour : int
         minute : int
         second : int
     inv time == time.hour >= 0 and time.hour < 24 and
             time.minute >=0 and time.minute < 60 and
             time.second >=0 and time.second <60;
 public DateTime :: date : Date
           time : Time;
operations
 -- Returns -1 if d1 is after d2, 1 if d2 is after d1, or 0 if same date
 pure public static compare: Date * Date ==> int
 compare(d1,d2) == (
  if d1.year > d2.year then return -1
  else if d1.year < d2.year then return 1
  else

   if d1.month > d2.month then return -1
   else if d1.month < d2.month then return 1
   else
    if d1.day > d2.day then return -1
    else if d1.day < d2.day then return 1
    else return 0;
 )
end Types
```

Function or operation	Line	Coverage	Calls
compare	23	100.0%	5

Types.vdmpp		100.0%	5

9 User

```
class User is subclass of Account
instance variables
 --Default
 country: seq of char;
 company: seq of char;
 companyRegNum: nat;
 vat: nat;

 --Others
 favorites: set of Accomodation := {};

 --Inv
 inv companyRegNum >= 0;
 inv vat >= 0;

operations
 public User: seq of char * seq of char * seq of char * seq of char * seq of char *seq of char *
     seq of char * int * int * real ==> User
 User(fName, name, em, pw, count, phNum, cp, cpRegNum, v, initWallet) == (
  firstName := fName;
  fullName := name;
  email := em;
  password := pw;
  country := count;
  phoneNumber := phNum;
  company := cp;
  companyRegNum := cpRegNum;
  vat := v;
  wallet := initWallet;
 );

 --TODO
  --Add post conditions

 public getCountry: () ==> seq of char
 getCountry() == return country;

 public getCompany: () ==> seq of char
 getCompany() == return company;

 public getCompanyRegNum: () ==> nat
 getCompanyRegNum() == return companyRegNum;

 public getVat: () ==> nat
 getVat() == return vat;

 public getFavorites: () ==> set of Accomodation
 getFavorites() == return favorites;


 public addFavorite: Accomodation ==> ()
 addFavorite(h) == favorites := favorites union {h}

 pre h not in set favorites;


 public addFavorites: set of Accomodation ==> ()
 addFavorites(acc) == (
  for all a in set acc do (
   favorites := favorites union {a};

  );
 );


 public removeFavorite: Accomodation ==> ()
 removeFavorite(h) == favorites := favorites \ {h}
 pre h in set favorites;



 public getUserType: () ==> seq of char
 getUserType() == return "User";

end User
```

Function or operation	Line	Coverage	Calls
User	30	100.0%	8
addFavorite	87	100.0%	3
addFavorites	52	100.0%	1
addMessage	98	0.0%	0
getCompany	65	100.0%	2
getCompanyRegNum	68	100.0%	2
getCountry	59	100.0%	2
getEmail	53	100.0%	1
getFavorites	80	100.0%	5
getFirstName	47	100.0%	3
getFullName	50	100.0%	3
getMessages	77	0.0%	0
getPassword	56	0.0%	0
getPhoneNumber	62	0.0%	0
getUserType	63	0.0%	0
getVat	71	100.0%	2
getWallet	74	0.0%	0
removeFavorite	92	100.0%	1
transaction	83	0.0%	0
User.vdmpp		96.9%	33

10 AccomodationTest

```
class AccomodationTest is subclass of Test
operations
 public createAccomodation: Landlord ==> Accomodation
 createAccomodation(landlord)==
  return new Accomodation("Porto",2,2,1,landlord,<House>, 120);

 public testCreateAccomodation: () ==> ()
 testCreateAccomodation() == (
  dcl l: Landlord := new Landlord("Bruno","Bruno Santos",<Male>,"922222222", "bruno@gmail.com","
      1234",<English>, "Rua das Flores", "Porto","4400-458","Portugal", "www.casas.com", <Owner>,
       0);

  dcl a: Accomodation := createAccomodation(l);
  assert(a.getArea() = "Porto");
  assert(a.getNumBedrooms() = 2);
  assert(a.getNumBeds() = 2);

  assert(a.getNumOfBathrooms() = 1);
  assert(a.getHost() = l);
  assert(a.getType() = <House>);
  assert(a in set l.getHouses());
 );

 public test: () ==> ()
 test() == (
  testCreateAccomodation();
 );
end AccomodationTest
```

Function or operation	Line	Coverage	Calls
createAccomodation	10	100.0%	1
test	28	100.0%	1
testCreateAccomodation	14	100.0%	3
AccomodationTest.vdmpp		100.0%	5

11 LandlordTest

