PROGRAMAÇÃO ORIENTADA A OBJETOS - POO

Sistemas para Internet

Prof.º: Paulo Silva

Contato:

(96) 99142-4429

FACULDADE META

Polimorfismo

Aula 4

FACULDADE META

Polimorfismo

Polimorfismo é a capacidade de um objeto poder ser referenciado de várias formas. (cuidado, polimorfismo não quer dizer que o objeto fica se transformando, muito pelo contrário, um objeto nasce de um tipo e morre daquele tipo, o que pode mudar é a maneira como nos referimos a ele).

De forma genérica, polimorfismo significa **várias formas**. No caso da Orientação a Objetos, polimorfismo denota uma situação na qual um objeto pode se comportar de maneiras diferentes ao receber uma mensagem, dependendo do seu tipo de criação.

Polimorfismo

Na herança, vimos que todo **Gerente** é um **Funcionario**, pois é uma extensão deste.

Podemos nos referir a um **Gerente** como sendo um **Funcionario**. Se alguém precisa falar com um **Funcionario** do banco, pode falar com um **Gerente**! Porque? Pois Gerente é um Funcionario. Essa é a semântica da herança.

Exemplo - Exercício

Imagine que vamos modelar um sistema para a faculdade que controle as despesas com funcionários e professores. Crie a Classe abaixo:

EmpregadoDaFaculdade
- nome: String - salario: float
+ setSalario(salario:float) + getSalario(): float + setNome(nome:String) + getNome(): String + getInfo() :String

```
getInfo () {
return "nome: " + $this->nome + " com salário "
+ $this->salario;
}
```

