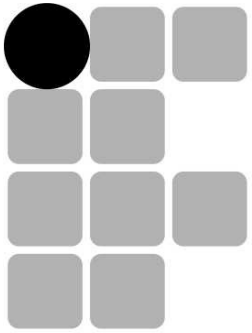


INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
RIO GRANDE DO SUL
Câmpus Feliz

# PHP

*Orientado a Objetos*

Prof. Moser Fagundes

Programação III
ADS

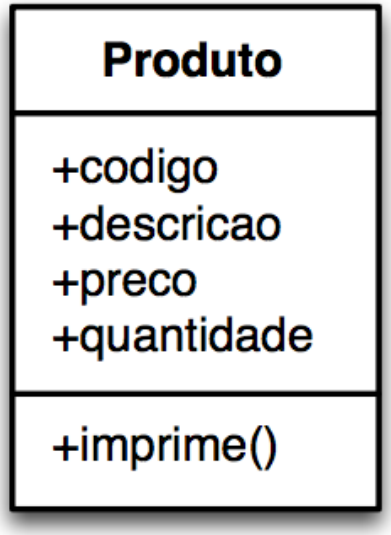
# Sumário

- Classes
- Atributos
- Métodos
- Objetos
- Construtores
- Indução de tipos
- Herança

# Classe

- Uma **classe** representa um **tipo abstrato de dado** que é usado para representar objetos complexos que possuem estado e comportamento;
- Uma classe é um **gabarito** para a definição de objetos;
- É composta por:
  - **Atributos** – descreve as propriedades da classe e armazena o seu estado.
  - **Métodos** – define as funcionalidades (comportamento) e permitem a modificação do seu estado.

# Exemplo de classe em PHP




```php
<?php
class Produto {
      public $codigo;
      public $descricao;
      public $preco;
      public $quantidade;

      function imprime() {
      ...
      }
}
?>
```

# Exemplo de atributos

- Define as **propriedades** que o objeto possui.
- Cada **atributo** é identificado por um **nome** e inclui um **modificador de acesso** (veremos mais adiante):

```
<?php
class Produto {
    public $codigo;
    public $descricao;
    public $preco;
    public $quantidade;

    function imprime() {
    ...
    }
}
?>
```

# Exemplo de método

- Define as **funcionalidades/ações** da classe, ou seja, o que os objetos dessa classe fazem;
- É composto por um **identificador** para o método (o nome do método) e uma **lista de argumentos** (como nas funções).

```
<?php
class Produto {
      public $codigo;
      public $descricao;
      public $preco;
      public $quantidade;

      function imprime() {
          echo "Codigo:".$this->codigo."<BR>\n";
          echo "Descricao:".$this->descricao."<BR>\n";
      }
}
?>
```

# Retorno de um método

- Podemos retornar um determinado valor através do comando **return**, o qual encerra a execução do método imediatamente.
- O que vier depois do **return** nunca é executado!

```
<?php
class Produto {
      public $codigo;
      public $descricao;

      function getCodigo() {
          return $this->codigo;
      }
}
?>
```

# Modificadores de acessibilidade

- Define a acessibilidade dos atributos e métodos:
  - **public**: o atributo ou método de um objeto dessa classe pode ser acessado por qualquer outro objeto (visibilidade externa total).
  - **private**: o atributo ou método de um objeto dessa classe não pode ser acessado por nenhum outro objeto (nenhuma visibilidade externa);
  - **protected**: o atributo ou método de um objeto dessa classe poderá ser acessado apenas por objetos de classes que sejam derivadas dessa através do mecanismo de herança (veremos mais tarde este mecanismo).

# Métodos get e set

- Definimos um atributo como **private**/**protected** para impedir o seu acesso indiscriminado.
  - Ao tornarmos o acesso de um atributo mais controlado, podemos criar métodos especiais que permitem a sua definição (set) e obtenção (get).

- **Vantagens**:
  - Uso de atributos somente leitura pela omissão do set.
  - Controle do atributo para valores anormais/inválidos.
  - Facilita mudanças futuras nas regras dos atributos.

# Exemplo com get e set

```php
<?php
class Produto {
    private $codigo;

    function getCodigo() {
        return $this->codigo;
    }

    function setCodigo($novo_codigo) {
        $this->codigo = $novo_codigo;
    }
}
?>
```

# Convenção de nomes

- **Nomes de classes** são normalmente **substantivos** ou **expressões curtas** no **singular**. Por convenção, cada palavra existente no nome da classe inicia com **letra maiúscula**.
- **Atributos** costumam ser **substantivos** que descrevem **características** enquanto **métodos** normalmente são **verbos** representando **ações**.
- O **nome** dos **atributos**, **métodos** e **variáveis** se escrevem com cada palavra começando por maiúsculo, **exceto a primeira letra**.

