# MFES

December 28, 2017

## Contents

# 1 Agenda

```
class Agenda
instance variables

private healthProfessional : [HealthProfessional];
private agenda : set of (Schedule);

inv healthProfessional <> nil;
inv card agenda >= 0;

operations

 public Agenda : HealthProfessional ==> Agenda
  Agenda(h) == (healthProfessional := h; agenda := {}; return self)
 pre healthProfessional <> nil
 post healthProfessional = h and agenda = {};


 pure public getHealthProfessional : () ==> HealthProfessional
  getHealthProfessional() == (return healthProfessional);


 pure public getAgenda : () ==> set of (Schedule)
  getAgenda() == (return agenda);


 public addSchedule : Schedule ==> ()
  addSchedule(s) == (agenda := agenda union {s})
 pre s not in set agenda and forall sch in set agenda & not overlap(s, sch)
 post s in set agenda;


 public removeSchedule : Schedule ==> ()
  removeSchedule(s) == (agenda := agenda \ {s})
 pre s in set agenda
 post s not in set agenda;


 pure public overlap: Schedule * Schedule ==> bool
  overlap(t1, t2) == (return t1.overlap(t1, t2));
end Agenda
```

| Function or operation | Line | Coverage | Calls |
|---|---|---|---|
| Agenda | 11 | 100.0% | 240 |
| addSchedule | 22 | 100.0% | 285 |
| getAgenda | 19 | 100.0% | 503 |
| getHealthProfessional | 16 | 100.0% | 2207 |
| overlap | 32 | 100.0% | 52 |
| removeSchedule | 27 | 100.0% | 158 |
| Agenda.vdmpp | | 100.0% | 3445 |

# 2 Appointment

```
class Appointment is subclass of Task

instance variables
  private prescriptions:set of (Prescription);
  private priority : Types`Priority;

  inv priority <> nil;
  inv card prescriptions >= 0;
  inv medicalAssoc.getType() = <Doctor>;
operations

 public Appointment: HealthProfessional * Schedule * Patient * Hospital==> Appointment
  Appointment(d, s, p, h) == (medicalAssoc := d; priority := <Medium>; prescriptions := {}; Task(
      d, s, p, h, <Appointment>))
 post medicalAssoc = d and prescriptions = {} and priority = <Medium>;

 public Appointment: HealthProfessional * Types`Priority * Schedule * Patient * Hospital ==>
     Appointment
  Appointment(d, p, s, pat, h) == (medicalAssoc := d; priority := p; prescriptions := {}; Task(d,
       s, pat, h, <Urgencies>))
 pre p <> nil
 post medicalAssoc = d and prescriptions = {} and priority = p;


 pure public getPriority : () ==> Types`Priority
  getPriority() == (return priority);


  pure public getPrescriptions : () ==> set of (Prescription)
   getPrescriptions() == (return prescriptions);


  public setPriority : Types`Priority ==> ()
   setPriority(p) == (priority := p)
  pre type = <Urgencies>;


  public addPrescription : Prescription ==> ()
   addPrescription(p) == (prescriptions := prescriptions union {p})
  pre p not in set prescriptions
  post p in set prescriptions;


 public removePrescription : Prescription ==> ()
   removePrescription(p) == (prescriptions := prescriptions \ {p})
  pre p in set prescriptions
  post p not in set prescriptions;

end Appointment
```

| Function or operation | Line | Coverage | Calls |
|---|---|---|---|
| Appointment | 11 | 100.0% | 114 |
| addPrescription | 30 | 100.0% | 96 |
| getPrescriptions | 23 | 100.0% | 288 |
| getPriority | 20 | 100.0% | 72 |
| removePrescription | 35 | 100.0% | 96 |
| setPriority | 26 | 100.0% | 48 |
| Appointment.vdmpp | | 100.0% | 714 |

# 3 HealthProfessional

```
class HealthProfessional is subclass of Person

instance variables
  private medicalNumber: Types`String;
  private specialties:set of (Specialty);
  private patients : set of(Patient);
 private type : Types`Type;

 inv card patients >= 0;
  inv card specialties < 5;
 inv type <> nil;
operations

 public HealthProfessional: Types`String * Types`String * Types`String * Types`String * Types`
     String * Types`String * Types`Type ==> HealthProfessional
  HealthProfessional(a, fn, ln, c, pn, s, t) == (medicalNumber := s; type := t; specialties :=
      {}; patients := {}; Person(a, fn, ln, c, pn))
 pre t <> nil
 post medicalNumber = s and type = t and specialties = {} and patients = {};


 pure public getMedicalNumber: () ==> Types`String
  getMedicalNumber() == (return medicalNumber);


 pure public getSpecialties: () ==> set of (Specialty)
  getSpecialties() == (return specialties);


 pure public getPatients: () ==> set of (Patient)
  getPatients() == (return patients);


 pure public getType : () ==> Types`Type
  getType() == (return type);


 public removeSpecialty: Specialty ==> ()
  removeSpecialty(s) == (specialties := specialties \ {s})
 pre s in set specialties
 post s not in set specialties;


 public addSpecialty: Specialty ==> ()
  addSpecialty(s) == (specialties := specialties union {s})
 pre s not in set specialties
 post s in set specialties;


 public addPatient : Patient ==> ()
  addPatient(p) == (patients :=  patients union {p})
 pre p not in set patients
 post p in set patients;


 public removePatient : Patient ==> ()
  removePatient(p) == (patients := patients \ {p})
 pre p in set patients
 post p not in set patients;

end HealthProfessional
```

| Function or operation | Line | Coverage | Calls |
|---|---|---|---|
| HealthProfessional | 13 | 100.0% | 978 |
| addPatient | 40 | 100.0% | 183 |
| addSpecialty | 35 | 100.0% | 60 |
| getMedicalNumber | 18 | 100.0% | 120 |
| getPatients | 24 | 100.0% | 533 |
| getSpecialties | 21 | 100.0% | 330 |
| getType | 27 | 100.0% | 2168 |
| removePatient | 45 | 100.0% | 30 |
| removeSpecialty | 30 | 100.0% | 30 |
| HealthProfessional.vdmpp | | 100.0% | 4432 |

# 4 Hospital

```
class Hospital
instance variables
  private medicalAssociated: set of (HealthProfessional);
  private agenda : set of (Agenda);
  private name: Types`String;
  private address: Types`String;
  private tasks: set of(Task);
  private trainings: set of(Training);
  private safetyNet: [SafetyNetHospital];

 inv safetyNet <> nil;
 inv card medicalAssociated >= 0;
 inv card agenda >= 0;
 inv card tasks >= 0;
operations


 public Hospital: Types`String * Types`String * SafetyNetHospital ==> Hospital
  Hospital(n, a, s) == (name := n; address := a; safetyNet := s; medicalAssociated := {}; tasks
       := {}; trainings := {}; agenda := {};
  safetyNet.addHospital(self); return self)
 pre safetyNet <> nil
 post name = n and address = a and safetyNet = s and medicalAssociated = {} and tasks = {} and
      trainings = {} and agenda = {};


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


 pure public getAddress: () ==> Types`String
  getAddress() == (return address);


 pure public getAgendas : () ==> set of(Agenda)
  getAgendas() == (return agenda);


 pure public getAgenda : HealthProfessional ==> set of(Schedule)
```

```
 getAgenda(h) == (
  for all a in set agenda do
   if(a.getHealthProfessional() = h)
    then return a.getAgenda();
  return {});


public removeAgenda : Agenda ==> ()
 removeAgenda(a) == (agenda := agenda \ {a})
pre a in set agenda
post a not in set agenda;


public addMedAssociated: HealthProfessional ==> ()
 addMedAssociated(d) == (
  dcl agendaNew : Agenda;
  agendaNew := new Agenda(d);
  agenda := agenda union {agendaNew};
  medicalAssociated := {d} union medicalAssociated)
pre d not in set medicalAssociated
post d in set medicalAssociated;


public removeMedAssociated: HealthProfessional ==> ()
 removeMedAssociated(d) == (
  for all t in set tasks do
   if(d = t.getMedAssoc())
    then removeTask(t);
  for all t in set trainings do
   if(d = t.getMedAssoc())
    then removeTraining(t);
  for all a in set agenda do
   if(a.getHealthProfessional().getCC() = d.getCC())
    then removeAgenda(a);
  medicalAssociated := medicalAssociated \ {d})
pre d in set medicalAssociated
post d not in set medicalAssociated;


public addAgenda : HealthProfessional * Schedule ==> ()
 addAgenda(h, sch) == (
  for all a in set agenda do
   if(a.getHealthProfessional() = h)
    then a.addSchedule(sch);)
pre sch not in set getAgenda(h)
post sch in set getAgenda(h);


public addTask: Task ==> ()
 addTask(d) == (
  if(d.getPatient() not in set d.getMedAssoc().getPatients())
   then d.getMedAssoc().addPatient(d.getPatient());
  tasks := {d} union tasks;
  for all a in set agenda do
   if(a.getHealthProfessional().getCC() = d.getMedAssoc().getCC())
    then a.removeSchedule(d.getSchedule());)
pre d not in set tasks and d.getSchedule() in set getAgenda(d.getMedAssoc())
post d in set tasks and d.getPatient() in set d.getMedAssoc().getPatients() and d.getSchedule()
    not in set getAgenda(d.getMedAssoc());


public removeTask: Task ==> ()
 removeTask(d) == (
  for all a in set agenda do
   if(a.getHealthProfessional() = d.getMedAssoc())
```

```
      then a.addSchedule(d.getSchedule());
   tasks := tasks \ {d})
 pre d in set tasks and d.getSchedule() not in set getAgenda(d.getMedAssoc())
 post d not in set tasks and d.getSchedule() in set getAgenda(d.getMedAssoc());


 public addTraining: Training ==> ()
  addTraining(d) == (
   for all a in set agenda do
    if(a.getHealthProfessional() = d.getMedAssoc())
      then a.removeSchedule(d.getSchedule());
   trainings := {d} union trainings)
 pre d not in set trainings and d.getSchedule() in set getAgenda(d.getMedAssoc())
 post d in set trainings and d.getSchedule() not in set getAgenda(d.getMedAssoc());


 public removeTraining: Training ==> ()
  removeTraining(d) == (
   for all a in set agenda do
    if(a.getHealthProfessional() = d.getMedAssoc())
      then a.addSchedule(d.getSchedule());
   trainings := trainings \ {d})
 pre d in set trainings and d.getSchedule() not in set getAgenda(d.getMedAssoc())
 post d not in set trainings and d.getSchedule() in set getAgenda(d.getMedAssoc());


 pure public getTasksByType: Types`TaskType ==> set of (Task)
  getTasksByType(s) == (
   dcl tasksTotal: set of (Task);
   tasksTotal := {};
   for all t in set tasks do
    if(t.getType() = s)
     then tasksTotal := tasksTotal union {t};

   return tasksTotal);


 pure public getTrainingsByType: Types`Purpose ==> set of (Training)
  getTrainingsByType(s) == (
   dcl train: set of (Training);
   train := {};
   for all t in set trainings do
    if(t.getPurpose() = s)
     then train := train union {t};

   return train);


 pure public getMedicalAssociatedByType: Types`Type ==> set of (HealthProfessional)
  getMedicalAssociatedByType(type) == (
   dcl med: set of(HealthProfessional);
   med := {};
   for all d in set medicalAssociated do
    if(d.getType() = type)
     then med := med union {d};

   return med);
end Hospital
```

