OO em PHP

Prof.: Alisson G. Chiquitto
chiquitto@unipar.br

Orientação a Objetos

A orientação a objetos 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.

Orientação a Objetos no PHP

No PHP 5, lançado em 2004, foi adicionado vários recursos ao PHP para melhorar a performance e incluir novas funcionalidades:

- Classes e métodos abstratos e final;
- Visibilidade de métodos e atributos;
- Métodos mágicos;
- Interfaces;
- Clonagem;
- Indução de tipo;

Classes

- Toda definição de classe começa com a palavra-chave “class”, seguido por um nome de classe e um par de chaves;
- A classe pode conter suas variáveis chamadas de propriedades/atributos, e também suas funções, chamadas de métodos.
- Para acessar as propriedades / métodos use a sintaxe “->”;

```php
<?php

class Exemplo {
    // declaracao de propriedade
    public $var1;
    public $var2 = 'Valor padrão';

    // declaracao de metodo
    public function mostraVar() {
        echo $this->var2;
    }
}
```

Propriedades

- Variáveis de classes são chamadas de propriedades (atributos, fields, etc);
- Ao declarar uma propriedade é necessário definir o nível de visibilidade para *public*, *protected* ou *private*;
- A declaração pode vir seguida de um valor padrão;

```
class Carro {
    public $cor;
    protected $versao;
    private $valvulas;

    public $rodas = 4;
}

$carro = new Carro();
echo $this->rodas;
```

Constantes

- No contexto de uma classe é possível definir constantes;
- Não é necessário o simbolo $ na constante;
- O valor deve ser uma expressão constante;
- Para acessar a constante use o simbolo “::”;

```php
class Math {
    const PI = 3.1415;

    public function echoPI() {
        echo self::PI;
    }
}

echo Math::PI;

$m = new Math();
$m->echoPI();
```

Métodos

- Funções de classes são chamadas de métodos;
- Ao declarar um método é necessário definir o nível de visibilidade para *public*, *protected* ou *private*;
- A declaração deve seguir as regras de declaração de uma função;

```php
class Carro {
    public function andar() {
        // logica para andar
    }
}

$carro = new Carro();
echo $carro->andar();
```

Instanciando classes

- Para criar uma instancia da classe (objeto), use a palavra-chave *new*.
- Para acessar um atributo/método do **objeto**, use o operador “->”;

```
14    $instance = new Exemplo();
```

```
15    $instance->mostraVar();
```

Atividade 1

- Criar as seguintes classes: Escola e Aluno;
- A classe Escola deve conter:
  - Propriedade $alunos = array();
  - Método matricular($aluno) que irá adicionar $aluno à propriedade $alunos;
  - Método mostrarAlunos() que lista os nomes dos alunos matriculados;
- A classe Aluno deve conter:
  - Propriedade $nome;
- Matricular alguns alunos na Escola;
- Mostrar os nomes dos alunos matriculados;

Construtor de classe

- No PHP, o construtor de classe é um método com o nome "*__construct()*";
- O método construtor pode conter parâmetros;

```php
class Carro {
    private $cor;
    public function __construct() {
        echo 'Foi criado um carro';
    }
}

class Moto {
    private $cor;
    public function __construct($cor) {
        $this->cor = $cor;
        echo 'Foi criado um carro';
    }
}

$carro = new Carro();
$moto = new Moto('Vermelho');
```

Destrutor de classe

- No PHP, o destrutor de classe é um método com o nome "*__destruct()*";

```php
class Carro {
    public function __destruct() {
        // objeto destruido
    }
}

$carro = new Carro();
```

Estendendo classes

- Uma classe pode herdar atributos / métodos de outra classe através da palavra-chave *extends* na declaração da classe;
- Não é possível estender de várias classes;

```php
class Veiculo {

    public $cor = 'Vermelho';

    public function mostraCor() {
        echo $this->cor;
    }

}

class Carro extends Veiculo {
    public $rodas = 4;

    public function qtdRodas() {
        echo $this->rodas;
    }
}

$carro = new Carro();
$carro->mostraCor();
$carro->qtdRodas();
```

