Faculdade: Universidade Tecnológica Federal do Paraná

Curso: Especialização em Tecnologia Java

Nome: Erik Eduardo Valcezio RA: A2329611

DATA: 30/01/2022

Atividade: Teste de Software

Exercicio3

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.

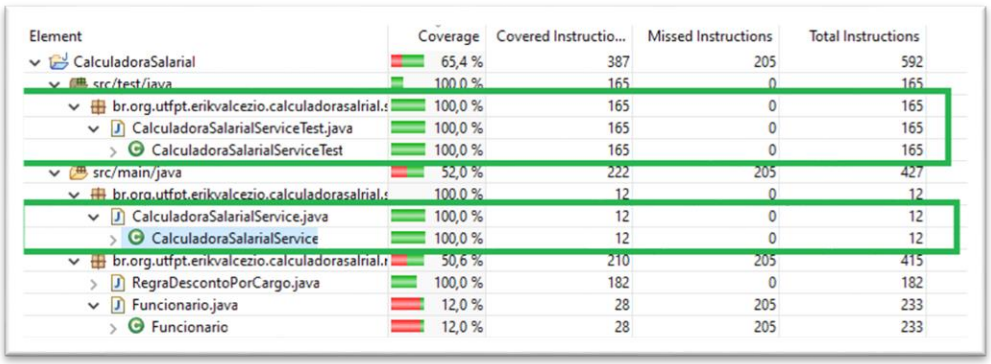
Caso o cargo seja GERENTE, o funcionário terá desconto de 30% caso o salário seja maior ou igual que 5.000,00, ou apenas 20% caso o salário seja menor que isso.

Exemplos de cálculo do salário:

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

GIT: https://github.com/erikvalcezio/java-projects-utfpr/tree/develop/Teste-De-Software/CalculadoraRegraNegocio_Strategy_TDD

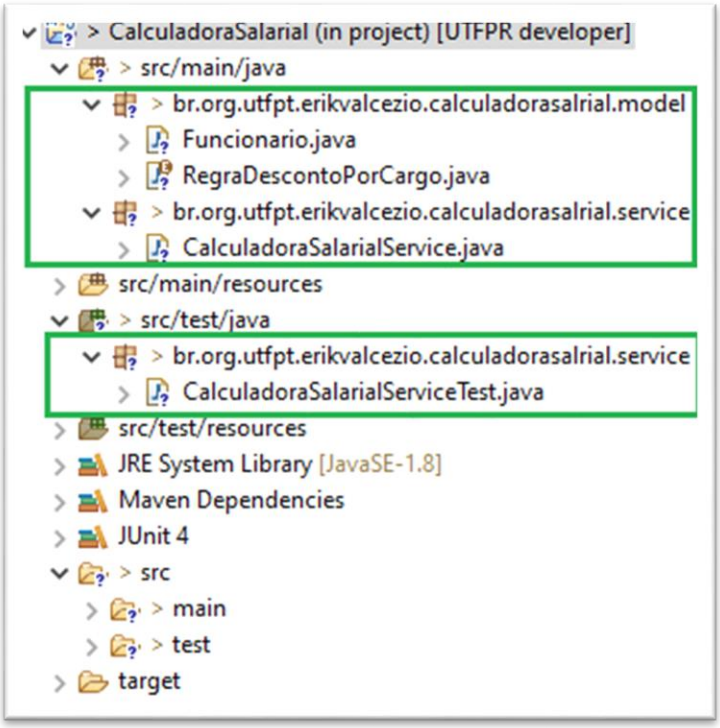
Cobertura de teste com 100% na regra de negócio conforme os cenários:



Element	Coverage	Covered Instructio...	Missed Instructions	Total Instructions
CalculadoraSalarial	65,4 %	387	205	592
src/test/java	100,0 %	165	0	165
br.org.utfpt.erikvalcezio.calculadorasalrial.s	100,0 %	165	0	165
CalculadoraSalarialServiceTest.java	100,0 %	165	0	165
CalculadoraSalarialServiceTest	100,0 %	165	0	165
src/main/java	52,0 %	222	205	427
br.org.utfpt.erikvalcezio.calculadorasalrial.s	100,0 %	12	0	12
CalculadoraSalarialService.java	100,0 %	12	0	12
CalculadoraSalarialService	100,0 %	12	0	12
br.org.utfpt.erikvalcezio.calculadorasalrial.r	50,6 %	210	205	415
RegraDescontoPorCargo.java	100,0 %	182	0	182
Funcionario.java	12,0 %	28	205	233
Funcionario	12,0 %	28	205	233

Workspace

O projeto foi aplicado um simples design pattern strategy, apenas para entrega a demanda com as classes na Model (Funcionario) Enum (RegraDescontoPorCargo) e Service(CalculadoraSalarialService). Classe de teste da regra de negócio CalculadoraSalarialServiceTest.