```
class LandlordTest is subclass of Test
operations
 public createLandlord: () ==> Landlord
 createLandlord() ==
  return new Landlord("Bruno","Bruno Santos",<Male>,"922222222", "bruno@gmail.com","1234",<
      English>, "Rua das Flores", "Porto","4400-458","Portugal", "www.casas.com", <Owner>, 0);

 public testCreateLandlord: () ==> ()
 testCreateLandlord() == (
  dcl l : Landlord := createLandlord();

  assert(l.getFirstName() = "Bruno");
  assert(l.getFullName() = "Bruno Santos");
  assert(l.getGender() = <Male>);
  assert(l.getPhoneNumber() = "922222222");

  assert(l.getEmail() = "bruno@gmail.com");
  assert(l.getPassword() = "1234");
  assert(l.getLanguage() = <English>);
  assert(l.getAddress() = "Rua das Flores");
  assert(l.getCity() = "Porto");
  assert(l.getPostalCode() = "4400-458");
  assert(l.getCountry() = "Portugal");
  assert(l.getWebsite() = "www.casas.com");
  assert(l.getType() = <Owner>);
 );
 public test: () ==> ()
 test() == (
  testCreateLandlord();
 );
end LandlordTest
```

Function or operation	Line	Coverage	Calls
createLandlord	10	100.0%	1
test	32	100.0%	1
testCreateLandlord	14	100.0%	1
LandlordTest.vdmpp		100.0%	3

12 MagicStayTest

```
class MagicStayTest
operations
 public static main: () ==> ()
 main() == (
  IO`println("Starting MagicStay Tests");

  IO`print("Testing User...");
  new UserTest().test();
  IO`println("successfully!");


  IO`print("Testing Landlord...");
  new LandlordTest().test();
  IO`println("successfully!");

  IO`print("Testing Accomodation...");
  new AccomodationTest().test();
  IO`println("successfully!");

  IO`print("Testing Reservation...");
  new ReservationTest().test();
  IO`println("successfully!");

  IO`print("Testing Review...");
  new ReviewTest().test();
  IO`println("successfully!");

  IO`print("Testing Search...");
  new SearchTest().test();
  IO`println("successfully!");

  IO`print("Testing Message...");
  new MessageTest().test();
  IO`println("successfully!");

  IO`print("Testing a real case use of Magic Stay...");
  new RealCaseTest().test();
  IO`println("successfully!");
 );
end MagicStayTest
```

Function or operation	Line	Coverage	Calls
main	10	100.0%	1
MagicStayTest.vdmpp		100.0%	1

13 MessageTest

```
class MessageTest is subclass of Test
operations
 public createUser: () ==> User
 createUser() ==
  return new User("Antonio","Antonio Melo", "antonio@gmail.com", "12345", "Portugal", "911111111"
      , "Empresa", 123, 123, 50000);

 public createLandlord: () ==> Landlord
 createLandlord() ==
  return new Landlord("Bruno","Bruno Santos",<Male>,"922222222", "bruno@gmail.com","1234",<
      English>, "Rua das Flores", "Porto","4400-458","Portugal", "www.casas.com", <Owner>, 0);


 public testCreateMessage: () ==> ()
 testCreateMessage() == (
  dcl u1: User := createUser();

  dcl u2: User := createUser();
  dcl l1: Landlord := createLandlord();
  dcl l2: Landlord := createLandlord();


