

## Módulo 2 – Fundamentos OO

Neste workbook do módulo 2, revisaremos os principais conceitos vistos nas videoaulas.

### Programação procedural

O software é construído por um conjunto de procedimentos, nem sempre organizados em um contexto. No PHP, declarados com a palavra-chave function.

```
<?php
function total_debitos_cliente($id_cliente) {}
function registra_venda($id_cliente, $data) {}
function registra_venda_item($id_venda, $item) {}
function consulta_estoque($produto) {}
function consulta_valor($produto) {}
function gera_financeiro($id_cliente, $total, $parcelas) {}
```

### Orientação a Objetos

O software é modelado e construído por um conjunto de objetos relacionados. Cada objeto possui atributos e métodos. Cada objeto é criado com o operador new.

```
<?php
class Produto {
    public $descricao;
    public $estoque;
    public $preco;
}

$p1 = new Produto;
$p1->descricao = 'Chocolate';
$p1->estoque = 10;
$p1->preco = 7;

$p2 = new Produto;
$p2->descricao = 'Café';
$p2->estoque = 20;
$p2->preco = 4;
```

## Método construtor

É executado na instanciação do objeto. Sempre que você executar o operador new, o `__construct()` é executado, recebendo os parâmetros de criação.

```
<?php
class Produto
{
    public function __construct($descricao, $estoque, $preco) {
```

## Conversão de tipo

Objetos podem ser convertidos para array, por meio de operação de cast.

```
<?php
$produto = new stdClass;
$produto->descricao = 'Chocolate Amargo';
$produto->estoque = 100;
$produto->preco = 7;

$vetor1 = (array) $produto;
```

## Herança

A capacidade de herdar comportamento entre classes é acionada pela palavra-chave `extends`, indicando a classe pai da qual vamos absorver o comportamento.

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

## Classe abstrata

Uma classe usada como base para outras. Uma classe abstrata não pode ser instanciada diretamente.

```
<?php
abstract class Conta {
    // ...
}
```

## Encapsulamento

Define como um atributo ou método será acessado.

- `public`: acessado livremente.
- `private`: acessado internamente à classe.
- `protected`: acessado internamente e por classes filhas.

## Constantes de classe

Armazenam valores fixos relacionados à classe. Declarada com a palavra-chave `const`.

```
<?php
class Pessoa {
    private $nome;
    private $genero;
    const GENEROS = array('M'=>'Masculino', 'F'=> 'Feminino');
```

## Propriedade estática

Retém um atributo valor em nível de classe. Todos objetos daquela classe podem acessá-lo.

```
<?php
class Software {
    private $id;
    private $nome;
    public static $contador;
```

## Interfaces

Fornecer uma especificação que deve ser implementada. Esta especificação é formada por um conjunto de métodos com seus respectivos parâmetros (assinatura).

```
<?php
interface OrcavelInterface {
    public function getPreco();
}
class Servico implements OrcavelInterface {
```

## Outras referências:

- Funções para objetos: [https://www.php.net/manual/pt\\_BR/ref.classobj.php](https://www.php.net/manual/pt_BR/ref.classobj.php)
- Padrões de projetos: [https://en.wikipedia.org/wiki/Design\\_Patterns](https://en.wikipedia.org/wiki/Design_Patterns)
- GAMMA, Erich, HELM, Richard, JOHNSON, Ralph, VLISSIDES, John. Design patterns: elements of reusable object-oriented software, Addison-Wesley Longman Publishing Co., Inc., Boston, MA, 1995

Livro PHP Programando com Orientação a Objetos



[www.adianti.com.br/phpoo](http://www.adianti.com.br/phpoo)