| Function or operation | Line | Coverage | Calls |
|---|---|---|---|

| | | | |
|---|---|---|---|
| Hospital | 17 | 100.0% | 186 |
| addAgenda | 68 | 0.0% | 0 |
| addMedAssociated | 44 | 100.0% | 171 |
| addTask | 76 | 100.0% | 103 |
| addTraining | 96 | 100.0% | 3 |
| getAddress | 26 | 100.0% | 72 |
| getAgenda | 32 | 100.0% | 5 |
| getAgendas | 29 | 100.0% | 42 |
| getMedicalAssociatedByType | 134 | 100.0% | 223 |
| getName | 23 | 100.0% | 114 |
| getTasksByType | 114 | 100.0% | 124 |
| getTrainingsByType | 124 | 100.0% | 34 |
| removeAgenda | 39 | 100.0% | 12 |
| removeMedAssociated | 53 | 65.8% | 60 |
| removeTask | 87 | 100.0% | 4 |
| removeTraining | 105 | 100.0% | 45 |
| Hospital.vdmpp | | 91.9% | 1198 |

# 5 Medicament

```
class Medicament

instance variables
  private name:Types`String;
operations

 public Medicament: Types`String ==> Medicament
  Medicament(n) == (name := n; return self)
 post name = n;


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

end Medicament
```

| Function or operation | Line | Coverage | Calls |
|---|---|---|---|
| Medicament | 6 | 100.0% | 60 |
| getName | 10 | 100.0% | 24 |
| Medicament.vdmpp | | 100.0% | 84 |

# 6 Patient

```
class Patient is subclass of Person
instance variables
  private healthNumber: Types`String;
operations
```

```
 public Patient: Types`String * Types`String * Types`String * Types`String * Types`String * Types
     `String ==> Patient
  Patient(a, fn, ln, c, pn, n) == ( healthNumber := n; Person(a, fn, ln, c, pn))
 post healthNumber = n;


 pure public getHealthNumber : () ==> Types`String
  getHealthNumber() == (return healthNumber);

end Patient
```

| Function or operation | Line | Coverage | Calls |
|---|---|---|---|
| Patient | 5 | 100.0% | 516 |
| getHealthNumber | 9 | 100.0% | 30 |
| Patient.vdmpp | | 100.0% | 546 |

# 7 Person

```
class Person

instance variables
  protected address: Types`String;
  protected firstName: Types`String;
  protected lastName: Types`String;
  protected cc : Types`String;
  protected phoneNumber: Types`String;
operations

 public Person: Types`String * Types`String * Types`String * Types`String * Types`String ==>
     Person
  Person(a, fn, ln, c, pn) == ( address := a; firstName := fn; lastName := ln; cc := c;
      phoneNumber := pn; return self)
 post address = a and firstName = fn and lastName = ln and cc = c and phoneNumber = pn;


 pure public getCC : () ==> Types`String
  getCC() == (return cc);


 pure public getInfo: () ==> Types`String
  getInfo() == (return "Name: " ^ firstName ^ " " ^ lastName ^ "\nAddress: " ^ address ^ "\nPhone
       Number: " ^ phoneNumber ^ "\nCC: " ^ cc);

end Person
```

| Function or operation | Line | Coverage | Calls |
|---|---|---|---|
| Person | 10 | 100.0% | 1494 |
| getCC | 14 | 100.0% | 2538 |
| getInfo | 17 | 100.0% | 150 |
| Person.vdmpp | | 100.0% | 4182 |

# 8 Prescription

```
class Prescription

instance variables
  private medicaments:set of (Medicament);
  private code:Types`String;

operations

 public Prescription: Types`String ==> Prescription
  Prescription(c) == (code := c; medicaments := {}; return self)
 post code = c and medicaments = {};


 pure public getCode : () ==> Types`String
  getCode() == (return code);


 public addMedicament: Medicament ==> ()
  addMedicament(m) == (medicaments := {m} union medicaments)
 pre m not in set medicaments
 post m in set medicaments;


 public removeMedicament: Medicament ==> ()
  removeMedicament(m) == (medicaments := medicaments \ {m})
 pre m in set medicaments
 post m not in set medicaments;


 pure public getMedicaments: () ==> set of (Medicament)
  getMedicaments() == (return medicaments);

end Prescription
```

| Function or operation | Line | Coverage | Calls |
|---|---|---|---|
| Prescription | 8 | 100.0% | 60 |
| addMedicament | 15 | 100.0% | 24 |
| getCode | 12 | 100.0% | 24 |
| getMedicaments | 25 | 100.0% | 168 |
| removeMedicament | 20 | 100.0% | 48 |
| Prescription.vdmpp | | 100.0% | 324 |

# 9 SafetyNetHospital

```
class SafetyNetHospital
instance variables
 private hospitals: set of (Hospital);

 inv card hospitals >= 0;
operations
```

```
public SafetyNetHospital : () ==> SafetyNetHospital
 SafetyNetHospital() == (hospitals := {}; return self)
post hospitals = {};


public addHospital : Hospital ==> ()
 addHospital(h) == (hospitals := hospitals union {h})
pre h not in set hospitals
post h in set hospitals;


public removeHospital : Hospital ==> ()
 removeHospital(h) == (hospitals := hospitals \ {h})
pre h in set hospitals
post h not in set hospitals;


pure public getHospitals : () ==> set of (Hospital)
 getHospitals() == (return hospitals);


pure public getHospitalsMoreAppointments : Types`TaskType ==> Hospital
 getHospitalsMoreAppointments(t) == (
                   dcl max: int, hosp: Hospital;
                   max := -1;
                   for all h in set hospitals do
                    if((card h.getTasksByType(t)) > max)
                     then (max := (card h.getTasksByType(t)); hosp := h);
                   return hosp);


pure public getMedMoreHospitals : Types`Type ==> set of(HealthProfessional)
 getMedMoreHospitals(t) == (
                  dcl doctors: set of(HealthProfessional);
                  doctors := {};
                  for all h in set hospitals do (
                   dcl med: set of (HealthProfessional), list: set of(Hospital);
                   med := h.getMedicalAssociatedByType(t);

                   list := hospitals \ {h};
                   for all m in set med do(
                    for all l in set list do
                     if(m.getType() = t and m in set l.getMedicalAssociatedByType(t) and m not in
                          set doctors)
                      then doctors := doctors union {m};
                   );
                  );

                  return doctors;
                 );


pure public getMedAssociatedByPatient: Patient * Types`Type ==> map Hospital to set of(
    HealthProfessional)
 getMedAssociatedByPatient(p, t) == (
                    dcl maps: map Hospital to set of(HealthProfessional), med : set of (
                        HealthProfessional);
                    maps := { |-> };
                    med := {};
                    for all h in set hospitals do (
                     for all m in set h.getMedicalAssociatedByType(t) do
                      if(p in set m.getPatients())
                       then med := med union {m};

                     maps := maps munion {h |-> med};
```

```
                    med := {};);
                    return maps);


 pure public getMedByHospital: Types`Type ==> map Hospital to set of(HealthProfessional)
  getMedByHospital(t) == (
                    dcl maps: map Hospital to set of(HealthProfessional);
                    maps := { |-> };
                    for all h in set hospitals do
                     maps := maps munion {h |-> h.getMedicalAssociatedByType(t)};
                    return maps);
end SafetyNetHospital
```

| Function or operation | Line | Coverage | Calls |
|---|---|---|---|
| SafetyNetHospital | 8 | 100.0% | 114 |
| addHospital | 12 | 100.0% | 186 |
| getHospitals | 22 | 100.0% | 123 |
| getHospitalsMoreAppointments | 25 | 100.0% | 12 |
| getMedAssociatedByPatient | 53 | 100.0% | 3 |
| getMedByHospital | 67 | 100.0% | 2 |
| getMedMoreHospitals | 34 | 100.0% | 18 |
| removeHospital | 17 | 100.0% | 48 |
| SafetyNetHospital.vdmpp | | 100.0% | 506 |