  dcl m1: Message := new Message(u1, u2, "Hello", nil);
  dcl m2: Message := new Message(u1, l1, "Hello", nil);
  dcl m3: Message := new Message(l1, u1, "Hello", nil);
  dcl m4: Message := new Message(l1, l2, "Hello", nil);

  assert(m1 in set u2.getMessages());
  assert(m2 in set l1.getMessages());
  assert(m3 in set u1.getMessages());
  assert(m4 in set l2.getMessages());
 );


 public testConversation: () ==> ()
 testConversation() == (
  dcl u1: User := createUser();
  dcl u2: User := createUser();

  dcl m1: Message := new Message(u1, u2, "Hello", nil);
  dcl m2: Message := new Message(u2, u1, "Hello", m1);

  dcl m3: Message := new Message(u1, u2, "How are you?", m2);
  dcl m4: Message := new Message(u2, u1, "Good", m3);

  dcl c: seq of Message := m4.getConversation();

  assert([m1, m2, m3, m4] = c);
 );

 public test: () ==> ()
 test() == (
  testCreateMessage();
 );
```

```
end MessageTest
```

Function or operation	Line	Coverage	Calls
createLandlord	14	100.0%	2
createUser	10	100.0%	2
test	36	100.0%	1
testConversation	29	0.0%	0
testCreateMessage	18	100.0%	1
MessageTest.vdmpp		68.6%	6

14 RealCaseTest

```
class RealCaseTest is subclass of Test
types
-- TODO Define types here
values
-- TODO Define values here
instance variables
-- TODO Define instance variables here
 user: User;
 landlords: seq of Landlord := [];
 accomodations: set of Accomodation := {};
 results: set of Accomodation := {};
 accReserve: Accomodation;
 reservation: Reservation;
operations
-- TODO Define operations here


 public testSignUpLandlords: () ==> ()
 testSignUpLandlords() == (
  dcl landlord : Landlord := new Landlord("Bruno","Bruno Santos",<Male>,"922222222", "bruno@gmail
      .com","1234",<English>, "Rua das Flores", "Porto","4400-458","Portugal", "www.casas.com", <
      Owner>, 0);
  dcl landlord1 : Landlord := new Landlord("Louren o","Louren o Aires",<Male>,"933333333", "
      louren o@gmail.com","gg",<Spanish>, "Jardim das Rodas", "Madrid","4400-478","Spain", "www
      .casasbonitas.com", <Owner>, 0);
  assert(landlord.getFirstName() = "Bruno");
  assert(landlord.getGender() = <Male>);
  assert(landlord.getEmail() = "bruno@gmail.com");
  assert(landlord.getLanguage() = <English>);
  assert(landlord.getCity() = "Porto");
  assert(landlord.getCountry() = "Portugal");
  assert(landlord.getType() = <Owner>);
  assert(landlord1.getFullName() = "Louren o Aires");
  assert(landlord1.getPhoneNumber() = "933333333");
  assert(landlord1.getPassword() = "gg");
  assert(landlord1.getAddress() = "Jardim das Rodas");
  assert(landlord1.getPostalCode() = "4400-478");
  assert(landlord1.getWebsite() = "www.casasbonitas.com");
  assert(landlord1.getWallet() = 0);
  landlords:= landlords^[landlord,landlord1];
  assert(landlords(1) = landlord);
 );
```

```
public testCreateAccomodations: () ==> ()
testCreateAccomodations() == (
  dcl acc : Accomodation := new Accomodation("Porto",2,1,1,landlords(1),<Apartment>, 120);
 dcl acc1 : Accomodation := new Accomodation("Lisboa",1,2,1,landlords(1),<ApartaHotel>, 70);
 dcl acc2 : Accomodation := new Accomodation("Algarve",3,5,2,landlords(1),<House>, 320);
 dcl acc3 : Accomodation := new Accomodation("Porto",4,6,3,landlords(1),<House>, 440);
 dcl acc4 : Accomodation := new Accomodation("Madrid",2,2,1,landlords(2),<House>, 250);
 dcl acc5 : Accomodation := new Accomodation("Barcelona",3,4,2,landlords(2),<House>, 350);
 dcl acc6 : Accomodation := new Accomodation("Porto",4,6,3,landlords(2),<House>, 590);
 dcl acc7 : Accomodation := new Accomodation("Madrid",2,2,2,landlords(2),<Hotel>, 275);

