MFES_Safety_Net_Hospital

December 27, 2017

Contents

1 Appointment

```
class Appointment
values
 public APPOINTMENT_TIME_MIN = 30; -- duration of an appointment in minutes
instance variables
 private date : ModelUtils`Date;
  private hospitalId: nat;
  private doctorId: nat;
  private patientId: nat;

  -- inv
 inv hospitalId in set dom SafetyNetNetwork`getInstance().getHospitals() and   -- |
    doctorId in set dom SafetyNetNetwork`getInstance().getDoctors() and       -- |Check if they
        are in the network
    patientId in set dom SafetyNetNetwork`getInstance().getPatients() and     -- |
    doctorId in set SafetyNetNetwork`getInstance().getHospitals()(hospitalId).getDoctorsIds(); --
        check if the doctor works in that hospital

operations
 --constructor

 public Appointment: ModelUtils`Date * nat * nat * nat ==> Appointment
 Appointment(d,hos, doc, pat) == (
```

```
  atomic (
   date := d;
   hospitalId := hos;
   doctorId := doc;
   patientId:= pat;
  );
  return self;
 );

  --get appointment date

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

  --get appointment hospital

  public pure getHospitalId: () ==> nat
  getHospitalId() == (
   return hospitalId
  );

  --get appointment doctor

  public pure getDoctorId: () ==> nat
  getDoctorId() == (
   return doctorId
  );

  --get appointment patient

  public pure getPatientId: () ==> nat
  getPatientId() == (
   return patientId
  );

functions
 --checks if two dates are equal

 static public appointmentDatesDontOverlap: ModelUtils`Date * ModelUtils`Date -> bool
 appointmentDatesDontOverlap(d1,d2) == (
  (ModelUtils`dateToNat(getNextAppointmentDate(d1)) <= ModelUtils`dateToNat(d2)) or
  (ModelUtils`dateToNat(getNextAppointmentDate(d2)) <= ModelUtils`dateToNat(d1))
 );

 --get next appointment slot date

 static public getNextAppointmentDate: ModelUtils`Date -> ModelUtils`Date
 getNextAppointmentDate(d) == (

  if(d.month = 12 and d.day=30 and d.hour = 23 and d.min >= (ModelUtils`HOUR_MIN -
      APPOINTMENT_TIME_MIN)) then
   mk_ModelUtils`Date(d.year + 1, 1, 1, 00, (d.min + APPOINTMENT_TIME_MIN) mod ModelUtils`
       HOUR_MIN)
  elseif(d.day=30 and d.hour = 23 and d.min >= (ModelUtils`HOUR_MIN - APPOINTMENT_TIME_MIN)) then
   mk_ModelUtils`Date(d.year, d.month + 1, 1, 00, (d.min + APPOINTMENT_TIME_MIN) mod ModelUtils`
       HOUR_MIN)
  elseif (d.hour = 23 and d.min >= (ModelUtils`HOUR_MIN - APPOINTMENT_TIME_MIN)) then
   mk_ModelUtils`Date(d.year, d.month, d.day+1, 00, (d.min + APPOINTMENT_TIME_MIN) mod ModelUtils
       `HOUR_MIN)
  elseif (d.min >= (ModelUtils`HOUR_MIN - APPOINTMENT_TIME_MIN)) then
   mk_ModelUtils`Date(d.year, d.month, d.day, d.hour + 1, (d.min + APPOINTMENT_TIME_MIN) mod
       ModelUtils`HOUR_MIN)
  else
```

```
   mk_ModelUtils`Date(d.year, d.month, d.day, d.hour, d.min + APPOINTMENT_TIME_MIN)
 );


end Appointment
```

Function or operation	Line	Coverage	Calls
Appointment	18	100.0%	15
appointmentDatesDontOverlap	55	100.0%	15
getDate	30	100.0%	73
getDoctorId	42	100.0%	114
getHospitalId	36	100.0%	23
getNextAppointmentDate	62	100.0%	38
getPatientId	48	100.0%	81
Appointment.vdmpp		100.0%	359

2 Doctor

```
class Doctor is subclass of Person
instance variables
 private specialty: ModelUtils`Specialty;
 private static currId : nat := 0;
 private id : nat := currId;

 inv age > 18;
operations
 --contructor

  public Doctor : ModelUtils`String * nat * ModelUtils`Specialty ==> Doctor
  Doctor(n, a, s) == (
   atomic (
    name := n;
    age := a;
    specialty := s;
    currId := currId + 1;
   );
   return self;
  );

  --get doctor id

  public pure getId: () ==> nat
  getId() == (
   return id
  );


  --get doctor specialties

  public pure getSpecialty: () ==> ModelUtils`Specialty
  getSpecialty() == (
   return specialty
  );

  -- getName
```

```
  public getName: () ==> ModelUtils`String
  getName() == (
    return retName();
  );

  -- getAge

  public getAge: () ==> nat
  getAge() == (
    return retAge();
  );

end Doctor
```

Function or operation	Line	Coverage	Calls
Doctor	10	100.0%	34
getAge	41	100.0%	4
getId	22	100.0%	247
getName	35	100.0%	2
getSpecialty	29	100.0%	30
Doctor.vdmpp		100.0%	317

3 Hospital

```
class Hospital
instance variables
 private name: ModelUtils`String := [];
 private location: ModelUtils`Location;
 private agreements: set of ModelUtils`Agreement := {};
 private doctorsIds: set of nat := {};
 private static currId : nat := 0;
 private id : nat := currId;

 inv forall d in set doctorsIds & d in set dom SafetyNetNetwork`getInstance().getDoctors();

operations

 --constructor

 public Hospital: ModelUtils`String * ModelUtils`Location * set of ModelUtils`Agreement ==>
     Hospital
 Hospital(n,l,a) == (
  atomic (
   name := n;
   location := l;
   agreements := a;
   currId := currId + 1;
  );
  return self
 );

 --get hospital name

 public pure getName: () ==> ModelUtils`String
```

```
getName() == (
 return name
);

--get hospital id

public pure getId: () ==> nat
getId() == (
 return id
);

--get hospital doctors

public pure getDoctorsIds: () ==> set of nat
getDoctorsIds() == (
 return doctorsIds
);

--get hospital location

public pure getLocation: () ==> ModelUtils`Location
getLocation() == (
 return location
);

--get hospital agreements

public pure getAgreements: () ==> set of ModelUtils`Agreement
getAgreements() == (
 return agreements
);

--add agreement

public addAgreement: ModelUtils`Agreement ==> ()
addAgreement(a) == (
 agreements := agreements union {a};
)
pre a not in set agreements
post a in set agreements;



--remove agreement
public removeAgreement: ModelUtils`Agreement ==> ()
removeAgreement(a) == (
 agreements := agreements \ {a};
)
pre a in set agreements
post a not in set agreements;


--add doctor
public addDoctor: nat ==> ()
addDoctor(doctorId) == (
 doctorsIds := doctorsIds union {doctorId}
)
pre doctorId not in set doctorsIds
post doctorId in set doctorsIds;



--remove doctor
public removeDoctor: nat ==> ()
removeDoctor(doctorId) == (
```

```
  doctorsIds := doctorsIds \ {doctorId}
 )
 pre doctorId in set doctorsIds
 post doctorId not in set doctorsIds;

end Hospital
```

Function or operation	Line	Coverage	Calls
Hospital	15	100.0%	41
addAgreement	57	100.0%	1
addDoctor	72	100.0%	27
getAgreements	51	100.0%	15
getDoctorsIds	39	100.0%	95
getId	33	100.0%	245
getLocation	45	100.0%	6
getName	27	100.0%	2409
removeAgreement	65	100.0%	1
removeDoctor	81	100.0%	3
Hospital.vdmpp		100.0%	2843

4 ModelUtils

```
class ModelUtils
values
 public HOUR_MIN = 60; -- minutes in an hour
 public DAY_HOURS = 24; -- hour in a day
 public MONTH_DAYS = 30; -- days in a month
 public YEAR_MONTHS = 12; -- months in a year
types
 public String = seq of char;
 public Location :: city: String
           address: String
           postalCode: String;
 public Date :: year : nat1
         month: nat1
         day : nat1
         hour : nat
         min : nat
 inv d == d.year > 2017 and d.month <= YEAR_MONTHS and d.day <= MONTH_DAYS and d.hour < DAY_HOURS
      and d.min < HOUR_MIN;
 public Date_DoctorId :: date : Date
              doctorId: nat;

 public Agreement = <ADSE> | <MEDICARE> | <MEDIS> | <MULTICARE>;
 public Specialty = <ORTOPEDIA> | <CARDIOLOGIA> | <OFTALMOLOGIA> |
           <DERMATOLOGIA> | <GINECOLOGIA> | <NEUROLOGIA> |
           <PEDIATRIA> | <REUMATOLOGIA> | <UROLOGIA> |
           <PNEUMOLOGIA>;

functions

 --checks if two dates are equal

 static public isDateLower: Date * Date -> bool
```

```
 isDateLower(d1,d2) == (
  dateToNat(d1) < dateToNat(d2)
 );

 --get max date

 static public getMaxDate: () -> Date
 getMaxDate() == (
  mk_ModelUtils`Date(99999,12,30,23,59)
 );

 --get min date

 static public getMinDate: () -> Date
 getMinDate() == (
  mk_ModelUtils`Date(2018,01,01,08,00)
 );

 --checks if two dates are equal

 static public dateToNat: Date -> nat
 dateToNat(d) == (
  d.year * 100000000 +
  d.month * 1000000 +
  d.day * 10000 +
  d.hour * 100 +
  d.min
 );

end ModelUtils
```