# 10 Schedule

```
class Schedule

instance variables
  private startHour: Types`Date;
  private endHour: Types`Date;

  inv lessThan(startHour, endHour);
operations

 public Schedule: Types`Date * Types`Date ==> Schedule
  Schedule(d, d2) == (startHour := d; endHour := d2; return self)
 pre lessThan(d, d2)
 post startHour = d and endHour = d2;


 public setSchedule : Types`Date * Types`Date ==> ()
  setSchedule(d1, d2) == (startHour := d1; endHour := d2;)
 pre lessThan(d1, d2)
 post startHour = d1 and endHour = d2;


 pure public getScheduleStart : () ==> Types`Date
  getScheduleStart() == (return startHour);


 pure public getScheduleEnd : () ==> Types`Date
  getScheduleEnd() == (return endHour);
```

```
 pure public overlap : Schedule * Schedule ==> bool
  overlap(d1, d2) == (
          if((lessThan(d1.startHour, d2.startHour) and greaterThan(d1.endHour, d2.startHour)) or
          (not lessThan(d1.startHour, d2.startHour) and lessThan(d1.startHour, d2.endHour)))
           then return true;
          return false;);


 pure static public lessThan : Types`Date * Types`Date ==> bool
  lessThan(d1, d2) == (
          if(d1.year < d2.year)
           then return true
          else if(d1.year > d2.year)
           then return false;
          if(d1.month < d2.month)
           then return true
          else if(d1.month > d2.month)
           then return false;
          if(d1.day < d2.day)
           then return true
          else if(d1.day > d2.day)
           then return false;
          if(d1.time.hour < d2.time.hour)
           then return true
          else if(d1.time.hour > d2.time.hour)
           then return false;
          return (d1.time.min < d2.time.min););


 pure static public greaterThan : Types`Date * Types`Date ==> bool
  greaterThan(d1, d2) == (
          if(d1.year < d2.year)
           then return false
          else if(d1.year > d2.year)
           then return true;
          if(d1.month < d2.month)
           then return false
          else if(d1.month > d2.month)
           then return true;
          if(d1.day < d2.day)
           then return false
          else if(d1.day > d2.day)
           then return true;
          if(d1.time.hour < d2.time.hour)
           then return false
          else if(d1.time.hour > d2.time.hour)
           then return true;
          return (d1.time.min > d2.time.min););
end Schedule
```

| Function or operation | Line | Coverage | Calls |
|---|---|---|---|
| Schedule | 9 | 100.0% | 726 |
| getScheduleEnd | 22 | 100.0% | 672 |
| getScheduleStart | 19 | 100.0% | 1104 |
| greaterThan | 52 | 96.0% | 0 |
| lessThan | 32 | 92.0% | 24 |
| overlap | 25 | 83.3% | 52 |
| setSchedule | 14 | 100.0% | 24 |

| | | | |
|---|---|---|---|
| Schedule.vdmpp | | 93.5% | 2602 |

# 11 Specialty

```
class Specialty

instance variables
  private name: Types`String;
operations

 public Specialty : Types`String ==> Specialty
  Specialty(n) == (name := n; return self)
 post name = n;


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

end Specialty
```

| Function or operation | Line | Coverage | Calls |
|---|---|---|---|
| Specialty | 6 | 100.0% | 60 |
| getName | 10 | 100.0% | 60 |
| Specialty.vdmpp | | 100.0% | 120 |

# 12 Surgery

```
class Surgery is subclass of Task
instance variables
  private secondaryDoctors:set of (HealthProfessional);
  private other:set of (HealthProfessional);

  inv card secondaryDoctors >= 0;
  inv card other >= 0;
operations

 public Surgery: HealthProfessional * Schedule * Patient * Hospital ==> Surgery
  Surgery(s, sch, p, h) == (medicalAssoc := s ; other := {}; secondaryDoctors := {}; Task(s, sch,
       p, h, <Surgery>))
 post medicalAssoc = s and other = {} and secondaryDoctors = {};


 public addSecondaryDoctor : HealthProfessional ==> ()
  addSecondaryDoctor(s) == (secondaryDoctors := secondaryDoctors union {s})
 pre s <> medicalAssoc and s.getType() = <Surgeon> and  s not in set secondaryDoctors
 post s in set secondaryDoctors;


 public removeSecondaryDoctor : HealthProfessional ==> ()
  removeSecondaryDoctor(s) == (secondaryDoctors := secondaryDoctors \ {s})
 pre s.getType() = <Surgeon> and s in set secondaryDoctors
 post s not in set secondaryDoctors;
```

```
 public addOther : HealthProfessional ==> ()
  addOther(s) == (other := other union {s})
 pre s.getType() = <Nurse> and s not in set other
 post s in set other;


 public removeOther : HealthProfessional ==> ()
  removeOther(s) == (other := other \ {s})
 pre s.getType() = <Nurse> and s in set other
 post s not in set other;


 public setMainDoctor : HealthProfessional ==> ()
  setMainDoctor(s) == (medicalAssoc := s)
 pre s.getType() = <Surgeon> and s not in set secondaryDoctors;


 public getMainDoctor : () ==> HealthProfessional
  getMainDoctor() == (return medicalAssoc);


 pure public getSurgeryPersons : Types`Type ==> set of (HealthProfessional)
  getSurgeryPersons(t) == (
               dcl med : set of (HealthProfessional);
               if(t = <Surgeon>)
                then med := secondaryDoctors
               else
                med := other;
               return med);
end Surgery
```

| Function or operation | Line | Coverage | Calls |
|---|---|---|---|
| Surgery | 9 | 100.0% | 114 |
| addOther | 23 | 100.0% | 24 |
| addSecondaryDoctor | 13 | 100.0% | 24 |
| getMainDoctor | 37 | 100.0% | 48 |
| getSurgeryPersons | 40 | 100.0% | 72 |
| removeOther | 28 | 100.0% | 24 |
| removeSecondaryDoctor | 18 | 100.0% | 24 |
| setMainDoctor | 33 | 100.0% | 72 |
| Surgery.vdmpp | | 100.0% | 402 |

# 13 Task

```
class Task
instance variables
  protected schedule:[Schedule];
  protected patient:[Patient];
  protected hospital:[Hospital];
  protected medicalAssoc:[HealthProfessional];
  protected type : Types`TaskType;

  inv patient <> nil;
```

```
  inv hospital <> nil;
  inv type <> nil;
operations

 public Task: HealthProfessional * Schedule * Patient * Hospital * Types`TaskType ==> Task
  Task(med, s, p, h, t) == (schedule := s; patient := p; hospital := h; type := t; medicalAssoc
      := med; return self)
 pre med.getCC() <> p.getCC()
 post schedule = s and patient = p and hospital = h and medicalAssoc = med;


 pure public getSchedule: () ==> Schedule
  getSchedule() == (return schedule);


 pure public getPatient: () ==> Patient
  getPatient() == (return patient);


 pure public getHospital: () ==> Hospital
  getHospital() == (return hospital);


 pure public getType: () ==> Types`TaskType
  getType() == (return type);


 pure public getMedAssoc : () ==> HealthProfessional
  getMedAssoc() == (return medicalAssoc);


 public setSchedule : Schedule ==> ()
  setSchedule(s) == (schedule := s);


 pure public getSurgeryPersons : Types`Type ==> set of (HealthProfessional)
  getSurgeryPersons(t) == ( return {}; );

end Task
```

| Function or operation | Line | Coverage | Calls |
|---|---|---|---|
| Task | 13 | 100.0% | 564 |
| getHospital | 24 | 100.0% | 24 |
| getMedAssoc | 30 | 100.0% | 1281 |
| getPatient | 21 | 100.0% | 401 |
| getSchedule | 18 | 100.0% | 1484 |
| getSurgeryPersons | 36 | 100.0% | 24 |
| getType | 27 | 100.0% | 520 |
| setSchedule | 33 | 100.0% | 24 |
| Task.vdmpp | | 100.0% | 4322 |

