INSTITUTO FEDERAL
Rio Grande do Norte | Campus Parnamirim

Introdução à Orientação a Objetos
Parte 02

Prof. Fábio Procópio

Prof. João Nascimento

Técnico Integrado em Informática – Programação com Acesso a Banco de Dados

Relembrando...

- Na aula passada, relembramos alguns conceitos estudados na disciplina de Programação Estruturada e Orientada a Objetos;
- Relembramos os conceitos de **classe**, **objeto**, **atributos** e **métodos**;
- Nesta aula, implementaremos esses conceitos.

Relembrando...

- Faremos alguns exemplos usando a classe Cachorro, definida na aula passada. Vamos relembrar o modelo da classe:

Cachorro
- nome - idade - raca
+ latir() + brincar()

Declaração de classes

- A criação de classes se dá por meio da palavra-chave **`class`**;
- Observe o exemplo da definição de uma classe chamada Cachorro;
- Crie um arquivo chamado **Cachorro.php** (por enquanto, ela nada faz):

```
class Cachorro{
}
```

Declaração de atributos

- Para declarar um atributo, utilizamos a palavra-chave **`var`**;

- Vamos declarar os atributos da classe Cachorro:

```
class Cachorro{

    var $nome;
    var $idade;
    var $raca;
}
```

Declaração de métodos

- Para declarar um método, utilizamos a palavra-chave **`function`**;

- Vamos declarar os métodos da classe Cachorro. Observe que, agora, a classe começa a ter funcionalidades:

```
class Cachorro{

   var $nome;
   var $idade;
   var $raca;

   function latir(){
      return "au au au";
   }

   function brincar(){
      return "Estou cavando o jardim.";
   }
}
```

Relembrando...

O que é um método construtor?

- Método especial chamado quando um objeto é criado;
- Normalmente um construtor é usado para inicializar atributos ou criar objetos para os atributos;
- É utilizado para inicializar o estado de um objeto;
- Em PHP, utiliza-se a palavra chave **`__construct`**.

Instanciação de objetos – 1 de 2

- Para instanciar um objeto, utilizamos a palavra-chave **`new`**;
- Vamos criar um arquivo chamado **TestaClasses.py** o qual será utilizado para validar nossas classes;
- Para que não ocorram erros de chamada, **salve** TestaClasses.php **na mesma pasta** do arquivo Cachorro.py;
- Para que a classe Cachorro possa ser utilizada normalmente, **não devemos esquecer** de referenciá-la indicando em qual arquivo ela foi definida;
  - Para referenciar um arquivo, utiliza-se **`include`**

Exercício Resolvido 01 – 1 de 2

- Modifique a classe Cachorro adicionando os métodos **__construct()** e **imprimir()**:

```
class Cachorro{
   function __construct($p_nome, $p_idade, $p_raca){
       $this->nome  = $p_nome;
       $this->idade = $p_idade;
       $this->raca  = $p_raca;
   }
   function latir(){
      return "au au au";
   }
   function brincar(){
      return "Estou cavando o jardim.";
   }
   function imprimir_dados(){
      $dados  = "Nome.: ".$this->nome."</br>";
      $dados .= "Idade: ".$this->idade."</br>";
      $dados .= "Raça.: ".$this->raca;
      return $dados;
   }
 }
```

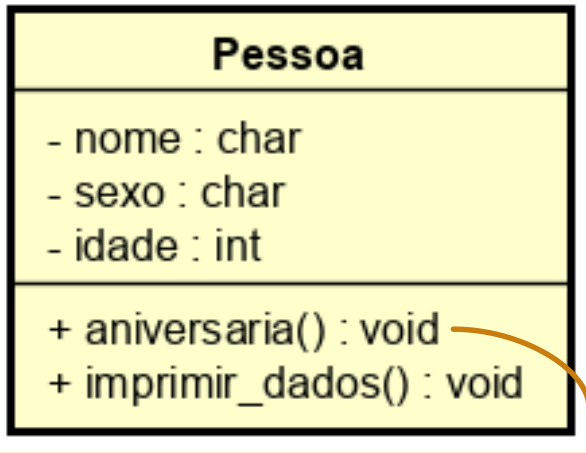
Exercício Resolvido 01 – 2 de 2

- Arquivo TestaClasses.php:

```
<?php
   include("Cachorro.php");

   /*Observe que informamos 3 argumentos para instanciar
   o objeto de Cachorro. Isso foi feito porque o seu método
   construtor aguarda a passagem desses parâmetros*/

   $dog = new Cachorro("core", 10, "SRD");

   echo $dog->latir();
   echo "<br>".$dog->brincar();

   echo "<br>"."********** Dados do cachorro **********";
   echo "<br>".$dog->imprimir_dados();
   echo "<br>"."**************************************";
?>
```

Exercício Resolvido 02 – 1 de 3

- Implemente a seguinte classe:



Este método deve fazer com que o atributo **idade** seja incrementado cada vez que ele for executado.

