Hokify

December 26, 2017

Contents

1 MyTestCase

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

operations

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

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

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

 protected assertEqual: ? * ? ==> ()
 assertEqual(expected, actual) ==
  if expected <> actual then (
     IO`print("Actual value (");
     IO`print(actual);
     IO`print(") different from expected (");
     IO`print(expected);
     IO`println(")\n")
  )
 post expected = actual

end MyTestCase
```

Function or operation	Line	Coverage	Calls
assertEqual	20	38.8%	55
assertTrue	12	0.0%	0
MyTestCase.vdmpp		35.0%	55

2 TestUtilizador

```
class TestUtilizador 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() ==
(
 dcl user : Utilizador := new Utilizador("Pedro","email@email.com",<Masculino>,"Portugal","Porto"
     ,27,123456789);
 user.addInteresse("Informatica");
 user.addInteresse("Cinema");
 user.addSkills("Java");
 user.addSkills("VDM");

 assertEqual(user.getNome(),"Pedro");
 assertEqual(user.getEmail(),"email@email.com");
 assertEqual(user.getSexo(),<Masculino>);
 assertEqual(user.getPais(),"Portugal");
 assertEqual(user.getLocalidade(),"Porto");
 assertEqual(user.getIdade(),27);
 assertEqual(user.getTelefone(),123456789);
 assertEqual(user.getInteresses(),{"Informatica","Cinema"});
 assertEqual(user.getSkills(),{"Java","VDM"});

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

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

 user.setNome("Pedro Faria");
 user.setEmail("PedroFaria@gmail.com");

 user.setTelefone(987654321);
 assertEqual(user.getNome(),"Pedro Faria");
 assertEqual(user.getEmail(),"PedroFaria@gmail.com");
 assertEqual(user.getTelefone(),987654321);

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

 return;
);

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

```
main() ==
(
   new TestUtilizador().TestPlayerFirst();
);

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

Function or operation	Line	Coverage	Calls
TestPlayerFirst	10	100.0%	1
main	37	100.0%	1
TestUtilizador.vdmpp		100.0%	2

3 Utilizador

```
class Utilizador
types
-- TODO Define types here
 public String = seq of char;
 public Sexo = <Masculino> | <Feminino>;
 public Interesses = set of String;
 public Skills = set of String;

values
-- TODO Define values here

instance variables
-- TODO Define instance variables here
 private nome: String;
 private email: String;
 private sexo: Sexo;
 private idade: nat1;
 private telefone: nat1;
 private pais : String;
 private localidade: String;
 private interesses: Interesses := {};
 private skills: Skills := {};


operations
-- TODO Define operations here

 --Construtor
 public Utilizador: String * String * Sexo * String * String * nat1 * nat1 ==> Utilizador
 Utilizador(nm,emailC,sexoC,paisC,localidadeC,idadeC,telefoneC) == (
  nome := nm;
  email := emailC;
  sexo := sexoC;
  idade := idadeC;
  telefone := telefoneC;
  pais := paisC;
  localidade := localidadeC;
  return self
```

```
)
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;

-- Adicionar interesses

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

-- Remover interesses
public removeInteresse: String ==> ()

removeInteresse(String) == interesses := interesses \ {String}
pre String in set interesses
post interesses = interesses~ \ {String};

-- Retorna os interesses
public pure getInteresses : () ==> Interesses

getInteresses() ==
(
 return interesses;
);

-- Adicionar skills

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

-- Remover skills
public removeSkills: String ==> ()

removeSkills(String) == skills := skills \ {String}
pre String in set skills
post skills = skills~ \ {String};

-- Retorna as skills
public pure getSkills : () ==> Skills
getSkills() ==
(
 return skills;
);


-- Retorna o nome
public pure getNome : () ==> String
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 : () ==> String
getEmail() ==
(
 return email;
);

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

getSexo() ==
(
 return sexo;
);

-- Retorna o pais


public pure getPais : () ==> String
getPais() ==
(
 return pais;
);


-- Retorna a localidade

public pure getLocalidade : () ==> String
getLocalidade() ==
(
 return localidade;
);

-- Editar Nome
public setNome: String ==> ()
setNome(newName) == nome := newName
pre newName <> undefined
post nome = newName;

-- Editar Email
public setEmail: String ==> ()
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;

functions
-- TODO Define functiones here

traces
-- TODO Define Combinatorial Test Traces here

end Utilizador
```

Function or operation	Line	Coverage	Calls
Utilizador	24	100.0%	1
addInteresse	38	100.0%	5
addSkills	57	100.0%	4
getEmail	103	100.0%	2
getIdade	86	100.0%	1
getInteresses	50	100.0%	2
getLocalidade	130	100.0%	1
getNome	76	100.0%	2
getPais	123	100.0%	1
getSexo	100	100.0%	1
getSkills	69	100.0%	2
getTelefone	93	100.0%	2
removeInteresse	44	100.0%	2
removeSkills	63	100.0%	1
setEmail	123	100.0%	1
setNome	117	100.0%	1
setTelefone	129	100.0%	1
Utilizador.vdmpp		100.0%	30