O que é Programação Orientada a Objetos?

POO (Programação Orientada a Objetos) é um paradigma de programação que organiza o código em "objetos", que são instâncias de classes.

Estrutura de uma Classe em PHP

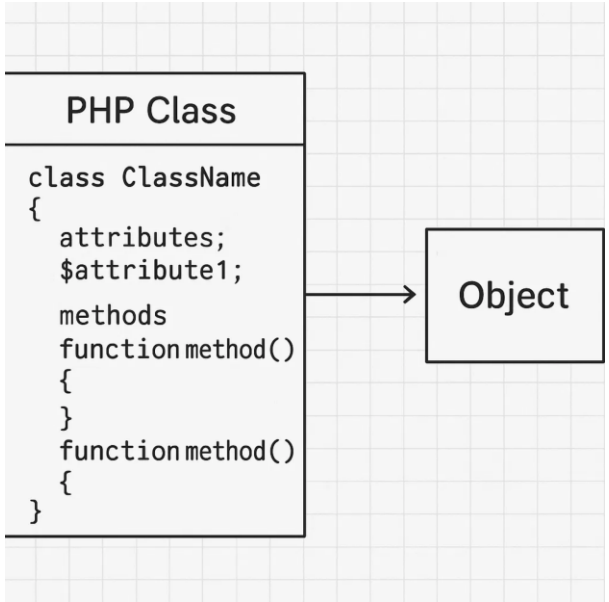
```php
<?php
class Pessoa {
    // Atributos
    public $nome;
    public $idade;

    // Método (função dentro da classe)
    public function dizerOla() {
        echo "Olá, meu nome é $this->nome!";
    }
}
?>
```

Classe: Modelo/estrutura.

Atributos: Variáveis da classe.

Métodos: Funções da classe.

$this: Referência ao próprio objeto.



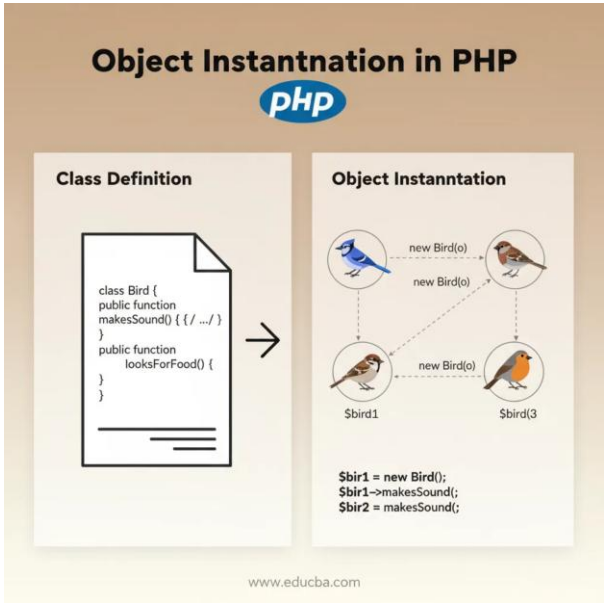
Criando e Usando Objetos

```php
<?php
$p1 = new Pessoa();
$p1->nome = "João";
$p1->idade = 30;

$p1->dizerOla();
                                // Saída: Olá, meu nome é João!

?>
```

Após definir uma classe, podemos:

- Criar instâncias usando o operador **new**
- Acessar atributos usando o operador **->**
- Chamar métodos usando o operador **->**

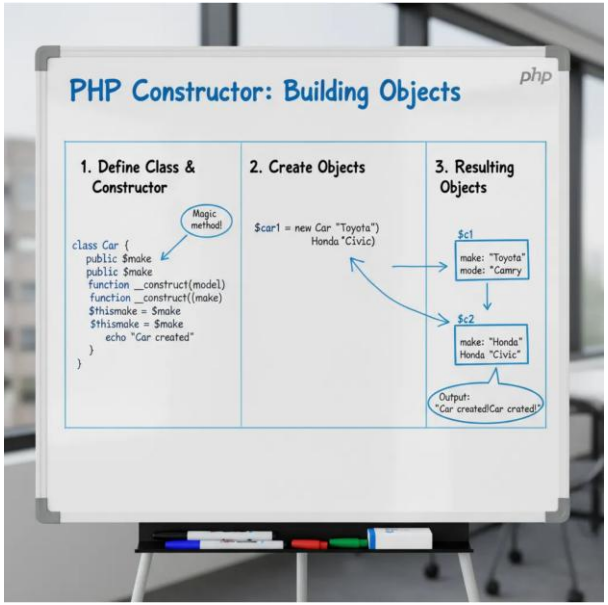
  Criar múltiplos objetos da mesma classe



Construtor (__construct())

Construtores são métodos especiais chamados automaticamente ao criar um objeto.

```php
<?php
class Pessoa {
    public $nome;
    public $idade;

    public function __construct($nome, $idade) {
        $this->nome = $nome;
        $this->idade = $idade;
    }

    public function apresentar() {
        echo "Sou $this->nome e tenho $this->idade anos.";
    }
}

$p2 = new Pessoa("Maria", 25);
$p2->apresentar(); // Saída: Sou Maria e tenho 25 anos.
?>
```

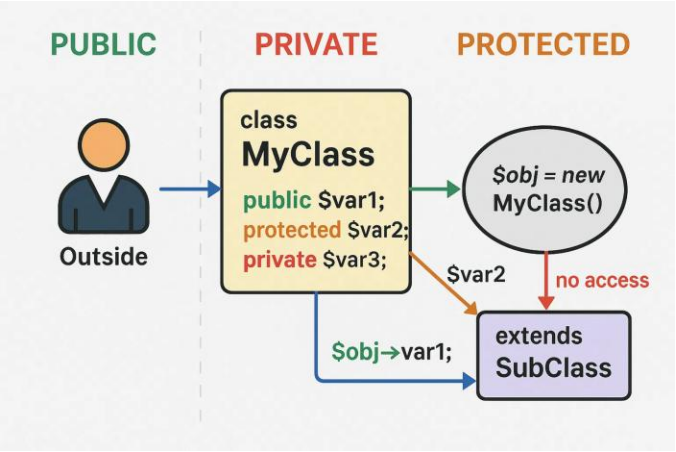


Visibilidade (public, private, protected)

Define o nível de acesso aos atributos e métodos.