Exercício Resolvido 02 – 2 de 3

- Crie uma classe chamada Pessoa e salve-a no arquivo **Pessoa.php**:

```
class Pessoa{
    function __construct($p_nome, $p_sexo, $p_idade){
      $this->nome  = $p_nome;
      $this->sexo  = $p_sexo;
      $this->idade = $p_idade;
    }
    function imprimir_dados(){
      $dados  = "********** DADOS **********<br>";
      $dados .= "Nome.: $this->nome <br>";
      $dados .= "Sexo.: $this->sexo <br>";
      $dados .= "Idade: $this->idade anos <br>";
      $dados .= "**************************<br>";
      echo $dados;
    }
    function aniversaria(){
      # Toda vez que a pessoa aniversaria, sua idade é acrescida em 1 ano
      $this->idade += 1;
      echo "Parabéns, $this->nome pelos seus $this->idade anos.<br>";
    }
}
```

Exercício Resolvido 02 – 2 de 3

- Em TestaClasses.php, escreva o seguinte código:

```
<?php
   include("Pessoa.php");

   $paolla = new Pessoa("Paolla de Oliveira", "F", 37);

   $paolla->imprimir_dados();
   $paolla->aniversaria();
   $paolla->aniversaria();
   $paolla->aniversaria();
?>
```

Exercício Resolvido 03 – 1 de 4

- Implemente a classe `Porta`.

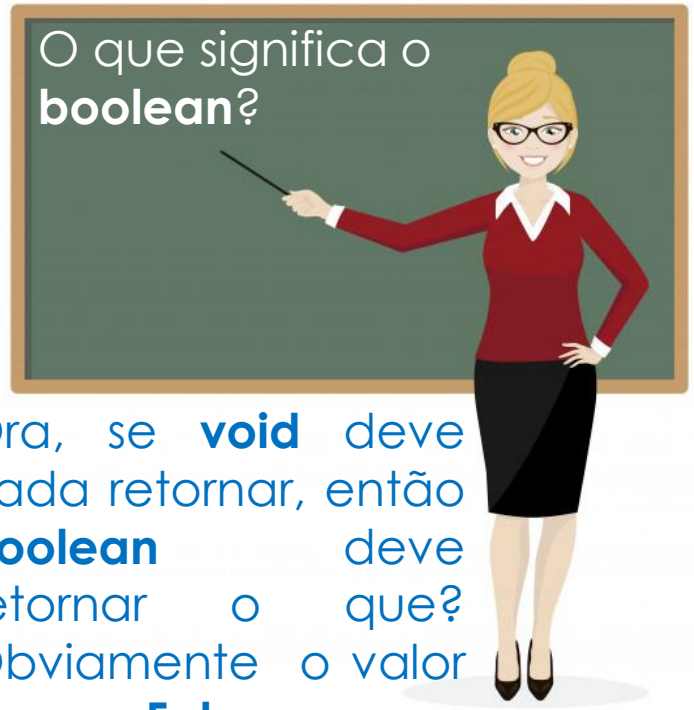
Porta
- aberta : boolean - cor : char - dimensaoX : float - dimensaoY : float - dimensaoZ : float
+ pinta(p_cor : char) : void + mostra_estado() : boolean + abre() : void + fecha() : void

Método **`mostra_estado()`**
Verifica se a porta está aberta ou fechada

Método **`abre()`**
Só poderá abrir a porta se ela estiver fechada.
Caso contrário, um erro deve ser exibido

Método **`fecha()`**
Só poderá fechar a porta se ela estiver aberta.
Caso contrário, um erro deve ser exibido

ATENÇÃO
Antes de abrir ou fechar a porta, use o método **`mostra_estado()`**



Ora, se **void** deve nada retornar, então **boolean** deve retornar o que? Obviamente o valor **True** ou **False**.

