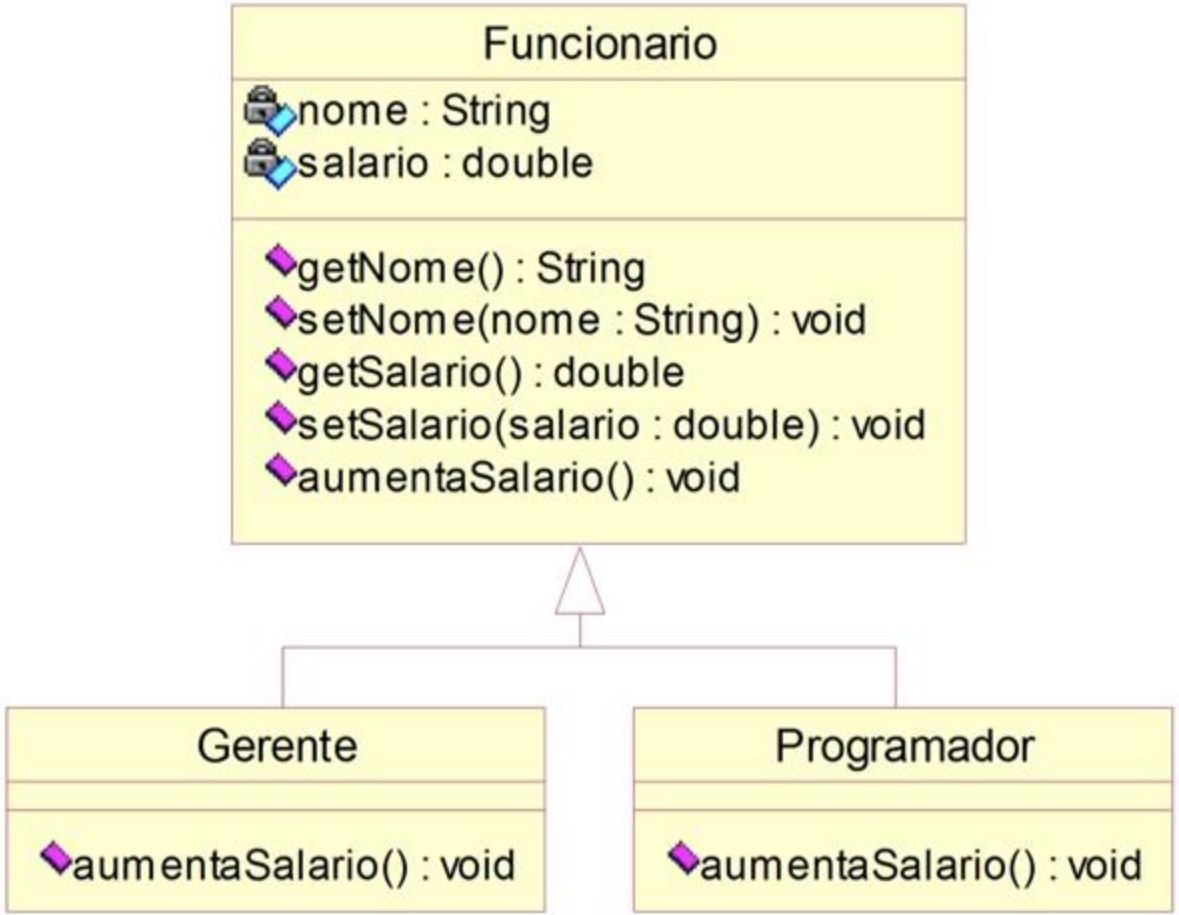
Paradigmas de Linguagem de Programação em Python

Paradigma Orientado a Objetos: Overriding

Prof. Henrique Mota



Sobreposição de Métodos

Sobreposição de Métodos
Contextualização

- Vamos supor agora que o método ***aumentaSalario( )*** da classe Funcionario no exercício anterior possui uma implementação, ou seja, o método não é mais abstrato.

- E que as subclasses Gerente e Programador mantêm suas implementações para o método ***aumentaSalario( )*** .

Sobreposição de Métodos Definição

- Quando uma subclasse declara um método com o mesmo nome, mesmo tipo de retorno e mesma lista de parâmetros de um método da sua superclasse, dizemos que ocorreu uma **sobreposição ou redefinição de método (overriding)**.
- Um método redefinido em uma subclasse oculta o método da classe ancestral a partir da subclasse.
- **Métodos privados não podem ser sobrepostos.**

Sobreposição de Métodos Observação

- O nome e a lista de parâmetros de um método é chamado de **assinatura do método**.

Sobreposição x Sobrecarga

- **Sobrecarga (*overloading*)** significa que pode-se ter métodos de **mesmo nome,** mas que **difiram na lista de parâmetros**.
- Ou seja, **métodos sobrecarregados não possuem mesma assinatura**.