# 14 Training

```
class Training
```

```
instance variables
 private medicalAssociated:[HealthProfessional];
 private purpose:[Types`Purpose];
 private schedule:[Schedule];

 inv medicalAssociated <> nil;
 inv purpose <> nil;
 inv schedule <> nil;

operations

 public Training: Types`Purpose * Schedule * HealthProfessional ==> Training
   Training(p, s, h) == (purpose := p; schedule := s; medicalAssociated := h; return self)
 post purpose = p and schedule = s and medicalAssociated = h;


 pure public getSchedule : () ==> Schedule
   getSchedule() == (return schedule);


  pure public getPurpose : () ==> Types`Purpose
  getPurpose() == (return purpose);


 pure public getMedAssoc : () ==> HealthProfessional
  getMedAssoc() == (return medicalAssociated);


 public setSchedule : Schedule ==> ()
   setSchedule(s) == (schedule := s);


 public setPurpose : Types`Purpose ==> ()
   setPurpose(p) == (purpose := p);

 end Training
```

| Function or operation | Line | Coverage | Calls |
|---|---|---|---|
| Training | 13 | 100.0% | 131 |
| getMedAssoc | 23 | 100.0% | 69 |
| getPurpose | 20 | 100.0% | 57 |
| getSchedule | 17 | 100.0% | 501 |
| setPurpose | 29 | 100.0% | 24 |
| setSchedule | 26 | 100.0% | 24 |
| Training.vdmpp | | 100.0% | 806 |

# 15 Treatment

```
class Treatment is subclass of Task
instance variables
  public med: [HealthProfessional];
  public name: Types`String;

  inv med.getType() = <Nurse> or med.getType() = <Technician>;
operations
```

```
 public Treatment: HealthProfessional * Types`String * Schedule * Patient * Hospital ==>
     Treatment
  Treatment(m, n, s, p, h) == (name := n; med := m; Task(m, s, p, h, <Other>))
 post name = n and med = m;


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


 pure public getMed : () ==> HealthProfessional
  getMed() == (return med);

end Treatment
```

| Function or operation | Line | Coverage | Calls |
|---|---|---|---|
| Treatment | 9 | 100.0% | 114 |
| getMed | 16 | 100.0% | 24 |
| getName | 13 | 100.0% | 24 |
| Treatment.vdmpp | | 100.0% | 162 |

# 16 Types

```
class Types
types
 public String = seq1 of (char);
 public Priority = <High> | <Medium> | <Low>;
 public Type = <Doctor> | <Surgeon> | <Nurse> | <Technician>;
 public TaskType = <Appointment> | <Urgencies> | <Surgery> | <Other>;
 public Purpose = <Training> | <AddSkills>;
 public Time :: hour : nat
         min: nat
 inv t == t.hour >= 0 and t.hour < 24 and t.min >= 0 and t.min < 60;
 public Date ::  year: nat1
         month: nat1
         day: nat1
         time: Time
 inv d == d.month <= 12 and d.day <= daysOfMonth(d.month);

operations

 public static pure daysOfMonth : nat1 ==> nat1
  daysOfMonth(month) == (
              if(month = 1 or month = 3 or month = 5 or month = 7 or month = 8 or month = 10 or
                  month = 12)
               then return 31
              else if(month = 4 or month = 6 or month = 9 or month = 11)
               then return 30
              else
               return 28;);

end Types
```

| Function or operation | Line | Coverage | Calls |
|---|---|---|---|
| daysOfMonth | 18 | 100.0% | 1476 |
| Types.vdmpp | | 100.0% | 1476 |

# 17 PersonTest

```
class PersonTest
instance variables
 private patient: Patient := new Patient("Rua 1 Maio", "Rui", "Andrade", "123456789", "223456111"
     , "0987654321");
 private doctor: HealthProfessional := new HealthProfessional("Rua de Cima", "Ana", "Marques", "
     123432156", "921349076", "111111222", <Doctor>);
 private surgeon: HealthProfessional := new HealthProfessional("Rua 2", "Diogo", "Viana", "
     234512389", "921349134", "111111232", <Surgeon>);
 private nurse: HealthProfessional := new HealthProfessional("Rua de Baixo", "Lisete", "Antunes",
      "123444654", "921378643", "111222333", <Nurse>);
 private technician: HealthProfessional := new HealthProfessional("Rua Antero Marques", "Ins", "
     Pinto", "123432151", "921348765", "123432578", <Technician>);
operations

 private assertTrue: bool ==> ()
  assertTrue(cond) == return
 pre cond;


 public testGetInformation: () ==> ()
  testGetInformation() == (
   IO`print("\n Getting patient informations \n");
   assertTrue(patient.getHealthNumber() = "0987654321");
   assertTrue(patient.getCC() = "123456789");
   assertTrue(patient.getInfo() = "Name: " ^ "Rui" ^ " " ^ "Andrade" ^ "\nAddress: " ^ "Rua 1
       Maio" ^ "\nPhone Number: " ^ "223456111" ^ "\nCC: " ^ "123456789");

   IO`print("\n Getting doctor informations \n");
   assertTrue(doctor.getMedicalNumber() = "111111222");
   assertTrue(doctor.getCC() = "123432156");
   assertTrue(doctor.getInfo() = "Name: " ^ "Ana" ^ " " ^ "Marques" ^ "\nAddress: " ^ "Rua de
       Cima" ^ "\nPhone Number: " ^ "921349076" ^ "\nCC: " ^ "123432156");
   assertTrue(doctor.getType() = <Doctor>);

   IO`print("\n Getting surgeon informations \n");
   assertTrue(surgeon.getMedicalNumber() = "111111232");
   assertTrue(surgeon.getCC() = "234512389");
   assertTrue(surgeon.getInfo() = "Name: " ^ "Diogo" ^ " " ^ "Viana" ^ "\nAddress: " ^ "Rua 2" ^
       "\nPhone Number: " ^ "921349134" ^ "\nCC: " ^ "234512389");
   assertTrue(surgeon.getType() = <Surgeon>);

   IO`print("\n Getting nurse informations \n");
   assertTrue(nurse.getMedicalNumber() = "111222333");
   assertTrue(nurse.getCC() = "123444654");
   assertTrue(nurse.getInfo() = "Name: " ^ "Lisete" ^ " " ^ "Antunes" ^ "\nAddress: " ^ "Rua de
       Baixo" ^ "\nPhone Number: " ^ "921378643" ^ "\nCC: " ^ "123444654");
   assertTrue(nurse.getType() = <Nurse>);

   IO`print("\n Getting technician informations \n");
   assertTrue(technician.getMedicalNumber() = "123432578");
   assertTrue(technician.getCC() = "123432151");
   assertTrue(technician.getInfo() = "Name: " ^ "Ins" ^ " " ^ "Pinto" ^ "\nAddress: " ^ "Rua
       Antero Marques" ^ "\nPhone Number: " ^ "921348765" ^ "\nCC: " ^ "123432151");
   assertTrue(technician.getType() = <Technician>);
```

```
 );


public testAddRemovePatient : () ==> ()
 testAddRemovePatient() == (
  IO`print("\n Number of patients: ");
  IO`print(card doctor.getPatients());
  assertTrue(card doctor.getPatients() = 0);

  IO`print("\n Adding a patient \n");
  doctor.addPatient(patient);

  IO`print("\n Number of patients: ");
  IO`print(card doctor.getPatients());
  assertTrue(card doctor.getPatients() = 1);

  IO`print("\nRemoving a patient \n");
  doctor.removePatient(patient);

  IO`print("\n Number of patients: ");
  IO`print(card doctor.getPatients());
  assertTrue(card doctor.getPatients() = 0);

  IO`print("\n Adding a patient \n");
  assertTrue(card surgeon.getPatients() = 0);

  surgeon.addPatient(patient);
  IO`print("\n Number of patients: ");
  IO`print(card surgeon.getPatients());
  assertTrue(card surgeon.getPatients() = 1);
 );


public testAddRemoveSpecialty : () ==> ()
 testAddRemoveSpecialty() == (
  dcl specialty1: Specialty := new Specialty("General"), specialty2: Specialty := new Specialty(
      "Cardio");

  IO`print("\n Number of specialties: ");
  IO`print(card doctor.getSpecialties());
  assertTrue(card doctor.getSpecialties() = 0);

  IO`print("\n Adding a specialty \n");
  doctor.addSpecialty(specialty1);

  IO`print("\n Number of specialties: ");
  IO`print(card doctor.getSpecialties());

  assertTrue(specialty1.getName() = "General");
  assertTrue(card doctor.getSpecialties() = 1);
  assertTrue(doctor.getSpecialties() = {specialty1});

  IO`print("\n Adding a specialty \n");
  doctor.addSpecialty(specialty2);

  IO`print("\n Number of specialties: ");
  IO`print(card doctor.getSpecialties());

  assertTrue(specialty2.getName() = "Cardio");
  assertTrue(card doctor.getSpecialties() = 2);
  assertTrue(doctor.getSpecialties() = {specialty1, specialty2});

  IO`print("\n Removing a specialty \n");
  doctor.removeSpecialty(specialty1);
```

```
   IO`print("\n Number of specialties: ");
   IO`print(card doctor.getSpecialties());

   assertTrue(card doctor.getSpecialties() = 1);
   assertTrue(doctor.getSpecialties() = {specialty2});
  );


 public static main: () ==> ()
   main() == (
    dcl personTest: PersonTest := new PersonTest();
    IO`print("\n *****Running PersonTest***** \n");
    personTest.testGetInformation();
    personTest.testAddRemovePatient();
    personTest.testAddRemoveSpecialty();
   );

end PersonTest
```

| Function or operation | Line | Coverage | Calls |
|---|---|---|---|
| assertTrue | 9 | 100.0% | 1980 |
| main | 112 | 100.0% | 30 |
| testAddRemovePatient | 45 | 100.0% | 30 |
| testAddRemoveSpecialty | 74 | 100.0% | 30 |
| testGetInformation | 13 | 100.0% | 30 |
| PersonTest.vdmpp | | 100.0% | 2100 |

