

Hokify

December 30, 2017

Contents

| | | |
|----------|-------------------|-----------|
| 1 | Hokify | 1 |
| 2 | MyTestCase | 5 |
| 3 | TestAll | 6 |
| 4 | Trabalho | 10 |
| 5 | Utilizador | 15 |

1 Hokify

```
class Hokify
types
-- TODO Define types here
public String = seq of char;
public Utilizadores = set of Utilizador;
public Trabalhos = set of Trabalho;
public Escolaridade = <Secundario> | <Licenciatura> | <Mestrado> | <Doutoramento>;

values
-- TODO Define values here
instance variables
-- TODO Define instance variables here
private utilizadores: set of Utilizador := {};
private trabalhos: set of Trabalho := {};

operations
-- TODO Define operations here

--Construtor

public Hokify: () ==> Hokify
Hokify() == (return self)
post utilizadores = {} and
    trabalhos = {};

--Adicionar Utilizadores

public addUtilizadores: Utilizador ==> ()
addUtilizadores(utilizador) == utilizadores := utilizadores union {utilizador}
pre sameUser(utilizador)
post utilizadores = utilizadores~ union {utilizador};
```

```

-- Remover Utilizadores

public removeUtilizadores: Utilizador ==> ()
removeUtilizadores(utilizador) == utilizadores := utilizadores \ {utilizador}
pre not sameUser(utilizador)
post utilizadores = utilizadores~ \ {utilizador};

--Adicionar Trabalhos

public addTrabalhos: Trabalho ==> ()
addTrabalhos(trabalho) == trabalhos := trabalhos union {trabalho}
pre sameTrabalho(trabalho)
post trabalhos = trabalhos~ union {trabalho};

-- Remover Trabalhos

public removeTrabalhos: Trabalho ==> ()
removeTrabalhos(trabalho) == trabalhos := trabalhos \ {trabalho}
pre not sameTrabalho(trabalho)
post trabalhos = trabalhos~ \ {trabalho};

-- Retorna os utilizadores

public pure getUtilizadores : () ==> set of Utilizador
getUtilizadores() ==
(
    return utilizadores;
);

-- Retorna os trabalhos

public pure getTrabalhos : () ==> set of Trabalho
getTrabalhos() ==
(
    return trabalhos;
);

-- Retorna os trabalhos por nome

public pure getTrabalhosPorNome: seq of char ==> set of Trabalho
getTrabalhosPorNome(nome) == (return {trabalhos | trabalhos in set trabalhos & trabalhos.
    nomeSemelhante(nome)})
pre len nome > 0;

--Retorna os trabalhos por interesses

public pure getTrabalhosPorInteresses: seq of char ==> set of Trabalho
getTrabalhosPorInteresses(nome) == (
    dcl results: set of Trabalho := {};
    for all tr in set trabalhos do
        if nome in set tr.getInteresses() then
            results := results union {tr};
    return results;
)
pre len nome > 0;

--Retorna os trabalhos por skills

public pure getTrabalhosPorSkills: seq of char ==> set of Trabalho
getTrabalhosPorSkills(nome) == (
    dcl results: set of Trabalho := {};
    for all tr in set trabalhos do
        if nome in set tr.getSkills() then
            results := results union {tr};

```

```

    return results;
)
pre len nome > 0;

--Retorna os trabalhos por Escolaridade

public pure getTrabalhosPorEscolaridade: Escolaridade ==> set of Trabalho
getTrabalhosPorEscolaridade(nome) == (
    dcl results: set of Trabalho := {};
    for all tr in set trabalhos do
        if nome in set tr.getlistaEscolaridades() then
            results := results union {tr};
    return results;
)
pre nome <> undefined;

--Retorna os trabalhos por Utilizador (Escolaridade, Skills, interesses)

public pure getTrabalhosPorUtilizador: Utilizador ==> set of Trabalho
getTrabalhosPorUtilizador(usr) == (
    dcl results_escolaridade: set of Trabalho := {};
    dcl results_skills: set of Trabalho := {};
    dcl results_interesses: set of Trabalho := {};
    dcl trabalhos_temp: set of Trabalho := {};
    results_escolaridade := getTrabalhosPorEscolaridade(usr.getEscolaridade());

    for all skill in set usr.getSkills() do
        trabalhos_temp := getTrabalhosPorSkills(skill);
        for all skill_temp in set trabalhos_temp do
            if skill_temp not in set results_skills then
                results_skills := results_skills union {skill_temp};

    for all interesse in set usr.getInteresses() do
        trabalhos_temp := getTrabalhosPorInteresses(interesse);
        for all interesse_temp in set trabalhos_temp do
            if interesse_temp not in set results_interesses then
                results_interesses := results_interesses union {interesse_temp};
    return (results_escolaridade inter results_skills inter results_interesses);
)
pre usr <> undefined;

--Retorna os trabalhos por pais

public pure getTrabalhosPorPais: seq of char ==> set of Trabalho
getTrabalhosPorPais(nome) == (
    dcl results: set of Trabalho := {};
    for all tr in set trabalhos do
        if nome = tr.getPais() then
            results := results union {tr};
    return results;
)
pre len nome > 0;

--Retorna os trabalhos por localidade

public pure getTrabalhosPorLocalidade: seq of char ==> set of Trabalho
getTrabalhosPorLocalidade(nome) == (
    dcl results: set of Trabalho := {};
    for all tr in set trabalhos do
        if nome = tr.getLocalidade() then
            results := results union {tr};
    return results;
)
pre len nome > 0;

