Exercício: Sistema de Funcionários como uma Sintaxe de Animais de exemplo (Herança em Python)

Exercício: Sistema de Funcionários

Crie um mini-sistema para modelar funcionários de uma empresa usando herança.

Requisitos:

Classe base: Funcionario

Atributos: nome (str), salario (float)

Métodos:

• descrever(): imprime nome e salário

• calcular_bonus(): retorna um bônus genérico de 5% do salário

Classe derivada: Gerente (herda de Funcionario)

Atributos adicionais: equipe (lista de strings com nomes)

Regras:

• calcular_bonus(): sobrescrever para 10% do salário

• adicionar_membro(nome): adiciona um nome à equipe

• descrever(): incluir também o tamanho da equipe

Classe derivada: Desenvolvedor (herda de Funcionario)

Atributos adicionais: linguagem (str)

Regras:

• calcular_bonus(): sobrescrever para 8% do salário

• descrever(): incluir também a linguagem principal

Testes obrigatórios:

• Instanciar 1 Gerente e 1 Desenvolvedor.

• Chamar descrever() para ambos.

• Mostrar o bônus calculado para cada um usando calcular_bonus().

• Para o Gerente, adicionar pelo menos 2 membros à equipe e exibir novamente a descrição.

Extras opcionais:

• Criar uma função imprimir_resumo(funcionario) que aceite qualquer Funcionario (polimorfismo) e chame descrever() e calcular_bonus().

• Validar que salário não pode ser negativo (levantar ValueError).

Sintaxe de exemplo (Animais com Herança):

```
class Animal:
    def __init__(self, nome, idade):
        self.nome = nome
        self.idade = idade

    def fazer_som(self):
        print("O animal faz um som.")

    def apresentar(self):
        print(f"Olá, eu sou {self.nome} e tenho {self.idade} anos.")

class Cachorro(Animal):  # Cachorro herda de Animal
    def __init__(self, nome, idade, raca):
        super().__init__(nome, idade)  # Chama o construtor da classe
base
        self.raca = raca

    def fazer_som(self):  # Sobrescreve o método fazer_som
        print("Au au!")

class Gato(Animal):  # Gato herda de Animal
    def __init__(self, nome, idade, cor_pelo):
        super().__init__(nome, idade)  # Chama o construtor da classe
base
        self.cor_pelo = cor_pelo

    def fazer_som(self):  # Sobrescreve o método fazer_som
        print("Miau!")
```