/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

/\*\*

\*

\* @author Andreea

\*/

package optDist;

import OptDist.Student;

import OptDist.StudentAdministration;

import java.util.ArrayList;

import org.junit.After;

import org.junit.AfterClass;

import org.junit.Before;

import org.junit.BeforeClass;

import org.junit.Test;

import static org.junit.Assert.\*;

public class StudentAdministrationTest {

//daca se introduce in lista de studenti un student fara nume, prenume etc

@Test

public void addStudentTestStudentNull() {

StudentAdministration instance = new StudentAdministration();

StudentAdministration instance2 = new StudentAdministration();

Student s = new Student(null, null, null, null, 0);

instance.addStudent(s);

if (s == null)

{

fail("The Student should not be null");

}

}

//testul reuseste daca studentul null nu a fost adaugat

@Test

public void addStudentTestStudentNullAdaugat() {

StudentAdministration instance = new StudentAdministration();

StudentAdministration instance2 = new StudentAdministration();

Student s = new Student(null, null, null, null, 0);

instance.addStudent(s);

assertEquals(instance, instance2);

}

//daca se introduce in lista de studenti un student cu nrmatricol care nu este string

@Test

public void addStudentTestNrMatricolNotString() {

StudentAdministration instance = new StudentAdministration();

Student s = new Student(2,"Radu","Andrei","A3",10);

if (!(s.getNrMatricol() instanceof String))

{

fail("NrMatricol should be a string");

}

instance.addStudent(s);

}

//testul reuseste daca studentul cu nrmatricol!=string nu a fost adaugat

@Test

public void addStudentTestNrMatricolNotStringAdaugat() {

StudentAdministration instance = new StudentAdministration();

StudentAdministration instance2 = new StudentAdministration();

Student s = new Student(null, null, null, null, 0);

instance.addStudent(s);

assertEquals(instance, instance2);

}

//daca se introduce in lista de studenti un student cu nume care nu este string

@Test

public void addStudentTestNumeNotString() {

StudentAdministration instance = new StudentAdministration();

Student s = new Student("123",12,"Andrei","A3",10);

if (!(s.getName() instanceof String))

{

fail("Name should be a string");

}

instance.addStudent(s);

}

//testul reuseste daca studentul cu nume!=string nu a fost adaugat

@Test

public void addStudentTestNumeNotStringAdaugat() {

StudentAdministration instance = new StudentAdministration();

StudentAdministration instance2 = new StudentAdministration();

Student s = new Student(null, null, null, null, 0);

instance.addStudent(s);

assertEquals(instance, instance2);

}

//daca se introduce in lista de studenti un student cu prenume care nu este string

@Test

public void addStudentTestPrenumeNotString() {

StudentAdministration instance = new StudentAdministration();

Student s = new Student("123","Radu",123,"A3",10);

if (!(s.getSurname() instanceof String))

{

fail("Surname should be a string");

}

instance.addStudent(s);

}

//testul reuseste daca studentul cu prenume!=string nu a fost adaugat

@Test

public void addStudentTestPrenumeNotStringAdaugat() {

StudentAdministration instance = new StudentAdministration();

StudentAdministration instance2 = new StudentAdministration();

Student s = new Student(null, null, null, null, 0);

instance.addStudent(s);

assertEquals(instance, instance2);

}

//daca se introduce in lista de studenti un student cu grupa care nu este string

@Test

public void addStudentTestGroupNotString() {

StudentAdministration instance = new StudentAdministration();

Student s = new Student("123","Radu","Andrei",3,10);

if (!(s.getGroup() instanceof String))

{

fail("Group should be a string");

}

instance.addStudent(s);

}

//testul reuseste daca studentul cu grupa!=string nu a fost adaugat

@Test

public void addStudentTestGroupNotStringAdaugat() {

StudentAdministration instance = new StudentAdministration();

StudentAdministration instance2 = new StudentAdministration();

Student s = new Student(null, null, null, null, 0);

instance.addStudent(s);

assertEquals(instance, instance2);

}

//daca se introduce in lista de studenti un student cu grade care nu este float

@Test

public void addStudentTestGradeNotFloat() {

StudentAdministration instance = new StudentAdministration();

Student s = new Student("123","Radu","Andrei","A3","8");

if (!(s.getGrade() instanceof Float))

{

fail("Grade should be a string");

}

instance.addStudent(s);

}

//testul reuseste daca studentul cu grade!=float nu a fost adaugat

@Test

public void addStudentTestGradeNotFloatAdaugat() {

StudentAdministration instance = new StudentAdministration();

StudentAdministration instance2 = new StudentAdministration();

Student s = new Student(null, null, null, null, 0);

instance.addStudent(s);

assertEquals(instance, instance2);

}

//verifica daca compare returneaza tot float

@Test

public void compareTestCompareReturnFloat() {

Student s1 = new Student("123aa","ana","popescu","A3",7);

Student s2 = new Student("123bb","alex","ionescu","B3",8);

if (!(compare(s1, s2)) instanceof Float)

fail("Compare does not return a float value");

}

//verifica daca compare returneaza nota mai mare

@Test

public void compareTestBiggestGrade() {

Student s1 = new Student("123aa","ana","popescu","A3",7);

Student s2 = new Student("123bb","alex","ionescu","B3",8);

float expectedResult = 8;

float actualResult = compare(s1, s2);

if (expectedResult != actualResult)

fail("Compare does not return the biggest value");

}

}