```

```

--Retorna os utilizadres por interesses

public pure getUtilizadoresPorInteresses: seq of char ==> set of Utilizador
getUtilizadoresPorInteresses(nome) == (
  dcl results: set of Utilizador := {};
  for all tr in set utilizadores do
    if nome in set tr.getInteresses() then
      results := results union {tr};
  return results;
)
pre len nome > 0;

--Retorna os utilizadres por skills

public pure getUtilizadoresPorSkills: seq of char ==> set of Utilizador
getUtilizadoresPorSkills(nome) == (
  dcl results: set of Utilizador := {};
  for all tr in set utilizadores do
    if nome in set tr.getSkills() then
      results := results union {tr};
  return results;
)
pre len nome > 0;

--Retorna os utilizadres por Escolaridade

public pure getUtilizadoresPorEscolaridade: Escolaridade ==> set of Utilizador
getUtilizadoresPorEscolaridade(nome) == (
  dcl results: set of Utilizador := {};
  for all tr in set utilizadores do
    if nome in set tr.getlistaEscolaridades() then
      results := results union {tr};
  return results;
)
pre nome <> undefined;

--Retorna os utilizadores por Trabalhos (Escolaridade, Skills, interesses)
public pure getTrabalhosPorUtilizador: Trabalho ==> set of Utilizador
getTrabalhosPorUtilizador(trab) == (
  dcl results_escolaridade: set of Utilizador := {};
  dcl results_skills: set of Utilizador := {};
  dcl results_interesses: set of Utilizador := {};
  dcl utilizadores_temp: set of Utilizador := {};
  results_escolaridade := getUtilizadoresPorEscolaridade(trab.getEscolaridade());

  for all skill in set trab.getSkills() do
    utilizadores_temp := getUtilizadoresPorSkills(skill);
    for all skill_temp in set utilizadores_temp do
      if skill_temp not in set results_skills then
        results_skills := results_skills union {skill_temp};

  for all interesse in set trab.getInteresses() do
    utilizadores_temp := getUtilizadoresPorInteresses(interesse);
    for all interesse_temp in set utilizadores_temp do
      if interesse_temp not in set results_interesses then
        results_interesses := results_interesses union {interesse_temp};

  return (results_escolaridade inter results_skills inter results_interesses);
)
pre trab <> undefined;

-- Verifica se o utilizador existe por email ou telefone

public pure sameUser: Utilizador ==> bool
sameUser(user) ==(

```

```

for all u in set utilizadores do
  if (u.getEmail() = user.getEmail() or
    u.getTelefone() = user.getTelefone()) then
    return false;
  return true;
)
pre user <> undefined;

-- Verifica se o trabalho existe por email ou nome ou entidade

public pure sameTrabalho: Trabalho ==> bool
sameTrabalho(trab) == (
  for all u in set trabalhos do
    if (u.getEmail() = trab.getEmail() or
      u.getNome() = trab.getNome() or
      u.getEntidade() = trab.getEntidade()) then
      return false;
    return true;
)
pre trab <> undefined;

functions
-- TODO Define functiones here
traces
-- TODO Define Combinatorial Test Traces here
end Hokify

```

| Function or operation | Line | Coverage | Calls |
|--------------------------------|------|----------|-------|
| Hokify | 20 | 100.0% | 1 |
| addTrabalhos | 39 | 100.0% | 3 |
| addUtilizadores | 27 | 100.0% | 3 |
| getTrabalhos | 58 | 100.0% | 4 |
| getTrabalhosPorEscolaridade | 92 | 100.0% | 6 |
| getTrabalhosPorInteresses | 70 | 100.0% | 7 |
| getTrabalhosPorLocalidade | 138 | 100.0% | 2 |
| getTrabalhosPorNome | 65 | 100.0% | 3 |
| getTrabalhosPorPais | 127 | 100.0% | 3 |
| getTrabalhosPorSkills | 81 | 100.0% | 7 |
| getTrabalhosPorUtilizador | 103 | 100.0% | 2 |
| getUtilizadores | 51 | 100.0% | 4 |
| getUtilizadoresPorEscolaridade | 171 | 100.0% | 6 |
| getUtilizadoresPorInteresses | 149 | 100.0% | 7 |
| getUtilizadoresPorSkills | 160 | 100.0% | 7 |
| removeTrabalhos | 45 | 100.0% | 1 |
| removeUtilizadores | 33 | 100.0% | 1 |
| sameTrabalho | 218 | 100.0% | 5 |
| sameUser | 207 | 100.0% | 2 |
| Hokify.vdmpp | | 100.0% | 74 |

