# PHP Orientado a Objetos para Iniciantes

Programação orientada a objetos é um estilo de programação que permite os desenvolvedores agruparem tarefas semelhantes em **classes**. Isso ajuda a mantermos-nos dentro do princípio ***não se repita***, além de facilitar a manutenção do código.

## Compreendendo Objetos e Classes

Há uma grande confusão na POO: quando programadores de longa data falam sobre objetos e classes, esses termos parecem permutáveis entre si. Porém, não é bem assim, mesmo que a diferença entre eles seja um pouco complicada de perceber, no início.

Uma classe, por exemplo, é como uma **planta baixa para uma casa**. Ela define a forma da casa no papel, com as relações entre as diferentes partes da casa, claramente definidas e planejadas, mesmo a casa ainda não existindo.

Um objeto, por outro lado, **seria a casa de verdade**, construída de acordo com a planta baixa.

### Estruturando Classes

A sintaxe para criar uma classe é bem direta: declare-a usando a palavra chave **class**, seguida do nome da classe e um par de chaves ({}):

```php
<?php

class MinhaClasse
{
      // As propriedades e métodos da Classe vem aqui
}
?>
```

Após criar a classe, ela pode ser instanciada e guardada em alguma variável usando a palavra chave new:

```php
$obj = new MinhaClasse;
```

Para vermos o conteúdo da classe, usamos var_dump():

> Esta função exibe informações estruturadas sobre uma ou mais expressões que incluem seu tipo e valor.

```php
var_dump($obj);
```

Experimente isso, colocando todo o código anterior em um único arquivo php, chamado teste.php na sua pasta local de /htdocs/**Aula2and3**

```php
<?php

class MinhaClasse
{
  // As propriedades e métodos da Classe vem aqui
}

$obj = new MinhaClasse;

var_dump($obj);

?>
```

Teste no navegador: http://localhost/Aula2and3/teste.php

De uma forma bem simples, você acabou de criar seu primeiro código em POO.

## Definindo as Propriedades da Classe

Para adicionar dados à classe, usamos as **propriedades**, que são variáveis específicas à classe.

Elas funcionam de forma parecida às variáveis normais, exceto que elas estão ligadas ao objeto e só podem ser acessadas usando o objeto.

Para adicionar uma propriedade a **MinhaClasse,** adicione o seguinte trecho de código ao seu código:

```
<?php

class MinhaClasse
{
        public $prop1 = "Sou um propriedade de classe!";
}

$obj = new MinhaClasse;

var_dump($obj);

?>
```

Teste no navegador: http://localhost/Aula2and3/teste.php

A palavra chave public determina a visibilidade da propriedade, a qual você aprenderá mais sobre, nas próximas seções.  Membros de classes declarados como públicos podem ser acessados de qualquer lugar.

Modifique o código do arquivo teste.php para ler a propriedade ao invés de mostrar todo o conteúdo da classe, dessa forma:

```
<?php

class MinhaClasse
{
  // As propriedades e métodos da Classe vem aqui
        public $prop1 = "Sou um propriedade de classe!";
}

$obj = new MinhaClasse;

echo $obj->prop1; // Mostra a saída/conteúdo da propriedade
?>
```

Teste no navegador: http://localhost/Aula2and3/teste.php

## Definindo Métodos de Classe

**Métodos** são funções específicas das classes. Ações particulares que os objetos serão capazes de executar são definidas dentro das classes na forma de métodos.

Por exemplo, para criar métodos que atribuam e retornem o valor de uma propriedade, crie um novo arquivo em Aula2and3 chamado **TesteMetodo.php** e adicione o código a seguir:

```php
<?php

class MinhaClasse2
{
    public $propriedade1 = "Sou uma propriedade de classe";

    public function setPropriedade1($v)
    {
      $this->propriedade1 = $v;
    }

    public function getPropriedade1()
    {
      return "Método que imprime: ". $this->propriedade1 . "<br />";
    }
}//fim da classe

$obj = new MinhaClasse2;

echo "Olá.  $obj->propriedade1. <br>";
echo $obj->getPropriedade1();

  //atualizando a propriedade
  $obj->setPropriedade1("Novo valor a propriedade");
  echo "Olá ".$obj->propriedade1."<br>";
  echo $obj->getPropriedade1();
?>
```

"O poder da POO mostra-se ao usar múltiplas instâncias da mesma classe."

```php
<?php

class MinhaClasse2
{
  public $propriedade1 = "Sou uma propriedade de classe!";

  public function setPropriedade1($v)
  {
     $this->propriedade1 = $v;
  }

     public function getPropriedade1()
    {
        return "Método que imprime: ". $this->propriedade1 . "<br />";
    }
}// fim da minhaClasse2

   $obj = new MinhaClasse2;
   $obj2 = new MinhaClasse2;

   //atualizando a propriedade 1
   $obj->setPropriedade1("Novo valor a propriedade do objeto 1");
   echo "Olá ". $obj->propriedade1"<br>";
   echo $obj->getPropriedade1();
   echo "<br><br>";
   //atualizando a propriedade 2
   $obj2->setPropriedade1("Novo valor a propriedade do objeto 2");
   echo "Olá ". $obj2->propriedade1"<br>";
   echo $obj2->getPropriedade1();
?>
```

## Usando Construtores e Destruidores

Quando um objeto é instanciado, é desejável que algumas coisas ocorram de cara.

Para lidar com isso, o PHP provê o método __construct(), que é chamado automaticamente quando um novo objeto é criado.

Para ilustrar os conceitos dos construtores, crie um novo arquivo chamado **TesteConstrutor.php** e adicione um construtor à classe MinhaClasse3 que mostre uma mensagem toda vez que uma nova instância for criada:

```
<?php

class MinhaClasse3
{
  public $prop1="Sou uma propriedade de classe!";

  public function __construct()
  {
     echo "A classe, ".  __CLASS__." ,  foi instanciada!<br />";
  }

  public function setProp1($newval)
  {
     $this->prop1 = $newval;
  }

  public function getProp1()
  {
     return $this->prop1 . "<br />";
  }
}

// Cria um novo objeto
$obj = new MinhaClasse3;

// Mostra o valor de $prop1
echo $obj->getProp1();

// Mostra uma mensagem ao final do arquivo
echo "Fim do arquivo.<br />";

?>
```

**Nota** :  __CLASS__ retorna o nome da classe na qual foi usado.