Em abaixo as classes:

```java
package br.org.utfpt.erikvalcezio.calculadorasalrial.service;

import static org.junit.Assert.assertEquals;

@RunWith (value = JUnit4.class)
public class CalculadoraSalarialServiceTest {
    /**
     * TDD Exercício 3
     */

    /**
     * Canario 1
     * Caso o cargo seja DESENVOLVEDOR,
     * o funcionário terá desconto de 20% caso o salário seja maior ou igual que 3.000,00
     */
    @Test
    public void funcionarioTeraDescontoVintePorcentoCasoSalarioMaiorOuIgualTresMil() {
        CalculadoraSalarialService service = new CalculadoraSalarialService();
        Funcionario funcionario = new Funcionario("Joao","joao@utfpr.org.br",new BigDecimal(3000.00),"DESENVOLVEDOR");

        service.obterDesconto(funcionario, RegraDescontoPorCargo.DESENVOLVEDOR);

        assertEquals(new BigDecimal("2400.00"), funcionario.getSalarioBase());
    }

    /**
     * Canario 2
     * Caso o cargo seja DESENVOLVEDOR,
     * ou apenas 10% caso o salário seja menor que isso.
     */

    @Test
    public void funcionarioTeraDescontoDezPorcentoCasoSalarioMaiorOuIgualTresMil() {
        CalculadoraSalarialService service = new CalculadoraSalarialService();
        Funcionario funcionario = new Funcionario("Joao","joao@utfpr.org.br",new BigDecimal(2500.00),"DESENVOLVEDOR");

        service.obterDesconto(funcionario, RegraDescontoPorCargo.DESENVOLVEDOR);

        assertEquals(new BigDecimal("2250.00"), funcionario.getSalarioBase());
```

```
49      }
50
51      /**
52       * Canario 3
53       * Caso o cargo seja DBA ou TESTADOR,
54       * o funcionário terá desconto de 25% caso o salário seja maior ou
    igual que 2.000,00,
55       */
56      @Test
57      public void
    funcionarioTeraDescontoVinteCincoPorcentoCasoSalarioMaiorOuIgualDoisMil() {
58          CalculadoraSalarialService service = new CalculadoraSalarialService
    ();
59          Funcionario funcionario = new Funcionario
    ("Maria","maria@utfpr.org.br",new BigDecimal(2000.00),"DBA");
60
61          service.obterDesconto(funcionario, RegraDescontoPorCargo.DBA);
62
63          assertEquals(new BigDecimal("1500.00"), funcionario.getSalarioBase
    ());
64      }
65
66      /**
67       * Canario 4
68       * Caso o cargo seja DBA ou TESTADOR,
69       * ou apenas 15% caso o salário seja menor que isso.
70       */
71
72      @Test
73      public void
    funcionarioTeraDescontoQuinzePorcentoCasoSalarioMenorDoisMil() {
74          CalculadoraSalarialService service = new CalculadoraSalarialService
    ();
75          Funcionario funcionario = new Funcionario
    ("Joao","joao@utfpr.org.br",new BigDecimal(1500.00),"TESTADOR");
76
77          service.obterDesconto(funcionario, RegraDescontoPorCargo.TESTADOR);
78
79          assertEquals(new BigDecimal("1275.00"), funcionario.getSalarioBase
    ());
80      }
81
82      /**
83       * Canario 5
84       * Caso o cargo seja GERENTE,
85       * o funcionário terá desconto de 30% caso o salário seja maior ou
    igual que 5.000,00,
86       */
87      @Test
88      public void
```

```
   funcionarioTeraDescontoTrintaPorcentoCasoSalarioMaiorOuIgualCincoMil() {
 89        CalculadoraSalarialService service = new CalculadoraSalarialService
   ();
 90        Funcionario funcionario = new Funcionario
   ("Maria","maria@utfpr.org.br",new BigDecimal(5000.00),"GERENTE");
 91
 92        service.obterDesconto(funcionario, RegraDescontoPorCargo.GERENTE);
 93
 94        assertEquals(new BigDecimal("3500.00"), funcionario.getSalarioBase
   ());
 95    }
 96
 97    /**
 98     * Canario 6
 99     * Caso o cargo seja DBA ou GERENTE,
100     * ou apenas 20% caso o salário seja menor que isso.
101     */
102
103    @Test
104    public void
   funcionarioTeraDescontoVintePorcentoCasoSalarioMenorCincoMil() {
105        CalculadoraSalarialService service = new CalculadoraSalarialService
   ();
106        Funcionario funcionario = new Funcionario
   ("Joao","joao@utfpr.org.br",new BigDecimal(4500.00),"GERENTE");
107
108        service.obterDesconto(funcionario, RegraDescontoPorCargo.GERENTE);
109
110        assertEquals(new BigDecimal("3600.00"), funcionario.getSalarioBase
   ());
111    }
112
113}
114
```

```java
package br.org.utfpt.erikvalcezio.calculadorasalrial.service;

import java.math.BigDecimal;

public class CalculadoraSalarialService {

    public void obterDesconto(Funcionario funcionario, RegraDescontoPorCargo desconto) {
        BigDecimal reajuste = desconto.percentualDeDesconto(funcionario.getSalarioBase());
        funcionario.ajustarSalarioComDesconto(reajuste);
    }
}