Function or operation	Line	Coverage	Calls
dateToNat	48	100.0%	152
getMaxDate	36	100.0%	1
getMinDate	42	100.0%	1
isDateLower	30	100.0%	8
ModelUtils.vdmpp		100.0%	162

5 Patient

```
class Patient is subclass of Person
instance variables
 private clinicalObservations: seq of ModelUtils`String := [];
 private static currId : nat := 0;
 private id : nat := currId;

operations

 public Patient : ModelUtils`String * nat * ModelUtils`String ==> Patient
 Patient(n, a, d) == (
  atomic (
   name := n;
   age:= a;
   clinicalObservations:= clinicalObservations ^ [d];
   currId := currId + 1;
  );
```

```
 return self;
);

--get patient id

public pure getId: () ==> nat
getId() == (
 return id
);

-- getName

 public getName: () ==> ModelUtils`String
 getName() == (
   return retName();
 );

 -- getAge

 public getAge: () ==> nat
 getAge() == (
   return retAge();
 );

--get clinical observations

public pure getClinicalObservations: () ==> seq of ModelUtils`String
getClinicalObservations() == (
 return clinicalObservations
);


--add clinical observation

public addObservation: ModelUtils`String ==> ()
addObservation(d) == (
 clinicalObservations := clinicalObservations ^ [d];
)
post len clinicalObservations = len clinicalObservations~ + 1 and
   exists i in set inds clinicalObservations & clinicalObservations(i) = d;


end Patient
```

Function or operation	Line	Coverage	Calls
Patient	8	100.0%	14
addObservation	45	100.0%	2
getAge	32	100.0%	4
getClinicalObservations	38	100.0%	4
getId	20	100.0%	102
getName	26	100.0%	4
Patient.vdmpp		100.0%	130

6 Person

```
class Person
instance variables
  protected name: ModelUtils`String := [];
  protected age: nat;

operations


  protected retName: () ==> ModelUtils`String
  retName() == (
    return name;
  );


  protected retAge: () ==> nat
  retAge() == (
    return age;
  );


end Person
```

Function or operation	Line	Coverage	Calls
retAge	13	100.0%	4
retName	8	100.0%	4
Person.vdmpp		100.0%	8