Exercício Resolvido 03 – 2 de 4

- Crie uma classe chamada `Porta` e salve-a no arquivo **Porta.php**:

```
class Porta{
    function __construct($p_dimensaoX, $p_dimensaoY, $p_dimensaoZ){
      $this->aberta    = False; //Sempre que criada uma porta ela estará fechada
      $this->cor       = null;  //Sempre que criada uma porta ela estará sem cor
      $this->dimensaoX = $p_dimensaoX;
      $this->dimensaoY = $p_dimensaoY;
      $this->dimensaoZ = $p_dimensaoZ;
    }

    function pinta($p_cor){
      $this->cor = $p_cor;
      echo "Porta pintada de $this->cor<br> ";
    }

    function mostra_estado(){
      return $this->aberta;
    }

    // Continua no outro slide
```

Exercício Resolvido 03 – 3 de 4

- Continuação do arquivo **Porta.php**:

```
    // Continuação...
    function abre(){
      if ($this->mostra_estado() == False) {
         $this->aberta = True;
         echo "OK: Porta aberta.<br>";
      }
      else{
         echo "ERRO: Porta já está aberta.<br>";
      }
    }

    function fecha(){
      if ($this->mostra_estado() == True){
         $this->aberta = False;
         echo "OK: Porta fechada.<br>";
      }
      else{
         echo "ERRO: Porta já está fechada.";}}
}
```

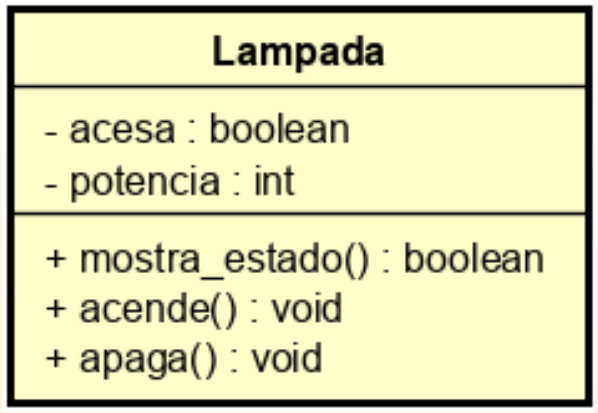
Exercício Resolvido 03 – 4 de 4

- Crie um arquivo chamado **TestaClasses.php**:

```
include("Porta.php");

$p1 = new Porta(0.9, 2.5, 0.05);
$p1->pinta("cinza");
$p1->abre();
$p1->fecha();

# Observe que será gerado um erro porque a porta já foi fechada
$p1->fecha();
```

Exercício de Fixação 01

- Implemente a classe `Lampada`.



Método **`mostra_estado()`**
Verifica se a lâmpada está acesa ou apagada e retorna um valor boolean

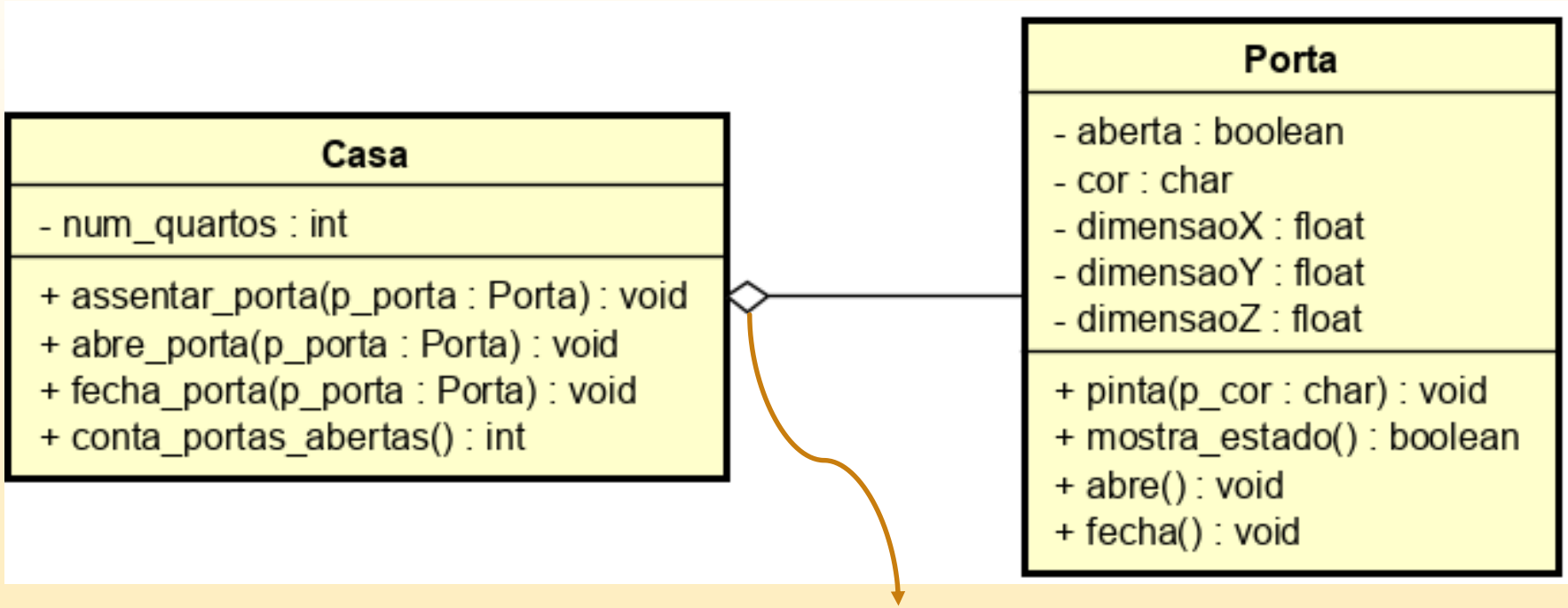
Método **`acende()`**
Só poderá acender a lâmpada se ela estiver apagada. Caso contrário, um erro deve ser disparado

