

# **Técnico em Desenvolvimento de Sistemas**

## **Programação WEB 3**

**Linguagem PHP – Orientação a Objetos**

**Prof. Laércio Silva**

[laercio.silva31@etec.sp.gov.br](mailto:laercio.silva31@etec.sp.gov.br)

[Lndsilva@Hotmail.com](mailto:Lndsilva@Hotmail.com)

### Encapsulamento

- Um dos recursos mais interessantes na orientação a objetos é o encapsulamento, um mecanismo que provê proteção de acesso aos membros internos de um objeto.
- Lembre-se que uma classe possui responsabilidade sobre os atributos que contém.

## Encapsulamento

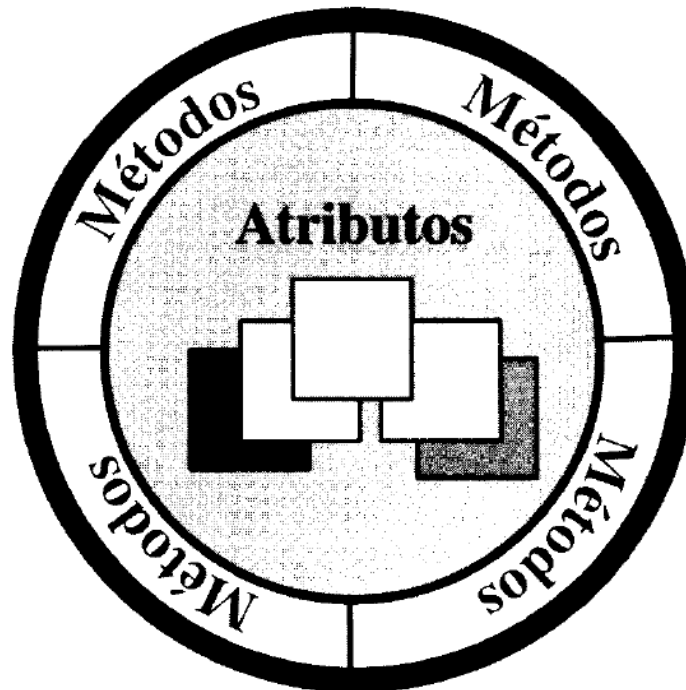
- Existem certas propriedades de uma classe que devem ser tratadas exclusivamente por métodos dela mesma
- São implementações projetadas para manipular essas propriedades da forma correta.
- As propriedades nunca devem ser acessadas diretamente de fora do escopo de uma classe, pois dessa forma a classe não fornece mais garantias sobre os atributos que contém, perdendo, assim, a responsabilidade sobre eles.

### Encapsulamento

- As propriedades são os pequenos blocos no núcleo do objeto; os métodos são os círculos maiores na parte externa do mesmo.

## Encapsulamento

- A figura foi construída para demonstrar que os métodos devem ser projetados de forma a protegerem suas propriedades, fornecendo interfaces para a manipulação destas.



### Encapsulamento

- Para atingir o encapsulamento, uma das formas é definindo a visibilidade das propriedades e dos métodos de um objeto.
- A visibilidade define a forma como essas propriedades devem ser acessadas.

# Programação a Objetos em PHP:

- Existem três formas de acesso:

Visibilidade	Descrição
Private	Membros declarados como private somente podem ser acessados dentro da própria classe em que foram declarados. Não poderão ser acessados a partir de classes descendentes nem a partir do programa que faz uso dessa classe (manipulando o objeto em si). Na UML, simbolizamos com um (-) em frente à propriedade.

# Programação a Objetos em PHP:

Visibilidade	Descrição
Protected	Membros declarados como protected somente podem ser acessados dentro da própria classe em que foram declarados e a partir de classes descendentes, mas não poderão ser acessados a partir do programa que faz uso dessa classe (manipulando o objeto em si). Na UML, simbolizamos com um(#) em frente à propriedade.



# Programação a Objetos em PHP:

Visibilidade	Descrição
Public	Membros declarados como public e poderão ser acessados livremente a partir da própria classe em que foram declarados, a partir de classes descendentes e a partir do programa que faz uso dessa classe (manipulando o objeto em si). Na UML, simbolizamos com um ( + ) em frente à propriedade.

## Classe Funcionário

- Para demonstrar a visibilidade `private`, criaremos a classe `Funcionaria` e marcaremos a maioria das propriedades como `private`.
- Dessa forma, elas só poderão ser alteradas por métodos da mesma classe. A única propriedade que deixaremos para livre acesso será o nome, marcado como `public`.

