Faculdade de Informática e Administração Paulista

Domain Driven Design

Checkpoint 2

Prof. Me. Orlando C. Patriarcha

"Ciência é um conhecimento que compreendemos tão bem que podemos ensiná-lo a um computador;
Se não entendemos algo completamente,
seria uma arte lidar com isso"
- Donald Knuth - The Art of Computer programming - vol 1

Descrição do Problema

Você foi contratado para desenvolver um **sistema de gestão de funcionários** para uma empresa que possui diferentes tipos de funcionários em seu departamento. O sistema deve representar diferentes categorias de funcionários, como **Funcionários Comuns**, **Gerentes** e **Diretores**, cada um com comportamentos e atributos específicos. Além disso, deve ser possível calcular o **salário** com base no tipo de funcionário e suas particularidades, como bônus ou comissões.

Requisitos

1. Classe `Funcionario` (Superclasse)

A classe `Funcionario` deve possuir os seguintes atributos encapsulados:

- `nome`: nome do funcionário.
- `cpf`: número de CPF do funcionário.
- `salarioBase`: salário base do funcionário.

Métodos:

- Um método `getSalarioFinal()`, que retorna o salário do funcionário. Inicialmente, esse método retorna o salário base.
- Sobrecarga de construtores: a classe deve ter um construtor padrão e outro que recebe nome, CPF e salário base.

2. Classe `Gerente` (Subclasse de `Funcionario`)

A classe `Gerente` herda de `Funcionario` e possui os seguintes atributos adicionais:

- `bonus`: valor de bônus mensal do gerente.

Métodos:

- O método `getSalarioFinal()` deve ser **sobrescrito** para adicionar o bônus ao salário base.
- A classe deve possuir um construtor que receba nome, CPF, salário base e bônus, e utilize o construtor da superclasse para inicializar esses valores.

3. Classe `Diretor` (Subclasse de `Gerente`)

A classe `Diretor` herda de `Gerente` e possui os seguintes atributos adicionais:

- `porcentagemParticipacaoLucros`: porcentagem de participação nos lucros da empresa.

Métodos:

- O método `getSalarioFinal()` deve ser **sobrescrito** para calcular o salário com base no salário base, bônus e participação nos lucros. A fórmula pode ser: `salarioBase + bonus + salarioBase * porcentagemParticipacaoLucros`.
- A classe deve possuir um construtor que receba nome, CPF, salário base, bônus e participação nos lucros, e utilize o construtor da superclasse para inicializar esses valores.

4. Sobrecarga de Métodos para Cálculo de Aumento

Na classe `Funcionario`, implemente um método `aumentarSalario()` que suporte **sobrecarga**:

- Um método sem parâmetros que aumenta o salário em uma porcentagem fixa de 10%.
- Um método com um parâmetro `porcentagem` para permitir que a empresa defina o percentual de aumento.
- Um método com dois parâmetros `porcentagem` e `bonusExtra`, que aumenta o salário e adiciona um valor extra ao salário.

5. Encapsulamento

Todos os atributos das classes devem ser avaliados quanto À acessibilidade, inclusive por métodos `getter` e `setter`, garantindo o encapsulamento.

6. Erro

Quaisquer erros ou ambiguidades encontradas devem ser declaradas em arquivo README.md na raiz do projeto e serão analisados.

Desafio Extra

Crie uma **classe Departamento**, que tenha um array de funcionários com capacidade fixa. A classe deve possuir métodos para:

- Adicionar funcionários ao departamento.
- Calcular a folha de pagamento total de todos os funcionários do departamento (usando o método `getSalarioFinal()` de cada funcionário).

Exemplo de Implementação