

Orientação a Objetos com PHP

Alisson Chiquitto¹

¹Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas

Cianorte, 2014

- 1 Introdução
- 2 Classes, Métodos e Atributos
 - Declaração de classes
 - Criando objetos
 - Herança de classes
 - Modificadores de acesso
 - Elementos estáticos
- 3 Atividades

O que é o Orientação a Objetos?

A **orientação a objetos** (OO) é um **paradigma** de análise, projeto e programação de sistemas de software baseado na **composição e interação** entre diversas unidades de software chamadas de **objetos**.

Vocabulário basico de OO

classe

A receita ou modelo para criação de um objeto;

objeto

Uma coisa (coiso);

instanciar

A ação de criação de um objeto a partir de uma classe;

método

Uma função que pertence à um objeto;

atributo

Uma variável que pertence à um objeto;

- 1 Introdução
- 2 Classes, Métodos e Atributos
 - Declaração de classes
 - Criando objetos
 - Herança de classes
 - Modificadores de acesso
 - Elementos estáticos
- 3 Atividades

- Uma **classe** é um modelo, ou um conjunto de instruções de como um objeto deve ser criado.
- Uma **classe** podem conter **atributos** e **métodos**;

```
3  class Conta {  
4  
5  }
```

Figura: Classe Conta (classe-conta.php)

- Um **método** é uma função que pertence à uma **classe**;
- Um **método** é *tudo o que a classe é capaz de fazer*.
- A declaração de um **método** segue as mesmas regras de declaração de **funções**. Podemos passar **parametros** e **retornar** valores do mesmo jeito que é feito com uma função;


```
3  class Conta {  
4      // retorna o número da conta  
5      function getNumero() {  
6          return 123;  
7      }  
8  
9      // muda o dono da conta  
10     // não retorna valor  
11     function setDono() {  
12     }  
13 }
```

Figura: Classe Conta com métodos (classe-conta.php)

- Um **atributo** (ou **propriedade**) é uma variável que é declarada dentro do escopo de uma classe, mas fora de métodos;
- Um atributo pode possuir um valor padrão. Quando a classe for instanciada, o atributo receberá este valor padrão;

```
3 class Conta {  
4     public $numero;  
5     public $dono;  
6     public $saldo;  
7     public $limite;  
8 }
```

Figura: Classe Conta com vários atributos

Atributos com valor padrão

```
3  class Conta {  
4      public $numero;  
5      public $dono;  
6      public $saldo = 0;  
7      public $limite = 1000;  
8  }
```

Figura: Classe com atributos com valor padrão

- 1 Introdução
- 2 **Classes, Métodos e Atributos**
 - Declaração de classes
 - **Criando objetos**
 - Herança de classes
 - Modificadores de acesso
 - Elementos estáticos
- 3 Atividades

Criando instâncias de objetos

```
3  require './classe-conta.php';  
4  
5  $conta = new Conta();  
6  $conta->numero = '1234-5';
```

Figura: Uma nova instância de Conta

A variável *\$this*

- A variável *\$this* é uma variável especial que está sempre disponível dentro do escopo de um objeto.
- Se refere ao **objeto atual**;
- É utilizada para acessar **propriedades** ou chamar **métodos** de dentro do objeto;

A variável *\$this*

```
3  class Conta {
4      public $numero;
5      public $dono;
6      public $saldo;
7      public $limite;
8
9      // retorna o número da conta
10     function getNumero() {
11         return $this->numero;
12     }
13
14     // muda o dono da conta
15     // não retorna valor
16     function setDono($dono) {
17         $this->dono = $dono;
18     }
19 }
```

Figura: A variável *\$this* (classe-conta.php)

Exemplo de uso da classe Conta

```
3  require './classe-conta.php';
4
5  $conta = new Conta();
6  $conta->setDono('Jose Aparecido');
7  $conta->numero = '123456-7';
8
9  echo 'Cliente: '
10      . $conta->dono
11      . '<br>Conta: '
12      . $conta->numero
13      ;
```

Figura: Exemplo de uso da classe Conta

Resolver as atividades:

- ▶ Atividade: Família feliz
- ▶ Atividade: Uma família um pouco maior

- 1 Introdução
- 2 Classes, Métodos e Atributos
 - Declaração de classes
 - Criando objetos
 - **Herança de classes**
 - Modificadores de acesso
 - Elementos estáticos
- 3 Atividades

- **Herança** é uma maneira de uma classe se relacionar com outra;
- Classes (filhas) podem **herdar/estender** somente de uma classe (pai) por vez;
- Uma **classe filha** possui todas as características da **classe pai**;

Exemplo: Classe Conta I

```
3  class Conta {
4      public $numero;
5      public $dono;
6      private $saldo = 0;
7      protected $limite = 0;
8
9      public function fazerDeposito($vlr) {
10         $this->saldo += $vlr;
11     }
12
13     public function fazerSaque($vlr) {
14         $saldo2 = $this->saldo - $vlr;
15         if ($saldo2 < $this->getLimite()) {
16             return false;
17         }
18         $this->saldo = $saldo2;
19         return true;
20     }
```

Figura: A classe Conta

Exemplo: Classe Conta II

```
21  
22     protected function getLimite() {  
23         return $this->limite;  
24     }  
25  
26     public function getSaldo() {  
27         return $this->saldo;  
28     }  
29 }
```