Funcionario
-Codigo: integer +Nome: string -Nascimento: date -Salario: float

## Classe Funcionário

```
Funcionario.php
1  <?php
2
3      class Funcionario{
4
5          private    $codigo;
6          public     $nome;
7          private    $nascimento;
8          protected  $salario;
9
10     }
11  ?>
12
```

# Programação a Objetos em PHP:

## Classe testa\_funcionario

```
testa_funcionario.php x
1  <?php
2
3      # carrega a classe
4      include_once 'Funcionario.php';
5
6      $pedro= new Funcionario();
7      $pedro->Salario = 'Oitocentos e setenta e seis';
8
9
10  ?>
```

**Fatal error:** Uncaught Error: Cannot access private property Funcionario::\$Salario in C:\xampp\htdocs\ProjetoPHP3H2\_2022\aula03\testa\_funcionario.php:7 Stack trace: #0 {main} thrown in C:\xampp\htdocs\ProjetoPHP3H2\_2022\aula03\testa\_funcionario.php on line 7

## Classe Funcionário

- Veja que o PHP resulta em um erro de acesso à propriedade, como esperado, mas se não podemos alterar o valor da propriedade Salario dessa forma, como faremos?
- Simples! Criaremos métodos pertencentes à classe Funcionario para manipular essa propriedade.
- No exemplo a seguir criaremos o método SetSalario() para atribuir o valor da propriedade Salario e GetSalario() para retornar o valor. Aproveitamos para realizar uma validação no método SetSalario(), que somente atribuirá o valor de Salario caso este seja de um tipo numérico e positivo.

# Programação a Objetos em PHP:

## Métodos de acesso Get e Set

```
Funcionario.php
1  <?php
2
3      class Funcionario{
4
5          private $codigo;
6          public $nome;
7          private $nascimento;
8          private $salario;
9
10         /* método SetSalario
11          * atribui o parâmetro $Salario à propriedade $Salario
12          */
13         function setSalario($salario){
14             //verifica se é numérico e positivo
15             if(is_numeric($salario)&&($salario > 0))
16             {
17                 $this->salario = $salario;
18             }
19         }
20
21         /* método GetSalario
22          * retorna o valor da propriedade $Salario
23          */
24         function getSalario(){
25             return $this->salario;
26         }
27     }
28 ?>
29
```

## Métodos de acesso Get e Set

```
testa_funcionario.php x
1  <?php
2
3      # carrega a classe
4      include_once 'Funcionario.php';
5
6      //instancia novo Funcionario
7      $pedro= new Funcionario();
8
9      //atribui novo Salário
10     $pedro->setSalario(876);
11
12     //obtem o Salário
13     echo 'Salário: R$ ' . $pedro->getSalario();
14
15     ?>
16
```