2 MyTestCase

```

class MyTestCase
/*
  Superclass for test classes, simpler but more practical than VDMUnit`TestCase.
  For proper use, you have to do: New -> Add VDM Library -> IO.
  JPF, FEUP, MFES, 2014/15.
*/

operations

-- Simulates assertion checking by reducing it to pre-condition checking.
-- If 'arg' does not hold, a pre-condition violation will be signaled.

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

-- Simulates assertion checking by reducing it to post-condition checking.
-- If values are not equal, prints a message in the console and generates
-- a post-conditions violation.

protected assertEquals: ? * ? ==> ()
assertEquals(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 |
|-----------------------|------|----------|-------|
| assertEquals | 20 | 38.8% | 0 |
| assertTrue | 12 | 100.0% | 7 |
| MyTestCase.vdmpp | | 45.0% | 7 |

3 TestAll

```

class TestAll is subclass of MyTestCase
types
-- TODO Define types here
values
-- TODO Define values here
instance variables
-- TODO Define instance variables here
operations
-- TODO Define operations here

public TestPlayerFirst :() ==> ()
TestPlayerFirst() ==
(
  decl user : Utilizador := new Utilizador("Pedro", "email@email.com", <Masculino>, <Mestrado>, "
    Portugal", "Porto", 27, 123456789, 1990, 09, 08);

```

```

user.addInteresse("Informatica");
user.addInteresse("Cinema");
user.addSkills("Java");
user.addSkills("VDM");

-- Testes simples de verificacao dos dados do utiliza o - Inserir, editar e eliminar.
assertEquals(user.getNome(), "Pedro");
assertEquals(user.getEmail(), "email@email.com");
assertEquals(user.getSexo(), <Masculino>);
assertEquals(user.getEscolaridade(), <Mestrado>);
assertEquals(user.getPais(), "Portugal");
assertEquals(user.getLocalidade(), "Porto");
assertEquals(user.getIdade(), 27);
assertEquals(user.getTelefone(), 123456789);
assertEquals(user.getInteresses(), {"Informatica", "Cinema"});
assertEquals(user.getSkills(), {"Java", "VDM"});
assertEquals(user.getDataNascimento(), mk_Utilizador.Date(1990, 09, 08));

user.removeInteresse("Cinema");
assertEquals(user.getInteresses(), {"Informatica"});

user.removeSkills("VDM");
assertEquals(user.getSkills(), {"Java"});

user.setNome("Pedro Faria");
user.setEmail("PedroFaria@gmail.com");
user.setPais("Espanha");
user.setLocalidade("Madrid");
user.setEscolaridade(<Secundario>);
user.setTelefone(987654321);
assertEquals(user.getNome(), "Pedro Faria");
assertEquals(user.getEmail(), "PedroFaria@gmail.com");
assertEquals(user.getPais(), "Espanha");
assertEquals(user.getLocalidade(), "Madrid");
assertEquals(user.getEscolaridade(), <Secundario>);
assertEquals(user.getTelefone(), 987654321);

--deve falhar pois estamos adicionando um duplicado
--user.addInteresse("Informatica");
--user.addSkills("Java");

--Verifica a escolaridade de um curso
assertTrue(not (user.verificarEscolaridade(<Secundario>, 10)));
return;
);

public TestTrabalhoFirst :() ==> ()
TestTrabalhoFirst() ==
(
    ddl trabalho : Trabalho := new Trabalho("Programador de java", "Google", "google@google.pt", <
        Mestrado>, 123456789, "Portugal", "Porto");
    trabalho.addInteresse("Informatica");
    trabalho.addInteresse("Cinema");
    trabalho.addSkills("Java");
    trabalho.addSkills("VDM");

    -- Testes simples de verificacao dos dados do trabalho - Inserir, editar e eliminar.
    assertEquals(trabalho.getNome(), "Programador de java");
    assertEquals(trabalho.getEntidade(), "Google");
    assertEquals(trabalho.getEmail(), "google@google.pt");
    assertEquals(trabalho.getEscolaridade(), <Mestrado>);
    assertEquals(trabalho.getTelefone(), 123456789);
    assertEquals(trabalho.getPais(), "Portugal");

```

```

assertEqual(trabalho.getLocalidade(), "Porto");
assertEqual(trabalho.getInteresses(), {"Informatica", "Cinema"});
assertEqual(trabalho.getSkills(), {"Java", "VDM"});

trabalho.removeInteresse("Cinema");
assertEqual(trabalho.getInteresses(), {"Informatica"});

trabalho.removeSkills("VDM");
assertEqual(trabalho.getSkills(), {"Java"});

trabalho.setNome("Programador de C++");
trabalho.setEntidade("apple");
trabalho.setEmail("apple@apple.pt");
trabalho.setPais("Espanha");
trabalho.setLocalidade("Madrid");
trabalho.setTelefone(987654321);
trabalho.setEscolaridade(<Secundario>);
assertEqual(trabalho.getNome(), "Programador de C++");
assertEqual(trabalho.getEmail(), "apple@apple.pt");
assertEqual(trabalho.getPais(), "Espanha");
assertEqual(trabalho.getLocalidade(), "Madrid");
assertEqual(trabalho.getTelefone(), 987654321);
assertEqual(trabalho.getEscolaridade(), <Secundario>);