7 SafetyNetNetwork

```
class SafetyNetNetwork
instance variables
  private hospitals: inmap nat to Hospital := { |-> };
  private doctors: inmap nat to Doctor := { |-> };
  private patients: inmap nat to Patient := { |-> };
  private appointments: set of Appointment := {};
  private static networkInstance: SafetyNetNetwork := new SafetyNetNetwork();

  inv not exists h1, h2 in set rng hospitals &
  h1 <> h2 and h1.getName() = h2.getName();

operations

  --constructor
  private SafetyNetNetwork: () ==> SafetyNetNetwork
  SafetyNetNetwork() == return self
  post hospitals = { |-> } and  doctors = { |-> };

  --get network instance
  public pure static getInstance: () ==> SafetyNetNetwork
  getInstance() == return networkInstance
  post isofclass(SafetyNetNetwork,RESULT);


  --clear network instance
  public static clearInstance: () ==> ()
  clearInstance() == (
```

```
 networkInstance := new SafetyNetNetwork();
);

--associate doctor to hospital
public associateDoctorToHospital : nat * nat ==> ()
 associateDoctorToHospital(hospitalId, doctorId) == (
  hospitals(hospitalId).addDoctor(doctorId)
)
pre hospitalId in set dom hospitals and doctorId in set dom doctors and doctorId not in set
    hospitals(hospitalId).getDoctorsIds()
 post hospitalId in set dom hospitals and doctorId in set dom doctors and doctorId in set
     hospitals(hospitalId).getDoctorsIds();


--disassociate doctor from hospital
public disassociateDoctorFromHospital : nat * nat ==> ()
 disassociateDoctorFromHospital(hospitalId, doctorId) == (
  hospitals(hospitalId).removeDoctor(doctorId)
)
pre hospitalId in set dom hospitals and doctorId in set dom doctors and doctorId in set
    hospitals(hospitalId).getDoctorsIds()
post hospitalId in set dom hospitals and doctorId in set dom doctors and doctorId not in set
    hospitals(hospitalId).getDoctorsIds();

------------------------------------------------------------
------------------Hospital----------------------------------
------------------------------------------------------------
 -- get hospitals
public pure getHospitals : () ==> inmap nat to Hospital
getHospitals() == (
 return hospitals;
);


 --add hospital
public addHospital: Hospital ==> ()
addHospital(hospital) == (
 hospitals := hospitals ++  { hospital.getId() |-> hospital};
)
pre {hospital.getId() } <: hospitals = { |-> }
post {hospital.getId() } <: hospitals = { hospital.getId() |-> hospital } ;

--remove an hospital
public removeHospital: Hospital ==> ()
removeHospital(hospital) == (
 hospitals := {hospital.getId()} <-: hospitals;
 --cancel appointments in that hospital
 for all a in set appointments do
  if(a.getHospitalId() = hospital.getId()) then
   removeAppointment(a);
)
 pre {hospital.getId()} <: hospitals = { hospital.getId() |-> hospital }
post {hospital.getId()} <: hospitals = { |-> } and
   forall a in set appointments & a.getHospitalId() <> hospital.getId();

-- add agreement to hospital
public addAgreementToHospital: nat * ModelUtils`Agreement ==> ()
addAgreementToHospital(hospitalId, agreement) == (
 hospitals(hospitalId).addAgreement(agreement);
)
pre {hospitalId} <: hospitals = { hospitalId |-> hospitals(hospitalId) };


-- remove agreement from hospital
public removeAgreementFromHospital: nat * ModelUtils`Agreement ==> ()
```

```
removeAgreementFromHospital(hospitalId, agreement) == (
 hospitals(hospitalId).removeAgreement(agreement);
)
pre {hospitalId} <: hospitals = { hospitalId |-> hospitals(hospitalId) };

-------search hospitals-----------------------------------

----------get hospitals by id
public pure getHospitalsById: nat ==> Hospital
 getHospitalsById(hospitalId) == (
  return hospitals(hospitalId);
 )
pre hospitalId in set dom hospitals
post RESULT.getId() = hospitalId;

----------get hospitals by city

public pure getHospitalsByCity: ModelUtils`String ==> set of Hospital
 getHospitalsByCity(city) == (
  dcl res : set of Hospital := {};
  for all h in set rng hospitals do
    if(h.getLocation().city = city) then

     res := res union {h};
  return res
 );

----------get hospitals by name
public pure getHospitalsByName: ModelUtils`String ==> set of Hospital

 getHospitalsByName(name) == (
  dcl res : set of Hospital := {};
  for all h in set rng hospitals do
    if(h.getName() = name) then
     res := res union {h};
  return res

 );

----------get hospitals by agreement
public pure getHospitalsByAgreement: ModelUtils`Agreement ==> set of Hospital
 getHospitalsByAgreement(agreement) == (
  dcl res : set of Hospital := {};
  for all h in set rng hospitals do
    if(agreement in set h.getAgreements()) then
     res := res union {h};

  return res
 );

------------------End hospital search--------------------------------

-- get hospitals specialties
public pure getHospitalSpecialties: nat ==> set of ModelUtils`Specialty
 getHospitalSpecialties(hospitalId) == (
  dcl res : set of ModelUtils`Specialty := {};
   for all doctorId in set hospitals(hospitalId).getDoctorsIds() do
     res := res union {doctors(doctorId).getSpecialty()};

  return res
 );


-------------------------------------------------------------
-----------------------end hospital --------------------------
```

```
-------------------------------------------------------------



-------------------------------------------------------
----------------------Doctors --------------------------
-------------------------------------------------------
-- get doctors
public pure getDoctors : () ==> inmap nat to Doctor
getDoctors() == (

 return doctors;
);

--add doctor
public addDoctor: Doctor ==> ()
addDoctor(doctor) == (
 doctors := doctors ++  { doctor.getId() |-> doctor};
)
pre {doctor.getId() } <: doctors = { |-> }
post {doctor.getId() } <: doctors = { doctor.getId() |-> doctor };

--remove doctor from the network and from all the hospitals where he works
public removeDoctor: Doctor ==> ()

removeDoctor(doctor) == (
 --remove doctor from network
 doctors := {doctor.getId()} <-: doctors;
 --remove doctor from hospitals where he works
 for all h in set rng hospitals do
  if(doctor.getId() in set h.getDoctorsIds()) then
   h.removeDoctor(doctor.getId());
 --cancel doctor appointments

 for all a in set appointments do
  if(a.getDoctorId() = doctor.getId()) then
   removeAppointment(a);
)
 pre {doctor.getId()} <: doctors = { doctor.getId() |-> doctor }
post {doctor.getId()} <: doctors = { |-> } and
   forall h in set rng hospitals & doctor.getId() not in set h.getDoctorsIds() and
   forall a in set appointments & a.getDoctorId() <> doctor.getId();



--search doctors--------------------------------
----------get doctor by specialty
public pure getDoctorsBySpecialty: ModelUtils`Specialty ==> set of Doctor
 getDoctorsBySpecialty(s) == (
  dcl res : set of Doctor := {};
  for all d in set rng doctors do
    if(d.getSpecialty() = s) then

     res := res union {d};
  return res
 );

----------get doctor by id
public pure getDoctorById: nat ==> Doctor
 getDoctorById(doctorId) == (
  return doctors(doctorId);
 )
pre doctorId in set dom doctors

post RESULT.getId() = doctorId;
```

```
----------get hospitals where a doctor works
public pure getDoctorHospitals: nat ==> set of Hospital
 getDoctorHospitals(doctorId) == (
  dcl res : set of Hospital := {};
  for all h in set rng hospitals do
   if(doctorId in set h.getDoctorsIds()) then
    res := res union {h};
  return res

 );

------------------------------------------------------------
----------------------End Doctors ----------------------
------------------------------------------------------------


------------------------------------------------------------
------------------Patients--------------------------------
------------------------------------------------------------

--add patient

public addPatient: Patient ==> ()
addPatient(patient) == (
 patients := patients ++  { patient.getId() |-> patient};
)
pre {patient.getId() } <: patients = { |-> }
post {patient.getId() } <: patients = { patient.getId() |-> patient };

--remove patient
public removePatient: Patient ==> ()
removePatient(patient) == (
 --remove patient appointments
 for all a in set appointments do
  if(a.getPatientId() = patient.getId()) then
   removeAppointment(a);
 --remove patient from network
 patients := {patient.getId()} <-: patients;
)
 pre {patient.getId()} <: patients = { patient.getId() |-> patient }

post {patient.getId()} <: patients = { |-> } and forall a in set appointments & a.getPatientId
    () <> patient.getId();

--add clinical observation
public addClinicalObservation: nat * ModelUtils`String ==> ()
addClinicalObservation(patientId, obs) == (
 patients(patientId).addObservation(obs);

)
 pre {patientId} <: patients = { patientId |-> patients(patientId) } ;

 -- get patients
public pure getPatients : () ==> inmap nat to Patient
getPatients() == (
 return patients;
);



------------------------------------------------------------
------------------End Patients----------------------------
------------------------------------------------------------
```

```
-------------------------------------------------------
---------------------Appointments----------------------
-------------------------------------------------------

-- get appointments
public pure getAppointments : () ==> set of Appointment
getAppointments() == (
 return appointments;
);

--get hospital appointments
public pure getHospitalAppointments: nat ==> set of Appointment
getHospitalAppointments(hospitalId) == (
 dcl res: set of Appointment := {};
 for all a in set appointments do

  if(a.getHospitalId() = hospitalId) then
    res := res union {a};

 return res
)
pre {hospitalId} <: hospitals = { hospitalId |-> hospitals(hospitalId) }
post forall a in set RESULT & isofclass(Appointment,a) and a.getHospitalId() = hospitalId;

-- total number of appointments in a hospital
public pure getHospitalNumberOfAppointments: nat ==> nat

 getHospitalNumberOfAppointments(hospitalId) == (
  return card getHospitalAppointments(hospitalId);
)
pre hospitalId in set dom hospitals;



--get doctor appointments

public pure getDoctorAppointments: nat ==> set of Appointment
getDoctorAppointments(doctorId) == (
 dcl res: set of Appointment := {};
 for all a in set appointments do
  if(a.getDoctorId() = doctorId) then
    res := res union {a};
 return res
)
pre {doctorId} <: doctors = { doctorId |-> doctors(doctorId) }
post forall a in set RESULT & isofclass(Appointment,a) and a.getDoctorId() = doctorId;

--get patient appointments
public pure getPatientAppointments: nat ==> set of Appointment
getPatientAppointments(patientId) == (
 dcl res: set of Appointment := {};
 for all a in set appointments do
  if(a.getPatientId() = patientId) then
    res := res union {a};
 return res

)
pre {patientId} <: patients = { patientId |-> patients(patientId) }
post forall a in set RESULT & isofclass(Appointment,a) and a.getPatientId() = patientId;

--get specialty appointments
public pure getSpecialtyAppointments: ModelUtils`Specialty ==> set of Appointment
getSpecialtyAppointments(specialty) == (
 dcl res: set of Appointment := {};
```

```
 for all a in set appointments do
  if(doctors(a.getDoctorId()).getSpecialty() = specialty) then
    res := res union {a};
 return res
)
post forall a in set RESULT & isofclass(Appointment,a) and doctors(a.getDoctorId()).
    getSpecialty() = specialty;


--add an Appointment
public addAppointment: Appointment ==> ()
addAppointment(a) == (
 appointments := appointments union {a};
)

pre forall ap in set getDoctorAppointments(a.getDoctorId()) union getPatientAppointments(a.
    getPatientId()) & Appointment`appointmentDatesDontOverlap(ap.getDate(), a.getDate())
post a in set appointments;

--remove an Appointment
public removeAppointment: Appointment ==> ()
removeAppointment(a) == (
 appointments := appointments \ {a};

)
pre a in set appointments
post a not in set appointments;



-- get closest appointment date available given a set of appointments
-- example: pass the appointments related to a specialty and it will return the closest
    appointment date available for that specialty as well as the doctor
public getHospitalClosestAvailableDate: set of Appointment ==> ModelUtils`Date_DoctorId
 getHospitalClosestAvailableDate(appointmentSet) == (
 dcl minDate: ModelUtils`Date := ModelUtils`getMaxDate();
 dcl doctorId: nat;
 dcl availableDoctors: set of nat := {};
 dcl occupiedDates: set of ModelUtils`Date := {};


 for all ap in set appointmentSet do
 (
  availableDoctors := availableDoctors union {ap.getDoctorId()};
  occupiedDates := occupiedDates union {ap.getDate()};
 );


 -- have to check if the first date is available
 occupiedDates := occupiedDates union {ModelUtils`getMinDate()};

 for all date in set occupiedDates do
 (
  dcl auxDate : ModelUtils`Date := Appointment`getNextAppointmentDate(date);
  if(ModelUtils`isDateLower(auxDate, minDate)) then
   for all docId in set availableDoctors do
     if(forall docAp in set getDoctorAppointments(docId) &  Appointment`
         appointmentDatesDontOverlap(docAp.getDate(),auxDate)) then -- if the next appointment
         is on a closer date than the actual minimum and one of the doctor has the date slot
         available, update the minimum
     (
      doctorId := docId;
      minDate := auxDate;
     )
```

```
    );

    return mk_ModelUtils`Date_DoctorId(minDate, doctorId);
  )
  pre forall ap in set appointmentSet & ap in set appointments
  post RESULT.doctorId in set dom doctors;

  ------------------------------------------------------------
  ----------------------End Appointments--------------------

  ------------------------------------------------------------


end SafetyNetNetwork
```

Function or operation	Line	Coverage	Calls
SafetyNetNetwork	105	100.0%	26
addAgreementToHospital	170	100.0%	1
addAppointment	424	100.0%	19
addClinicalObservation	338	100.0%	2
addDoctor	251	100.0%	35
addHospital	149	100.0%	41
addPatient	317	100.0%	15
associateDoctorToHospital	122	100.0%	27
clearInstance	116	100.0%	25
disassociateDoctorFromHospital	131	100.0%	1
getAppointments	360	100.0%	10
getDoctorAppointments	388	100.0%	35
getDoctorById	290	100.0%	3
getDoctorHospitals	298	100.0%	3
getDoctors	245	100.0%	59
getDoctorsBySpecialty	280	100.0%	3
getHospitalAppointments	366	100.0%	5
getHospitalClosestAvailableDate	443	100.0%	2
getHospitalNumberOfAppointments	379	100.0%	8
getHospitalSpecialties	227	100.0%	2
getHospitals	142	100.0%	49
getHospitalsByAgreement	215	100.0%	12
getHospitalsByCity	195	100.0%	2
getHospitalsById	187	100.0%	3
getHospitalsByName	205	100.0%	3
getInstance	110	100.0%	240
getPatientAppointments	400	100.0%	18
getPatients	345	100.0%	17
getSpecialtyAppointments	412	100.0%	3
removeAgreementFromHospital	178	100.0%	1
removeAppointment	432	100.0%	8
removeDoctor	259	100.0%	2
removeHospital	157	100.0%	2
removePatient	325	100.0%	2

SafetyNetNetwork.vdmpp		100.0%	684

8 MyTestCase

```
class MyTestCase

operations



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


 protected assertFalse: bool ==> ()
 assertFalse(arg) ==
  return
 pre not arg;


 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	16	38.8%	0
assertFalse	11	100.0%	3
assertNotEqual	27	0.0%	0
assertTrue	6	100.0%	22
MyTestCase.vdmpp		52.1%	25

9 SystemTest

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

```
  private safetyNet: SafetyNetNetwork := SafetyNetNetwork`getInstance();

operations



public static main: () ==> ()
main() == (

  dcl systemTest: SystemTest := new SystemTest();

  IO`println("network ");

  -- association hospital - doctor
  IO`print("test associateADoctorToAnHospital -> ");
  systemTest.testAssociateDoctorToAnHospital();
  IO`println("Success");