# 18 RunTests

```
class RunTests

operations

 public static main: () ==> ()
   main() == (
    dcl taskTest: TaskTest := new TaskTest(), personTest: PersonTest := new PersonTest(),
     trainingTest: TrainingTest := new TrainingTest(), safetyNetTest: SafetyNetHospitalTest :=
         new SafetyNetHospitalTest();

    personTest.main();
    taskTest.main();
    trainingTest.main();
    safetyNetTest.main();
   );
end RunTests
```

| Function or operation | Line | Coverage | Calls |
|---|---|---|---|
| main | 4 | 100.0% | 30 |
| RunTests.vdmpp | | 100.0% | 30 |

# 19 SafetyNetHospitalTest

```
class SafetyNetHospitalTest
instance variables
 private safetyNet: SafetyNetHospital := new SafetyNetHospital();

 private time1: Types`Time := mk_Types`Time(12, 10);
 private date1: Types`Date := mk_Types`Date(2017, 12, 25, time1);
 private time2: Types`Time := mk_Types`Time(12, 30);
 private date2: Types`Date := mk_Types`Date(2017, 12, 25, time2);
 private schedule: Schedule := new Schedule(date1, date2);

 private time3: Types`Time := mk_Types`Time(12, 15);
 private date3: Types`Date := mk_Types`Date(2017, 12, 25, time3);
 private time4: Types`Time := mk_Types`Time(12, 35);
 private date4: Types`Date := mk_Types`Date(2017, 12, 25, time4);
 private schedule2: Schedule := new Schedule(date3, date4);

 private time5: Types`Time := mk_Types`Time(12, 40);
 private date5: Types`Date := mk_Types`Date(2017, 12, 25, time5);
 private time6: Types`Time := mk_Types`Time(12, 50);
 private date6: Types`Date := mk_Types`Date(2017, 12, 25, time6);
 private schedule3: Schedule := new Schedule(date5, date6);

 private time7: Types`Time := mk_Types`Time(12, 10);
 private date7: Types`Date := mk_Types`Date(2017, 11, 22, time7);
 private time8: Types`Time := mk_Types`Time(12, 30);
 private date8: Types`Date := mk_Types`Date(2017, 11, 22, time8);
 private schedule4: Schedule := new Schedule(date7, date8);

 private time9: Types`Time := mk_Types`Time(12, 35);
 private date9: Types`Date := mk_Types`Date(2017, 11, 23, time9);
 private time10: Types`Time := mk_Types`Time(12, 45);
 private date10: Types`Date := mk_Types`Date(2017, 11, 23, time10);
 private schedule5: Schedule := new Schedule(date9, date10);

 private patient: Patient := new Patient("Rua 1 Maio", "Rui", "Andrade", "123456789", "223456111"
     , "0987654321");
 private patient2: Patient := new Patient("Rua 1 Maio", "Diogo", "Andrade", "123321123", "
     911112345", "908765123");
 private patient3: Patient := new Patient("Rua 1 Maio", "Vitor", "Andrade", "135790864", "
     912345334", "123432130");
 private patient4: Patient := new Patient("Rua 1 Maio", "Simone", "Andrade", "234123765", "
     931238654", "0987654143");

  private hospital: Hospital := new Hospital("Hospital das Camlias", "Rua de Cima", safetyNet);

  private doctor: HealthProfessional := new HealthProfessional("Rua de Cima", "Ana", "Marques", "
      123432156", "921349076", "111111222", <Doctor>);
  private doctor2: HealthProfessional := new HealthProfessional("Rua de Cima", "Anabela", "
      Marques", "123432157", "921349077", "111111223", <Doctor>);
  private surgeon: HealthProfessional := new HealthProfessional("Rua 2", "Diogo", "Viana", "
      234512389", "921349134", "111111232", <Surgeon>);
  private secSurgeon: HealthProfessional := new HealthProfessional("Rua 2", "Diana", "Viana", "
      234512390", "921349135", "111111235", <Surgeon>);
 private nurse: HealthProfessional := new HealthProfessional("Rua de Baixo", "Lisete", "Antunes",
      "123444654", "921378643", "111222333", <Nurse>);
 private technician: HealthProfessional := new HealthProfessional("Rua de Baixo", "Lu s", "
     Antunes", "123444655", "921377654", "111222345", <Technician>);

 private appointment: Appointment := new Appointment(doctor, schedule, patient, hospital);
 private appointment2: Appointment := new Appointment(doctor, schedule3, patient4, hospital);
 private appointment3: Appointment := new Appointment(doctor2, schedule3, patient, hospital);
```

```
 private urgencies: Appointment := new Appointment(doctor2, <High>, schedule, patient2, hospital)
     ;
 private surgery: Surgery := new Surgery(surgeon, schedule, patient3, hospital);
 private treatment: Treatment := new Treatment(technician, "Fisioterapia", schedule, patient4,
     hospital);

 private purpose: Types`Purpose := <Training>;
 private training : Training := new Training(purpose, schedule3, nurse);
operations

 private assertTrue: bool ==> ()
  assertTrue(cond) == return
 pre cond;


 public testAddRemoveHospitals: () ==> ()
  testAddRemoveHospitals() == (
      dcl h1: Hospital, h2: Hospital, h3: Hospital;
      h1 := new Hospital("Hospital dos Lusadas", "Rua de Cima", safetyNet);
      h2 := new Hospital("Hospital Novo", "Rua 1 de Maio", safetyNet);
      h3 := new Hospital("Hospital da Trofa", "Rua da Trofa", safetyNet);
      IO`print("\n Number of hospitals: ");
      IO`print(card safetyNet.getHospitals());

      IO`print("\n\n Getting hospitals information \n");
      assertTrue(h1.getName() = "Hospital dos Lusadas");
      assertTrue(h2.getName() = "Hospital Novo");
      assertTrue(h3.getName() = "Hospital da Trofa");

      assertTrue(h1.getAddress() = "Rua de Cima");
      assertTrue(h2.getAddress() = "Rua 1 de Maio");
      assertTrue(h3.getAddress() = "Rua da Trofa");

      IO`print("\n Removing hospitals \n");
      assertTrue(card safetyNet.getHospitals() = 4);
      safetyNet.removeHospital(h1);

      IO`print("\n Removing hospitals \n");
      assertTrue(card safetyNet.getHospitals() = 3);
      safetyNet.removeHospital(h2);
      assertTrue(card safetyNet.getHospitals() = 2);

      IO`print("\n Number of hospitals: ");
      IO`print(card safetyNet.getHospitals());
 );


 public testAddRemoveMedHospital : () ==> ()
 testAddRemoveMedHospital() == (
    dcl agenda1 : Agenda, agenda2 : Agenda, agenda3 : Agenda, agenda4: Agenda, agenda5: Agenda;

    IO`print("\n Adding health professionals \n");
    hospital.addMedAssociated(doctor);
    hospital.addMedAssociated(doctor2);
    hospital.addMedAssociated(surgeon);
    hospital.addMedAssociated(nurse);
    hospital.addMedAssociated(technician);

   for all a in set hospital.getAgendas() do(
    if(a.getHealthProfessional() = doctor)
     then agenda1 := a
    else if(a.getHealthProfessional() = doctor2)
     then agenda2 := a
    else if(a.getHealthProfessional() = surgeon)
     then agenda3 := a
```

```
    else if(a.getHealthProfessional() = nurse)
     then agenda4 := a
    else
     agenda5 := a;);

     agenda1.addSchedule(schedule);
     agenda1.addSchedule(schedule3);

     agenda2.addSchedule(schedule);
     agenda2.addSchedule(schedule3);

     agenda3.addSchedule(schedule);

     agenda4.addSchedule(schedule3);

     agenda5.addSchedule(schedule);

     assertTrue(card agenda1.getAgenda() = 2);
     assertTrue(card agenda2.getAgenda() = 2);
     assertTrue(card agenda3.getAgenda() = 1);
     assertTrue(card agenda4.getAgenda() = 1);
     assertTrue(card agenda5.getAgenda() = 1);

     IO`print("\n Total number of doctors: ");
     IO`print(card hospital.getMedicalAssociatedByType(<Doctor>));
     IO`print("\n Total number of surgeons: ");
     IO`print(card hospital.getMedicalAssociatedByType(<Surgeon>));
     IO`print("\n Total number of nurses: ");
     IO`print(card hospital.getMedicalAssociatedByType(<Nurse>));
     IO`print("\n Total number of technicians: ");
     IO`print(card hospital.getMedicalAssociatedByType(<Technician>));

     assertTrue(card hospital.getMedicalAssociatedByType(<Doctor>) = 2);
     assertTrue(card hospital.getMedicalAssociatedByType(<Surgeon>) = 1);
     assertTrue(card hospital.getMedicalAssociatedByType(<Nurse>) = 1);
     assertTrue(card hospital.getMedicalAssociatedByType(<Technician>) = 1);

     IO`print("\n Total number of doctors: ");
     IO`print(card hospital.getMedicalAssociatedByType(<Doctor>));

     assertTrue(card hospital.getMedicalAssociatedByType(<Doctor>) = 2);

     IO`print("\n Removing a doctor \n");
     hospital.removeMedAssociated(doctor);
     assertTrue(card hospital.getMedicalAssociatedByType(<Doctor>) = 1);

     IO`print("\n Total number of doctors: ");
     IO`print(card hospital.getMedicalAssociatedByType(<Doctor>));

     hospital.addMedAssociated(doctor);

     for all a in set hospital.getAgendas() do
      if(a.getHealthProfessional().getCC() = doctor.getCC())
       then agenda1 := a;

     agenda1.addSchedule(schedule);
     agenda1.addSchedule(schedule3);
     assertTrue(card agenda1.getAgenda() = 2);

);


   public testAddRemoveTaskHospital : () ==> ()
   testAddRemoveTaskHospital() == (
     hospital.addTask(appointment);
```

```
    hospital.addTask(appointment2);
    hospital.addTask(appointment3);
    hospital.addTask(urgencies);
    hospital.addTask(surgery);
    hospital.addTask(treatment);

    IO`print("\n\n Total number of appointments: ");
    IO`print(card hospital.getTasksByType(<Appointment>));
    IO`print("\n Total number of urgencies: ");
    IO`print(card hospital.getTasksByType(<Urgencies>));
    IO`print("\n Total number of surgeries: ");
    IO`print(card hospital.getTasksByType(<Surgery>));
    IO`print("\n Total number of other treatments: ");
    IO`print(card hospital.getTasksByType(<Other>));

    assertTrue(card hospital.getTasksByType(<Appointment>) = 3);
    assertTrue(card hospital.getTasksByType(<Urgencies>) = 1);
    assertTrue(card hospital.getTasksByType(<Surgery>) = 1);
    assertTrue(card hospital.getTasksByType(<Other>) = 1);

    IO`print("\n Removing an appointment \n");
    hospital.removeTask(appointment);
    assertTrue(card hospital.getTasksByType(<Appointment>) = 2);

    IO`print("\n Total number of appointments: ");
    IO`print(card hospital.getTasksByType(<Appointment>));

    IO`print("\n Adding an appointment \n");
    hospital.addTask(appointment);
    assertTrue(card hospital.getTasksByType(<Appointment>) = 3);

    IO`print("\n Total number of appointments: ");
    IO`print(card hospital.getTasksByType(<Appointment>));
);


public testAddRemoveTrainingHospital : () ==> ()
 testAddRemoveTrainingHospital() == (
  IO`print("\n\n Total number of trainings: ");
  IO`print(card hospital.getTrainingsByType(<Training>) + card hospital.getTrainingsByType(<
      AddSkills>));

  assertTrue(card hospital.getTrainingsByType(<Training>) = 0);
  assertTrue(card hospital.getTrainingsByType(<AddSkills>) = 0);

  IO`print("\n Adding a training \n");
  hospital.addTraining(training);
  assertTrue(card hospital.getTrainingsByType(<Training>) = 1);

  IO`print("\n Total number of trainings: ");
  IO`print(card hospital.getTrainingsByType(<Training>) + card hospital.getTrainingsByType(<
      AddSkills>));

  IO`print("\n Removing a training \n");
  hospital.removeTraining(training);
  assertTrue(card hospital.getTrainingsByType(<Training>) = 0);

  IO`print("\n\n Total number of trainings: ");
  IO`print(card hospital.getTrainingsByType(<Training>) + card hospital.getTrainingsByType(<
      AddSkills>));
);


public testGetHospitalsMoreAppointments : () ==> ()
 testGetHospitalsMoreAppointments() == (
```

```
  IO`print("\n Checking Safety Net Hospitals with more appointments, etc \n");
  assertTrue(safetyNet.getHospitalsMoreAppointments(<Appointment>).getName() = "Hospital das
      Camlias");
  assertTrue(safetyNet.getHospitalsMoreAppointments(<Urgencies>).getName() = "Hospital das
      Camlias");
  assertTrue(safetyNet.getHospitalsMoreAppointments(<Surgery>).getName() = "Hospital das
      Camlias");
  assertTrue(safetyNet.getHospitalsMoreAppointments(<Other>).getName() = "Hospital das Camlias
      ");
 );


public testGetMedMoreHospitals : () ==> ()
 testGetMedMoreHospitals() == (
  for all t in set safetyNet.getHospitals() do
   if(t.getName() <> "Hospital das Camlias")
    then t.addMedAssociated(doctor);

  IO`print("\n Checking Safety Net Doctors that works in more than 1 hospital \n");
  IO`print("\n Number of Doctors: ");
  IO`print(card safetyNet.getMedMoreHospitals(<Doctor>));
  assertTrue(card safetyNet.getMedMoreHospitals(<Doctor>) = 1);
  assertTrue(safetyNet.getMedMoreHospitals(<Doctor>) = {doctor});
 );


public testGetMedAssociatedByPatient : () ==> ()
 testGetMedAssociatedByPatient() == (
  dcl mapTest : map Hospital to set of (HealthProfessional);
  IO`print("\n\n Getting Doctors associated by patient by hospital \n");
  mapTest := safetyNet.getMedAssociatedByPatient(patient, <Doctor>);

  assertTrue(card mapTest(hospital) = 2);
  assertTrue(mapTest(hospital) = {doctor, doctor2});
 );


public testGetMedByHospital : () ==> ()
 testGetMedByHospital() == (
  dcl mapTest : map Hospital to set of (HealthProfessional);
  IO`print("\n\n Getting Doctors associated by hospital \n");
  mapTest := safetyNet.getMedByHospital(<Doctor>);

  assertTrue(card mapTest(hospital) = 2);
  assertTrue(mapTest(hospital) = {doctor, doctor2});

  mapTest := safetyNet.getMedByHospital(<Surgeon>);

  assertTrue(card mapTest(hospital) = 1);
  assertTrue(mapTest(hospital) = {surgeon});
 );


public static main: () ==> ()
 main() == (
  dcl safetyNetTest: SafetyNetHospitalTest := new SafetyNetHospitalTest();
  IO`print("\n *****Running SafetyNetHospitalTest***** \n");
  safetyNetTest.testAddRemoveHospitals();
  safetyNetTest.testAddRemoveMedHospital();
  safetyNetTest.testAddRemoveTaskHospital();
  safetyNetTest.testAddRemoveTrainingHospital();
  safetyNetTest.testGetHospitalsMoreAppointments();
  safetyNetTest.testGetMedMoreHospitals();
  safetyNetTest.testGetMedAssociatedByPatient();
  safetyNetTest.testGetMedByHospital();
```

```
    );

end SafetyNetHospitalTest
```