Figura: A classe Conta

Exemplo: Classe ContaPoupanca

```
3 class ContaPoupanca
4 extends Conta
5 {
6     protected $limite = -100;
7 }
```

Figura: A classe ContaPoupanca

Exemplo: Usando a classe ContaPoupanca

```
3  require './classe-conta-2.php';
4  require './classe-conta-poupanca-2.php';
5
6  $conta = new ContaPoupanca();
7
8  $conta->fazerDeposito(100);
9
10 if (!$conta->fazerSaque(100)) {
11     echo 'Voce nao pode sacar este valor';
12 }
13 else {
14     echo 'Valor retirado da conta';
15 }
16
17 echo '<br><br>'
18     . 'Voce tem: R$'
19     . $conta->getSaldo()
20 ;
```

Figura: Exemplo de uso da classe ContaPoupanca

1 Introdução

2 Classes, Métodos e Atributos

- Declaração de classes
- Criando objetos
- Herança de classes
- **Modificadores de acesso**
- Elementos estáticos

3 Atividades

Modificadores de acesso

- Os modificadores de acesso alteram a visibilidade de um elemento;
- Os modificadores de acesso existentes no PHP são: *public*, *protected* e *private*;
- São aplicados somente a **métodos** e **atributos**, não a **classes**;

Modificadores de acesso

public

Os membros públicos podem ser acessados de qualquer local;

protected

Os membros protegidos podem ser acessados somente de dentro da classe de origem ou classes filhas;

private

Os membros privados podem ser acessados somente de dentro da classe de origem;

Modificadores de acesso

```
3  class Conta {
4      public $numero;
5      public $dono;
6      private $saldo;
7      protected $limite;
8
9      // retorna o número da conta
10     public function getNumero() {
11         return $this->numero;
12     }
13
14     // muda o dono da conta
15     // não retorna valor
16     public function setDono($dono) {
17         $this->dono = $dono;
18     }
19 }
```

Figura: Modificadores de acesso na classe Conta

- 1 Introdução
- 2 Classes, Métodos e Atributos
 - Declaração de classes
 - Criando objetos
 - Herança de classes
 - Modificadores de acesso
 - Elementos estáticos
- 3 Atividades

- **Métodos** e **Atributos** podem ser estáticos;
- Os elementos estáticos podem ser utilizados sem que a classe seja instanciada;

Uma classe com elementos estáticos

```
3  class Matematica {  
4      public static $resultado;  
5  
6      public static function somar($n1, $n2) {  
7          self::$resultado = $n1 + $n2;  
8          return self::$resultado;  
9      }  
10 }
```

Figura: Classe Matemática com métodos e atributos estáticos

Acessando elementos estáticos

Um elemento estático pode ser acessado pelo **Operador de Resolução de Escopo (::)**

```
3  require './classe-matematica.php';  
4  
5  $soma = Matematica::somar(1,2);  
6  echo $soma;
```

Figura: Exemplo de uso dos elementos estáticos

Características dos elementos estáticos

- Os **atributos estáticos** usam o caractere *dolar* (\$), ao contrário de **atributos não estáticos**;
- De dentro da classe, utilize **self::** ao invés de **\$this** para acessar elementos estáticos desta classe;

Atividade: Família feliz

1. Em um arquivo chamado **classe-pessoa.php**, criar uma classe Pessoa
 - 1.1 Com os atributos *nome*, *anoNascimento*, *pai* e *mae*;
 - 1.2 Com o método `pegarNome()`: retorna o nome em caixa-alta;
 - 1.3 O método `pegarIdade()`: retorna (`anoAtual` - `anoNascimento` da pessoa);
2. Em um arquivo chamado *pessoa1.php*:
 - 2.1 Instanciar a classe Pessoa e atribuir a instancia à variável **\$pai**.
 - 2.2 Instanciar a classe Pessoa e atribuir a instancia à variável **\$mae**.
 - 2.3 Atribuir nomes e ano de nascimento aos objetos *\$pai* e *\$mae*;
 - 2.4 Criar um objeto *\$filho* da classe Pessoa, e atribuir nome e ano de nascimento;
 - 2.5 Atribuir *\$pai* e *\$mae* ao *\$filho*;
 - 2.6 A partir deste ponto utilize apenas a variável *\$filho* para fazer o que é pedido;
 - 2.7 Escrever a diferenca de idade entre pai, mãe e o filho;
 - 2.8 Escrever os nomes do pai e mãe;

Atividade: Uma família um pouco maior

1. Caracterização do problema: Um homem teve dois filhos, cada filho com uma mulher distinta;
2. Como na ► Atividade: Família feliz :
 - 2.1 Criar os objetos de cada personagem do problema citado;
 - 2.2 Fazer as respectivas atribuições de pais e mães;
3. Utilizando apenas as variáveis que armazenam as referências aos filhos:
 - 3.1 Mostrar o nome do filho mais velho, e sua respectiva idade;
 - 3.2 Mostrar a diferença de idade entre os dois filhos;
 - 3.3 Mostrar o nome da mãe do filho mais novo;



Hypertext Markup Language (HTML).
W3C.

<http://www.w3.org/MarkUp/draft-ietf-iiir-html-01.txt>