return;
);

public TestHokify :() ==> ()
TestHokify() ==
(
dcl hokify : Hokify := new Hokify();
dcl utilizador : Utilizador := new Utilizador("Pedro", "email@email.com", <Masculino>, <
Licenciatura>, "Portugal", "Porto", 27, 123456789, 1990, 10, 31);
dcl utilizador2 : Utilizador := new Utilizador("Fabiola", "gmail@gmail.com", <Feminino>, <
Doutoramento>, "Portugal", "Lisboa", 26, 123123123, 1994, 02, 28);
dcl utilizador3 : Utilizador := new Utilizador("Francisca", "asd@asd.com", <Feminino>, <Secundario
>, "Portugal", "Lisboa", 26, 123132323, 1992, 02, 29);
dcl sameuser : Utilizador := new Utilizador("Pedro", "email@email.com", <Masculino>, <Licenciatura
>, "Portugal", "Porto", 27, 123456789, 1990, 10, 31);
dcl trabalho : Trabalho := new Trabalho("Programador de java", "Google", "google@google.pt", <
Licenciatura>, 123456789, "Portugal", "Porto");
dcl trabalho2 : Trabalho := new Trabalho("Programador de c++", "Apple", "apple@apple.pt", <
Doutoramento>, 4562343434, "Portugal", "Lisboa");
dcl trabalho3 : Trabalho := new Trabalho("Programador de php", "Apple2", "apple2@apple.pt", <
Secundario>, 357864, "Portugal", "Funchal");
dcl sametrabalho : Trabalho := new Trabalho("Programador de java", "Google", "google@google.pt", <
Licenciatura>, 123456789, "Portugal", "Porto");

hokify.addUtilizadores(utilizador);
hokify.addTrabalhos(trabalho);
assertEqual(card hokify.getUtilizadores(), 1);
assertEqual(card hokify.getTrabalhos(), 1);

hokify.addUtilizadores(utilizador2);
hokify.addTrabalhos(trabalho2);
assertEqual(card hokify.getUtilizadores(), 2);
assertEqual(card hokify.getTrabalhos(), 2);

hokify.addUtilizadores(utilizador3);
hokify.addTrabalhos(trabalho3);

--Deve falhar pois estamos adicionando um utilizador igual
--hokify.addUtilizadores(sameuser);

```



```

--hokify.addTrabalhos(sametrabalho);
assertTrue(not hokify.sameUser(sameuser));
assertTrue(not hokify.sameTrabalho(sametrabalho));

-- Testes para a pesquisa por nome do trabalho em que retorne os trabalhos
assertEquals(hokify.getTrabalhosPorNome("Programador"),{trabalho,trabalho2,trabalho3});
assertEquals(hokify.getTrabalhosPorNome("java"),{trabalho});
assertEquals(hokify.getTrabalhosPorNome("c++"),{trabalho2});

-- Testes para a pesquisa por interesses em que retorne os trabalhos
trabalho.addInteresse("Informatica");
trabalho.addInteresse("Cinema");
assertEquals(hokify.getTrabalhosPorInteresses("Informatica"),{trabalho});
assertEquals(hokify.getTrabalhosPorInteresses("Cinema"),{trabalho});
assertTrue(not (hokify.getTrabalhosPorInteresses("Cinema")={trabalho2}));

-- Testes para a pesquisa por skills em que retorne os trabalhos
trabalho.addSkills("Java");
trabalho.addSkills("VDM");
assertEquals(hokify.getTrabalhosPorSkills("Java"),{trabalho});
assertEquals(hokify.getTrabalhosPorSkills("VDM"),{trabalho});
assertTrue(not (hokify.getTrabalhosPorSkills("Java")={trabalho2}));

-- Testes para a pesquisa por localidade em que retorne os trabalhos
assertEquals(hokify.getTrabalhosPorLocalidade("Porto"),{trabalho});
assertEquals(hokify.getTrabalhosPorLocalidade("Lisboa"),{trabalho2});

-- Testes para a pesquisa por pais em que retorne os trabalhos
assertEquals(hokify.getTrabalhosPorPais("Portugal"),{trabalho,trabalho2,trabalho3});

-- Testes para a escolaridade em que retorne os trabalhos
assertEquals(hokify.getTrabalhosPorEscolaridade(<Secundario>),{trabalho3});
assertEquals(hokify.getTrabalhosPorEscolaridade(<Licenciatura>),{trabalho,trabalho3});
assertEquals(hokify.getTrabalhosPorEscolaridade(<Mestrado>),{trabalho,trabalho3});
assertEquals(hokify.getTrabalhosPorEscolaridade(<Doutoramento>),{trabalho,trabalho2,trabalho3});