| Function or operation | Line | Coverage | Calls |
|---|---|---|---|
| assertTrue | 59 | 100.0% | 714 |
| main | 281 | 100.0% | 4 |
| testAddRemoveHospitals | 63 | 100.0% | 17 |
| testAddRemoveMedHospital | 94 | 100.0% | 3 |
| testAddRemoveTaskHospital | 173 | 100.0% | 8 |
| testAddRemoveTrainingHospital | 211 | 100.0% | 3 |
| testGetHospitalsMoreAppointments | 234 | 100.0% | 3 |
| testGetMedAssociatedByPatient | 256 | 100.0% | 6 |
| testGetMedByHospital | 266 | 100.0% | 1 |
| testGetMedMoreHospitals | 243 | 100.0% | 18 |
| SafetyNetHospitalTest.vdmpp | | 100.0% | 777 |

# 20 TaskTest

```
class TaskTest

instance variables
 private safetyNet: SafetyNetHospital := new SafetyNetHospital();

 private time1: Types`Time := mk_Types`Time(12, 10);
 private date: Types`Date := mk_Types`Date(2017, 11, 25, time1);
 private d: Types`Date := mk_Types`Date(2017, 2, 25, time1);
 private date1: Types`Date := mk_Types`Date(2017, 12, 25, time1);
 private time2: Types`Time := mk_Types`Time(12, 30);
 private date2: Types`Date := mk_Types`Date(2017, 12, 25, time2);
 private schedule: Schedule := new Schedule(date1, date2);

 private time3: Types`Time := mk_Types`Time(12, 15);
 private date3: Types`Date := mk_Types`Date(2017, 12, 25, time3);
 private time4: Types`Time := mk_Types`Time(12, 35);
 private date4: Types`Date := mk_Types`Date(2017, 12, 25, time4);
 private schedule2: Schedule := new Schedule(date3, date4);

 private time5: Types`Time := mk_Types`Time(12, 40);
 private date5: Types`Date := mk_Types`Date(2017, 12, 25, time5);
 private time6: Types`Time := mk_Types`Time(12, 50);
 private date6: Types`Date := mk_Types`Date(2017, 12, 25, time6);
 private schedule3: Schedule := new Schedule(date5, date6);

 private time7: Types`Time := mk_Types`Time(12, 10);
 private date7: Types`Date := mk_Types`Date(2018, 11, 22, time7);
 private time8: Types`Time := mk_Types`Time(12, 30);
 private date8: Types`Date := mk_Types`Date(2018, 11, 22, time8);
 private schedule4: Schedule := new Schedule(date7, date8);

 private time9: Types`Time := mk_Types`Time(12, 35);
 private date9: Types`Date := mk_Types`Date(2017, 11, 22, time9);
 private time10: Types`Time := mk_Types`Time(12, 45);
 private date10: Types`Date := mk_Types`Date(2017, 11, 22, time10);
```

```
 private schedule5: Schedule := new Schedule(date9, date10);

 private patient: Patient := new Patient("Rua 1 Maio", "Rui", "Andrade", "123456789", "223456111"
     , "0987654321");
 private patient2: Patient := new Patient("Rua 1 Maio", "Diogo", "Andrade", "123321123", "
     911112345", "908765123");
 private patient3: Patient := new Patient("Rua 1 Maio", "Vitor", "Andrade", "135790864", "
     912345334", "123432130");
 private patient4: Patient := new Patient("Rua 1 Maio", "Simone", "Andrade", "234123765", "
     931238654", "0987654143");

  private hospital: Hospital := new Hospital("Hospital dos Lusadas", "Rua de Cima", safetyNet);
  private doctor: HealthProfessional := new HealthProfessional("Rua de Cima", "Ana", "Marques", "
      123432156", "921349076", "111111222", <Doctor>);
  private doctor2: HealthProfessional := new HealthProfessional("Rua de Cima", "Anabela", "
      Marques", "123432157", "921349077", "111111223", <Doctor>);
  private surgeon: HealthProfessional := new HealthProfessional("Rua 2", "Diogo", "Viana", "
      234512389", "921349134", "111111232", <Surgeon>);
  private secSurgeon: HealthProfessional := new HealthProfessional("Rua 2", "Diana", "Viana", "
      234512390", "921349135", "111111235", <Surgeon>);
 private nurse: HealthProfessional := new HealthProfessional("Rua de Baixo", "Lisete", "Antunes",
      "123444654", "921378643", "111222333", <Nurse>);
 private technician: HealthProfessional := new HealthProfessional("Rua de Baixo", "Lus", "
     Antunes", "123444655", "921377654", "111222345", <Technician>);

 private appointment: Appointment := new Appointment(doctor, schedule, patient, hospital);
 private urgencies: Appointment := new Appointment(doctor2, <High>, schedule, patient2, hospital)
     ;
 private surgery: Surgery := new Surgery(surgeon, schedule3, patient3, hospital);
 private treatment: Treatment := new Treatment(technician, "Fisioterapia", schedule, patient4,
     hospital);

 private medicament: Medicament := new Medicament("Brufen");
 private prescription: Prescription := new Prescription("123");
operations

 private assertTrue: bool ==> ()
  assertTrue(cond) == return
 pre cond;


 public testGetsSetsTask : () ==> ()
  testGetsSetsTask() == (
   dcl agenda1 : Agenda, agenda2 : Agenda, agenda3 : Agenda, agenda4: Agenda;
   hospital.addMedAssociated(doctor);
   hospital.addMedAssociated(doctor2);
   hospital.addMedAssociated(surgeon);
   hospital.addMedAssociated(technician);

   for all a in set hospital.getAgendas() do(
    if(a.getHealthProfessional() = doctor)
     then agenda1 := a
    else if(a.getHealthProfessional() = doctor2)
     then agenda2 := a
    else if(a.getHealthProfessional() = surgeon)
     then agenda3 := a
    else
     agenda4 := a;);

   agenda1.addSchedule(schedule);

   assertTrue(agenda1.getHealthProfessional().getCC() = doctor.getCC());

   assertTrue(card agenda1.getAgenda() = 1);
```

```
agenda2.addSchedule(schedule);

assertTrue(card agenda2.getAgenda() = 1);

agenda3.addSchedule(schedule3);

assertTrue(card agenda3.getAgenda() = 1);

agenda4.addSchedule(schedule);

assertTrue(card agenda4.getAgenda() = 1);

agenda4.removeSchedule(schedule);

assertTrue(card agenda4.getAgenda() = 0);

agenda4.addSchedule(schedule);

assertTrue(card agenda4.getAgenda() = 1);

hospital.addTask(appointment);
hospital.addTask(urgencies);
hospital.addTask(surgery);
hospital.addTask(treatment);

IO`print("\n Getting appointment informations \n");
assertTrue(appointment.getPatient().getCC() = "123456789");
assertTrue(appointment.getHospital().getName() = "Hospital dos Lusadas");

assertTrue(appointment.getType() = <Appointment>);
assertTrue(urgencies.getType() = <Urgencies>);
assertTrue(surgery.getType() = <Surgery>);
assertTrue(treatment.getType() = <Other>);

IO`print("\n Getting tasks informations \n");
assertTrue(appointment.getMedAssoc().getCC() = "123432156");
assertTrue(card appointment.getSurgeryPersons(<Nurse>) = 0);
assertTrue(urgencies.getMedAssoc().getCC() = "123432157");
assertTrue(surgery.getMedAssoc().getCC() = "234512389");

IO`print("\n Checking schedules \n");
assertTrue(appointment.getSchedule().getScheduleStart().year = 2017);
assertTrue(appointment.getSchedule().getScheduleStart().month = 12);
assertTrue(appointment.getSchedule().getScheduleStart().day = 25);
assertTrue(appointment.getSchedule().getScheduleStart().time.hour = 12);
assertTrue(appointment.getSchedule().getScheduleStart().time.min = 10);

assertTrue(appointment.getSchedule().getScheduleEnd().year = 2017);
assertTrue(appointment.getSchedule().getScheduleEnd().month = 12);
assertTrue(appointment.getSchedule().getScheduleEnd().day = 25);
assertTrue(appointment.getSchedule().getScheduleEnd().time.hour = 12);