localhost/ProjetoPHP3H2\_2022/aula03/testa\_funcionario.php

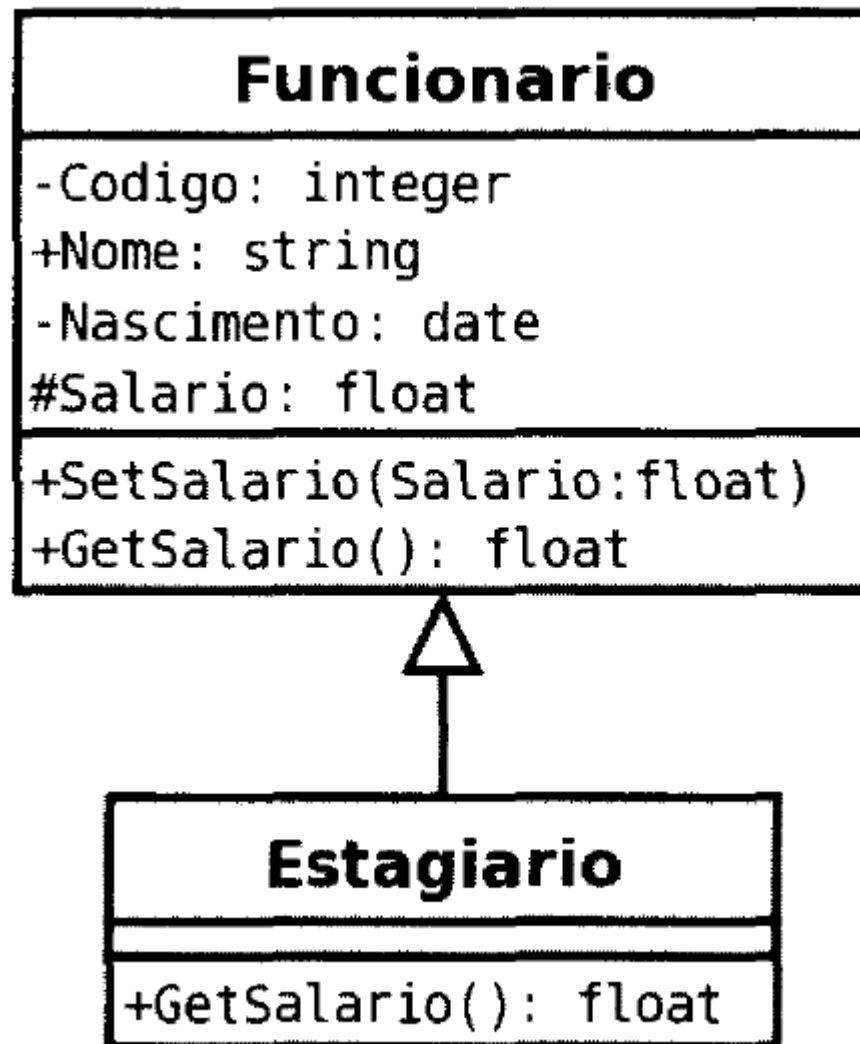
Salário: R\$ 876

## Protected

- Para demonstrar a visibilidade protected, vamos especializar a classe Funcionário, criando a classe Estagiaria.
- A única característica exclusiva de um estagiário é que o seu salário é acrescido de 12% de bônus.
- Para tanto, sobrescreveremos o método GetSalario() para que este retorne o salário já reajustado.



## Classe estagiário



# Programação a Objetos em PHP:

## Protected

```
Estagiario.php x
1  <?php
2
3      class Estagiario extends Funcionario{
4
5          /*método GetSalario sobreescrito retorna o $Salário com 12% de bônus.*/
6          function getSalario(){
7              return $this->salario * 1.12;
8          }
9      }
10
11  ?>
12
```

## Classe Estagiário

```
Estagiario.php x
1 <?php
2
3 class Estagiario extends Funcionario{
4
5     /*método GetSalario sobreescritoreturna o $Salário com 12% de bônus.*/
6     function getSalario(){
7         return $this->salario * 1.12;
8     }
9 }
10
11 ?>
12
```

# Programação a Objetos em PHP:

## Classe testa\_estagiário

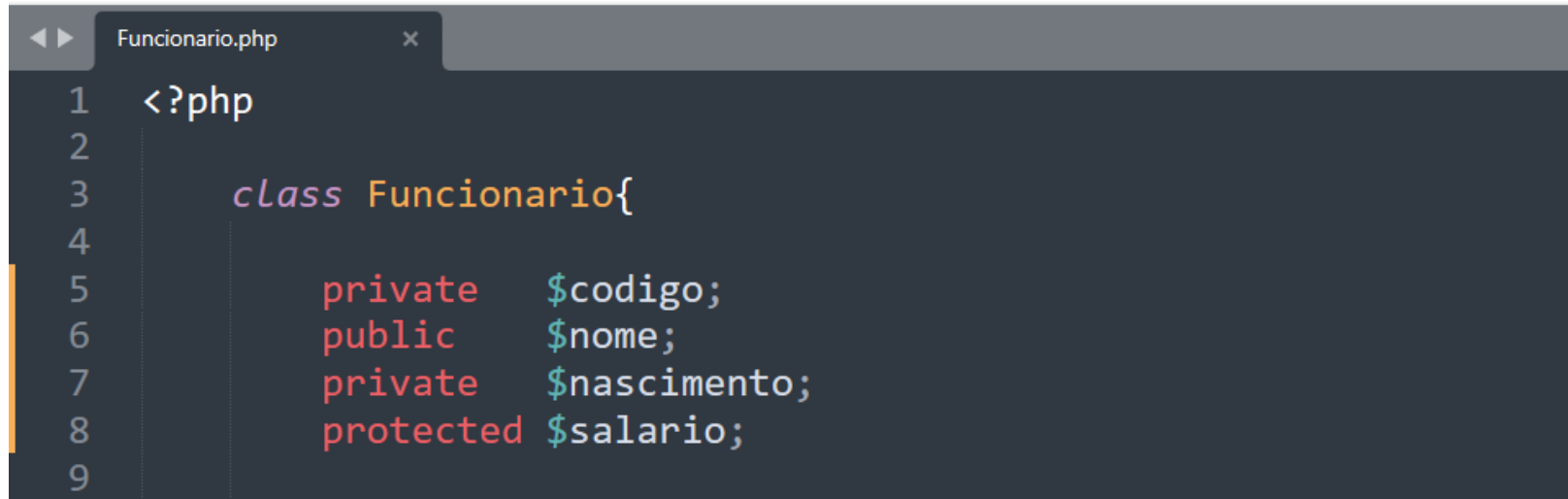
```
testa_estagiario.php x
1  <?php
2      # carrega as classes
3      include 'Funcionario.php';
4      include 'Estagiario.php';
5
6      $pedrinho= new Estagiario();
7
8      $pedrinho->SetSalario(248);
9
10     echo 'O Salário do Pedrinho é R$: ' . $pedrinho->getSalario() . "<br>";
11
12
13  ?>
```