FACULDADE MEIA

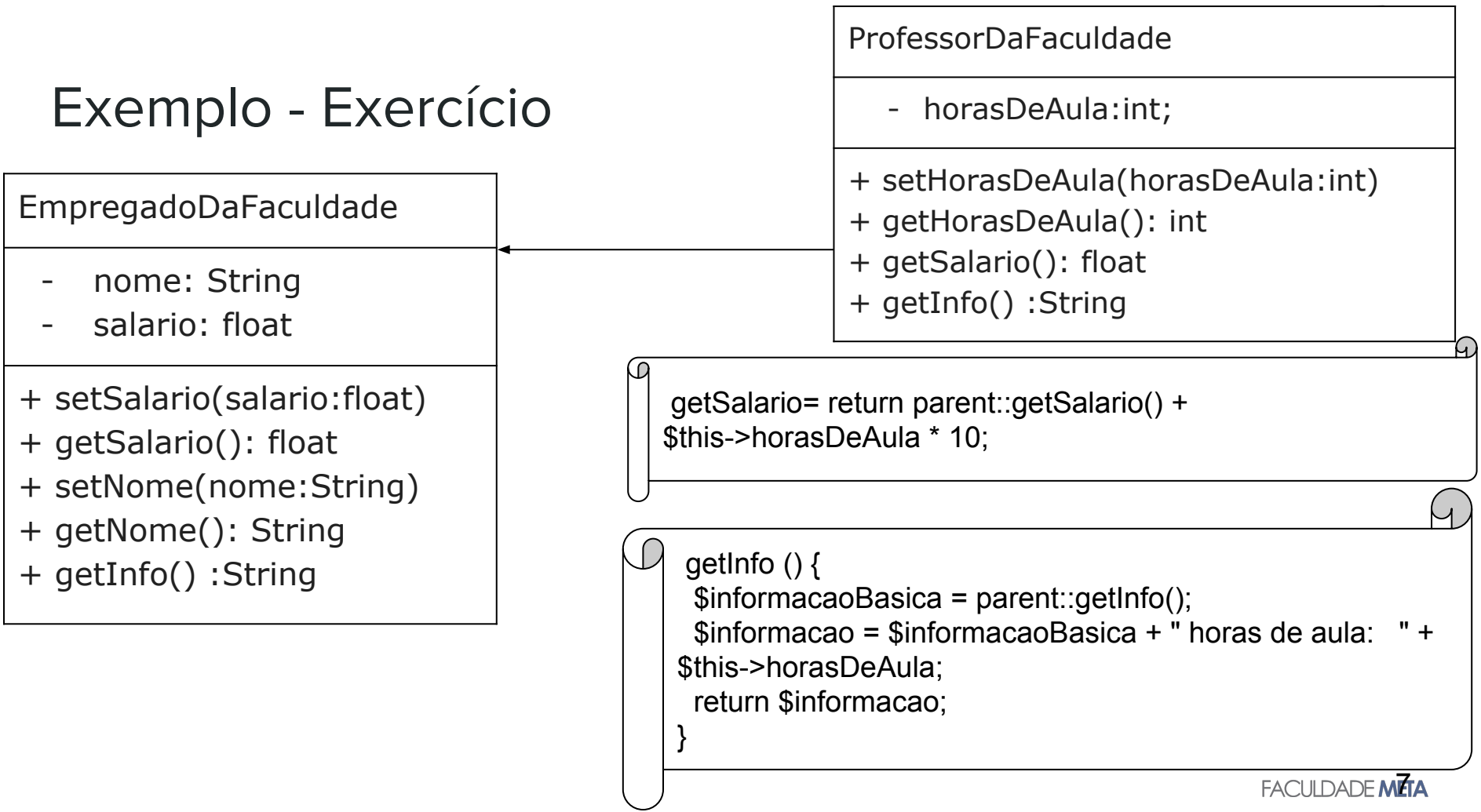
Exemplo - Exercício

O gasto que temos com o professor não é apenas seu salário. Temos de somar um bônus de 10 reais por hora/aula.

O que fazemos então? Reescrevemos o método. Assim como o getSalario é diferente, o getInfo também será, pois temos de mostrar as horas/aula também.

Crie a seguinte estrutura de classes.

Exemplo - Exercício



Como tiramos proveito do polimorfismo?

Como tiramos proveito do polimorfismo?

Imagine que temos uma classe de relatório:

GeradorDeRelatorio
+ adiciona(EmpregadoDaFaculdade $funcionario)

```
adiciona () {
        echo $funcionario.getInfo();
        echo $funcionario.getSalario();
}
```

Podemos passar para nossa classe qualquer EmpregadoDaFaculdade! Vai funcionar tanto para professor, quanto para funcionário comum.

Classes Abstratas

Classes Abstratas

Pode-se dizer que as **classes abstratas** servem como “modelo” para outras classes que dela herdem, não podendo ser instanciada por si só.

Para ter um objeto de uma classe abstrata é necessário criar uma classe mais especializada herdando dela e então instanciar essa nova classe. Os métodos da classe abstrata devem então serem sobrescritos nas classes filhas.

Por exemplo, é definido que a classe “Animal” seja herdada pelas subclasses “Gato”, “Cachorro”, “Cavalo”, mas ela mesma nunca pode ser instanciada.

Nossa classe Funcionario

O que, exatamente, vem a ser a nossa classe Funcionario? Nossa empresa tem apenas Diretores, Gerentes, Secretárias, etc. Ela é uma classe que apenas idealiza um tipo, define apenas um rascunho.

Para o nosso sistema, é inadmissível que um objeto seja apenas do tipo Funcionario (pode existir um sistema em que faça sentido ter objetos do tipo Funcionario ou apenas Pessoa, mas, no nosso caso, não).

Usamos a palavra chave **abstract** para impedir que ela possa ser instanciada.

Exemplo


```php
<?php

abstract class Funcionario
{
    protected $nome;
    protected $cpf;
    protected $salario;

    public function getBonificacao() {
        return $this->salario * 0.10;
    }
}
```

Tente instanciar a classe Funcionario.

FACULDADE MEIA

Métodos abstratos

Métodos abstratos

Se o método getBonificacao não fosse reescrito, ele seria herdado da classe mãe, fazendo com que devolvesse o salário mais 10%.

Levando em consideração que cada funcionário em nosso sistema tem uma regra totalmente diferente para ser bonificado, faz algum sentido ter esse método na classe Funcionario? Será que existe uma bonificação padrão para todo tipo de Funcionario? Parece que não, cada classe filha terá um método diferente de bonificação pois, de acordo com nosso sistema, não existe uma regra geral.

Métodos abstratos

Poderíamos, então, jogar fora esse método da classe Funcionario? O problema é que, se ele não existisse, não poderíamos chamar o método apenas com uma referência a um Funcionario, pois ninguém garante que essa referência aponta para um objeto que possui esse método.