```

```java
package br.org.utfpt.erikvalcezio.calculadorasalrial.model;

import java.math.BigDecimal;

public enum RegraDescontoPorCargo {
    DESENVOLVEDOR(new BigDecimal(3000), 0.20, 0.10) {
        @Override
        public BigDecimal percentualDeDesconto(BigDecimal salarioFuncionario) {
            return calcularBaseDesconto(this.baseSalarial, this.basePorcetagemMaior, this.basePorcetagemMenor, salarioFuncionario);
        }
    },
    DBA(new BigDecimal(2000), 0.25, 0.15) {
        @Override
        public BigDecimal percentualDeDesconto(BigDecimal salarioFuncionario) {
            return calcularBaseDesconto(this.baseSalarial, this.basePorcetagemMaior, this.basePorcetagemMenor, salarioFuncionario);
        }
    },
    TESTADOR(new BigDecimal(2000), 0.25, 0.15) {
        @Override
        public BigDecimal percentualDeDesconto(BigDecimal salarioFuncionario) {
            return calcularBaseDesconto(this.baseSalarial, this.basePorcetagemMaior, this.basePorcetagemMenor, salarioFuncionario);
        }
    },
    GERENTE(new BigDecimal(5000), 0.30, 0.20) {
        @Override
        public BigDecimal percentualDeDesconto(BigDecimal salarioFuncionario) {
            return calcularBaseDesconto(this.baseSalarial, this.basePorcetagemMaior, this.basePorcetagemMenor, salarioFuncionario);
        }
    };

    protected BigDecimal baseSalarial;
    protected Double basePorcetagemMaior;
    protected Double basePorcetagemMenor;

    private RegraDescontoPorCargo(BigDecimal baseSalarial, Double basePorcetagemMaior, Double basePorcetagemMenor) {
        this.baseSalarial = baseSalarial;
        this.basePorcetagemMaior = basePorcetagemMaior;
        this.basePorcetagemMenor = basePorcetagemMenor;
    }

    public abstract BigDecimal percentualDeDesconto(BigDecimal
```

```
  salarioFuncionario);
42
43    private static BigDecimal calcularBaseDesconto(final BigDecimal
  baseSalarial,
44            final Double basePorcetagemMaior, final Double
  basePorcetagemMenor, BigDecimal salarioFuncionario) {
45        if (salarioFuncionario.compareTo(baseSalarial) >= 0) {
46            return salarioFuncionario.multiply(new BigDecimal
  (basePorcetagemMaior));
47        } else {
48            return salarioFuncionario.multiply(new BigDecimal
  (basePorcetagemMenor));
49        }
50    }
51}
52
```

```java
package br.org.utfpt.erikvalcezio.calculadorasalrial.model;

import java.math.BigDecimal;
import java.math.RoundingMode;

import lombok.Data;

@Data
public class Funcionario {

    private String nome;
    private String email;
    private BigDecimal salarioBase;
    private String cargo;

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


    public void ajustarSalarioComDesconto(BigDecimal reajuste) {
        this.salarioBase = this.salarioBase.subtract(reajuste).setScale(2, RoundingMode.HALF_UP);
    }

}

```