Disciplina: **Teste**

Antonio Carlos Ferreira de Almeida - RA:2438194

Aberto: segunda, 30 jan 2023, 08:00
Vencimento: domingo, 12 fev 2023, 23:59
Seguindo um ciclo *Test Driven Development* (TDD), desenvolva as classes necessárias (usando a linguagem Java e o *framework* JUnit) para resolver o problema descrito abaixo:

"O participante deve implementar uma calculadora de salário de funcionários. Um funcionário contém nome, e-mail, salário-base e cargo. De acordo com seu cargo, a regra para cálculo do salário líquido é diferente:

- *Caso o cargo seja DESENVOLVEDOR, o funcionário terá desconto de 20% caso o salário seja maior ou igual que 3.000,00, ou apenas 10% caso o salário seja menor que isso.*
- *Caso o cargo seja DBA, o funcionário terá desconto de 25% caso o salário seja maior ou igual que 2.000,00, ou apenas 15% caso o salário seja menor que isso.*
- *Caso o cargo seja TESTADOR, o funcionário terá desconto de 25% caso o salário seja maior ou igual que 2.000,00, ou apenas 15% caso o salário seja menor que isso.*
- *Caso o cargo seja GERENTE, o funcionário terá desconto de 30% caso o salário seja maior ou igual que 5.000,00, ou apenas 20% caso o salário seja menor que isso.*

Exemplos de cálculo do salário:

- *DESENVOLVEDOR com salário-base 5,000.00. Salário final = 4.000,00*
- *GERENTE com salário-base de 2.500,00. Salário final: 2.000,00*
- *TESTADOR com salário de 550.00. Salário final: 467,50*

Entregue o código fonte das classes requisitadas em um arquivo único .pdf (coloque o código necessário para responder os itens usando seu editor de texto favorito e salve em um arquivo .pdf para entrega).

Respostas:

01.

```java

package br.calcSalarFunc;

import java.math.BigDecimal;

public class CalculadoraDeSalarioDeFuncionários {

    public void calcularReajuste(Funcionarios funcionario) {

        final int menorque = -1;

        final BigDecimal salarioBaseDesenvolvedor = new BigDecimal("3000.0").setScale(2, RoundingMode.HALF_UP);
        final BigDecimal salarioBaseDBA = new BigDecimal("2000.0").setScale(2, RoundingMode.HALF_UP);
        final BigDecimal salarioBaseTestador = new BigDecimal("2000.0").setScale(2, RoundingMode.HALF_UP);
        final BigDecimal salarioBaseGerente = new BigDecimal("5000.0").setScale(2, RoundingMode.HALF_UP);

        final BigDecimal noventaPercent = new BigDecimal(".9");
        final BigDecimal oitentaECincoPercent = new BigDecimal(".85");
        final BigDecimal oitentaPercent = new BigDecimal(".8");
        final BigDecimal setentaECincoentaPercent = new BigDecimal(".75");
        final BigDecimal setentaPercent = new BigDecimal(".7");

        switch (funcionario.getCargo()) {
        case "Desenvolvedor":
            if (funcionario.getSalarioBase().compareTo(salarioBaseDesenvolvedor) == menorque) {
                funcionario.setSalarioBase(
                        funcionario.getSalarioBase().multiply(noventaPercent
                                .setScale(2, RoundingMode.HALF_UP)));
            } else {
                funcionario.setSalarioBase(
                        funcionario.getSalarioBase().multiply(oitentaPercent
                                .setScale(2, RoundingMode.HALF_UP)));
            }

              break;

         case "DBA":
             if (funcionario.getSalarioBase().compareTo(salarioBaseDBA) == menorque) {
                 funcionario.setSalarioBase(
                         funcionario.getSalarioBase().multiply(oitentaECincoPercent
                                 .setScale(2, RoundingMode.HALF_UP)));
             } else {
                 funcionario.setSalarioBase(
                         funcionario.getSalarioBase().multiply(setentaECincoentaPercent
                                 .setScale(2, RoundingMode.HALF_UP)));
             }
             break;

         case "Testador":
             if (funcionario.getSalarioBase().compareTo(salarioBaseTestador) == menorque) {
                 funcionario.setSalarioBase(
                         funcionario.getSalarioBase().multiply(oitentaECincoPercent
                                 .setScale(2, RoundingMode.HALF_UP)));
             } else {
                 funcionario.setSalarioBase(
                         funcionario.getSalarioBase().multiply(setentaECincoentaPercent
                                 .setScale(2, RoundingMode.HALF_UP)));
             }
             break;

         case "Gerente":
             if (funcionario.getSalarioBase().compareTo(salarioBaseGerente) == menorque) {
                 funcionario.setSalarioBase(
                         funcionario.getSalarioBase().multiply(oitentaPercent
                                 .setScale(2, RoundingMode.HALF_UP)));

               } else {
                   funcionario.setSalarioBase(
                           funcionario.getSalarioBase().multiply(setentaPercent
                                   .setScale(2, RoundingMode.HALF_UP)));
               }
               break;
           }
       }
}
```