 assert(acc.getArea() = "Porto");
 assert(acc1.getNumBedrooms() = 1);
 assert(acc2.getNumBeds() = 5);
 assert(acc3.getNumOfBathrooms() = 3);
 assert(landlords(1).getFirstName() = "Bruno");
 assert(acc4.getHost() = landlords(2));
 assert(acc5.getType() = <House>);
 assert(acc6.getPrice() = 590);
 accomodations := accomodations union {acc,acc1,acc2,acc3,acc4,acc5,acc6,acc7};
 results := results union {acc3,acc6};
 accReserve := acc3;
);


public testSignUpUser: () ==> ()
testSignUpUser() == (
 user:= new User("Ant nio","Ant nio Teixeira de Melo", "antonio@gmail.com", "12345", "
     Portugal", "911111111", "Empresa", 123, 123, 50000);
 assert(user.getFirstName() = "Ant nio");
 assert(user.getFullName() = "Ant nio Teixeira de Melo");
 assert(user.getEmail() = "antonio@gmail.com");
 assert(user.getPassword() = "12345");
 assert(user.getCountry() = "Portugal");
 assert(user.getPhoneNumber() = "911111111");
 assert(user.getCompany() = "Empresa");
 assert(user.getCompanyRegNum() = 123);
 assert(user.getVat() = 123);
 assert(user.getWallet() = 50000);
);


public testSearchAccomodationsAndAddFavorites: () ==> ()
testSearchAccomodationsAndAddFavorites() == (
 dcl search : Search := new Search("Porto",mk_Types`Date(2018,1,19),mk_Types`Date(2018,1,30),<
     House>,4,0,6,3,500,1000);
 assert(search.searchResults(accomodations) subset results);
 user.addFavorites(results);
 assert(user.getFavorites() = results);
);


public testMakeConversationWithLandlord: () ==> ()
testMakeConversationWithLandlord() == (
 dcl msg : Message := new Message(user,landlords(1),"Does it have car park?",nil);
 dcl msg1 : Message := new Message(landlords(1), user, "Yes!!", msg);
 assert(msg.getSender() = user);
 assert(msg1.getContent() = "Yes!!");
);


public testReserve: () ==> ()
testReserve() == (
  reservation := new Reservation(user,accReserve,mk_Types`DateTime(mk_Types`Date(2018,1,19),
      mk_Types`Time(14, 00, 00)),mk_Types`DateTime(mk_Types`Date(2018,1,30), mk_Types`Time(11,
      00, 00)));
  assert(reservation.getPrice() = 440);
 );


 public testReviewReserve: () ==> ()
 testReviewReserve() == (
  dcl review: Review := new Review(user, accReserve, "Awesome",4);
  assert(review.getUser() = user);
  assert(review.getContent() = "Awesome");
  assert(accReserve.getNumStars() = 4 );
 );


 public test: () ==> ()
 test() == (
  testSignUpLandlords();
  testCreateAccomodations();
  testSignUpUser();
  testSearchAccomodationsAndAddFavorites();
  testMakeConversationWithLandlord();
  testReserve();
  testReviewReserve();
 );

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

Function or operation	Line	Coverage	Calls
test	108	100.0%	1
testCreateAccomodations	39	100.0%	2
testMakeConversationWithLandlord	86	100.0%	3
testReserve	94	100.0%	1
testReviewReserve	100	100.0%	1
testSearchAccomodationsAndAddFavorites	78	100.0%	1
testSignUpLandlords	17	100.0%	1
testSignUpUser	63	100.0%	3
RealCaseTest.vdmpp		100.0%	13

