Laboratório 20 - Coleções II (30/10/2014)

1.

```
package lab20;
import java.util.Collection;

/**
 * Interface that defines the contract to be implemented by all
 * containers of {@link Person} objects.
 * @author Challenge.IT
 * Copyright (c) 2014, Challenge.IT and/or its affiliates. All rights reserved.
 * DO NOT ALTER OR REMOVE COPYRIGHT NOTICES OR THIS FILE HEADER.
 * This code is distributed in the hope that it will be useful for learning purposes, but WITHOUT
 * ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or
 * FITNESS FOR A PARTICULAR PURPOSE.
 **/
public interface PersonsContainer
//interface que define o contrato a ser implementado por todos os contentores de objectos Person.
{
	/**
	 * Add operation.
	 * @param person The person for save in the container.
	 * @return True if the operation succeeds.
	 */
	public boolean add(Person person);   // adicionar um objeto Person ao contentor

	/**
	 * @return All the persons.
	 */
	public Collection<Person> getAll();   //ver todos os objejos person da coleção

	/**
	 * @param nif The person's nif number for search.
	 * @return The person with the nif equals to the nif passed in the arguments or
null if not exists.
	 */
	public Person getByNif(String nif);
	//ver o objecto Person que tem como argumento determinado nif ou a inexistência
desse objecto Person
}
```

```
package lab20;
/**
 * Class that defines the Person object.
 * @author Challenge.IT
 * Copyright (c) 2014, Challenge.IT and/or its affiliates. All rights reserved.
 * DO NOT ALTER OR REMOVE COPYRIGHT NOTICES OR THIS FILE HEADER.
 * This code is distributed in the hope that it will be useful for learning purposes, but WITHOUT
 * ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or
 * FITNESS FOR A PARTICULAR PURPOSE.
 * */
	public class Person
	{
	   private final String _nif;          // nºcontribuinte
	   private String _name;                      // nome
	   private int _age;                                // idade

	/**
	 * Creates an instance of {@link Person}.
	 * @param nif The person's nif number.
	 * @param name The person's name.
```

```
         * @param age The person's age.
         */
          public Person(String nif, String name, int age)   // construtor
          {
              _nif = nif;
              _name = name;
              _age = age;
          }

          /**
           * @return The person's nif number.
           */
          public String getNif() { return _nif; }

          /**
           * @return The person's name.
           */
          public String getName() { return _name; }

          /**
           * Set the person's name.
           * @param name The new name.
           */
          public void setName(String name) { _name = name; }  //Alterar o nome

          /**
           * @return The person's age.
           */
          public int getAge() { return _age; }

          /**
           * Increments the person's age.
           */
          public void incrementAge() { _age++; }    // aumenta a idade ao longo dos anos
        }
```

```
package lab20;

import java.util.ArrayList;
import java.util.Collection;
import java.util.List;
import org.junit.Assert;
import lab20.Person;
import lab20.PersonsContainer;

/**
 * Class that implements the interface {@link PersonsContainer} for keep in the memory
 * {@link Person} objects using one {@link List}.
 * @author Challenge.IT
 * Copyright (c) 2014, Challenge.IT and/or its affiliates. All rights reserved.
 * DO NOT ALTER OR REMOVE COPYRIGHT NOTICES OR THIS FILE HEADER.
 * This code is distributed in the hope that it will be useful for learning purposes, but WITHOUT
 * ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or
 * FITNESS FOR A PARTICULAR PURPOSE.
 * */
public class ListPersonsContainer implements PersonsContainer
{
       private List<Person> _persons = new ArrayList<>();
```

```
        public ListPersonsContainer(){
           super();
        }

        @Override
        public boolean add(Person person) {
           if (_persons.contains(person))
                            // a forma mais fácil de encontar uma Person numa base de dados
               return false;//é pelos seus números de identificação, uma vez que são únicos
           else{
                     persons.add(person);
                     return true;
               }
        }

        @Override
        public Collection<Person> getAll() {
           return _persons;
        }

        @Override
        public Person getByNif(String nif) {
           for (int idx=0;idx<_persons.size(); idx++){
               if(_persons.get(idx).getNif()== nif)
                        //Se já existe um objecto com este nif, apresenta os dados da Person

                   // System.out.println(_persons.get(idx).getName())   //só para testar a
impressão
                   return _persons.get(idx);
           }
           return null;  // caso não exista, retorna null
        }

        public static void main (String[] args){
               ListPersonsContainer _container = new ListPersonsContainer();
               Person p1 = new Person("123456", "Ricardo Sousa", 25);

               _container.add(p1);
               Person person = _container.getByNif(p1.getNif());

               System.out.println(p1.getNif());
               System.out.println(p1.getName());
               System.out.println(p1.getAge());
        }
}
```

```
package lab20;

import java.util.List;
import org.junit.Assert;
import org.junit.Before;
import org.junit.Test;
import lab20.Person;
import lab20.PersonsContainer;

/**
 * Test cases for {@link ListPersonsContainer} class.
 *
 * @author Challenge.IT
 * Copyright (c) 2014, Challenge.IT and/or its affiliates. All rights reserved.
 * DO NOT ALTER OR REMOVE COPYRIGHT NOTICES OR THIS FILE HEADER.
 * This code is distributed in the hope that it will be useful for learning purposes, but WITHOUT
 * ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or
 * FITNESS FOR A PARTICULAR PURPOSE.
 * */
public class ListPersonsContainerTest
{
       private PersonsContainer _container;

       @Before
       public void beforeTests()
       {
              _container = new ListPersonsContainer();
       }

       @Test
       public void shouldAddOnePersonToContainer()
       {
          // Arrange
          // Act

          // Assert
          Assert.assertTrue(_container.add(new Person("123456", "Ricardo Sousa", 25)));
       }

       @Test
       public void shouldGetOnePersonByNifFromTheContainer()
       {
          // Arrange
          Person p1 = new Person("123456", "Ricardo Sousa", 25);

          // Act
          _container.add(p1);
          Person person = _container.getByNif(p1.getNif());

              // Assert
          Assert.assertNotNull(person);
          Assert.assertEquals(p1.getNif(), person.getNif());
          Assert.assertEquals(p1.getName(), person.getName());
          Assert.assertEquals(p1.getAge(), person.getAge());
       }

       @Test
       public void shouldGetAllPersonsFromTheContainer()
       {
              // Arrange
              Person p1 = new Person("123456", "Ricardo Sousa", 25);
              Person p2 = new Person("1234", "Diogo Matos", 25);

              // Act
              _container.add(p1);
              _container.add(p2);

              List<Person> persons = (List<Person>) _container.getAll();

              // Assert
              Assert.assertNotNull(persons);
              Assert.assertEquals(2, persons.size());

              Assert.assertEquals(p1.getNif(), persons.get(0).getNif());
              Assert.assertEquals(p1.getName(), persons.get(0).getName());
              Assert.assertEquals(p1.getAge(), persons.get(0).getAge());

              Assert.assertEquals(p2.getNif(), persons.get(1).getNif());
              Assert.assertEquals(p2.getName(), persons.get(1).getName());
              Assert.assertEquals(p2.getAge(), persons.get(1).getAge());
       }
}
```