2.

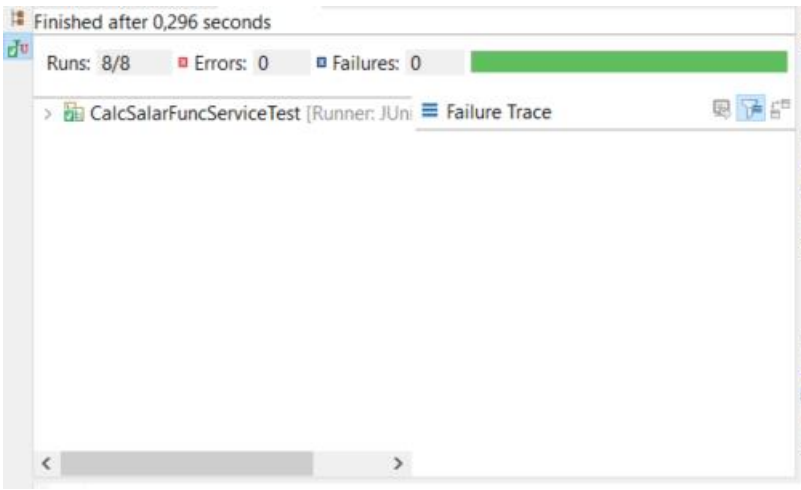
```java
package br.calcSalarFunc;

import java.math.BigDecimal;

public class Funcionarios {
    private String nome;
    private String email;
    private BigDecimal salarioBase;
    private String cargo;

    public Funcionarios(String nome, String email, BigDecimal salarioBase, String cargo) {
        this.nome = nome;
        this.email = email;
        this.salarioBase = salarioBase;
        this.cargo = cargo;
    }

    public String getNome() {return nome;}

    public void setNome(String nome) {this.nome = nome;}

    public String getEmail() {return email;}

    public void setEmail(String email) {this.email = email;}

    public BigDecimal getSalarioBase() {return salarioBase;}

    public void setSalarioBase(BigDecimal salarioBase) {this.salarioBase = salarioBase;}

    public String getCargo() {return cargo;}

    public void setCargo(String cargo) {this.cargo = cargo;}
}
```