15 ReservationTest

```
class ReservationTest is subclass of Test
operations
 public createReservation: User * Accomodation * Types`DateTime * Types`DateTime ==> Reservation
 createReservation(u, a, cI, cO) ==
  return new Reservation(u, a, cI, cO) ;

 public testCreateReservation: () ==> ()
 testCreateReservation() == (
  dcl price: real := 120;
  dcl userWallet: real := 50000;
  dcl hostWallet: real := 0;
  dcl cI: Types`DateTime := mk_Types`DateTime(mk_Types`Date(2018,1,19), mk_Types`Time(14, 00, 00)
      );
  dcl cO: Types`DateTime := mk_Types`DateTime(mk_Types`Date(2018,1,30), mk_Types`Time(11, 00, 00)
      );

  dcl u: User := new User("Antonio","Antonio Melo", "antonio@gmail.com", "12345", "Portugal", "
      911111111", "Empresa", 123, 123, userWallet);
  dcl l: Landlord := new Landlord("Bruno","Bruno Santos",<Male>,"922222222", "bruno@gmail.com","
      1234",<English>, "Rua das Flores", "Porto","4400-458","Portugal", "www.casas.com", <Owner>,
       hostWallet);
  dcl a : Accomodation := new Accomodation("Porto",4,2,1,l,<House>, price);
  dcl r : Reservation := createReservation(u,a,cI,cO);
  assert(r.getAccomodation() = a);
  assert(r.getCheckIn() = cI);
  assert(r.getCheckOut() = cO);
  assert(r.getPrice() = price);
  assert(r in set a.getReservations());
  assert(r.getUser().getWallet() = userWallet-price);
  assert(r.getAccomodation().getHost().getWallet() = hostWallet+price);
 );

 public testOverlappingReservations: () ==> ()
 testOverlappingReservations() == (
  dcl price: real := 120;
  dcl userWallet: real := 50000;
  dcl hostWallet: real := 0;
  dcl u: User := new User("Antonio","Antonio Melo", "antonio@gmail.com", "12345", "Portugal", "
      911111111", "Empresa", 123, 123, userWallet);
  dcl l: Landlord := new Landlord("Bruno","Bruno Santos",<Male>,"922222222", "bruno@gmail.com","
      1234",<English>, "Rua das Flores", "Porto","4400-458","Portugal", "www.casas.com", <Owner>,
       hostWallet);

  dcl a : Accomodation := new Accomodation("Porto",4,2,1,l,<House>, price);

  dcl cI1: Types`DateTime := mk_Types`DateTime(mk_Types`Date(2018,1,19), mk_Types`Time(14, 00,
      00));
  dcl cO1: Types`DateTime := mk_Types`DateTime(mk_Types`Date(2018,1,30), mk_Types`Time(11, 00,
      00));
  dcl r1 : Reservation := createReservation(u,a,cI1,cO1);

  dcl cI2: Types`DateTime := mk_Types`DateTime(mk_Types`Date(2018,1,8), mk_Types`Time(14, 00, 00)
      );
  dcl cO2: Types`DateTime := mk_Types`DateTime(mk_Types`Date(2018,1,19), mk_Types`Time(11, 00,
      00));
  dcl r2 : Reservation := createReservation(u,a,cI2,cO2);

  dcl cI3: Types`DateTime := mk_Types`DateTime(mk_Types`Date(2017,10,6), mk_Types`Time(14, 00,
      00));
  dcl cO3: Types`DateTime := mk_Types`DateTime(mk_Types`Date(2017,11,5), mk_Types`Time(11, 00,
      00));
  dcl r3 : Reservation := createReservation(u,a,cI3,cO3);

  dcl cI4: Types`DateTime := mk_Types`DateTime(mk_Types`Date(2017,12,26), mk_Types`Time(14, 00,
      00));
  dcl cO4: Types`DateTime := mk_Types`DateTime(mk_Types`Date(2018,1,4), mk_Types`Time(11, 00, 00)
      );
  dcl r4 : Reservation := createReservation(u,a,cI4,cO4);

  dcl cI5: Types`DateTime := mk_Types`DateTime(mk_Types`Date(2018,3,4), mk_Types`Time(14, 00, 00)
      );
  dcl cO5: Types`DateTime := mk_Types`DateTime(mk_Types`Date(2018,3,9), mk_Types`Time(11, 00, 00)
      );
```

```
  dcl r5 : Reservation := createReservation(u,a,cI5,cO5);

  dcl cI6: Types`DateTime := mk_Types`DateTime(mk_Types`Date(2018,1,3), mk_Types`Time(14, 00, 00)
      );
  dcl cO6: Types`DateTime := mk_Types`DateTime(mk_Types`Date(2018,1,8), mk_Types`Time(11, 00, 00)
      );
  dcl r6 : Reservation;

  a.cancelReservation(r4);
  r6 := createReservation(u,a,cI6,cO6);

  assert(r1 in set a.getReservations());
  assert(r2 in set a.getReservations());
  assert(r3 in set a.getReservations());
  assert(r4 not in set a.getReservations());
  assert(r5 in set a.getReservations());
  assert(r6 in set a.getReservations());

  assert(r1.getUser().getWallet() = userWallet-price*5);
  assert(r1.getAccomodation().getHost().getWallet() = hostWallet+price*5);
 );

 public test: () ==> ()
 test() == (
  testCreateReservation();
  testOverlappingReservations();
 );
end ReservationTest
```