-- Testes para procurar por trabalhos para um utilizador
utilizador.addInteresse("Informatica");
utilizador.addInteresse("Cinema");
utilizador.addSkills("Java");
utilizador.addSkills("VDM");
assertEquals(hokify.getTrabalhosPorUtilizador(utilizador),{trabalho});

utilizador2.addInteresse("Informatica");
utilizador2.addInteresse("Cinema");
utilizador2.addSkills("Java");
utilizador2.addSkills("VDM");
assertEquals(hokify.getTrabalhosPorUtilizador(utilizador2),{trabalho});

-- Testes para a pesquisa por interesses em que retorne os utilizadores
assertEquals(hokify.getUtilizadoresPorInteresses("Informatica"),{utilizador,utilizador2});
utilizador2.removeInteresse("Cinema");
assertEquals(hokify.getUtilizadoresPorInteresses("Cinema"),{utilizador});
assertTrue(not (hokify.getUtilizadoresPorInteresses("Cinema")={}));

-- Testes para a pesquisa por skills em que retorne os utilizadores
assertEquals(hokify.getUtilizadoresPorSkills("Java"),{utilizador,utilizador2});
utilizador2.removeSkills("VDM");
assertEquals(hokify.getUtilizadoresPorSkills("VDM"),{utilizador});
assertTrue(not (hokify.getUtilizadoresPorSkills("VDM")={}));

-- Testes para a escolaridade em que retorne os utilizadores
assertEquals(hokify.getUtilizadoresPorEscolaridade(<Secundario>),{utilizador,utilizador2,
    utilizador3});
assertEquals(hokify.getUtilizadoresPorEscolaridade(<Licenciatura>),{utilizador,utilizador2});

```

```

assertEquals(hokify.getUtilizadoresPorEscolaridade(<Mestrado>), {utilizador2});
assertEquals(hokify.getUtilizadoresPorEscolaridade(<Doutoramento>), {utilizador2});

-- Testes para procurar por utilizadores para um trabalho
assertEquals(hokify.getTrabalhosPorUtilizador(trabalho), {utilizador});
utilizador2.addInteresse("Cinema");
utilizador2.addSkills("VDM");
assertEquals(hokify.getTrabalhosPorUtilizador(trabalho), {utilizador, utilizador2});

assertEquals(card hokify.getUtilizadores(), 3);
assertEquals(card hokify.getTrabalhos(), 3);
hokify.removeUtilizadores(utilizador2);
hokify.removeTrabalhos(trabalho2);
assertEquals(card hokify.getUtilizadores(), 2);
assertEquals(card hokify.getTrabalhos(), 2);
return;
);

public static main: () ==> ()
main() ==
(
  IO`print("TestPlayerFirst -> ");
  new TestAll().TestPlayerFirst();
  IO`println("Passed");

  IO`print("TestTrabalhoFirst -> ");
  new TestAll().TestTrabalhoFirst();
  IO`println("Passed");

  IO`print("TestHokify -> ");
  new TestAll().TestHokify();
  IO`println("Passed");
);

functions
-- TODO Define functiones here
traces
-- TODO Define Combinatorial Test Traces here
end TestAll

```

| Function or operation | Line | Coverage | Calls |
|-----------------------|------|----------|-------|
| TestHokify | 104 | 100.0% | 1 |
| TestPlayerFirst | 10 | 100.0% | 1 |
| TestTrabalhoFirst | 60 | 100.0% | 1 |
| main | 215 | 100.0% | 1 |
| TestAll.vdmpp | | 100.0% | 4 |

4 Trabalho

```

class Trabalho
types
-- TODO Define types here
public String = seq of char;
public Escolaridade = <Secundario> | <Licenciatura> | <Mestrado> | <Doutoramento>;

```

```

public ListaEscolaridade = set of Escolaridade;
public Interesses = set of String;
public Skills = set of String;

values
-- TODO Define values here
instance variables
-- TODO Define instance variables here
private nome: seq of char;
private entidade: seq of char;
private email: seq of char;
private escolaridade: Escolaridade;
private telefone: nat1;
private pais: seq of char;
private localidade: seq of char;
private interesses: set of String := {};
private skills: set of String := {};
private listaEscolaridade: ListaEscolaridade := {};

operations
-- TODO Define operations here

--Construtor

public Trabalho: seq of char * seq of char * seq of char * Escolaridade * nat1 * seq of char *
    seq of char ==> Trabalho
Trabalho(nomeC,entidadeC,emailC,escolaridadeC,telefoneC,paisC,localidadeC) == (
    nome := nomeC;
    entidade := entidadeC;
    email := emailC;
    escolaridade := escolaridadeC;
    listaEscolaridades(escolaridadeC);
    telefone := telefoneC;
    pais := paisC;
    localidade := localidadeC;
    return self;
)
post interesses = {} and
    skills = {} and
    nome = nomeC and
    entidade = entidadeC and
    email = emailC and
    escolaridade = escolaridadeC and
    telefone = telefoneC and
    pais = paisC and
    localidade = localidadeC;