  IO`print("test disassociateADoctorToAnHospital -> ");
  systemTest.testDisassociateDoctorToAnHospital();
  IO`println("Success");


  -- Hospital
  IO`print("test addHospital -> ");
  systemTest.testAddHospital();
  IO`println("Success");

  IO`print("test removeHospital -> ");
  systemTest.testRemoveHospital();
  IO`println("Success");

  IO`print("test getAllHospitalsByLocation -> ");
  systemTest.testGetHospitalsByLocation();
  IO`println("Success");

  IO`print("test getAllHospitals -> ");
  systemTest.testGetAllHospitals();
  IO`println("Success");

  IO`print("test getHospitalsByName -> ");
  systemTest.testGetHospitalsByName();
  IO`println("Success");

  IO`print("test getHospitalsById -> ");
  systemTest.testGetHospitalsById();
  IO`println("Success");

  IO`print("test getHospitalsByAgreement -> ");
  systemTest.testGetHospitalsByAgreement();
  IO`println("Success");

  IO`print("test getHospitalSpecialties -> ");
  systemTest.testGetHospitalSpecialties();
  IO`println("Success");

  -- Doctor
  IO`print("test addDoctor -> ");
  systemTest.testAddDoctor();
  IO`println("Success");

  IO`print("test getDoctors -> ");
  systemTest.testGetAllDoctors();
  IO`println("Success");
```

```
  IO`print("test removeDoctor -> ");
  systemTest.testRemoveDoctor();
  IO`println("Success");

  IO`print("test getDoctorHospitals -> ");
  systemTest.testGetDoctorHospitals();
  IO`println("Success");

  IO`print("test getDoctorBySpecialtie-> ");
  systemTest.testGetDoctorBySpecialtie();
  IO`println("Success");

  IO`print("test getDoctorById -> ");
  systemTest.testGetDoctorById();
  IO`println("Success");

  -- Patient
  IO`print("test addPatient -> ");
  systemTest.testAddPatient();
  IO`println("Success");

  IO`print("test removePatient -> ");
  systemTest.testRemovePatient();
  IO`println("Success");

  IO`print("test addObservation -> ");
  systemTest.testAddObservation();
  IO`println("Success");


  -- Appointement

  IO`print("test addAppointment -> ");
  systemTest.testAddAppointment();
  IO`println("Success");

  IO`print("test removeAppointment -> ");
  systemTest.testRemoveAppointment();
  IO`println("Success");

  IO`print("test getSpecialtyAppointments -> ");
  systemTest.testGetSpecialtyAppointments();
  IO`println("Success");

  IO`print("test getHospitalClosestAvailableDate -> ");

  systemTest.testGetHospitalClosestAvailableDate();
  IO`println("Success");

  IO`print("test getNextAppointmentDate -> ");
  systemTest.testGetNextAppointmentDate();
  IO`println("Success");

  -- Agreement

  IO`print("test addAgreement-> ");
  systemTest.testAddAgreement();
  IO`println("Success");

  IO`print("test removeAgreement -> ");
  systemTest.testRemoveAgreement();
  IO`println("Success");

);
```

```
private testAddDoctor: () ==> ()
 testAddDoctor () == (

  dcl doc1: Doctor := new Doctor("jose",35,<ORTOPEDIA>);
  dcl doc2: Doctor := new Doctor("marcelo", 40 ,<CARDIOLOGIA>);


  safetyNet := SafetyNetNetwork`getInstance();

  safetyNet.addDoctor(doc1);
  safetyNet.addDoctor(doc2);

  assertEqual(doc1.getName(), "jose");
  assertEqual(doc2.getName(), "marcelo");

  assertEqual(doc1.getAge(), 35);
  assertEqual(doc2.getAge(), 40);

  assertEqual(doc1.getSpecialty(), <ORTOPEDIA>);
  assertEqual(doc2.getSpecialty(), <CARDIOLOGIA>);

  assertEqual( safetyNet.getDoctors(), {doc1.getId() |-> doc1, doc2.getId() |->doc2});

  safetyNet.clearInstance();

);


private testGetAllDoctors: () ==> ()
 testGetAllDoctors () == (

  dcl doc1: Doctor := new Doctor("jose",35,<ORTOPEDIA>);
  dcl doc2: Doctor := new Doctor("marcelo", 40 ,<CARDIOLOGIA>);

  safetyNet := SafetyNetNetwork`getInstance();

  assertEqual(rng safetyNet.getDoctors(), {});

  safetyNet.addDoctor(doc1);
  safetyNet.addDoctor(doc2);

  assertEqual( safetyNet.getDoctors(), {doc1.getId() |-> doc1, doc2.getId() |->doc2});

  safetyNet.clearInstance();

);

private testRemoveDoctor: () ==> ()
 testRemoveDoctor () == (

  dcl doc1: Doctor := new Doctor("jose",35,<ORTOPEDIA>);
  dcl doc2: Doctor := new Doctor("marcelo", 40 ,<CARDIOLOGIA>);

  dcl hos1: Hospital := new Hospital("Hospital Sao Joao",mk_ModelUtils`Location("Porto", "Alameda
       Prof. Hernni Monteiro","4200-319"), {<ADSE>,<MEDICARE>});
  dcl hos2: Hospital := new Hospital("Hospital da Luz Lisboa",mk_ModelUtils`Location("Lisboa", "
      Avenida Lusada, n  100","4700-959"), {<ADSE>,<MEDIS>, <MULTICARE>});

  safetyNet := SafetyNetNetwork`getInstance();

  safetyNet.addHospital(hos1);
  safetyNet.addHospital(hos2);
```

```
  safetyNet.addDoctor(doc1);
  safetyNet.addDoctor(doc2);

  safetyNet.associateDoctorToHospital(hos1.getId(), doc1.getId());
  safetyNet.associateDoctorToHospital(hos2.getId(), doc2.getId());

  safetyNet.removeDoctor(doc2);

  -- check if was removed from the system
  assertEqual( safetyNet.getDoctors(), {doc1.getId() |-> doc1});

  -- check if the doctor was removed from all hospitals where he worked
  for all hs in set rng safetyNet.getHospitals() do
   assertFalse(doc2.getId() in set hs.getDoctorsIds());

  safetyNet.clearInstance();


);

private testAddHospital: () ==> ()
 testAddHospital () == (

  dcl hos1: Hospital := new Hospital("Hospital Sao Joao",mk_ModelUtils`Location("Porto", "Alameda
       Prof. Hernni Monteiro","4200-319"), {<ADSE>,<MEDICARE>});
  dcl hos2: Hospital := new Hospital("Hospital da Luz Lisboa",mk_ModelUtils`Location("Lisboa", "
      Avenida Lusada, n  100","4700-959"), {<ADSE>,<MEDIS>, <MULTICARE>});

  safetyNet := SafetyNetNetwork`getInstance();

  safetyNet.addHospital(hos1);
  safetyNet.addHospital(hos2);

  assertEqual(safetyNet.getHospitals(), { hos1.getId() |-> hos1, hos2.getId() |-> hos2});

  safetyNet.clearInstance();

);


private testRemoveHospital: () ==> ()
 testRemoveHospital () == (

  dcl hos1: Hospital := new Hospital("Hospital Sao Joao",mk_ModelUtils`Location("Porto", "Alameda
       Prof. Hernni Monteiro","4200-319"), {<ADSE>,<MEDICARE>});
  dcl hos2: Hospital := new Hospital("Hospital da Luz Lisboa",mk_ModelUtils`Location("Lisboa", "
      Avenida Lusada, n  100","4700-959"), {<ADSE>,<MEDIS>, <MULTICARE>});

  safetyNet := SafetyNetNetwork`getInstance();

  safetyNet.addHospital(hos1);
  safetyNet.addHospital(hos2);

  safetyNet.removeHospital(hos1);

  assertEqual(safetyNet.getHospitals(), { hos2.getId() |-> hos2});

  safetyNet.clearInstance();

);

-- change test
private testAssociateDoctorToAnHospital: () ==> ()
 testAssociateDoctorToAnHospital () == (
```

```
  dcl hos1: Hospital := new Hospital("Hospital Sao Joao",mk_ModelUtils`Location("Porto", "Alameda
       Prof. Hernni Monteiro","4200-319"), {<ADSE>,<MEDICARE>});
  dcl hos2: Hospital := new Hospital("Hospital da Luz Lisboa",mk_ModelUtils`Location("Lisboa", "
      Avenida Lusada, n  100","4700-959"), {<ADSE>,<MEDIS>, <MULTICARE>});

  dcl doc1: Doctor := new Doctor("jose",35,<ORTOPEDIA>);
  dcl doc2: Doctor := new Doctor("marcelo", 40 ,<CARDIOLOGIA>);

  safetyNet := SafetyNetNetwork`getInstance();


  safetyNet.addHospital(hos1);
  safetyNet.addHospital(hos2);

  safetyNet.addDoctor(doc1);
  safetyNet.addDoctor(doc2);

  safetyNet.associateDoctorToHospital(hos1.getId(), doc1.getId());
  safetyNet.associateDoctorToHospital(hos2.getId(), doc1.getId());
  safetyNet.associateDoctorToHospital(hos2.getId(), doc2.getId());

  assertEqual(hos1.getDoctorsIds(), {doc1.getId()});
  assertEqual(hos2.getDoctorsIds(), {doc1.getId(),doc2.getId()});

  assertEqual( safetyNet.getDoctors(), {doc1.getId() |-> doc1, doc2.getId() |->doc2});

  safetyNet.clearInstance();

