PHP OO

Paradigma de programação

Programação orientada a objetos

Paradigma?

Um **paradigma de programação** fornece e determina a visão que o programador possui sobre a estruturação e execução do programa.

Principais paradigmas de programação:

Imperativa

Estruturada

Orientada a objetos

Entre outros ...

Programação imperativa

Programas imperativos são uma sequência de comandos para o computador executar. O nome do paradigma, Imperativo, está ligado ao tempo verbal imperativo, onde o programador diz ao computador: faça isso, depois isso, depois aquilo...

Programação estruturada

Programas estruturados preconiza que todos os programas possíveis podem ser reduzidos a apenas três estruturas: sequência, decisão ou seleção e iteração[1] (esta última também é chamada de repetição).

Agora, programação orientada a objetos...

Programação orientada a objetos

É um modelo de análise, projeto e programação de sistemas de *software* baseado na composição e interação entre diversas unidades de software chamadas de objetos.

Conceitos essenciais

Programação orientada a objetos

- Classes
- Instância / Objeto
- Atributo
- Método
- Herança
- Encapsulamento
- Polimorfismo
- Interface
- Pacotes ou namespaces

Vamos abstrair...

Classe no PHP (Fora Java!)

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. Exemplo de classe: Os humanos

Subclasse é uma nova classe que herda características de sua(s) classe(s) ancestral(is)

Definição Simples de uma Classe

```php
<?php
class SimpleClass
{
    // declaração de propriedade
    public $var = 'um valor padrão';

    // declaração de método
    public function displayVar() {
        echo $this->var;
    }
}
?>
```

Atributo

Atributo são características de um objeto. Basicamente a estrutura de dados que vai representar a classe. Exemplos: Funcionário: nome, endereço, telefone, CPF,...; Carro: nome, marca, ano, cor, …; Livro: autor, editora, ano. Por sua vez, os atributos possuem valores. Por exemplo, o atributo cor pode conter o valor azul. O conjunto de valores dos atributos de um determinado objeto é chamado de estado.

Declaração de propriedades

```php
<?php
class SimpleClass
{
   // valid as of PHP 5.6.0:
   public $var1 = 'hello ' . 'world';
   public $var3 = 1+2;

   // valid property declarations:
   public $var6 = myConstant;
   public $var7 = array(true, false);
}
?>
```

Método

Método definem as habilidades dos objetos. Um método em uma classe é apenas uma definição. A ação só ocorre quando o método é invocado através do objeto. Dentro do programa, a utilização de um método deve afetar apenas um objeto em particular; Todos os cachorros podem latir, mas você quer que apenas Bidu dê o latido. Normalmente, uma classe possui diversos métodos, que no caso da classe Cachorro poderiam ser sente, coma e morda.

Definindo e usando um método

```php
<?php
class MinhaClasse
{
    const constante = 'valor constante';

    // declaração de método
    function mostrarConstante() {
        echo  self::constante . "\n";
    }
}
```

Objeto / Instância

Objeto / **Instância** de uma classe. Um objeto é capaz de armazenar estados através de seus atributos e reagir a mensagens enviadas a ele, assim como se relacionar e enviar mensagens a outros objetos. Exemplo de objetos da classe Humanos: João, José, Maria.

```php
<?php
class BaseClass {
   function __construct() {
       print "In BaseClass constructor\n";
   }
}

class SubClass extends BaseClass {
   function __construct() {
       parent::__construct();
       print "In SubClass constructor\n";
   }
}

class OtherSubClass extends BaseClass {
    // inherits BaseClass's constructor
}
?>
```

```php
<?php
// In BaseClass constructor
$obj = new BaseClass();
// In BaseClass constructor
// In SubClass constructor
$obj = new SubClass();
// In BaseClass constructor
$obj = new OtherSubClass();
?>
```

Herança

Herança é o mecanismo pelo qual uma classe pode estender outra classe ou, ainda, ser estendida de outra classe. A grande vantagem deste mecanismo, durante o desenvolvimento de uma aplicação, é evitar a duplicação desnecessária de código, o que pode levar a reduzir o tempo gasto para desenvolver o projeto. **Generalização** é o processo de herança, no qual é criada uma superclasse, a partir de subclasses já existentes. **Especialização** é o processo no qual é criada uma subclasse a partir de superclasse(s) já existentes. Neste último caso, a herança é **simples**, quando uma subclasse herda características de uma única superclasse; a herança é **múltipla**, quando a subclasse herda características de mais de uma superclasse. Um exemplo de herança: Mamífero é super-classe de Humano. Ou seja, um Humano **é um** mamífero.

```php
<?php
class Foo
{
    public function printItem($string)
    {
        echo 'Foo: ' . $string . PHP_EOL;
    }

    public function printPHP()
    {
        echo 'PHP is great.' . PHP_EOL;
    }
}

class Bar extends Foo
{
    public function printItem($string)
    {
        echo 'Bar: ' . $string . PHP_EOL;
    }
}
?>
```

```php
<?php

$foo = new Foo();
$bar = new Bar();
$foo->printItem('baz');
// Output: 'Foo: baz'
$foo->printPHP();
// Output: 'PHP is great'
$bar->printItem('baz');
// Output: 'Bar: baz'
$bar->printPHP();
// Output: 'PHP is great'

?>
```

Encapsulamento

Encapsulamento consiste na separação de aspectos internos e externos de um objeto. Este mecanismo é utilizado amplamente para impedir o acesso direto ao estado de um objeto (seus atributos), disponibilizando externamente os métodos que acessam (getters) e alteram (setters) estes estados. Exemplo: você não precisa conhecer os detalhes dos circuitos de um telefone para utilizá-lo. A carcaça do telefone encapsula esses detalhes, provendo a você uma interface mais amigável (os botões, o monofone e os sinais de tom).

```php
<?php
class MyClass
{
    private $var1 = 'value 1';
    protected $var2 = 'value 2';
    public $var3 = 'value 3';

    protected $protected = 'protected var';
    private   $private   = 'private var';

    private function privado() {}
    protected function protegido() {}
    public function publico() {}
}

$class = new MyClass();

?>
```

Interface

Interface é um contrato entre a classe e o mundo externo. Quando uma classe implementa uma interface, ela está comprometida a fornecer o comportamento publicado pela interface.

```php
<?php
// Declara a interface 'iTemplate'
interface iTemplate
{
    public function setVariable($name, $var);
    public function getHtml($template);
}

// Implementa a interface
class Template implements iTemplate
{
    private $vars = array();

    public function setVariable($name, $var)
    {
        $this->vars[$name] = $var;
    }

    public function getHtml($template)
    {
        foreach($this->vars as $name => $value) {
            $template = str_replace('{' . $name . '}', $value, $template);
        }

        return $template;
    }
}
?>
```

Pacotes ou namespaces

Pacotes (ou Namespaces) são referências para organização lógica de classes e interfaces.