-- Adiciona uma lista com as escolaridades no qual possui

public listaEscolaridades: Escolaridade ==> ()
listaEscolaridades(tipo)==(
if tipo = <Secundario> then (
    listaEscolaridade := listaEscolaridade union {<Secundario>};
    listaEscolaridade := listaEscolaridade union {<Licenciatura>};
    listaEscolaridade := listaEscolaridade union {<Mestrado>};
    listaEscolaridade := listaEscolaridade union {<Doutoramento>};
)elseif tipo = <Licenciatura> then(
    listaEscolaridade := listaEscolaridade union {<Licenciatura>};
    listaEscolaridade := listaEscolaridade union {<Mestrado>};
    listaEscolaridade := listaEscolaridade union {<Doutoramento>};
)elseif tipo = <Mestrado> then(
    listaEscolaridade := listaEscolaridade union {<Mestrado>};
    listaEscolaridade := listaEscolaridade union {<Doutoramento>};
)else(
    listaEscolaridade := listaEscolaridade union {<Doutoramento>};

```

```

););

-- Adicionar interesses

public addInteresse: seq of char ==> ()
addInteresse(String) == interesses := interesses union {String}
pre String not in set interesses
post interesses = interesses~ union {String};

-- Remover interesses

public removeInteresse: seq of char ==> ()
removeInteresse(String) == interesses := interesses \ {String}
pre String in set interesses
post interesses = interesses~ \ {String};

-- Retorna os interesses

public pure getInteresses : () ==> set of String
getInteresses() ==
(
    return interesses;
);

-- Adicionar skills

public addSkills: seq of char ==> ()
addSkills(String) == skills := skills union {String}
pre String not in set skills
post skills = skills~ union {String};

-- Remover skills

public removeSkills: seq of char ==> ()
removeSkills(String) == skills := skills \ {String}
pre String in set skills
post skills = skills~ \ {String};

-- Retorna as skills

public pure getSkills : () ==> set of String
getSkills() ==
(
    return skills;
);

-- Retorna o nome

public pure getNome : () ==> seq of char
getNome() ==
(
    return nome;
);

-- Retorna o entidade

public pure getEntidade : () ==> seq of char
getEntidade() ==
(
    return entidade;
);

-- Retorna o email

public pure getEmail : () ==> seq of char
getEmail() ==
(

```

```

    return email;
);
-- Retorna o escolaridade

public pure getEscolaridade : () ==> Escolaridade
getEscolaridade() ==
(
    return escolaridade;
);
-- Retorna o telefone

public pure getTelefone : () ==> nat1
getTelefone() ==
(
    return telefone;
);
-- Retorna o pais

public pure getPais : () ==> seq of char
getPais() ==
(
    return pais;
);
-- Retorna o localidade

public pure getLocalidade : () ==> seq of char
getLocalidade() ==
(
    return localidade;
);
-- Retorna o listaEscolaridades

public pure getlistaEscolaridades : () ==> ListaEscolaridade
getlistaEscolaridades() ==
(
    return listaEscolaridade;
);

-- Editar Nome

public setName: seq of char ==> ()
setName(newName) == nome := newName
pre newName <> undefined
post nome = newName;

-- Editar Entidade

public setEntidade: seq of char ==> ()
setEntidade(newEntidade) == entidade := newEntidade
pre newEntidade <> undefined
post entidade = newEntidade;

-- Editar Email

public setEmail: seq of char ==> ()
setEmail(newEmail) == email := newEmail
pre newEmail <> undefined
post email = newEmail;

-- Editar Telefone

public setTelefone: nat1 ==> ()
setTelefone(newTelefone) == telefone := newTelefone
pre newTelefone <> undefined
post telefone = newTelefone;

```

```

-- Editar Pais

public setPais: seq of char ==> ()
setPais(newPais) == pais := newPais
pre newPais <> undefined
post pais = newPais;

-- Editar Localidade

public setLocalidade: seq of char ==> ()
setLocalidade(newLocalidade) == localidade := newLocalidade
pre newLocalidade <> undefined
post localidade = newLocalidade;

-- Editar Escolaridade

public setEscolaridade: Escolaridade ==> ()
setEscolaridade(newEscolaridade) == escolaridade := newEscolaridade
pre newEscolaridade <> undefined
post escolaridade = newEscolaridade;

public pure nomeSemelhante: seq of char ==> bool
nomeSemelhante(n) == (
  dcl nameS: seq of char := nome;
  dcl found: bool := false;

  while len nameS >= len n and not found do (
    found := true;

    for index = 1 to len n do
      if found and n(index) <> nameS(index) then (
        found := false;
      );

    if found then
      return true
    else (
      nameS := tl nameS;
      found := false;
    );
  );

  return false;
)
pre len n > 0;