);
-- change test
private testDisassociateDoctorToAnHospital: () ==> ()
 testDisassociateDoctorToAnHospital () == (

  dcl hos1: Hospital := new Hospital("Hospital Sao Joao",mk_ModelUtils`Location("Porto", "Alameda
       Prof. Hernni Monteiro","4200-319"), {<ADSE>,<MEDICARE>});
  dcl hos2: Hospital := new Hospital("Hospital da Luz Lisboa",mk_ModelUtils`Location("Lisboa", "
      Avenida Lusada, n  100","4700-959"), {<ADSE>,<MEDIS>, <MULTICARE>});

  dcl doc1: Doctor := new Doctor("jose",35,<ORTOPEDIA>);
  dcl doc2: Doctor := new Doctor("marcelo", 40 ,<CARDIOLOGIA>);

  safetyNet := SafetyNetNetwork`getInstance();

  safetyNet.addHospital(hos1);
  safetyNet.addHospital(hos2);


  safetyNet.addDoctor(doc1);
  safetyNet.addDoctor(doc2);

  safetyNet.associateDoctorToHospital(hos1.getId(), doc1.getId());
  safetyNet.associateDoctorToHospital(hos2.getId(), doc1.getId());
  safetyNet.associateDoctorToHospital(hos2.getId(), doc2.getId());


  safetyNet.disassociateDoctorFromHospital(hos2.getId(), doc1.getId());

  assertEqual(hos1.getDoctorsIds(), {doc1.getId()});
  assertEqual(hos2.getDoctorsIds(), {doc2.getId()});

  assertEqual( safetyNet.getDoctors(), {doc1.getId() |-> doc1, doc2.getId() |->doc2});

  safetyNet.clearInstance();

);
```

```
private testGetHospitalsByLocation: () ==> ()
 testGetHospitalsByLocation () == (

  dcl hos1: Hospital := new Hospital("Hospital Sao Joao",mk_ModelUtils`Location("Porto", "Alameda
       Prof. Hernni Monteiro","4200-319"), {<ADSE>,<MEDICARE>});
  dcl hos2: Hospital := new Hospital("Hospital da Luz Lisboa",mk_ModelUtils`Location("Lisboa", "
      Avenida Lusada, n  100","4700-959"), {<ADSE>,<MEDIS>, <MULTICARE>});
  dcl hos3: Hospital := new Hospital("Hospital de Santo Antonio",mk_ModelUtils`Location("Lisboa",
       "Avenida de Santo antonio, n   300","4750-559"), {<ADSE>});

  safetyNet := SafetyNetNetwork`getInstance();

  safetyNet.addHospital(hos1);
  safetyNet.addHospital(hos2);
  safetyNet.addHospital(hos3);

  assertEqual(safetyNet.getHospitalsByCity("Porto"), {hos1});
  assertEqual(safetyNet.getHospitalsByCity("Lisboa"), {hos2, hos3});

  safetyNet.clearInstance();
);



private testGetHospitalsByAgreement: () ==> ()
 testGetHospitalsByAgreement () == (

  dcl hos1: Hospital := new Hospital("Hospital Sao Joao",mk_ModelUtils`Location("Porto", "Alameda
       Prof. Hernni Monteiro","4200-319"), {<ADSE>,<MEDICARE>});
  dcl hos2: Hospital := new Hospital("Hospital da Luz Lisboa",mk_ModelUtils`Location("Lisboa", "
      Avenida Lusada, n  100","4700-959"), {<ADSE>,<MEDIS>, <MULTICARE>});
  dcl hos3: Hospital := new Hospital("Hospital de Santo Antonio",mk_ModelUtils`Location("Lisboa",
       "Avenida de Santo antonio, n   300","4750-559"), {<ADSE>});

  safetyNet := SafetyNetNetwork`getInstance();

  safetyNet.addHospital(hos1);
  safetyNet.addHospital(hos2);
  safetyNet.addHospital(hos3);

  assertEqual(safetyNet.getHospitalsByAgreement(<ADSE>), {hos1, hos2, hos3});
  assertEqual(safetyNet.getHospitalsByAgreement(<MEDIS>), {hos2});
  assertEqual(safetyNet.getHospitalsByAgreement(<MULTICARE>), {hos2});
  assertEqual(safetyNet.getHospitalsByAgreement(<MEDICARE>), {hos1});


  safetyNet.clearInstance();
);

private testGetHospitalsById: () ==> ()
 testGetHospitalsById () == (

  dcl hos1: Hospital := new Hospital("Hospital Sao Joao",mk_ModelUtils`Location("Porto", "Alameda
       Prof. Hernni Monteiro","4200-319"), {<ADSE>,<MEDICARE>});
  dcl hos2: Hospital := new Hospital("Hospital da Luz Lisboa",mk_ModelUtils`Location("Lisboa", "
      Avenida Lusada, n  100","4700-959"), {<ADSE>,<MEDIS>, <MULTICARE>});
  dcl hos3: Hospital := new Hospital("Hospital de Santo Antonio",mk_ModelUtils`Location("Lisboa",
       "Avenida de Santo antonio, n   300","4750-559"), {<ADSE>});

  safetyNet := SafetyNetNetwork`getInstance();

  safetyNet.addHospital(hos1);
  safetyNet.addHospital(hos2);
```

```
  safetyNet.addHospital(hos3);

  assertEqual(safetyNet.getHospitalsById(hos1.getId()), hos1);
  assertEqual(safetyNet.getHospitalsById(hos2.getId()), hos2);
  assertEqual(safetyNet.getHospitalsById(hos3.getId()), hos3);


  safetyNet.clearInstance();
);

private testGetAllHospitals: () ==> ()
 testGetAllHospitals () == (

  dcl hos1: Hospital := new Hospital("Hospital Sao Joao",mk_ModelUtils`Location("Porto", "Alameda
       Prof. Hernni Monteiro","4200-319"), {<ADSE>,<MEDICARE>});
  dcl hos2: Hospital := new Hospital("Hospital da Luz Lisboa",mk_ModelUtils`Location("Lisboa", "
      Avenida Lusada, n  100","4700-959"), {<ADSE>,<MEDIS>, <MULTICARE>});
  dcl hos3: Hospital := new Hospital("Hospital de Santo Antonio",mk_ModelUtils`Location("Lisboa",
       "Avenida de Santo antonio, n   300","4750-559"), {<ADSE>});

  safetyNet := SafetyNetNetwork`getInstance();

  safetyNet.addHospital(hos1);
  safetyNet.addHospital(hos2);
  safetyNet.addHospital(hos3);

  assertEqual(safetyNet.getHospitals(), {hos1.getId() |-> hos1, hos2.getId() |-> hos2, hos3.getId
      () |->hos3});

  safetyNet.clearInstance();
);

private testGetHospitalsByName: () ==> ()
 testGetHospitalsByName () == (

  dcl hos1: Hospital := new Hospital("Hospital Sao Joao",mk_ModelUtils`Location("Porto", "Alameda
       Prof. Hernni Monteiro","4200-319"), {<ADSE>,<MEDICARE>});
  dcl hos2: Hospital := new Hospital("Hospital da Luz Lisboa",mk_ModelUtils`Location("Lisboa", "
      Avenida Lusada, n  100","4700-959"), {<ADSE>,<MEDIS>, <MULTICARE>});
  dcl hos3: Hospital := new Hospital("Hospital de Santo Antonio",mk_ModelUtils`Location("Lisboa",
       "Avenida de Santo antonio, n   300","4750-559"), {<ADSE>});

  safetyNet := SafetyNetNetwork`getInstance();

  safetyNet.addHospital(hos1);

  safetyNet.addHospital(hos2);
  safetyNet.addHospital(hos3);

  assertEqual(safetyNet.getHospitalsByName("Hospital Sao Joao"), {hos1});
  assertEqual(safetyNet.getHospitalsByName("Hospital da Luz Lisboa"), {hos2});
  assertEqual(safetyNet.getHospitalsByName("Hospital de Santo Antonio"), {hos3});

  safetyNet.clearInstance();
);

private testGetDoctorHospitals: () ==> ()
 testGetDoctorHospitals () == (

  dcl hos1: Hospital := new Hospital("Hospital Sao Joao",mk_ModelUtils`Location("Porto", "Alameda
       Prof. Hernni Monteiro","4200-319"), {<ADSE>,<MEDICARE>});
  dcl hos2: Hospital := new Hospital("Hospital da Luz Lisboa",mk_ModelUtils`Location("Lisboa", "
      Avenida Lusada, n  100","4700-959"), {<ADSE>,<MEDIS>, <MULTICARE>});

  dcl doc1: Doctor := new Doctor("jose",35,<ORTOPEDIA>);
```

```
  dcl doc2: Doctor := new Doctor("marcelo", 40 ,<CARDIOLOGIA>);
  dcl doc3: Doctor := new Doctor("joaquim",50,<CARDIOLOGIA>);

  safetyNet := SafetyNetNetwork`getInstance();


  safetyNet.addHospital(hos1);
  safetyNet.addHospital(hos2);

  safetyNet.addDoctor(doc1);
  safetyNet.addDoctor(doc2);
  safetyNet.addDoctor(doc3);

  safetyNet.associateDoctorToHospital(hos1.getId(), doc1.getId());
  safetyNet.associateDoctorToHospital(hos2.getId(), doc1.getId());
  safetyNet.associateDoctorToHospital(hos2.getId(), doc2.getId());

  assertEqual(safetyNet.getDoctorHospitals(doc1.getId()), {hos1, hos2});
  assertEqual(safetyNet.getDoctorHospitals(doc2.getId()), {hos2});
  assertEqual(safetyNet.getDoctorHospitals(doc3.getId()), {});

  safetyNet.clearInstance();