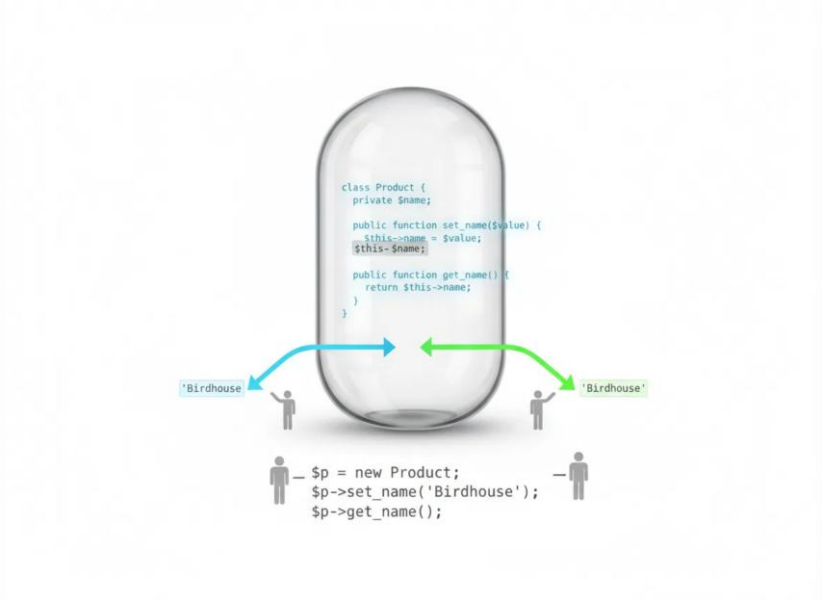
Modificador	Acesso na própria classe	Acesso fora da classe	Acesso em herança
public	✅	✅	✅
private	✅	❌	❌
protected	✅	❌	✅

```
class Conta {
    private $saldo = 0;

    public function depositar($valor) {
        $this->saldo += $valor;
    }

    public function verSaldo() {
        return $this->saldo;
    }
}
```



Encapsulamento com Getters e Setters

```
class Produto {
    private $preco;

    public function setPreco($p) {
        if ($p > 0) {
            $this->preco = $p;
        }
    }

    public function getPreco() {
        return $this->preco;
    }
}
```



Encapsulamento é um dos princípios fundamentais da POO que:

- Protege os dados de acesso direto
- Permite validação de dados antes de atribuir valores
- Oculta a implementação interna da classe
- Facilita a manutenção do código

Exercício Proposto

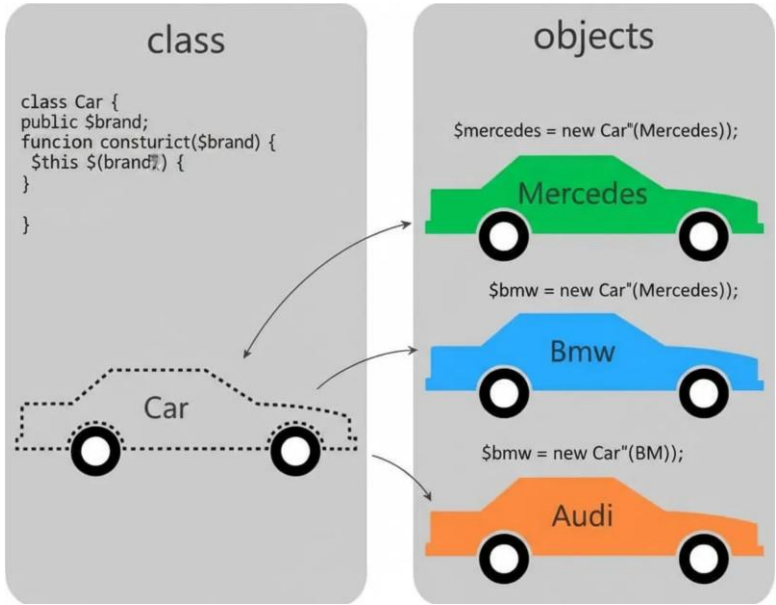
Crie uma classe Carro com os seguintes requisitos:

Atributos: marca , modelo , ano .

Método construtor.

Método exibirDetalhes() que imprime as informações do carro.

```
// Espaço reservado para o aluno

class Carro {
   // Seus atributos aqui

   // Seu construtor aqui

   // Seu método exibirDetalhes() aqui
}
```



Desafio Extra

Implemente uma classe chamada Aluno que tenha:

Atributos: nome , nota1 , nota2

Método: calcularMédia()

Verifique se o aluno está aprovado (média >= 7)

```
class Aluno {
   // Seus atributos aqui

   // Seu construtor aqui

   // Seu método calcularMédia() aqui

   // Seu método para verificar aprovação aqui
}

// Exemplo de uso:
$aluno = new Aluno("João", 8, 6);
echo "Média: " . $aluno->calcularMédia();
echo "Situação: " . ($aluno->estaAprovado() ? "Aprovado" : "Reprovado");
```