# Orientação a Objetos em PHP

## Classes, funções e objetos
## Construtores e Destrutores
## Herança e Encapsulamento

Material Didático organizado pela Profª. Aline de Campos

# Estruturada *versus* Orientação a Objetos

## Estrutural

Utiliza-se uma lógica de procedimentos onde um programa é composto por diversas funções chamadas de acordo com necessidade.

## Orientação a Objetos

Utiliza-se da lógica de criação de tipos genéricos descritos em classe e dentro destas atributos e métodos. Diversas classes podem compor um único programa.

# Estruturada *versus* Orientação a Objetos

- Modificação na organização de atributos e métodos
- Atributos e métodos podem ser agrupados
- Composições hierárquicas
- Maior robustez e qualidade dos programas
- Redução do esforço de desenvolvimento

# Programação Orientação a Objetos

Baseia-se na criação e utilização de objetos que entendem mensagens particulares e reagem sobre elas. Objetos são compostos por: identidade, estado e operações.

**Cada objeto é uma instância de uma classe.**

# Programação Orientação a Objetos

**Objeto é uma abstração dos objetos reais existentes.**

Aos objetos estão associados:

**estados:**
conjunto de propriedades

**comportamentos:**
ações possíveis

Considerando um objeto carro:

**estados:**
modelo, marca,
cor, ano

**comportamentos:**
acelerar, frear, buzinar

# Classes e Objetos

**Objetos** sempre possuem um **tipo**.
Essa **tipo** é o que chamamos de **Classe**.

Então Classe é a abstração de objetos no intuito de representá-los de forma genérica através de suas características e comportamentos em comum.

# Classes e Objetos

**Atributos:**

Representam o estado de uma classe, suas propriedades, características.

**Métodos:**

Representam ações, procedimentos, funções que alteram o estado dos objetos. Também servem para recuperar informações de objetos.

# Portanto objetos...

- São criados a partir de classes
- São formados por dados e ações
- Os termos objeto e instância de classe são sinônimos
- São elementos sobre os quais podem ser realizadas operações

# Construtores e Destrutores

Na Linguagem de Programação PHP em sua forma orientada a objetos existem duas palavras reservadas específicas para criação de objetos a partir de classes em sua criação e destruição.

**`__construct()`**

é chamada automaticamente quando um novo objeto da classe é construído.

**`__destruct()`**

é chamada automaticamente quando um novo objeto da classe é destruído (unset).

```php
class Cliente {
    public $nome;
    private $numero;
    private $quantidade;

    function __construct($nome,$numero) {
        $this->nome=$nome;
        $this->numero=$numero;
        $this->quantidade=array();
    }

    function getNumero() {
        return $this->numero;
    }
}
```

```php
class Cliente {
    public $nome;
    private $numero;
    private $quantidade;

    function __construct($nome,$numero) {
        $this->nome=$nome;
        $this->numero=$numero;
        $this->quantidade=array();
    }

    function getNumero() {
        return $this->numero;
    }
}
```

```php
class Pessoa {
  private $nome;

  function __destruct() {
    echo "O objeto foi destruído.";
  }
}

$cliente = new Pessoa();

unset($cliente);

echo "O objeto não existe mais";
```

# Encapsulamento e Visibilidade

Consiste em utilizar modificadores de acesso para indicar a visibilidade de atributos e métodos.

**private**
acesso apenas dentro da própria classe em que foram declarados.

**protected**
acesso dentro da classe ou a partir de subclasses (herança).

**public**
acesso de forma livre, a partir da própria classe, a partir de classes descendentes e a partir de programas que fazem uso dessa classe. Quando não declarada a visibilidade automaticamente esta será do tipo public.

# Encapsulamento e Visibilidade

| | Na classe | Na subclasse | Geral |
|---|---|---|---|
| Public | x | x | X |
| Private | x | | |
| Protected | x | x | |

```php
class Cliente {
    public $nome;
    private $numero;
    private $quantidade;

    function __construct($nome,$numero) {
        $this->nome=$nome;
        $this->numero=$numero;
        $this->quantidade=array();
    }

    function getNumero() {
        return $this->numero;
    }
}
```

```php
class Cliente {
    public $nome;
    private $numero;
    private $quantidade;

    function __construct($nome,$numero) {
        $this->nome=$nome;
        $this->numero=$numero;
        $this->quantidade=array();
    }

    function getNumero() {
        return $this->numero;
    }
}
```

```php
// instanciar dois objetos cliente
$cliente1 = new Cliente("Pedro", 1);
$cliente2 = new Cliente("Roberto", 564);


echo "O identificador do cliente 1 é: " .
$cliente1->getNumero();

echo "O identificador do cliente 2 é: " .
$cliente2->getNumero();
```

# Herança

Utilizando o paradigma de Orientação a Objetos outro conceito possível de ser implementado em PHP é o de Herança. Sendo assim, pode-se derivar classes gerando subclasses que herdam os atributos e métodos de sua classe base e superclasses.

**extends**

palavra reservada que designa a relação de herança entre uma classe e sua superclasse.

```php
class ClientePJ extends Cliente {
  private $cnpj;

  function __construct($nome, $numero, $cnpj)
  {
    parent::__construct($nome, $numero);
    $this->cnpj= $cnpj;
  }
}


$cliente3 = new ClientePJ("João", 45, 5234);
echo $cliente3->getNumero();
```

**Referências:**

DALL'OGLI, Pablo. PHP: Programando com Orientação a Objetos. São Paulo: Novatec, 2007.
Arquivos de Códigos. Disponível em: <http://www.arquivodecodigos.net>