Quiz 18 - Testes Unitários II (16/10/2014)

1. De acordo com o seguinte troço de código, quantos testes unitários irão ser executados com sucesso?

B. 1

//Apenas o segundo teste é executado, uma vez que o *Assert.assertTrue* verifica se a instrução tem o valor true. As instruções dos dois primeiros testes têm valor true, mas o primeiro teste está privado, logo dará erro, para correr deveria ser public.

2. Dado o seguinte troço de código, indique se todos os testes executam sem erros.

```
package Quizz14;

import org.junit.Assert;
import org.junit.Test;

public class PersonTester {
	private Person _person = new Person("Manuel Martins", 8);
	@BeforeClass   //tem de ser um método estático, tem como objectivo inicializar recursos como bases de dados
	public static void init() {
	}

	@Test
	public void methodOne() {
		Assert.assertNotNull(_person);
	}

	@Test
	public void methodTwo() {
	   Assert.assertTrue(_person.getId() == 8);
	   Assert.assertTrue(_person.getName().equals("Manuel Martins"));
	}
}
```

B. Falso  O método @Before Class tem como objectivo inicializar recursos como bases de dados e nunca criar objectos. Além disso tem de ser um método estático. Para que os testes executem sem erros, temos de criar o objecto person aquando da sua definição e colocar o médodo init como static.

3. Dado o seguinte troço de código, indique qual das seguintes afirmações é correcta.

```
public class Util {
    public static String sum(int a, short b) { return "One"; }
    public static String sum(int a, double b) { return "Two"; }
    public static String sum(double a, double b) { return "Three"; }
    public static String sum(short a, short b) { return "Four"; }
}
public class AppTest {
@Test
    public void methodOne() {
      Assert.assertSame("Two", Util.sum(1, 2.0f));
      Assert.assertSame("Three", Util.sum(1, 2)); //2 é promovido a double =>"Two"
```

```
        Assert.assertSame("One", Util.sum(1.0, 2)); //1.0 é do tipo double => "Three"
    }
}
```

C. As linhas 2 e 3 falham nas asserções;

4. Na escrita de testes unitários, é boa prática fazer uma classe de testes por cada classe criada na aplicação.

A. Verdadeiro; Sim, mas quando a quantidade de classes for muito grande pode não ser muito fácil de implementar

5. Dado o seguinte troço de código, indique quais as afirmações que estão correctas.

```
public class AppTest {
private int _value = 0;
@Before
public void init() {
        _value++;
}
@Test
public void methodOne() {
        Assert.assertTrue(_value == 1);
}
@Test
public void methodTwo() {
        Assert.assertTrue(_value == 2);
}
@Test
public void methodThree() {
        Assert.assertTrue(_value == 3);
}
@After
public void last() {
        _value++;
}
}
```

B. Apenas o teste methodOne corre com sucesso;

O programa corre o @Before e @After "dentro do teste", logo quando executa o primeiro teste dá:

Value = 1

True

Value = 2

Quando sai do teste o value volta a ser 0, logo nos próximos testes os valores repetem-se, logo

Value = 1

False

Value = 2