Visibilidade

- A visibilidade dos membros (métodos e propriedades) deve ser definida adicionando um prefixo *public*, *protected* ou *private* à declaração do mesmo;
  - Membros públicos (public): Pode ser acessado de qualquer lugar;
  - Membros protegidos (protected): Pode ser acessado somente de dentro da classe de origem ou então das classes que estenderam a classe de origem;
  - Membros privados (private): Pode ser acessado somente de dentro da classe de origem;

Visibilidade

```php
class CarroFord {
    public $cor;
    protected $versao;
    private $motor;
}

class Fiesta extends CarroFord {
    public $cor = 'Vermelho';
    protected $versao = 'Fiesta';

    public function mostraMotor() {
        // Notice: Undefined property: Fiesta::$motor
        echo $this->motor;
    }
}

$fiesta = new Fiesta();
$fiesta->mostraMotor();

// Fatal error: Cannot access protected property Fiesta::$versao
echo $fiesta->versao;

// OK
echo $fiesta->cor;
```

Atividade 2

- Criar uma classe Aritmetica com:
  - Propriedades protegidas $n1 e $n2;
  - Métodos públicos para atribuir valores à $n1 e $n2;
- Classe Soma que estende Aritmetica;
  - Método público somar() que retorna a soma entre $n1 e $n2;
- Classe Divisao que estende Aritmetica;
  - Método público dividir() que retorna a divisao de $n1 por $n2;
  - Método público mod() que retorna o mod de $n1 por $n2;
- Usar as classes Soma e Divisao para obter somas, divisões e restos entre 2 números qualquer;

Operador de resolução de escopo (::)

- O “*Operador de resolução de escopo*”, ou simplesmente *Duplo Colon*, é um simbolo que permite acesso a constantes, métodos / propriedades estáticos ou sobrescritas;

Palavra chave static

- A palavra-chave *static* pode ser utilizada para definir membros como estáticos;
- Um membro estático pode ser acessado sem a necessidade da classe ser instanciada.

```php
class Alfabeto {
    public static $letras = 'A..Z';
    public static function pegarLetras() {
        return self::$letras;
    }
}

echo Alfabeto::$letras;
echo Alfabeto::pegarLetras();
```

Palavra chave self

```php
class Math {
    const PI = 3.1415;

    public static function getPI() {
        return self::PI;
    }

    public static function getPI2() {
        return self::getPI() * 2;
    }
}

echo Math::getPI();
echo Math::getPI2();
```

- A palavra-chave *self* é utilizada nos seguintes casos:
  - Dentro de uma classe, para fazer referencia a uma constante dessa mesma classe;
  - Dentro de uma classe, para fazer referencia a um método estático dessa mesma classe (ou de uma classe pai);

Palavra chave parent

- A palavra-chave *parent* é utilizada nos seguintes casos:
  - Dentro de uma classe, para fazer referencia a um método da classe pai que foi sobrescrito;

```php
class Veiculo {
    public $velocidade = 1;
    public function andar() {
        // logica para andar
    }
}

class Carro extends Veiculo {
    public function andar() {
        $this->velocidade = 2;
        parent::andar();
    }
}

$carro = new Carro();
$carro->andar();
```

Sobrepondo propriedades/métodos

- As propriedades / métodos podem ser sobrepostos por redeclaração;
- Os métodos redeclarados podem ser acessados por "*parent::*";

```php
class Veiculo {
    public $velocidade = 1;
    public function andar() {
        // logica para andar
    }
}

class Carro extends Veiculo {
    public function andar() {
        $this->velocidade = 2;
        parent::andar();
    }
}

$carro = new Carro();
$carro->andar();
```

Clonando objetos

- Clonagem é processo de fazer clones de objetos;
- No PHP, existe a palavra-chave "*clone*" que faz clones de objetos, veja o exemplo ao lado:

```php
class Fruta {
    // implementacao
}

$fruta1 = new Fruta();
$fruta2 = clone $fruta1;
```

Classes e métodos abstratos

- Classes abstratas não podem ser **instanciadas**;
- Qualquer classe que contenha um **método abstrato** deve ser declarado como classe abstrata;
- Quando estendemos um classe abstrata, todos os métodos abstratos devem ser declarados;

