

Atividade : JUNIT

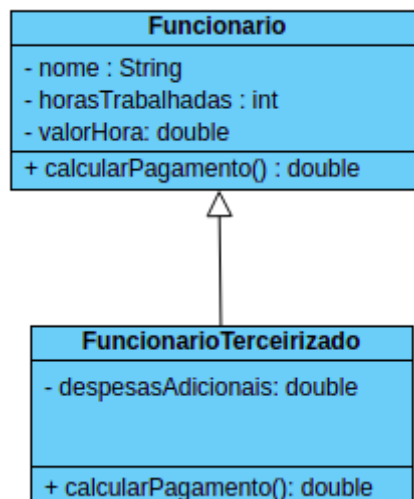
Implementar os testes para validar as classes que representam o pagamento de um funcionário.

Uma empresa possui funcionários próprios e terceirizados. Para cada funcionário, deseja-se registrar nome, horas trabalhadas e valor por hora. Funcionários terceirizados possuem ainda uma despesa adicional.

O pagamento dos funcionários corresponde ao valor da hora multiplicado pelas horas trabalhadas, sendo que os funcionários terceirizados ainda recebem um bônus correspondente a 110% de sua despesa adicional.

As regras para o cálculo do pagamento são as seguintes:

- O valor do pagamento dos funcionários deve ser maior ou igual ao valor atual do salário mínimo, que corresponde a R\$ 1.212,00. No funcionário terceirizado não considere o bônus nessa análise, ou seja, é a mesma verificação do funcionário.
- Os funcionários podem trabalhar por no máximo 40 horas.
- O valor por hora precisa ser entre 4% e 10% do salário mínimo.
- O valor das despesas adicionais não pode ultrapassar 1000,00.



Crie testes para validar os métodos da classe, inclusive o construtor e os setters. No caso do construtor e setters, você deve cobrir pelo menos dois cenários para cada: quando os dados satisfazem, e quando não satisfazem a regra de negócio (pelo menos 10 testes).

Sugestão: Tente fazer seguindo o princípio do TDD (escreva os testes antes de fazer as classes **Funcionario** e **FuncionarioTerceirizado**).

Dica1: se você estiver em dúvida sobre como o construtor e os setters devem lançar a exceção, seria algo como mostrado abaixo (você faz uma verificação no início do método, lançando uma exceção caso necessário - isso se chama “programação defensiva”):

```
public void setHorasTrabalhadas(int horasTrabalhadas) {
    if (horasTrabalhadas > 40) {
        throw new IllegalArgumentException("O número de horas trabalhadas por funcionários próprios deve ser menor ou igual a 40.");
    }
    this.horasTrabalhadas = horasTrabalhadas;
}
```

Dica2: não é muito recomendado chamar métodos setters no construtor, nesse caso, poderiam criar um método de validação e chamar tanto no construtor, como no método setter:

```
public Funcionario(String nome, int horasTrabalhadas, double valorHora) {
    this.nome = nome;
    this.horasTrabalhadas = validaHorasTrabalhadas(horasTrabalhadas);
    this.valorHora = validaValorHorasTrabalhadas(valorHora);
}

public void setHorasTrabalhadas(int horasTrabalhadas) {
    this.horasTrabalhadas = validaHorasTrabalhadas(horasTrabalhadas);
}

private int validaHorasTrabalhadas(int horasTrabalhadas) {
    if (horasTrabalhadas > 40) {
        throw new IllegalArgumentException("O número de horas trabalhadas por funcionários próprios deve ser menor ou igual a 40.");
    }
    return horasTrabalhadas;
}
```

Dica3: abaixo são listados os métodos que poderiam desenvolver:

```
public class FuncionarioTerceirizadoTest {

    public void testarConstrutorEntradaDespesasInvalida() {}

    public void testarConstrutorEntradasValida() {}

    public void testarModificarDespesasEntradaInvalida() {}

    public void testarModificarDespesasEntradaValida() {}
}

public class FuncionarioTest {

    public void testarConstrutorPagamentoInvalido() {}

    public void testarConstrutorEntradaValorHoraInvalida() {}

    public void testarConstrutorEntradaHorasInvalida() {}

    public void testarConstrutorEntradasValida() {}

    public void testarModificarHorasPagamentoInvalido() {}

    public void testarModificarHorasEntradaInvalida() {}

    public void testarModificarHorasEntradaValida() {}

    public void testarModificarValorPagamentoInvalido() {}

    public void testarModificarValorEntradaInvalida() {}

    public void testarModificarValorEntradaValida() {}

}
```