3.

```java
package serviceTest;

import static org.junit.Assert.assertEquals;
import java.math.BigDecimal;
import java.math.RoundingMode;

import org.junit.jupiter.api.Test;
import br.calcSalarFunc.CalculadoraDeSalarioDeFuncionários;
import br.calcSalarFunc.Funcionarios;

public class CalcSalarFuncServiceTest {

    CalculadoraDeSalarioDeFuncionários calculadoraDeSalarioDeFuncionários =
            new CalculadoraDeSalarioDeFuncionários();

    @Test
    public void casoCargoSejaDesenvolvedorDescontoEmFuncaoDaFaixaSalarialUp() {
        Funcionarios funcionario = new Funcionarios("ACFA", "acfa@email.com",
                new BigDecimal("5000.00").setScale(2, RoundingMode.HALF_UP), "Desenvolvedor");
        calculadoraDeSalarioDeFuncionários.calcularReajuste(funcionario);
        assertEquals(new BigDecimal("4000.0000"), funcionario.getSalarioBase());
    }

    @Test
    public void casoCargoSejaDesenvolvedorDescontoEmFuncaoDaFaixaSalarialDown() {

        Funcionarios funcionario = new Funcionarios("ACFA", "acfa@email.com",
                new BigDecimal("2500.00").setScale(2, RoundingMode.HALF_UP), "Desenvolvedor");
        calculadoraDeSalarioDeFuncionários.calcularReajuste(funcionario);
        assertEquals(new BigDecimal("2250.0000"), funcionario.getSalarioBase());
    }

}
```

```java
    @Test
    public void casoCargoSejaDBADescontoEmFuncaoDaFaixaSalarialUp() {
        Funcionarios funcionario = new Funcionarios("ACFA", "acfa@email.com",
                new BigDecimal("3000.00").setScale(2, RoundingMode.HALF_UP), "DBA");
        calculadoraDeSalarioDeFuncionários.calcularReajuste(funcionario);
        assertEquals(new BigDecimal("2250.0000"), funcionario.getSalarioBase());
    }

    @Test
    public void casoCargoSejaDBADescontoEmFuncaoDaFaixaSalarialDown() {
        Funcionarios funcionario = new Funcionarios("ACFA", "acfa@email.com",
                new BigDecimal("550.00").setScale(2, RoundingMode.HALF_UP), "DBA");
        calculadoraDeSalarioDeFuncionários.calcularReajuste(funcionario);
        assertEquals(new BigDecimal("467.5000"), funcionario.getSalarioBase());
    }

    @Test
    public void casoCargoSejaTestadorDescontoEmFuncaoDaFaixaSalarialUp() {
        Funcionarios funcionario = new Funcionarios("ACFA", "acfa@email.com",
                new BigDecimal("5000.00").setScale(2, RoundingMode.HALF_UP), "Testador");
        calculadoraDeSalarioDeFuncionários.calcularReajuste(funcionario);
        assertEquals(new BigDecimal("3750.0000"), funcionario.getSalarioBase());
    }

    @Test
    public void casoCargoSejaTestadorDescontoEmFuncaoDaFaixaSalarialDown() {
        Funcionarios funcionario = new Funcionarios("ACFA", "acfa@email.com",
                new BigDecimal("550.00").setScale(2, RoundingMode.HALF_UP), "Testador");
        calculadoraDeSalarioDeFuncionários.calcularReajuste(funcionario);
        assertEquals(new BigDecimal("467.5000"), funcionario.getSalarioBase());
    }
```

```java
    @Test
    public void casoCargoSejaGerenteDescontoEmFuncaoDaFaixaSalarialUp() {
        Funcionarios funcionario = new Funcionarios("ACFA", "acfa@email.com",
                new BigDecimal("5000.00").setScale(2, RoundingMode.HALF_UP), "Gerente");
        calculadoraDeSalarioDeFuncionários.calcularReajuste(funcionario);
        assertEquals(new BigDecimal("3500.0000"), funcionario.getSalarioBase());
    }

    @Test
    public void casoCargoSejaGerenteDescontoEmFuncaoDaFaixaSalarialDown() {
        Funcionarios funcionario = new Funcionarios("ACFA", "acfa@email.com",
                new BigDecimal("2500.00").setScale(2, RoundingMode.HALF_UP), "Gerente");
        calculadoraDeSalarioDeFuncionários.calcularReajuste(funcionario);
        assertEquals(new BigDecimal("2000.0000"), funcionario.getSalarioBase());
    }
}
```




```
getSalarioBase());

lUp() {
cfa@email.com",
Mode.HALF_UP), "DBA");
ncionario);
getSalarioBase());

lDown() {
cfa@email.com",
ode.HALF_UP), "DBA");
ncionario);
etSalarioBase());
```