);

private testGetDoctorBySpecialtie: () ==> ()
 testGetDoctorBySpecialtie () == (

  dcl doc1: Doctor := new Doctor("jose",35,<ORTOPEDIA>);
  dcl doc2: Doctor := new Doctor("marcelo", 40 ,<CARDIOLOGIA>);
  dcl doc3: Doctor := new Doctor("joaquim",50,<CARDIOLOGIA>);

  safetyNet := SafetyNetNetwork`getInstance();

  safetyNet.addDoctor(doc1);

  safetyNet.addDoctor(doc2);
  safetyNet.addDoctor(doc3);

  assertEqual(safetyNet.getDoctorsBySpecialty(<ORTOPEDIA>), {doc1});
  assertEqual(safetyNet.getDoctorsBySpecialty(<OFTALMOLOGIA>), {});
  assertEqual(safetyNet.getDoctorsBySpecialty(<CARDIOLOGIA>), {doc2, doc3});

  safetyNet.clearInstance();

);

private testGetHospitalSpecialties: () ==> ()
 testGetHospitalSpecialties () == (

  dcl hos1: Hospital := new Hospital("Hospital Sao Joao",mk_ModelUtils`Location("Porto", "Alameda
       Prof. Hernni Monteiro","4200-319"), {<ADSE>,<MEDICARE>});
  dcl hos2: Hospital := new Hospital("Hospital da Luz Lisboa",mk_ModelUtils`Location("Lisboa", "
      Avenida Lusada, n  100","4700-959"), {<ADSE>,<MEDIS>, <MULTICARE>});

  dcl doc1: Doctor := new Doctor("jose",35,<ORTOPEDIA>);
  dcl doc2: Doctor := new Doctor("marcelo", 40 ,<CARDIOLOGIA>);
  dcl doc3: Doctor := new Doctor("joaquim",50,<CARDIOLOGIA>);


  safetyNet := SafetyNetNetwork`getInstance();

  safetyNet.addHospital(hos1);
  safetyNet.addHospital(hos2);
```

```
  safetyNet.addDoctor(doc1);
  safetyNet.addDoctor(doc2);
  safetyNet.addDoctor(doc3);

  safetyNet.associateDoctorToHospital(hos1.getId(), doc1.getId());
  safetyNet.associateDoctorToHospital(hos2.getId(), doc1.getId());
  safetyNet.associateDoctorToHospital(hos2.getId(), doc2.getId());

  assertEqual(safetyNet.getHospitalSpecialties(hos1.getId()), {<ORTOPEDIA>});
  assertEqual(safetyNet.getHospitalSpecialties(hos2.getId()), {<ORTOPEDIA>, <CARDIOLOGIA>});

  safetyNet.clearInstance();

);

private testGetDoctorById: () ==> ()
 testGetDoctorById () == (


  dcl doc1: Doctor := new Doctor("jose",35,<ORTOPEDIA>);
  dcl doc2: Doctor := new Doctor("marcelo", 40 ,<CARDIOLOGIA>);
  dcl doc3: Doctor := new Doctor("joaquim",50,<CARDIOLOGIA>);

  safetyNet := SafetyNetNetwork`getInstance();

  safetyNet.addDoctor(doc1);
  safetyNet.addDoctor(doc2);
  safetyNet.addDoctor(doc3);

  assertEqual(safetyNet.getDoctorById(doc1.getId()), doc1);
  assertEqual(safetyNet.getDoctorById(doc2.getId()), doc2);
  assertEqual(safetyNet.getDoctorById(doc3.getId()), doc3);

  safetyNet.clearInstance();
);

private testAddPatient: () ==> ()

 testAddPatient () == (

  dcl pat1: Patient := new Patient("Susana", 26, "Gripe");
  dcl pat2: Patient := new Patient("Maria", 38, "Doen a pulmonar");

  safetyNet := SafetyNetNetwork`getInstance();

  safetyNet.addPatient(pat1);
  safetyNet.addPatient(pat2);

  assertEqual(pat1.getName(), "Susana");
  assertEqual(pat2.getName(), "Maria");

  assertEqual(pat1.getAge(), 26);
  assertEqual(pat2.getAge(), 38);

  assertEqual( safetyNet.getPatients(), {pat1.getId() |-> pat1, pat2.getId() |-> pat2});

  safetyNet.clearInstance();

);

private testRemovePatient: () ==> ()
 testRemovePatient () == (

  dcl pat1: Patient := new Patient("Susana", 26, "Gripe");
  dcl pat2: Patient := new Patient("Maria", 38, "Doen a pulmonar");
```

```
  safetyNet := SafetyNetNetwork`getInstance();

  safetyNet.addPatient(pat1);
  safetyNet.addPatient(pat2);

  safetyNet.removePatient(pat2);

  assertEqual( safetyNet.getPatients(), {pat1.getId() |-> pat1});

  safetyNet.clearInstance();

);

private testAddAppointment: () ==> ()
 testAddAppointment () == (

  dcl hos1: Hospital := new Hospital("Hospital Sao Joao",mk_ModelUtils`Location("Porto", "Alameda
       Prof. Hernni Monteiro","4200-319"), {<ADSE>,<MEDICARE>});
  dcl hos2: Hospital := new Hospital("Hospital da Luz Lisboa",mk_ModelUtils`Location("Lisboa", "
      Avenida Lusada, n  100","4700-959"), {<ADSE>,<MEDIS>, <MULTICARE>});

  dcl doc1: Doctor := new Doctor("jose",35,<ORTOPEDIA>);
  dcl doc2: Doctor := new Doctor("marcelo", 40 ,<CARDIOLOGIA>);
  dcl doc3: Doctor := new Doctor("joaquim",50,<CARDIOLOGIA>);

  dcl pat1: Patient := new Patient("Susana", 26, "Gripe");
  dcl pat2: Patient := new Patient("Maria", 38, "Doena pulmonar");

  dcl dt1: set of ModelUtils`Date := {};
  dcl dt2: set of ModelUtils`Date := {};

  dcl pt1: set of ModelUtils`Date := {};
  dcl pt2: set of ModelUtils`Date := {};

  safetyNet := SafetyNetNetwork`getInstance();

  safetyNet.addHospital(hos1);
  safetyNet.addHospital(hos2);

  safetyNet.addDoctor(doc1);
  safetyNet.addDoctor(doc2);
  safetyNet.addDoctor(doc3);

  safetyNet.addPatient(pat1);
  safetyNet.addPatient(pat2);

  safetyNet.associateDoctorToHospital(hos1.getId(), doc1.getId());
  safetyNet.associateDoctorToHospital(hos2.getId(), doc1.getId());
  safetyNet.associateDoctorToHospital(hos2.getId(), doc2.getId());

  assertEqual(card safetyNet.getDoctorAppointments(doc1.getId()), 0);
  assertEqual(card safetyNet.getDoctorAppointments(doc2.getId()), 0);

  safetyNet.addAppointment(new Appointment(mk_ModelUtils`Date(2018,12,21,8,30), hos1.getId(),
      doc1.getId(), pat1.getId()));
  safetyNet.addAppointment(new Appointment(mk_ModelUtils`Date(2018,12,21,8,30), hos2.getId(),
      doc2.getId(), pat2.getId()));

  -- doctor appointments
  assertEqual(card safetyNet.getDoctorAppointments(doc1.getId()), 1);
  for all a in set safetyNet.getDoctorAppointments(doc1.getId()) do (
   dt1 := dt1 union {a.getDate()};
   assertEqual(a.getDoctorId(), doc1.getId());
```

```
    assertTrue(a.getDate().year > 2017 and a.getDate().month <= 12 and a.getDate().day < 31 and a.
        getDate().hour < 24 and a.getDate().min < 60);
  );
  assertEqual(card dt1, card safetyNet.getDoctorAppointments(doc1.getId()));

  assertEqual(card safetyNet.getDoctorAppointments(doc2.getId()), 1);
  for all a in set safetyNet.getDoctorAppointments(doc2.getId()) do (
   dt2 := dt2 union {a.getDate()};
   assertEqual(a.getDoctorId(), doc2.getId());
   assertTrue(a.getDate().year > 2017 and a.getDate().month <= 12 and a.getDate().day < 31 and a.
        getDate().hour < 24 and a.getDate().min < 60);
  );
  assertEqual(card dt2, card safetyNet.getDoctorAppointments(doc2.getId()));


  assertEqual(card safetyNet.getDoctorAppointments(doc3.getId()), 0);