# Objetos

- **Objetos** são instâncias de classe.
  - Quando instanciamos um objeto de uma classe, estamos criando um novo item do conjunto representado por essa classe.

- **Exemplos** de objetos das classes:
  - Livro: *Pequeno Príncipe, Tutorial PHP,...*
  - Filme: *Matrix, Sexto Sentido, Avatar,...*
  - Carro: *Fit, Fiesta, CRV, Corolla,...*

# Exemplos de objetos

- Veja os arquivos:

  **exemplo_slide_13A.php**
  **classes/Produto.class.php**

  **exemplo_slide_13B.php**
  **classes/Carro.class.php**

- É uma boa prática criar uma pasta contendo as suas classes com a extensão **.class.php**

# Atividade 1

Realizar a Atividade 1 e fazer o upload do código em um arquivo ZIP no Moodle.

A descrição da atividade se encontra no arquivo `P3-ADS-Lista15.pdf`

# Comparação de objetos

- Qual é o resultado da execução do código abaixo?

```
<?php
    $prod1 = new Produto();
    $prod2 = new Produto();

    $prod1->codigo = 200;
    $prod2->codigo = 200;

    if ($prod1 == $prod2)
        echo "Iguais";
    else echo "Diferentes";
?>
```

**exemplo_slide_15.php**

# Comparação de objetos

- E do código abaixo? O que será impresso na tela?

```
<?php
    $prod1 = new Produto();
    $prod2 = new Produto();

    $prod1->codigo = 200;
    $prod2->codigo = 200;

    if ($prod1 === $prod2)
        echo "Iguais";
    else echo "Diferentes";
?>
```

**exemplo_slide_16.php**

# Construtores

- Caso não seja definido um método **construtor** na classe, automaticamente todas as propriedades do objeto criado são inicializadas com **null**.

- O método construtor tem o seguinte formato

```
__construct([$arg1 [, $arg2 ... ]])
```

# Construtores

- Exemplos:

**exemplo_slide_17A.php**
**classes/Cidade.class.php**

**exemplo_slide_17B.php**
**classes/Pais.class.php**

# Indução de tipos

- Considere o seguinte código da classe **Continente**:

```php
<?php
    class Continente {
        private $paises;

        function __construct() {
            $this->paises = array();
            echo "O continente foi criado.<br>";
        }

        // Para induzir um tipo no metodo, colocamos
        // o nome da classe na frente da variavel
        function registrar(Pais $_pais) {
            $paises[] = $_pais;
        }
    }
?>
```

**classes/Continente.class.php**

# Indução de tipos

- Considere o seguinte código onde criamos objetos e usamos o tipo adequado na chamada do método **registrar**.

```php
<?php
    include_once 'classes/Continente.class.php';
    include_once 'classes/Pais.class.php';

    $c1 = new Continente();
    $p1 = new Pais("Brasil");
    $c1->registrar($p1);
?>
```

**exemplo_slide_20.php**

# Indução de tipos

- Considere o seguinte código onde tentamos passar um tipo de dado **incorreto** na chamada do método **registrar**.

```php
<?php
    include_once 'classes/Continente.class.php';
    include_once 'classes/Pais.class.php';

    $c1 = new Continente();
    $p1 = new Pais("Brasil");
    $c1->registrar($p1);

    // Erro pois registrar() aceita somente Pais
    $c1->registrar("Argentina");
?>
```