2.

```
package lab20;

import java.util.Collection;
import java.util.Map;
import java.util.TreeMap;

/**
 * Class that implements the interface {@link PersonsContainer} for keep in the memory
 * {@link Person} objects using one {@link Map}.
 */
public class MapPersonsContainer implements PersonsContainer {

       public Map<String, Person> personContainerMap= new TreeMap<>();

       @Override
       public boolean add(Person person) { // como não permite chaves duplicadas
          if (personContainerMap.containsKey(person.getNif()))  // as chaves são os nif
              return false;     //não foi adicionado
          else{
              personContainerMap.put(person.getNif(), person);
              // adiciona-se ao contentor o novo objecto com a sua chave
              return true;    // foi adicionado
              }
       }

       @Override
       public Collection<Person> getAll() {    //obter todos os elementos
              return personContainerMap.values();
       }

       @Override
       public Person getByNif(String nif) {    //obter a partir da chave
              return personContainerMap.get(nif);
// Retorna o valor (objecto Person) da chave que está a ser verificada, ou null caso
// não contenha esta chave.
       }
}
```

3.

```
package lab20;

import static org.junit.Assert.*;
import java.util.Collection;
import java.util.List;
import java.util.Map;
import org.junit.Assert;
```

```java
import org.junit.Before;
import org.junit.Test;

public class MapPersonsContainerTest {

	private MapPersonsContainer _container;

	@Before
	public void beforeTests()
	{
		_container = new MapPersonsContainer();
	}

	@Test
	public void shouldAddOnePersonToContainer()
	{
		// Arrange
		Person p1 = new Person("654321", "Filipa Gonçalves", 31);
		// Act
		// Assert
		Assert.assertTrue(_container.add(p1));
	}

	@Test
	public void shouldGetOnePersonByNifFromTheContainer()
	{
		// Arrange
		Person p1 = new Person("123456", "Ricardo Sousa", 25);

		// Act
		_container.add(p1);
		Person person = _container.getByNif(p1.getNif());

		// Assert
		Assert.assertNotNull(person);
		Assert.assertEquals(p1.getNif(), person.getNif());
		Assert.assertEquals(p1.getName(), person.getName());
		Assert.assertEquals(p1.getAge(), person.getAge());
	}

	@Test
	public void shouldGetAllPersonsFromTheContainer()
	{
		// Arrange
		Person p1 = new Person("123456", "Ricardo Sousa", 25);
		Person p2 = new Person("1234", "Diogo Matos", 25);
		Person p3 = new Person("123456", "Filipa Gonçalves", 31);

		// Act
		_container.add(p1);
		_container.add(p2);

		Collection<Person> persons = _container.getAll();

		// Assert
		Assert.assertFalse(_container.add(p3));
	       //verifica que p3 não é adicionado a _container porque tem o mesmo nif que p1.
		Assert.assertNotNull(persons);   // verifica que a lista não está vazia
		Assert.assertEquals(2, persons.size());   // verifica tamanho da lista

		Assert.assertTrue(persons.contains(p1));
		Assert.assertTrue(persons.contains(p2));
	}
}
```