functions
-- TODO Define functiones here
traces
-- TODO Define Combinatorial Test Traces here
end Trabalho

```

| Function or operation | Line | Coverage | Calls |
|-----------------------|------|----------|-------|
| Trabalho | 29 | 100.0% | 5 |
| addInteresse | 71 | 100.0% | 4 |
| addSkills | 90 | 100.0% | 4 |
| getEmail | 121 | 100.0% | 14 |

| | | | |
|-----------------------|-----|--------|-----|
| getEntidade | 115 | 100.0% | 9 |
| getEscolaridade | 127 | 100.0% | 4 |
| getInteresses | 83 | 100.0% | 25 |
| getLocalidade | 145 | 100.0% | 8 |
| getNome | 109 | 100.0% | 10 |
| getPais | 139 | 100.0% | 5 |
| getSkills | 102 | 100.0% | 25 |
| getTelefone | 133 | 100.0% | 2 |
| getlistaEscolaridades | 151 | 100.0% | 18 |
| listaEscolaridades | 52 | 100.0% | 1 |
| nomeSemelhante | 199 | 100.0% | 4 |
| removeInteresse | 77 | 100.0% | 1 |
| removeSkills | 96 | 100.0% | 2 |
| setEmail | 170 | 100.0% | 1 |
| setEntidade | 164 | 100.0% | 2 |
| setEscolaridade | 194 | 100.0% | 1 |
| setLocalidade | 188 | 100.0% | 1 |
| setNome | 158 | 100.0% | 1 |
| setPais | 182 | 100.0% | 1 |
| setTelefone | 176 | 100.0% | 1 |
| Trabalho.vdmpp | | 100.0% | 149 |

5 Utilizador

```

class Utilizador
types
-- TODO Define types here
public String = seq of char;
public Sexo = <Masculino> | <Feminino>;
public Escolaridade = <Secundario> | <Licenciatura> | <Mestrado> | <Doutoramento>;
public ListaEscolaridade = set of Escolaridade;
public Interesses = set of String;
public Skills = set of String;
public Date :: year : nat month: nat1 day : nat
    inv mk_Date(y,m,d) == m <= 12 and d <= DaysOfMonth(m,y);

values
-- TODO Define values here

instance variables
-- TODO Define instance variables here
private nome: seq of char;
private email: seq of char;
private sexo: Sexo;
private escolaridade: Escolaridade;
private idade: nat1;
private telefone: nat1;
private pais: seq of char;
private localidade: seq of char;
private interesses: set of String := {};
private skills: set of String := {};
private listaEscolaridade: ListaEscolaridade := {};
private dataNascimento : Date;

```

operations

```
-- TODO Define operations here

--Construtor
public Utilizador: seq of char * seq of char * Sexo * Escolaridade * seq of char * seq of char *
    nat1 * nat1 * nat * nat1 * nat1 ==> Utilizador
Utilizador(nm,emailC,sexoC,escolaridadeC,paisC,localidadeC,idadeC,telefoneC,year,month,day) == (
    nome := nm;
    email := emailC;
    sexo := sexoC;
    escolaridade := escolaridadeC;
    listaEscolaridades(escolaridadeC);
    idade := idadeC;
    telefone := telefoneC;
    pais := paisC;
    localidade := localidadeC;
    dataNascimento := mk_Date(year, month, day);
    return self
)
pre verificarEscolaridade(escolaridadeC, idadeC)
post interesses = {} and
    skills = {} and
    nome = nm and
    email = emailC and
    sexo = sexoC and
    idade = idadeC and
    telefone = telefoneC and

    pais = paisC and
    localidade = localidadeC;

-- Verifica se o utilizador tem idade para a escolaridade
public pure verificarEscolaridade: Escolaridade * nat1 ==> bool
verificarEscolaridade(escola, idd)==(
if (escola = <Secundario> and idd > 17) then(
    return true;
) elseif(escola = <Licenciatura> and idd > 20) then(
    return true;
) elseif(escola = <Mestrado> and idd > 22) then(
    return true;
) elseif(escola = <Doutoramento> and idd > 24) then(
    return true;
) else
    return false;
)pre escolaridade <> undefined;

-- Adiciona uma lista com as escolaridades no qual possui
public listaEscolaridades: Escolaridade ==> ()
listaEscolaridades(tipo)==(
if tipo = <Doutoramento> then (
    listaEscolaridade := listaEscolaridade union {<Secundario>};
    listaEscolaridade := listaEscolaridade union {<Licenciatura>};
    listaEscolaridade := listaEscolaridade union {<Mestrado>};
    listaEscolaridade := listaEscolaridade union {<Doutoramento>};
) elseif tipo = <Mestrado> then(
    listaEscolaridade := listaEscolaridade union {<Secundario>};
    listaEscolaridade := listaEscolaridade union {<Licenciatura>};
    listaEscolaridade := listaEscolaridade union {<Mestrado>};
) elseif tipo = <Licenciatura> then(
    listaEscolaridade := listaEscolaridade union {<Secundario>};
    listaEscolaridade := listaEscolaridade union {<Licenciatura>};
) else(
    listaEscolaridade := listaEscolaridade union {<Secundario>};
```



```

);
)pre tipo <> undefined;

-- Adicionar interesses
public addInteresse: seq of char ==> ()
addInteresse(String) == interesses := interesses union {String}

pre String not in set interesses
post interesses = interesses~ union {String};

-- Remover interesses
public removeInteresse: seq of char ==> ()
removeInteresse(String) == interesses := interesses \ {String}

pre String in set interesses
post interesses = interesses~ \ {String};