Sobreposição de Métodos
Dynamic Binding

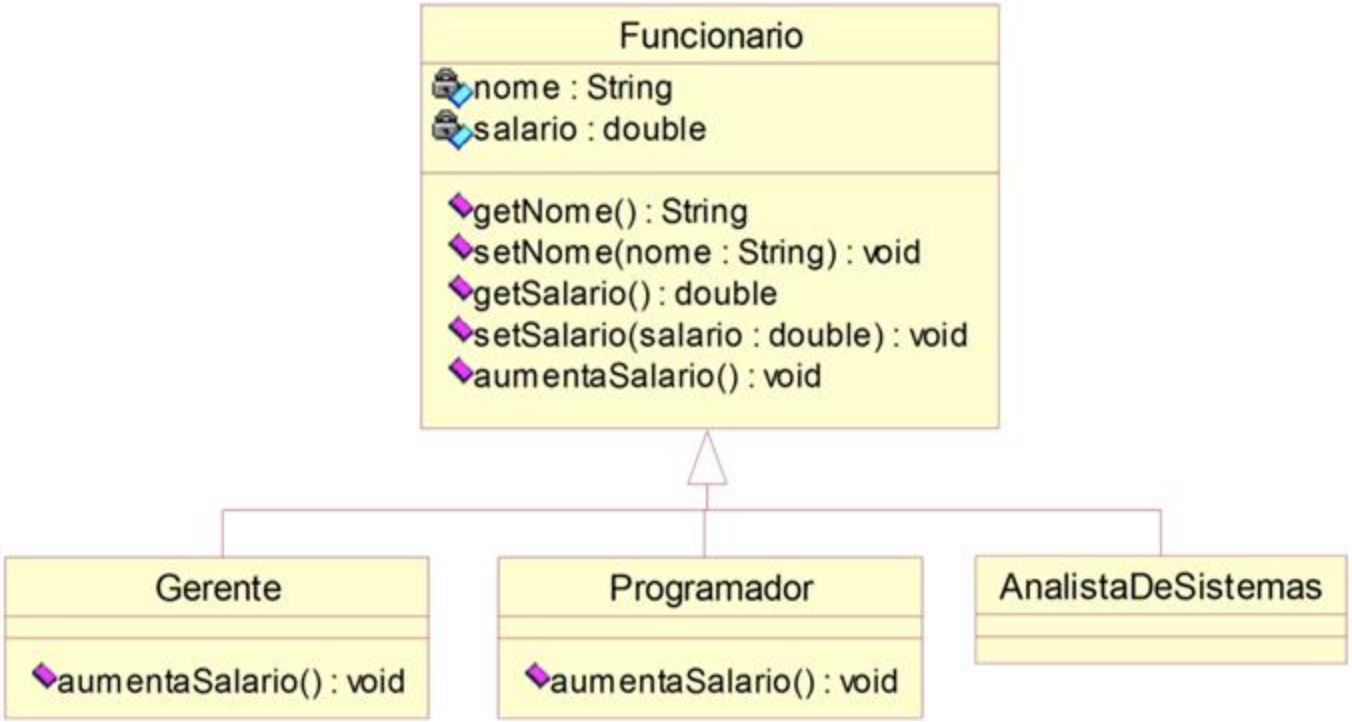
- A ligação entre a assinatura de um método e o método por ela designado efetua-se em tempo de execução.
- Este mecanismo é conhecido como ligação dinâmica (***dynamic binding***).

Lembrem-se da aula Introdução a POO...

- `__add__(self, outro)` → `self + outro`
- `__sub__(self, outro)` → `self - outro`
- `__eq__(self, outro)` → `self == outro`
- `__lt__(self, outro)` → `self < outro`
- `__len__(self)` → `len(self)`
- `__str__(self)` → `print(self)`

- Os métodos especiais **sobrescrevem** os métodos e operações originais, ou sejam, **fazem overriding**

Exercício 1


Funcionario
nome : String
salario : double
getNome() : String
setNome(nome : String) : void
getSalario() : double
setSalario(salario : double) : void
aumentaSalario() : void
Gerente
aumentaSalario() : void
Programador
aumentaSalario() : void
AnalistaDeSistemas

Exercício 1
Considerações

- A classe **Funcionario**, **Gerente** e **Programador**!
- Uma chamada ao ***aumentaSalario( )*** do Funcionario aumenta seu salário em 5%.
- Uma chamada ao ***aumentaSalario( )*** do Gerente aumenta seu salário em 10%.
- Uma chamada ao ***aumentaSalario( )*** do Programador aumenta seu salário em 20%.

Exercício 1
Aplicação

- Implemente uma aplicação que declara três variáveis do tipo **Funcionario** e cria três objetos um do tipo **Gerente** , outro do tipo **Programador** e o terceiro do tipo **AnalistaDeSistemas** . Em seguida, o programa deve oferecer um menu para o usuário com as seguintes opções:
  - **Imprimir dados** – O usuário deverá informar se ele deseja imprimir os dados do Gerente, do Programador ou do AnalistaDeSistemas.
  - **Aumentar salário** – O usuário deverá informar se ele deseja aumentar o salário do Gerente, do Programador ou do AnalistaDeSistemas.

Exercício 2

- Implemente uma classe Conta que contenha os atributos nome do cliente, número da conta, saldo e limite. Estes valores deverão ser informados no construtor, sendo que o limite não poderá ser maior que o valor do salário mensal do cliente.
- Implemente também um método depósito e outro método saque. O método saque retorna um booleano indicando se o saque pôde ser efetuado ou não.
- Implemente uma classe ContaEspecial que funciona da mesma forma que a classe Conta, mas que aceita um limite de até 3 vezes o valor do salário do cliente.

Obrigado!

Alguma dúvida?

Prof. Henrique Mota

profhenriquemota@gmail.com