Método **`apaga()`**
Só poderá apagar a lâmpada se ela estiver acesa. Caso contrário, um erro deve ser disparado

ATENÇÃO
Antes de acender ou apagar a lâmpada, use o método **`mostra_estado()`**

Exercício Resolvido 05 – 1 de 4

- Implemente o diagrama de classe abaixo considerando a associação do tipo Agregação entre as classes `Casa` e `Porta`.



Esse losango "aberto" significa **Agregação**, a qual tenta demonstrar que as informações de um objeto-todo precisam ser complementadas pelas informações contidas em um ou mais objetos de outra classe (objeto-parte).

Exercício Resolvido 05 – 2 de 4

- Crie uma classe chamada `Casa` e salve-a no arquivo **Casa.php**:

```
include("Porta.php");
class Casa{
    function __construct($p_num_quartos){
        $this->num_quartos = $p_num_quartos;
        $this->portas      = []; #array de portas
    }

    function assentar_porta($p_porta){
        array_push($this->portas, $p_porta); # Adiciona uma porta em $this->portas
    }

    function abre_porta($p_porta){
        foreach ($this->portas as $indice => $door){
            if ($door == $p_porta) {
                $this->portas[$indice]->abre();
                break;
            }
        }
    }
```

Exercício Resolvido 05 – 3 de 4

- Continuação do arquivo **Casa.php**:

```
    # Continuação da classe Casa
    function abre_porta($p_porta){
        foreach ($this->portas as $indice => $door){
            if ($door == $p_porta) {
                $this->portas[$indice]->abre();
                break;
            }
        }
    }
    function conta_portas_abertas(){
        $abertas = 0;
        foreach ($this->portas as $indice => $door){
            if ($door->mostra_estado() == True){
               $abertas += 1;
            }
        }
        return $abertas;
    }
}
```