  -- patient appointments
  assertEqual(card safetyNet.getPatientAppointments(pat1.getId()), 1);
  for all a in set safetyNet.getPatientAppointments(pat1.getId()) do (
   pt1 := pt1 union {a.getDate()};

   assertEqual(a.getPatientId(), pat1.getId());
   assertTrue(a.getDate().year > 2017 and a.getDate().month <= 12 and a.getDate().day < 31 and a.
        getDate().hour < 24 and a.getDate().min < 60);
  );
  assertEqual(card pt1, card safetyNet.getPatientAppointments(pat1.getId()));


  assertEqual(card safetyNet.getPatientAppointments(pat2.getId()), 1);
  for all a in set safetyNet.getPatientAppointments(pat2.getId()) do (
   pt2 := pt2 union {a.getDate()};
   assertEqual(a.getPatientId(), pat2.getId());
   assertTrue(a.getDate().year > 2017 and a.getDate().month <= 12 and a.getDate().day < 31 and a.
        getDate().hour < 24 and a.getDate().min < 60);
  );
  assertEqual(card pt2, card safetyNet.getPatientAppointments(pat2.getId()));

  -- system
  assertEqual(card safetyNet.getAppointments(), 2);

  -- hospital
  assertEqual(safetyNet.getHospitalNumberOfAppointments(hos1.getId()), 1);
  for all a in set safetyNet.getHospitalAppointments(hos1.getId()) do (
   pt1 := pt1 union {a.getDate()};
   assertEqual(a.getHospitalId(), hos1.getId());
   assertTrue(a.getDoctorId() in set hos1.getDoctorsIds());
   assertTrue(a.getDate().year > 2017 and a.getDate().month <= 12 and a.getDate().day < 31 and a.
        getDate().hour < 24 and a.getDate().min < 60);
  );

  assertEqual(safetyNet.getHospitalNumberOfAppointments(hos2.getId()), 1);
  for all a in set safetyNet.getHospitalAppointments(hos2.getId()) do (
   pt1 := pt1 union {a.getDate()};
   assertEqual(a.getHospitalId(), hos2.getId());
   assertTrue(a.getDoctorId() in set hos2.getDoctorsIds());
   assertTrue(a.getDate().year > 2017 and a.getDate().month <= 12 and a.getDate().day < 31 and a.
        getDate().hour < 24 and a.getDate().min < 60);
  );


  safetyNet.clearInstance();

);
```

```
private testRemoveAppointment: () ==> ()
 testRemoveAppointment () == (

  dcl hos1: Hospital := new Hospital("Hospital Sao Joao",mk_ModelUtils`Location("Porto", "Alameda
       Prof. Hernni Monteiro","4200-319"), {<ADSE>,<MEDICARE>});
  dcl hos2: Hospital := new Hospital("Hospital da Luz Lisboa",mk_ModelUtils`Location("Lisboa", "
      Avenida Lusada, n  100","4700-959"), {<ADSE>,<MEDIS>, <MULTICARE>});

  dcl doc1: Doctor := new Doctor("jose",35,<ORTOPEDIA>);
  dcl doc2: Doctor := new Doctor("marcelo", 40 ,<CARDIOLOGIA>);
  dcl doc3: Doctor := new Doctor("joaquim",50,<CARDIOLOGIA>);

  dcl pat1: Patient := new Patient("Susana", 26, "Gripe");
  dcl pat2: Patient := new Patient("Maria", 38, "Doena pulmonar");

  dcl app1: Appointment;
  dcl app2: Appointment;
  dcl app3: Appointment;
  dcl app4: Appointment;

  safetyNet := SafetyNetNetwork`getInstance();

  safetyNet.addHospital(hos1);
  safetyNet.addHospital(hos2);

  safetyNet.addDoctor(doc1);
  safetyNet.addDoctor(doc2);
  safetyNet.addDoctor(doc3);

  safetyNet.addPatient(pat1);
  safetyNet.addPatient(pat2);

  safetyNet.associateDoctorToHospital(hos1.getId(), doc1.getId());
  safetyNet.associateDoctorToHospital(hos2.getId(), doc1.getId());
  safetyNet.associateDoctorToHospital(hos2.getId(), doc2.getId());

  assertEqual(card safetyNet.getDoctorAppointments(doc1.getId()), 0);
  assertEqual(card safetyNet.getDoctorAppointments(doc2.getId()), 0);

  app1 := new Appointment(mk_ModelUtils`Date(2018,12,21,8,30), hos1.getId(), doc1.getId(), pat1.
      getId());
  app2 := new Appointment(mk_ModelUtils`Date(2018,12,21,8,30), hos2.getId(), doc2.getId(), pat2.
      getId());


  -- add Appointment (app1, app2)
  safetyNet.addAppointment(app1);
  safetyNet.addAppointment(app2);


  -- remove Appointment
  assertTrue(app2 in set safetyNet.getAppointments());

  safetyNet.removeAppointment(app2);

  assertTrue(app2 not in set safetyNet.getAppointments());

  -- verification of data

  -- doctor appointments
  assertEqual(safetyNet.getDoctorAppointments(doc1.getId()), {app1});
  assertEqual(safetyNet.getDoctorAppointments(doc2.getId()), {});
```

```
assertEqual(safetyNet.getDoctorAppointments(doc3.getId()), {});

-- patient appointments
assertEqual(safetyNet.getPatientAppointments(pat1.getId()), {app1});
assertEqual(safetyNet.getPatientAppointments(pat2.getId()), {});

-- system
assertEqual(card safetyNet.getAppointments(), 1);
assertEqual(safetyNet.getAppointments(), {app1});

-- hospital
assertEqual(safetyNet.getHospitalNumberOfAppointments(hos1.getId()), 1);
assertEqual(safetyNet.getHospitalNumberOfAppointments(hos2.getId()), 0);


-- remove a doctor and all his appointments
app3 := new Appointment(mk_ModelUtils`Date(2018,01,21,8,30), hos2.getId(), doc2.getId(), pat1.
    getId());
safetyNet.addAppointment(app3);


safetyNet.removeDoctor(doc2);

assertEqual(safetyNet.getAppointments(), {app1});


-- remove a patient and all his appointments

safetyNet.addDoctor(doc2);

safetyNet.associateDoctorToHospital(hos2.getId(), doc2.getId());

app2 := new Appointment(mk_ModelUtils`Date(2018,12,21,8,30), hos2.getId(), doc2.getId(), pat2.
    getId());
app4 := new Appointment(mk_ModelUtils`Date(2018,01,15,8,30), hos2.getId(), doc1.getId(), pat1.
    getId());

safetyNet.addAppointment(app2);
safetyNet.addAppointment(app4);

safetyNet.removePatient(pat1);

assertEqual(safetyNet.getAppointments(), {app2});


safetyNet.removeAppointment(app2);

-- remove an hospital and all his appointments

safetyNet.addPatient(pat1);
safetyNet.addAppointment(app1);
safetyNet.addAppointment(app2);
safetyNet.addAppointment(app3);
safetyNet.addAppointment(app4);

assertEqual(safetyNet.getAppointments(), {app1, app2, app3, app4});

safetyNet.removeHospital(hos2);

assertEqual(safetyNet.getAppointments(), {app1});

assertTrue(forall a in set safetyNet.getAppointments() & a.getHospitalId() <> hos2.getId());

safetyNet.clearInstance();
```

```
);

private testAddAgreement: () ==> ()
 testAddAgreement () == (

  dcl hos1: Hospital := new Hospital("Hospital Sao Joao",mk_ModelUtils`Location("Porto", "Alameda
       Prof. Hernni Monteiro","4200-319"), {<ADSE>,<MEDICARE>});
  dcl hos2: Hospital := new Hospital("Hospital da Luz Lisboa",mk_ModelUtils`Location("Lisboa", "
      Avenida Lusada, n  100","4700-959"), {<ADSE>,<MEDIS>, <MULTICARE>});

  safetyNet := SafetyNetNetwork`getInstance();

  safetyNet.addHospital(hos1);
  safetyNet.addHospital(hos2);

  assertEqual(hos1.getAgreements(), {<ADSE>,<MEDICARE>});

  safetyNet.addAgreementToHospital(hos1.getId(),<MULTICARE>);

  assertEqual(hos1.getAgreements(), {<ADSE>,<MEDICARE>, <MULTICARE>});

  safetyNet.clearInstance();

);

private testRemoveAgreement: () ==> ()
 testRemoveAgreement () == (

  dcl hos1: Hospital := new Hospital("Hospital Sao Joao",mk_ModelUtils`Location("Porto", "Alameda
       Prof. Hernni Monteiro","4200-319"), {<ADSE>,<MEDICARE>});
  dcl hos2: Hospital := new Hospital("Hospital da Luz Lisboa",mk_ModelUtils`Location("Lisboa", "
      Avenida Lusada, n  100","4700-959"), {<ADSE>,<MEDIS>, <MULTICARE>});

  safetyNet := SafetyNetNetwork`getInstance();

  safetyNet.addHospital(hos1);
  safetyNet.addHospital(hos2);

  safetyNet.removeAgreementFromHospital(hos1.getId(),<ADSE>);

  assertEqual(hos1.getAgreements(), {<MEDICARE>});

  safetyNet.clearInstance();