-- Retorna os interesses
public pure getInteresses : () ==> set of String
getInteresses() ==
(
    return interesses;
);

-- Adicionar skills
public addSkills: seq of char ==> ()
addSkills(String) == skills := skills union {String}

pre String not in set skills
post skills = skills~ union {String};

-- Remover skills
public removeSkills: seq of char ==> ()
removeSkills(String) == skills := skills \ {String}

pre String in set skills
post skills = skills~ \ {String};

-- Retorna as skills
public pure getSkills : () ==> set of String
getSkills() ==
(
    return skills;
);

-- Retorna o nome
public pure getNome : () ==> seq of char
getNome() ==
(
    return nome;
);

-- Retorna a idade
public pure getIdade : () ==> nat1
getIdade() ==
(
    return idade;
);

-- Retorna o telefone

```

```

public pure getTelefone : () ==> nat1
getTelefone() ==
(
    return telefone;
);

-- Retorna o email
public pure getEmail : () ==> seq of char
getEmail() ==
(
    return email;
);

-- Retorna o sexo
public pure getSexo : () ==> Sexo
getSexo() ==
(
    return sexo;
);

-- Retorna o pais
public pure getPais : () ==> seq of char
getPais() ==
(
    return pais;
);

-- Retorna a localidade
public pure getLocalidade : () ==> seq of char
getLocalidade() ==
(
    return localidade;
);

-- Retorna a escolaridade
public pure getEscolaridade : () ==> Escolaridade
getEscolaridade() ==
(
    return escolaridade;
);

-- Retorna o listaEscolaridades
public pure getlistaEscolaridades : () ==> ListaEscolaridade
getlistaEscolaridades() ==
(
    return listaEscolaridade;
);

-- Retorna a data de nascimento
public pure getDataNasccimento : () ==> Date
getDataNasccimento() ==
(
    return dataNascimento;
);

-- Editar Nome

```

```

public setName: seq of char ==> ()

setName(newName) == nome := newName
pre newName <> undefined
post nome = newName;

-- Editar Email
public setEmail: seq of char ==> ()

setEmail(newEmail) == email := newEmail
pre newEmail <> undefined
post email = newEmail;

-- Editar Telefone
public setTelefone: nat1 ==> ()

setTelefone(newTelefone) == telefone := newTelefone
pre newTelefone <> undefined
post telefone = newTelefone;

-- Editar Pais
public setPais: seq of char ==> ()

setPais(newPais) == pais := newPais
pre newPais <> undefined
post pais = newPais;

-- Editar Localidade
public setLocalidade: seq of char ==> ()
setLocalidade(newLocalidade) == localidade := newLocalidade
pre newLocalidade <> undefined
post localidade = newLocalidade;

-- Editar Escolaridade
public setEscolaridade: Escolaridade ==> ()
setEscolaridade(newEscolaridade) == escolaridade := newEscolaridade
pre newEscolaridade <> undefined
post escolaridade = newEscolaridade;

functions
-- TODO Define functiones here

-- Retorna o n mero de dias do mls num dado ano
public static DaysOfMonth(month,year : nat1) r : nat1 == (
  if month = 1 or month = 3 or month = 5 or month = 7 or month = 8 or month = 10 or month = 12
  then
    31
  else if month = 2 and ((year mod 4 = 0 and year mod 100 <> 0) or year mod 400 = 0) then
    29
  else if month = 2 then
    28
  else
    30
  )

traces
-- TODO Define Combinatorial Test Traces here

end Utilizador

```

| Function or operation | Line | Coverage | Calls |
|-----------------------|------|----------|-------|
| DaysOfMonth | 233 | 100.0% | 1 |
| Utilizador | 32 | 100.0% | 5 |
| addInteresse | 92 | 100.0% | 7 |
| addSkills | 111 | 100.0% | 7 |
| getDataNascimento | 197 | 100.0% | 1 |
| getEmail | 151 | 100.0% | 14 |
| getEscolaridade | 179 | 100.0% | 4 |
| getIdade | 137 | 100.0% | 1 |
| getInteresses | 104 | 100.0% | 25 |
| getLocalidade | 172 | 100.0% | 2 |
| getNome | 130 | 100.0% | 2 |
| getPais | 165 | 100.0% | 2 |
| getSexo | 158 | 100.0% | 1 |
| getSkills | 123 | 100.0% | 25 |
| getTelefone | 144 | 100.0% | 10 |
| getlistaEscolaridades | 186 | 100.0% | 18 |
| listaEscolaridades | 72 | 100.0% | 1 |
| removeInteresse | 98 | 100.0% | 4 |
| removeSkills | 117 | 100.0% | 2 |
| setEmail | 199 | 100.0% | 1 |
| setEscolaridade | 223 | 100.0% | 1 |
| setLocalidade | 217 | 100.0% | 1 |
| setNome | 193 | 100.0% | 2 |
| setPais | 211 | 100.0% | 1 |
| setTelefone | 205 | 100.0% | 1 |
| verificarEscolaridade | 57 | 100.0% | 3 |
| Utilizador.vdmpp | | 100.0% | 142 |