assertTrue(appointment.getSchedule().getScheduleEnd().time.min = 30);

assertTrue(appointment.getSchedule().lessThan(appointment.getSchedule().getScheduleStart(),
    appointment.getSchedule().getScheduleEnd()) = true);

appointment.getSchedule().setSchedule(date3, date4);
assertTrue(appointment.getSchedule().getScheduleStart().year = 2017);
assertTrue(appointment.getSchedule().getScheduleStart().month = 12);
assertTrue(appointment.getSchedule().getScheduleStart().day = 25);
assertTrue(appointment.getSchedule().getScheduleStart().time.hour = 12);
assertTrue(appointment.getSchedule().getScheduleStart().time.min = 15);

assertTrue(appointment.getSchedule().getScheduleEnd().year = 2017);
assertTrue(appointment.getSchedule().getScheduleEnd().month = 12);
```

```
  assertTrue(appointment.getSchedule().getScheduleEnd().day = 25);
  assertTrue(appointment.getSchedule().getScheduleEnd().time.hour = 12);
  assertTrue(appointment.getSchedule().getScheduleEnd().time.min = 35);

  appointment.setSchedule(schedule2);

  assertTrue(appointment.getSchedule().getScheduleStart().year = 2017);
  assertTrue(appointment.getSchedule().getScheduleStart().month = 12);
  assertTrue(appointment.getSchedule().getScheduleStart().day = 25);
  assertTrue(appointment.getSchedule().getScheduleStart().time.hour = 12);
  assertTrue(appointment.getSchedule().getScheduleStart().time.min = 15);

  assertTrue(appointment.getSchedule().getScheduleEnd().year = 2017);
  assertTrue(appointment.getSchedule().getScheduleEnd().month = 12);
  assertTrue(appointment.getSchedule().getScheduleEnd().day = 25);
  assertTrue(appointment.getSchedule().getScheduleEnd().time.hour = 12);
  assertTrue(appointment.getSchedule().getScheduleEnd().time.min = 35);
 );


public testAppointment : () ==> ()
 testAppointment() == (
  IO`print("\n Checking appointment priority \n");
  assertTrue(appointment.getPriority() = <Medium>);
  assertTrue(urgencies.getPriority() = <High>);

  urgencies.setPriority(<Low>);
  assertTrue(urgencies.getPriority() = <Low>);

  IO`print("\n Checking appointment prescriptions \n");
  IO`print("\n Number of prescriptions: ");
  IO`print(card appointment.getPrescriptions() + card urgencies.getPrescriptions());
  assertTrue(card appointment.getPrescriptions() = 0);
  assertTrue(card urgencies.getPrescriptions() = 0);

  IO`print("\n\n Getting prescription code and medicament name \n");
  assertTrue(medicament.getName() = "Brufen");
  assertTrue(prescription.getCode() = "123");
  assertTrue(card prescription.getMedicaments() = 0);

  IO`print("\n Adding medicament \n");
  IO`print("\n Number of medicaments: ");
  IO`print(card prescription.getMedicaments());
  prescription.addMedicament(medicament);
  assertTrue(card prescription.getMedicaments() = 1);
  assertTrue(prescription.getMedicaments() = {medicament});

  IO`print("\n\n Removing medicament \n");
  IO`print("\n Number of medicaments: ");
  IO`print(card prescription.getMedicaments());
  prescription.removeMedicament(medicament);
  assertTrue(card prescription.getMedicaments() = 0);
  assertTrue(prescription.getMedicaments() = {});

  IO`print("\n Adding a prescription \n");
  IO`print("\n Number of prescriptions: ");
  IO`print(card appointment.getPrescriptions() + card urgencies.getPrescriptions());
  appointment.addPrescription(prescription);
  urgencies.addPrescription(prescription);
  assertTrue(card appointment.getPrescriptions() = 1);
  assertTrue(card urgencies.getPrescriptions() = 1);

  IO`print("\n\n Removing a prescription \n");
  IO`print("\n Number of prescriptions: ");
  IO`print(card appointment.getPrescriptions() + card urgencies.getPrescriptions());
```

```
  appointment.removePrescription(prescription);
  urgencies.removePrescription(prescription);
  assertTrue(card appointment.getPrescriptions() = 0);
  assertTrue(card urgencies.getPrescriptions() = 0);
 );


public testSurgery: () ==> ()
 testSurgery() == (
  IO`print("\n Checking surgery informations \n");
  assertTrue(card surgery.getSurgeryPersons(<Surgeon>) = 0);

  surgery.addSecondaryDoctor(secSurgeon);
  assertTrue(card surgery.getSurgeryPersons(<Surgeon>) = 1);

  surgery.removeSecondaryDoctor(secSurgeon);
  assertTrue(card surgery.getSurgeryPersons(<Surgeon>) = 0);

  assertTrue(card surgery.getSurgeryPersons(<Nurse>) = 0);
  surgery.addOther(nurse);
  assertTrue(card surgery.getSurgeryPersons(<Nurse>) = 1);

  surgery.removeOther(nurse);
  assertTrue(card surgery.getSurgeryPersons(<Nurse>) = 0);

  assertTrue(surgery.getMainDoctor().getCC() = "234512389");
  surgery.setMainDoctor(secSurgeon);
  assertTrue(surgery.getMainDoctor().getCC() = "234512390");
);


public testTreatment : () ==> ()
 testTreatment() == (
  IO`print("\n Checking treatment informations \n");
  IO`print("\n Checking schedule functions \n");
  assertTrue(treatment.getName() = "Fisioterapia");
  assertTrue(treatment.getMed().getCC() = "123444655");
 );


 public testScheduleFunctions: () ==> ()
 testScheduleFunctions() == (
  dcl sch : Schedule, sch1 : Schedule, dateNew: Types`Date, dateNew2 : Types`Date;
  dateNew := mk_Types`Date(2017, 10, 25, time1);
  dateNew2 := mk_Types`Date(2017, 10, 25, time2);
  sch := new Schedule(dateNew, dateNew2);

  dateNew := mk_Types`Date(2017, 10, 26, time1);
  dateNew2 := mk_Types`Date(2017, 10, 26, time2);
  sch1 := new Schedule(dateNew, dateNew2);

  IO`print("\n Checking schedule functions \n");
  assertTrue(appointment.getSchedule().lessThan(appointment.getSchedule().getScheduleStart(),
      appointment.getSchedule().getScheduleEnd()));
  assertTrue(appointment.getSchedule().greaterThan(appointment.getSchedule().getScheduleEnd(),
      appointment.getSchedule().getScheduleStart()));

  assertTrue(appointment.getSchedule().lessThan(appointment.getSchedule().getScheduleStart(),
      schedule4.getScheduleStart()));
  assertTrue(not(schedule4.lessThan(schedule4.getScheduleStart(), schedule5.getScheduleStart()))
      );
  assertTrue(sch.lessThan(sch.getScheduleStart(), schedule.getScheduleStart()));
  assertTrue(not(sch1.lessThan(sch1.getScheduleStart(), sch.getScheduleStart())));
```

```
  assertTrue(appointment.getSchedule().greaterThan(schedule4.getScheduleStart(), schedule5.
      getScheduleStart()));
  assertTrue(not(appointment.getSchedule().greaterThan(appointment.getSchedule().
      getScheduleStart(), schedule4.getScheduleStart())));
  assertTrue(not(sch.greaterThan(sch.getScheduleStart(), schedule.getScheduleStart())));
  assertTrue(sch1.greaterThan(sch1.getScheduleStart(), sch.getScheduleStart()));
  assertTrue(schedule.greaterThan(schedule.getScheduleStart(), sch.getScheduleStart()));
 );


 public static main: () ==> ()
  main() == (
   dcl taskTest: TaskTest := new TaskTest();
   IO`print("\n *****Running TaskTest***** \n");
   taskTest.testGetsSetsTask();
   taskTest.testAppointment();
   taskTest.testSurgery();
   taskTest.testTreatment();
   taskTest.testScheduleFunctions();
  );

end TaskTest
```