);



private testAddObservation: () ==> ()
 testAddObservation () == (

  dcl pat1: Patient := new Patient("Susana", 26, "Gripe");
  dcl pat2: Patient := new Patient("Maria", 38, "Doenca pulmonar");

  safetyNet := SafetyNetNetwork`getInstance();

  safetyNet.addPatient(pat1);
  safetyNet.addPatient(pat2);

  assertEqual(pat1.getClinicalObservations(), ["Gripe"]);
  assertEqual(pat2.getClinicalObservations(), ["Doenca pulmonar"]);

  safetyNet.addClinicalObservation(pat1.getId(), "Pneumonia");
  safetyNet.addClinicalObservation(pat1.getId(), "Varicela");
```

```
  assertEqual(pat1.getClinicalObservations(), ["Gripe", "Pneumonia", "Varicela"]);
  assertEqual(pat2.getClinicalObservations(), ["Doenca pulmonar"]);

  safetyNet.clearInstance();

);



private testGetSpecialtyAppointments: () ==> ()
 testGetSpecialtyAppointments () == (

  dcl hos1: Hospital := new Hospital("Hospital Sao Joao",mk_ModelUtils`Location("Porto", "Alameda
       Prof. Hernni Monteiro","4200-319"), {<ADSE>,<MEDICARE>});
  dcl hos2: Hospital := new Hospital("Hospital da Luz Lisboa",mk_ModelUtils`Location("Lisboa", "
      Avenida Lusada, n  100","4700-959"), {<ADSE>,<MEDIS>, <MULTICARE>});

  dcl doc1: Doctor := new Doctor("jose",35,<ORTOPEDIA>);
  dcl doc2: Doctor := new Doctor("marcelo", 40 ,<CARDIOLOGIA>);
  dcl doc3: Doctor := new Doctor("joaquim",50,<CARDIOLOGIA>);

  dcl pat1: Patient := new Patient("Susana", 26, "Gripe");
  dcl pat2: Patient := new Patient("Maria", 38, "Doena pulmonar");

  dcl app1: Appointment;
  dcl app2: Appointment;
  dcl app3: Appointment;
  dcl app4: Appointment;

  safetyNet := SafetyNetNetwork`getInstance();

  safetyNet.addHospital(hos1);
  safetyNet.addHospital(hos2);

  safetyNet.addDoctor(doc1);
  safetyNet.addDoctor(doc2);
  safetyNet.addDoctor(doc3);

  safetyNet.addPatient(pat1);
  safetyNet.addPatient(pat2);

  safetyNet.associateDoctorToHospital(hos1.getId(), doc1.getId());
  safetyNet.associateDoctorToHospital(hos2.getId(), doc1.getId());
  safetyNet.associateDoctorToHospital(hos2.getId(), doc2.getId());

  app1 := new Appointment(mk_ModelUtils`Date(2018,12,21,8,30), hos1.getId(), doc1.getId(), pat1.
      getId());
  app2 := new Appointment(mk_ModelUtils`Date(2018,12,21,8,30), hos2.getId(), doc2.getId(), pat2.
      getId());
  app3 := new Appointment(mk_ModelUtils`Date(2018,01,21,8,30), hos2.getId(), doc2.getId(), pat1.
      getId());
  app4 := new Appointment(mk_ModelUtils`Date(2018,01,15,8,30), hos2.getId(), doc1.getId(), pat1.
      getId());

  -- add Appointment
  safetyNet.addAppointment(app1);
  safetyNet.addAppointment(app2);
  safetyNet.addAppointment(app3);
  safetyNet.addAppointment(app4);

  assertEqual(safetyNet.getSpecialtyAppointments(<ORTOPEDIA>), {app1, app4});
  assertEqual(safetyNet.getSpecialtyAppointments(<CARDIOLOGIA>), {app2, app3});
  assertEqual(safetyNet.getSpecialtyAppointments(<GINECOLOGIA>), {});
```

```
  safetyNet.clearInstance();

);


private testGetHospitalClosestAvailableDate: () ==> ()
 testGetHospitalClosestAvailableDate () == (

  dcl hos1: Hospital := new Hospital("Hospital Sao Joao",mk_ModelUtils`Location("Porto", "Alameda
       Prof. Hernni Monteiro","4200-319"), {<ADSE>,<MEDICARE>});
  dcl hos2: Hospital := new Hospital("Hospital da Luz Lisboa",mk_ModelUtils`Location("Lisboa", "
      Avenida Lusada, n  100","4700-959"), {<ADSE>,<MEDIS>, <MULTICARE>});

  dcl doc1: Doctor := new Doctor("jose",35,<ORTOPEDIA>);
  dcl doc2: Doctor := new Doctor("marcelo", 40 ,<CARDIOLOGIA>);
  dcl doc3: Doctor := new Doctor("joaquim",50,<CARDIOLOGIA>);

  dcl pat1: Patient := new Patient("Susana", 26, "Gripe");
  dcl pat2: Patient := new Patient("Maria", 38, "Doena pulmonar");

  dcl app1: Appointment;
  dcl app2: Appointment;
  dcl app3: Appointment;
  dcl app4: Appointment;

  dcl res: ModelUtils`Date_DoctorId;

  safetyNet := SafetyNetNetwork`getInstance();

  safetyNet.addHospital(hos1);
  safetyNet.addHospital(hos2);

  safetyNet.addDoctor(doc1);
  safetyNet.addDoctor(doc2);
  safetyNet.addDoctor(doc3);

  safetyNet.addPatient(pat1);
  safetyNet.addPatient(pat2);

  safetyNet.associateDoctorToHospital(hos1.getId(), doc1.getId());
  safetyNet.associateDoctorToHospital(hos2.getId(), doc1.getId());
  safetyNet.associateDoctorToHospital(hos2.getId(), doc2.getId());


  app1 := new Appointment(mk_ModelUtils`Date(2018,01,15,8,30), hos2.getId(), doc1.getId(), pat1.
      getId());
  app2 := new Appointment(mk_ModelUtils`Date(2018,01,21,8,30), hos2.getId(), doc2.getId(), pat1.
      getId());
  app3 := new Appointment(mk_ModelUtils`Date(2018,12,21,8,30), hos1.getId(), doc1.getId(), pat1.
      getId());
  app4 := new Appointment(mk_ModelUtils`Date(2018,12,21,8,30), hos2.getId(), doc2.getId(), pat2.
      getId());


  -- add Appointment
  safetyNet.addAppointment(app1);
  safetyNet.addAppointment(app2);
  safetyNet.addAppointment(app3);
  safetyNet.addAppointment(app4);

  assertFalse(Appointment`appointmentDatesDontOverlap(app1.getDate(), mk_ModelUtils`Date
      (2018,01,15,8,30)));

  res := safetyNet.getHospitalClosestAvailableDate({app1, app2, app3, app4});
```

```
  assertEqual(res, mk_ModelUtils`Date_DoctorId(mk_ModelUtils`Date(2018,1,1,8,30), doc2.getId()));

  safetyNet.clearInstance();

);



private testGetNextAppointmentDate: () ==> ()
 testGetNextAppointmentDate () == (

  -- next day
  assertEqual(Appointment`getNextAppointmentDate(mk_ModelUtils`Date(2018,12,12,23,45)) ,
      mk_ModelUtils`Date(2018,12,13,0,15));
  -- next year
  assertEqual(Appointment`getNextAppointmentDate(mk_ModelUtils`Date(2018,12,30,23,45)) ,
      mk_ModelUtils`Date(2019,1,1,0,15));
  -- next month
  assertEqual(Appointment`getNextAppointmentDate(mk_ModelUtils`Date(2018,10,30,23,45)) ,
      mk_ModelUtils`Date(2018,11,1,0,15));
  -- next 30 min
  assertEqual(Appointment`getNextAppointmentDate(mk_ModelUtils`Date(2018,10,15,10,30)) ,
      mk_ModelUtils`Date(2018,10,15,11,0));

);

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

Function or operation	Line	Coverage	Calls
main	13	100.0%	1
testAddAgreement	740	100.0%	3
testAddAppointment	516	100.0%	1
testAddDoctor	117	100.0%	1
testAddHospital	195	100.0%	1
testAddObservation	825	100.0%	1
testAddPatient	474	100.0%	3
testAssociateDoctorToAnHospital	232	100.0%	1
testDisassociateDoctorToAnHospital	262	100.0%	3
testGetAllDoctors	143	100.0%	2
testGetAllHospitals	334	100.0%	1
testGetDoctorById	454	100.0%	1
testGetDoctorBySpecialtie	403	100.0%	1
testGetDoctorHospitals	372	100.0%	1
testGetHospitalClosestAvailableDate	904	100.0%	1
testGetHospitalSpecialties	424	100.0%	1
testGetHospitalsByAgreement	334	100.0%	1
testGetHospitalsById	314	100.0%	1
testGetHospitalsByLocation	295	100.0%	1
testGetHospitalsByName	352	100.0%	1
testGetNextAppointmentDate	964	100.0%	1

testGetSpecialtyAppointments	850	100.0%	1
testRemoveAgreement	761	100.0%	1
testRemoveAppointment	623	100.0%	1
testRemoveDoctor	162	100.0%	1
testRemoveHospital	212	100.0%	1
testRemovePatient	497	100.0%	1
SystemTest.vdmpp		100.0%	34