**Catchable fatal error**: Argument 1 passed to Continente::registrar() must be an instance of Pais.

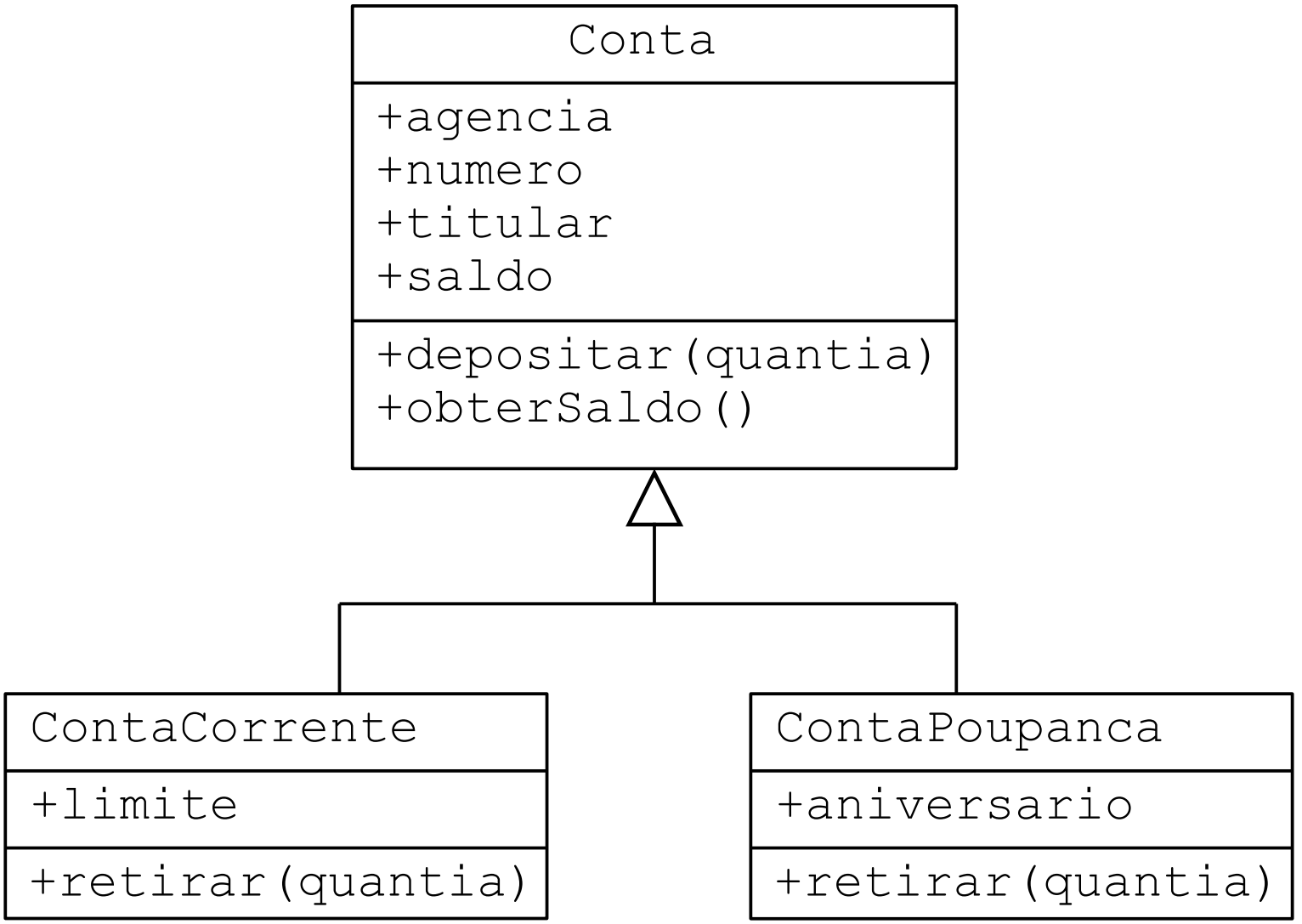
**exemplo_slide_21.php**

# O conceito de herança

- Na orientação a objetos, vemos a **herança** como a **passagem** (*compartilhamento*) de **características** (*atributos*) e **comportamentos** (*métodos*) entre classes de uma mesma **hierarquia** (*árvore*).

- As classes inferiores na hierarquia, chamadas **subclasses**, automaticamente herdan todos os **atributos** e **métodos** das classes superiores, chamadas **superclasses**.

# Representando herança em UML



# Herança no PHP

```php
<?php
    class Conta {
        public $agencia;
        public $numero;
        public $titular;
        public $saldo;


        function depositar($quantia) {
            // Código aqui
        }

        function obterSaldo() {
            // Código aqui
        }
    }
?>
```

# Herança no PHP

```php
<?php
   class ContaCorrente extends Conta {

      public $limite;

      function retirar($quantia) {
         // Código aqui
      }
   }
?>
```

# Herança no PHP

```php
<?php
   class ContaPoupanca extends Conta {

      public $aniversario;

      function retirar($quantia) {
         // Código aqui
      }
   }
?>
```

# Atividade 2

Realizar a Atividade 2 e fazer o upload do código em um arquivo ZIP no Moodle.

A descrição da atividade se encontra no arquivo `P3-ADS-Lista15.pdf`