Classes e métodos abstratos

- Para declarar uma classe como abstrata, somente adicione a palavra-chave “*abstract*” no inicio da declaração da classe;
- Para declara um método abstrato, somente adicione à declaração a palavra-chave “*abstract*” antes da declaração da visibilidade.
- Métodos definidos como abstrato não devem conter a implementação (corpo), somente a assinatura;

```php
abstract class Veiculo {
    abstract public function fabricar($ano);
}

class Ford extends Veiculo {
    public function fabricar($ano) {
        echo "Fabricado na Alemanha em $ano";
    }
}

class Fiat extends Veiculo {
    public function fabricar($ano) {
        echo "Fabricado no EUA em $ano";
    }
}

$ford = new Ford();
$ford->fabricar(2013);

$fiat = new Fiat();
$fiat->fabricar(2013);
```

Interfaces

- Interfaces especificam quais métodos uma classe devem implementar, sem definir como esses métodos irão funcionar.
- Quando uma classe implementa uma Interface, a classe (obrigatoriamente) deve implementar todos os métodos definidos na Interface.

Interfaces

- Para criar uma interface, use a palavra-chave "*interface*";
- Para implementar uma Interface na classe, use a palavra-chave "*implements*";

```
interface Veiculo {
    public function acelerar();
    public function freiar();
}

class Trem implements Veiculo {
    public function acelerar() {
        // implementacao
    }

    public function freiar() {
        // implementacao
    }
}
```

Métodos mágicos

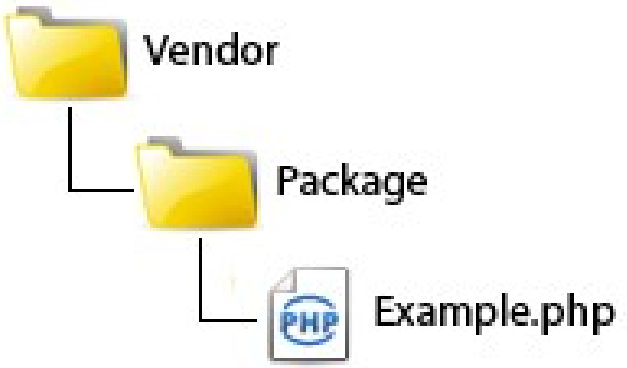
- No PHP existe o recurso chamado de métodos mágicos. Estes métodos gatilhos, que são executados após alguma ação;
- Os métodos mágicos que vamos estudar são as: __construct(), __destruct(), __toString(), __invoke(), __call(), __set(), __get() e __clone();

Função __autoload

- A função __autoload é responsável em carregar classes automaticamente;
- Se definida, ele será automaticamente chamada no caso de uso de uma classe/interface que não foi definida;
- Exemplo de função __autoload() https://github.com/php-fig/fig-standards/blob/maste

Facilitando o uso de __autoload

- 1 classe/interface por arquivo;
- Deve existir um diretório padrão para as classes. Este diretório pode conter subdiretórios;
- O nome da classe deve igual o endereço da classe, dentro do diretório padrão.
- Cada _ (underline) no nome da classe é entendido como um novo diretório;




```
class Vendor_Package_Example {

}
```

Facilitando o uso de __autoload

- Exemplos:
  - DAO_Cliente: CLASSES/DAO/Cliente.php
  - DO_Post: CLASSES/DO/Post.php
  - Conexao: CLASSES/Conexao.php

* CLASSES é o diretório padrão das classes;

Referencias

- “Orientação a Objetos - Wikipédia” - http://pt.wikipedia.org/wiki/Orienta%C3%A7%C3%A3o_a_objetos
- “Classes e Objetos – Manual PHP” - http://www.php.net/manual/en/language.oop5.php
- Normas de códificação PHP PSR e dos Frameworks representados. Disponível em <http://www.fititnt.org/padrao/php-psr.html>

Terminar slide ...

Definições

- Classe: *“Classe representa um conjunto de objetos com características afins. Uma classe define o comportamento dos objetos através de seus métodos, e quais estados ele é capaz de manter através de seus atributos.”* - Wikipédia
-