| Function or operation | Line | Coverage | Calls |
|---|---|---|---|
| assertTrue | 59 | 100.0% | 4136 |
| main | 279 | 100.0% | 18 |
| testAppointment | 170 | 100.0% | 18 |
| testGetsSetsTask | 63 | 100.0% | 18 |
| testScheduleFunctions | 252 | 100.0% | 18 |
| testSurgery | 221 | 100.0% | 18 |
| testTreatment | 244 | 100.0% | 18 |
| TaskTest.vdmpp | | 100.0% | 4244 |

# 21 TrainingTest

```
class TrainingTest
instance variables
 private doctor: HealthProfessional := new HealthProfessional("Rua de Cima", "Ana", "Marques", "
     123432156", "921349076", "111111222", <Doctor>);
  private purpose: Types`Purpose := <Training>;

 private time1: Types`Time := mk_Types`Time(12, 10);
 private date1: Types`Date := mk_Types`Date(2017, 12, 25, time1);
 private time2: Types`Time := mk_Types`Time(12, 30);
 private date2: Types`Date := mk_Types`Date(2017, 12, 25, time2);
 private schedule: Schedule := new Schedule(date1, date2);

 private time3: Types`Time := mk_Types`Time(12, 15);
 private date3: Types`Date := mk_Types`Date(2017, 12, 25, time3);
 private time4: Types`Time := mk_Types`Time(12, 35);
 private date4: Types`Date := mk_Types`Date(2017, 12, 25, time4);
 private schedule2: Schedule := new Schedule(date3, date4);

 private training : Training := new Training(purpose, schedule, doctor);
operations
```

```
 private assertTrue: bool ==> ()
  assertTrue(cond) == return
 pre cond;


 public testGetsSets : () ==> ()
  testGetsSets() == (
   IO`print("\n Testing Training gets and sets \n");
   assertTrue(training.getPurpose() = <Training>);
   assertTrue(training.getMedAssoc().getCC() = "123432156");

   training.setPurpose(<AddSkills>);
   assertTrue(training.getPurpose() = <AddSkills>);

   assertTrue(training.getSchedule().getScheduleStart().year = 2017);
   assertTrue(training.getSchedule().getScheduleStart().month = 12);
   assertTrue(training.getSchedule().getScheduleStart().day = 25);
   assertTrue(training.getSchedule().getScheduleStart().time.hour = 12);
   assertTrue(training.getSchedule().getScheduleStart().time.min = 10);

   assertTrue(training.getSchedule().getScheduleEnd().year = 2017);
   assertTrue(training.getSchedule().getScheduleEnd().month = 12);
   assertTrue(training.getSchedule().getScheduleEnd().day = 25);
   assertTrue(training.getSchedule().getScheduleEnd().time.hour = 12);
   assertTrue(training.getSchedule().getScheduleEnd().time.min = 30);

   training.setSchedule(schedule2);

   assertTrue(training.getSchedule().getScheduleStart().year = 2017);
   assertTrue(training.getSchedule().getScheduleStart().month = 12);
   assertTrue(training.getSchedule().getScheduleStart().day = 25);
   assertTrue(training.getSchedule().getScheduleStart().time.hour = 12);
   assertTrue(training.getSchedule().getScheduleStart().time.min = 15);

   assertTrue(training.getSchedule().getScheduleEnd().year = 2017);
   assertTrue(training.getSchedule().getScheduleEnd().month = 12);
   assertTrue(training.getSchedule().getScheduleEnd().day = 25);
   assertTrue(training.getSchedule().getScheduleEnd().time.hour = 12);
   assertTrue(training.getSchedule().getScheduleEnd().time.min = 35);
  );


 public static main: () ==> ()
   main() == (
    dcl trainingTest: TrainingTest := new TrainingTest();
    IO`print("\n *****Running TrainingTest***** \n");
    trainingTest.testGetsSets();
   );
end TrainingTest
```

| Function or operation | Line | Coverage | Calls |
|---|---|---|---|
| assertTrue | 21 | 100.0% | 552 |
| main | 61 | 100.0% | 12 |
| testGetsSets | 25 | 100.0% | 12 |
| TrainingTest.vdmpp | | 100.0% | 576 |