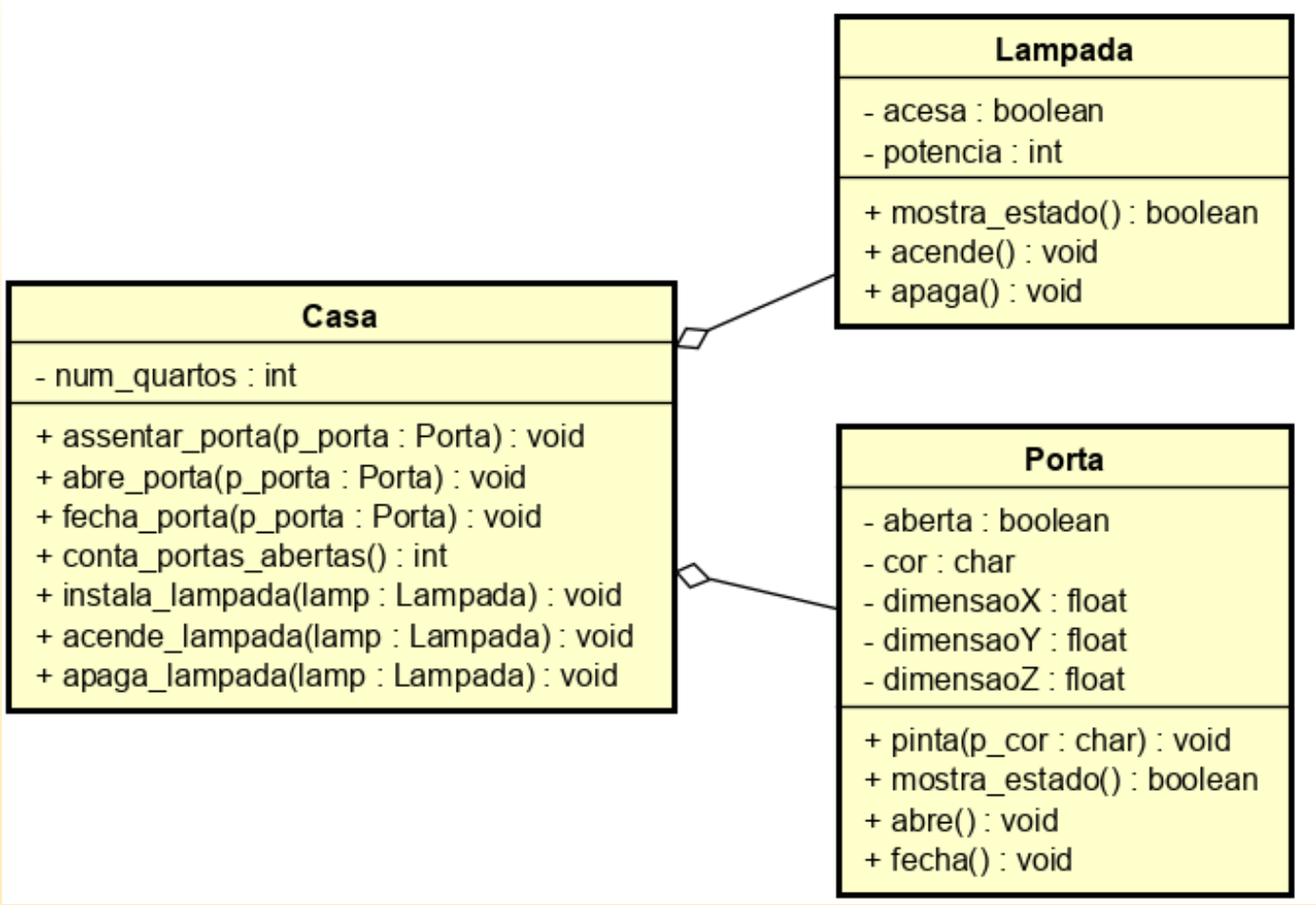
Exercício Resolvido 05 – 4 de 4

- Crie um arquivo chamado **TestaCasa.py**:

```
include 'Casa.php';

$p1 = new Porta(1, 1, 0.05);
$p2 = new Porta(1, 1, 0.05);
$p3 = new Porta(1, 1, 0.05);

$c = new Casa(3);

$c->assentar_porta($p1);
$c->assentar_porta($p2);
$c->assentar_porta($p3);

$c->abre_porta($p1);
$c->abre_porta($p2);
$c->abre_porta($p3);
$c->fecha_porta($p1);

$abertas = $c->conta_portas_abertas();
echo "Portas abertas: $abertas";
```

Exercício de Fixação 02

- Modifique a classe Casa, observando a nova associação com a classe **`Lampada`** e os métodos **`instala_lampada()`**, **`acende_lampada()`** e **`apaga_lampada()`**.



Referências

1. DEVMEDIA. **Introdução à Programação Orientada a Objetos em Java**. Disponível em: https://www.devmedia.com.br/introducao-a-programacao-orientada-a-objetos-em-java/26452. Acessado em: 12 jun. 2019.

2. PythonBrasil. **Python e Programação Orientada a Objetos**. Disponível em: http://wiki.python.org.br/ProgramacaoOrientadaObjetoPython. Acessado em: 12 jun. 2019.

3. PHP.NET. **array_push**. Disponível em: https://www.php.net/manual/pt_BR/function.array-push.php. Acessado em: 25 set. 2019.

4. PHP.NET. **foreach**. Disponível em: https://www.php.net/manual/en/control-structures.foreach.php. Acessado em: 25 set. 2019.