PROGRAMAÇÃO ORIENTADA A OBJETOS - POO

Sistemas para Internet

Prof.º: Paulo Silva

Contato:

(96) 99142-4429

FACULDADE META

Herança

Herança

A herança é um princípio da POO que permite a criação de novas classes a partir de outras previamente criadas.

Essas novas classes são chamadas de **subclasses**, ou classes derivadas; e as classes já existentes, que deram origem às subclasses, são chamadas de **superclasses**, ou classes base. Deste modo é possível criar uma hierarquia dessas classes, tornando, assim, classes mais amplas e classes mais específicas. Uma subclasse **herda** métodos e atributos de sua superclasse; apesar disso, pode escrevê-los novamente para uma forma mais específica de representar o comportamento do método herdado.

Herança

Como toda empresa, um Banco possui funcionários. Vamos modelar a classe Funcionario:

Funcionario
+ nome: string + cpf: string + salario: float

Além de um funcionário comum, há também outros cargos, como os **gerentes**. Os gerentes guardam a mesma informação que um funcionário comum, mas possuem outras informações, além de ter funcionalidades um pouco diferentes.

Herança

Um gerente no nosso banco possui também uma senha numérica que permite o acesso ao sistema interno do banco, além do número de funcionários que ele gerencia:

Gerente
+nome: string +cpf: string +salario: float +senha: int +numeroDeFuncionariosGerenciados: int
+boolean autentica(senha:int)

Se tivéssemos um outro tipo de funcionário que tem características diferentes do funcionário comum, precisaríamos criar uma outra classe e copiar o código novamente!

Herança

Existe um jeito, em PHP, de relacionarmos uma classe de tal maneira que uma delas **herda** tudo que a outra tem.

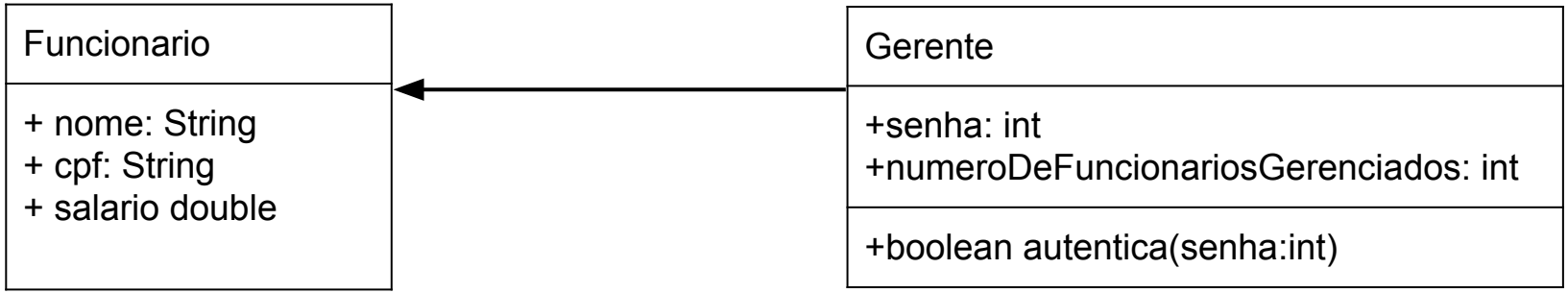
Isto é uma relação de classe mãe e classe filha. No nosso caso, gostaríamos de fazer com que o **Gerente** tivesse tudo que um **Funcionario** tem, gostaríamos que ela fosse uma extensão de **Funcionario**. Fazemos isto através da palavra chave **extends**

Exemplo:


```php
<?php

class Gerente extends Funcionario
{
    protected $senha;
    protected $numeroDeFuncionariosGerenciados;

    public function autentica($senha)
    {
        if($senha === $this->senha) {
            echo "Acesso permitido!";
            return true;
        } else {
            echo "Acesso Negado!";
            return false;
        }
    }

```

Herança

Em todo momento que criarmos um objeto do tipo **Gerente**, este objeto possuirá também os **atributos** definidos na **classe Funcionario**, pois um **Gerente** é um **Funcionario**:



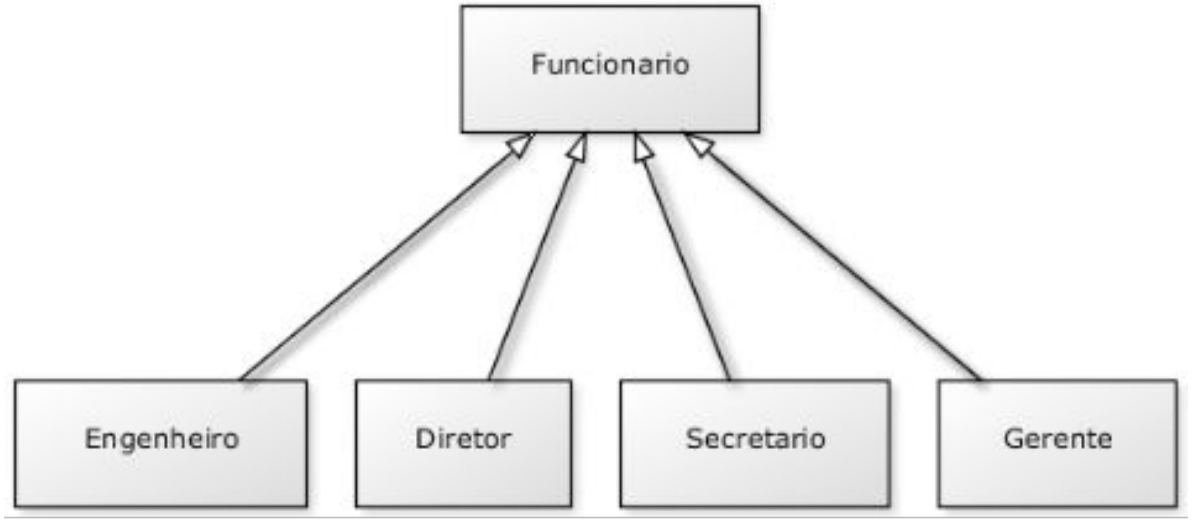
Herança

Funcionario herda os atributos e métodos privados, porém não consegue acessá-los diretamente.

Para acessar um membro privado na filha indiretamente, seria necessário que a mãe expusesse um outro método visível que invocasse esse atributo ou método privado.

Exercícios

1. Da mesma maneira, que foi criado a subclasse Gerente crie a estrutura de classe abaixo;



Reescrita de método

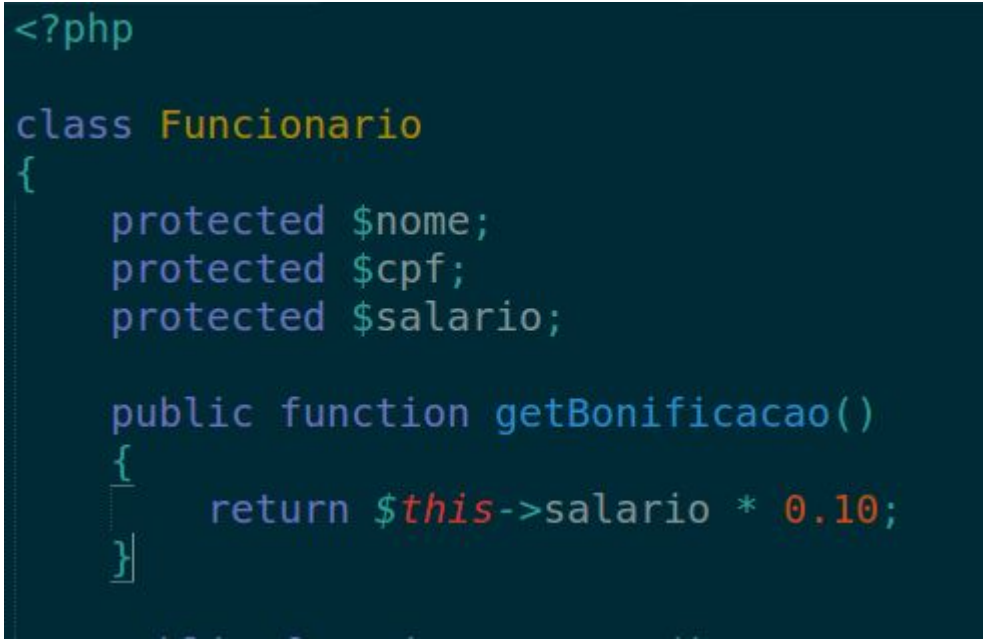
Reescrita de método

No PHP, quando herdamos um método, podemos alterar seu comportamento. Podemos reescrever (sobrescrever, **override**) este método:

Exemplo:

Todo fim de ano, os funcionários do nosso banco recebem uma bonificação. Os funcionários comuns recebem 10% do valor do salário e os gerentes, 15%.

classe Funcionario



Invocando o método reescrito

Depois de reescrito, não podemos mais chamar o método antigo que fora herdado da classe mãe: realmente alteramos o seu comportamento.

Mas podemos invocá-lo no caso de estarmos dentro da classe.

Imagine que para calcular a bonificação de um Diretor devemos fazer igual ao cálculo de um Funcionario porém adicionando R$ 1000.

```php
<?php

class Diretor extends Funcionario
{
    public function getBonificacao()
    {
        return $this->salario * 0.10 + 1000;
    }
}
```

FACULDADE MEIA

Problema:

O dia que o getBonificacao do Funcionario mudar, precisaremos mudar o método do Diretor para acompanhar a nova bonificação.

Para evitar isso, o getBonificacao do Diretor pode chamar o do Funcionario utilizando a palavra chave **parent**.

```php
<?php

class Diretor extends Funcionario
{
    public function getBonificacao()
    {
        return parent::getBonificacao() + 1000;
    }
}
```

Exercícios

1. Reescreva o método getBonificacao nas outras classes filhas de funcionário os valores da bonificação são:
   Engenherio: (salario + 20%) + R$ 1500;
   Secretario: (salario + 10%) + R$ 500;

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.

FACULDADE META