Function or operation	Line	Coverage	Calls
createReservation	10	100.0%	7
test	93	100.0%	1
testCreateReservation	14	100.0%	5
testOverlappingReservations	34	100.0%	1
ReservationTest.vdmpp		100.0%	14

16 ReviewTest

```
class ReviewTest is subclass of Test
operations
 public createReview: User * Accomodation * seq of char * nat ==> Review
 createReview(user, house, review, rating) ==
  return new Review(user, house, review, rating);

 public testCreateReview: () ==> ()
 testCreateReview() == (
  dcl user: User := new User("Antonio","Antonio Melo", "antonio@gmail.com", "12345", "Portugal",
      "911111111", "Empresa", 123, 123, 50000);

  dcl landlord: Landlord := new Landlord("Bruno","Bruno Santos",<Male>,"922222222", "bruno@gmail.
      com","1234",<English>, "Rua das Flores", "Porto","4400-458","Portugal", "www.casas.com", <
      Owner>, 0);
  dcl accomodation : Accomodation := new Accomodation("Porto",4,2,1,landlord,<House>, 120);
  dcl review: Review := new Review(user, accomodation, "Very good!", 4);
  assert(review.getContent() = "Very good!");
  assert(review.getUser() = user);
  assert(review in set rng accomodation.getReviews());
  accomodation.removeReview(review);
  assert(review not in set rng accomodation.getReviews());
 );

 public test: () ==> ()
 test() == (
  testCreateReview();
 );
end ReviewTest
```

Function or operation	Line	Coverage	Calls
createReview	10	0.0%	0
test	26	100.0%	1
testCreateReview	14	100.0%	1
ReviewTest.vdmpp		90.9%	2