**Warning:** Undefined property: Estagiario::\$salario in C:\xampp\htdocs\ProjetoPHP3H2\_2022\aula03\Estagiario.php on line 7  
O Salário do Pedrinho é R\$: 0

### Protected

- Como esperado, o programa não retornou o salário esperado.
- Isso ocorre porque a propriedade Salario é uma propriedade private, o que significa que ela somente pode ser acessada de dentro da classe em que ela foi declarada (Funcionario).
- Para permitir o acesso também nas classes-filha, alteraremos a classe Funcionario e marcaremos a propriedade Salario como protected.

## Protected



```
1 <?php
2
3 class Funcionario{
4
5     private $codigo;
6     public $nome;
7     private $nascimento;
8     protected $salario;
9 }
```

The image shows a code editor window titled 'Funcionario.php'. The code defines a PHP class named 'Funcionario'. Inside the class, there are four properties: '\$codigo' is private, '\$nome' is public, '\$nascimento' is private, and '\$salario' is protected. The code is displayed with syntax highlighting on a dark background.

### Protected

- Agora, ao executarmos novamente nosso trecho de programa, veremos que o comportamento desejado é atingido, sendo a propriedade Salario passível de alterações tanto no escopo da classe Funcionario quanto no escopo da classe Estagiario.

# Programação a Objetos em PHP:

## Protected

```
testa_estagiario.php x
1  <?php
2      # carrega as classes
3      include 'Funcionario.php';
4      include 'Estagiario.php';
5
6      $pedrinho= new Estagiario();
7
8      $pedrinho->SetSalario(248);
9
10     echo 'O Salário do Pedrinho é R$: ' . $pedrinho->getSalario() . "<br>";
11
12
13     ?>
```



localhost/ProjetoPHP3H2\_2022/aula03/testa\_estagiario.php

O Salário do Pedrinho é R\$: 277.76



## Public

- Demonstrar a visibilidade public é uma tarefa simples, pois o comportamento-padrão do PHP é tratar uma propriedade como public, ou seja, se não especificarmos a visibilidade, automaticamente ela será pública.
- No exemplo que segue, estamos criando dois objetos e alterando suas propriedades à vontade, as quais poderiam ser alteradas por métodos internos e por classes descendentes também.

# Programação a Objetos em PHP:

## Classe testa\_public

```
testa_public.php
1  <?php
2
3      # carrega as classes
4      include 'Funcionario.php';
5      include 'Estagiario.php';
6
7      //cria objeto Funcionaria
8      $pedrinho= new Funcionario();
9      $pedrinho->nome = 'Pedrinho';
10
11     //cria objeto Estagiaria
12     $mariana= new Estagiario();
13     $mariana->nome = '<br> Mariana';
14
15     //imprime propriedade nome
16
17     echo $pedrinho->nome;
18     echo $mariana->nome;
19
20  ?>
```



localhost/ProjetoPHP3H2\_2022/aula03/testa\_public.php

Pedrinho  
Mariana

# **Muito Obrigado**

Até a Próxima Aula

Prof. Laércio Silva  
Email: [Indsilva@hotmail.com](mailto:Indsilva@hotmail.com)