Existe um recurso em PHP que, em uma classe abstrata, podemos escrever que determinado método será sempre escrito pelas classes filhas. Isto é, um **método abstrato**.

Ele indica que todas as classes **filhas concretas** devem reescrever esse método.

Como declarar um método abstrato?

Basta escrever a palavra chave **abstract** na assinatura do método e colocar um ponto e vírgula em vez de abre e fecha chaves:

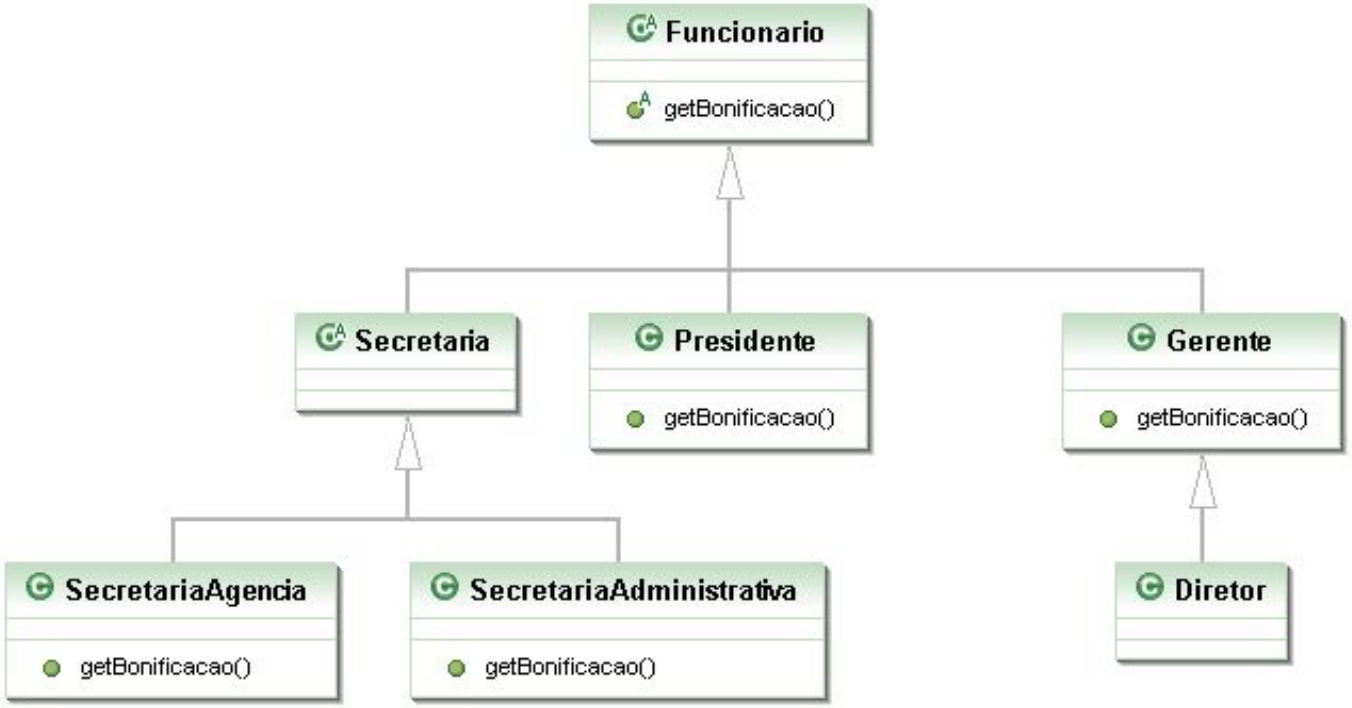
```php
<?php

abstract class Funcionario
{
    protected $nome;
    protected $cpf;
    protected $salario;

    public abstract function getBonificacao();
}
```

Exercício

Crie a seguinte estrutura abaixo;



Obrigado!

Referências Bibliográficas.

MANZANO, José Augusto G., COSTA JR., Roberto da. **Programação de Computadores com Java**. Érica, 2014.

FURGERI, Sérgio. **Java 8 - Ensino Didático - Desenvolvimento e Implementação de Aplicações**. Érica, 2015.