17 SearchTest

```
class SearchTest is subclass of Test
operations
 public createSearch: seq of char * Types`Date * Types`Date * Accomodation`TypeOf * nat  * nat *
     nat * nat * real * real ==> Search
 createSearch(a, cIn, cOut, typeOfAcco, nBedrooms, nOfStars,nOfBeds,nOfBathrooms,lowPrice,
     highPrice) ==
   return new Search(a, cIn, cOut, typeOfAcco, nBedrooms , nOfStars,nOfBeds,nOfBathrooms,lowPrice
       ,highPrice);
 public testCreateSearch: () ==> ()
 testCreateSearch() == (
  dcl s : Search := createSearch("Porto",mk_Types`Date(2018,1,19),mk_Types`Date(2018,1,30),<House
      >,1,4,2,1,500,1000);
  assert(s.getArea() = "Porto");

  assert(s.getCheckIn() = mk_Types`Date(2018,1,19));
  assert(s.getCheckOut() = mk_Types`Date(2018,1,30));
  assert(s.getNumBedrooms() = 1);

  assert(s.getTypeOfAccomodation() = <House>);
  assert(s.getNumOfStars() = 4);
  assert(s.getNumOfBeds() = 2);
  assert(s.getNumOfBathrooms() = 1);
  assert(s.getLowRangePrice() = 500);
  assert(s.getHighRangePrice() = 1000);
 );

 public testSearchResults: () ==> ()
 testSearchResults() == (
  dcl s : Search := createSearch("Porto",mk_Types`Date(2018,1,19),mk_Types`Date(2018,1,30),<House
      >,1,0,2,1,500,1000);
  dcl l: Landlord := new Landlord("Bruno","Bruno Santos",<Male>,"922222222", "bruno@gmail.com","
      1234",<English>, "Rua das Flores", "Porto","4400-458","Portugal", "www.casas.com", <Owner>,
       0);
  dcl ac : Accomodation := new Accomodation("Porto",1,2,1,1,<House>, 600);
  dcl al : Accomodation := new Accomodation("Porto",1,2,1,1,<House>, 120);
  dcl a: set of Accomodation := s.searchResults({ac,al});
  assert( a = {ac});
 );

 public test: () ==> ()
 test() == (
  testCreateSearch();
  testSearchResults();
 );
end SearchTest
```

Function or operation	Line	Coverage	Calls
createSearch	10	100.0%	2
test	38	100.0%	1
testCreateSearch	13	100.0%	1
testSearchResults	28	100.0%	1
SearchTest.vdmpp		100.0%	5

18 Test

```
class Test
operations
 protected assert : bool ==> ()
 assert(a) == return
 pre a
end Test
```

Function or operation	Line	Coverage	Calls
assert	9	100.0%	216
Test.vdmpp		100.0%	216

19 UserTest

```
class UserTest is subclass of Test
operations
 public createUser: () ==>  User
 createUser() == return new User("Antonio","Antonio Melo", "antonio@gmail.com", "12345", "
     Portugal", "911111111", "Empresa", 123, 123, 50000);

 public createLandlord: () ==> Landlord
 createLandlord() == return new Landlord("Bruno","Bruno Santos",<Male>,"922222222", "bruno@gmail.
     com","1234",<English>, "Rua das Flores", "Porto","4400-458","Portugal", "www.casas.com", <
     Owner>, 0);

 public testCreateUser: () ==> ()

 testCreateUser() == (
  dcl u: User := createUser();
  assert(u.getFirstName() = "Antonio");

  assert(u.getFullName() = "Antonio Melo");
  assert(u.getEmail() = "antonio@gmail.com");
  assert(u.getPassword() = "12345");

  assert(u.getCountry() = "Portugal");
  assert(u.getPhoneNumber() = "911111111");
  assert(u.getCompany() = "Empresa");
  assert(u.getCompanyRegNum() = 123);
  assert(u.getVat() = 123);
 );

 public testFavorites: () ==> ()
 testFavorites() == (
  dcl landlord: Landlord := createLandlord();
  dcl u1: User := createUser();
  dcl h1: Accomodation := new Accomodation("Porto",4,2,1,landlord,<House>, 120);
  dcl h2: Accomodation := new Accomodation("Porto",4,2,1,landlord,<House>, 80);
  dcl h3: Accomodation := new Accomodation("Porto",4,2,1,landlord,<House>, 90);

  u1.addFavorite(h1);
  u1.addFavorite(h2);
  assert(h2 in set u1.getFavorites());
  u1.addFavorite(h3);
  u1.removeFavorite(h2);
  assert(h1 in set u1.getFavorites());
  assert(h2 not in set u1.getFavorites());
  assert(h3 in set u1.getFavorites());
 );

 public test: () ==> ()
 test() == (
  testCreateUser();
  testFavorites();
 );
end UserTest
```

Function or operation	Line	Coverage	Calls
createLandlord	13	100.0%	1
createUser	10	100.0%	2
test	47	100.0%	1
testCreateUser	16	100.0%	3
testFavorites	30	100.0%	1
UserTest.vdmpp		100.0%	8