Laboratório 11 - Testes Unitários I (16/10/2014)

1. .Para que os testes corram no Junit é necessário que tenham a anotação @Test.

Depois de colocada esta anotação, apenas o teste de verificar a adição de números negativos falhou, por isso fiz a seguinte alteração:

```
package Lab11p1;

import static org.junit.Assert.*;
import org.junit.Test;

public class SumTester {
@Test
/**
* Verifies positive numbers additions.
*/
       public void testPositiveNumbers() {
          Sum s = new Sum(0);
          s.add(103);
          s.add(201);
          s.add(123);
          assertEquals(427, s.getTotal());
       }
   @Test
   /**
   * Verifies negative numbers additions
   */
       public void testNegativeNumbers() {
          Sum s = new Sum(0);
          s.add(-10);
         s.add(-20);                        //substitui: s.add((int)Math.pow(-10,2));
         s.add(-(int)Math.pow(-10,2));    // pow(-10 , 2)=100 =>-30 + 100 = 70 != -130
         assertEquals(-130, s.getTotal());
   }
   @Test
   /**
   * Verifies if the total is zero
   */
       public void testZeros() {
          Sum s = new Sum(0);
          s.add(40/2);
          s.add(-20);
          assertEquals(0, s.getTotal());
       }
}
```

2. .

```
package Lab11p2;

public class EmployeeBonus {
       double bonus = 0;
       public double getBonus(int pieces, int bonusLimit) {
          bonus = ((pieces > bonusLimit)?(0.10*pieces):(bonus));
          return bonus;
       }
```

```
	public double getBonus(int numUnits) {     // sistema de pagamento ao Departamento
de Produção
		bonus = ((numUnits > 25)?(150*numUnits):(100*numUnits));
		return bonus;
	}

	public double getBonus(double sales) {     // sistema de pagamento ao Departamento
de Vendas
		bonus = ((sales > 5000)?(0.05*sales):(bonus));
		return bonus;
	}
}
```

```
package Lab11p2;
import java.util.Scanner;

public class AppEmployee {
	final static int UNITS_PT = 250;    //unidades montadas por empregados em part-time
	final static int UNITS_FT = 700;     //unidades montadas por empregados em full-
time
	public static void main(String[] args){
		Scanner kbd = new Scanner(System.in);
		EmployeeBonus b = new EmployeeBonus();
		System.out.println("Enter department: ");
		int dept = kbd.nextInt();
		double bonus = 0;       //bonus inicial

		switch (dept)
		{
		    case 1:       //Departamento de Vendas
			System.out.print("Enter sales: ");
			double sales = kbd.nextDouble();
			bonus = b.getBonus(sales);
			break;

		    case 2:        //Departamento de Produção
			System.out.print("Enter number of units produced: ");
			int numUnits = kbd.nextInt();
			bonus = b.getBonus(numUnits);
			break;

		    case 3:         //Departamento de montagem
			System.out.print("Enter # of pieces completed: ");
			int pieces = kbd.nextInt();
			System.out.print("Full-time (1) or Part-Time (2)? ");
			int empType = kbd.nextInt();
			int bonusLimit = (empType == 1) ? UNITS_FT : UNITS_PT;
			bonus = b.getBonus(pieces, bonusLimit);
			break;

		    default:
			System.out.print("Error! ");
		}
		System.out.println("Bonus Amount: " + bonus + "€");
	}
}
```

3.

```
package Lab11p2;

import org.junit.Assert;
import org.junit.Test;
import junit.framework.TestCase;

public class EmployeeBonusTester {
    EmployeeBonus eB = new EmployeeBonus();
    @Test    // teste para bonus do Departamento de vendas
    public void testGetBonusDoubleParamBelowThreshould(){   //se o valor de vendas for
<= a 5000, não recebe bonus
        Assert.assertTrue(0.0 == eB.getBonus(4000.0));
    }

    @Test     // teste para bonus do Departamento de vendas
    public void testGetBonusDoubleParamAboveThreshould(){   //se o valor de vendas for
>= a 5000 recebe 5% sobre as vendas
        double sales = 6000.0;
        Assert.assertTrue((0.05 * sales) == eB.getBonus(sales));
    }

    @Test     // teste para bonus do Departamento de produção
    public void testGetBonusIntParamBelowThreshould(){    //se o nº peças produzidas
for <= 25, recebe bonus 100€/peça
        int numUnits = 1;
        Assert.assertTrue(100.0  == eB.getBonus(numUnits));
    }

    @Test     // teste para bonus do Departamento de produção
    public void testGetBonusIntParamAboveThreshould(){ //se o nº peças produzidas for
> 25, recebe bonus 150€/peça
        int numUnits = 30;
        Assert.assertTrue((150 * numUnits) == eB.getBonus(numUnits));
    }

    @Test    // teste para bonus do Departamento de montagem
    public void testGetBonus2ParamsBelowThreshould(){     //se o nº peças montadas for
<= 250 (part-time), não recebe bonus
        int UNITS_PT = 250;              //se o nº peças montadas for <= 700 (full-
time), não recebe bonus
        int UNITS_FT = 700;
        int numUnits = 200;
        Assert.assertTrue(0.0 == eB.getBonus(numUnits, UNITS_PT));
        Assert.assertTrue(0.0 == eB.getBonus(numUnits, UNITS_FT));
    }

    @Test    // teste para bonus do Departamento de montagem
    public void testGetBonus2ParamAboveThreshould(){//se o nº peças montadas for > 250
(part-time), recebe bonus 0.10€/peça
        int UNITS_PT = 250;              //se o nº peças montadas for > 700 (full-
time), recebe bonus 0.10€/peça
        int UNITS_FT = 700;
        int numUnits = 800;
        Assert.assertTrue((0.10 * numUnits)  == eB.getBonus(numUnits, UNITS_PT));
        Assert.assertTrue((0.10 * numUnits) == eB.getBonus(numUnits, UNITS_FT));
    }
}
```