PROGRAMAÇÃO ORIENTADA A OBJETOS EM PHP

Prof. Me. Hélio Lourenço Esperidião Ferreira

O que é POO?

- POO é um **paradigma de programação** baseado na modelagem do software através de **objetos que representam conceitos do mundo real ou do domínio do sistema**.
- **Domínio** = o problema que o software resolve.

Paradigma

Paradigma

- do latim tardio paradigma, do grego *παράδειγμα*, derivado de *παραδείκνυμι* «mostrar, apresentar, confrontar»)
- É um conceito das ciências que define **um exemplo típico ou modelo de algo**. É a representação de um padrão a ser seguido.

Paradigma

- Um exemplo que serve como modelo; padrão.
- É a representação de um padrão a ser seguido.

Paradigmas de programação

- O paradigma de programação é uma forma ou abordagem que guia a maneira como os desenvolvedores organizam e estruturam o código.
- Define a metodologia utilizada para conceber a execução e a arquitetura do programa.

O que é uma classe?

- Em **Programação Orientada a Objetos (POO)**, uma **classe** é um **MODELO** ou **MOLDE** para criar objetos.
- Ela define um conjunto de **atributos** (dados) e **métodos** (comportamentos) que os objetos dessa classe terão.

Atributos

- Os atributos são características.
- São utilizados para descrever alguma coisa.
- Quais os atributos da sala?
- Quais os atributos da mesa?
- Quais os atributos de um personagem de jogo?

Atributos

- São as características ou propriedades de um objeto.
- Representam o estado de um objeto e podem ser de diferentes tipos de dados, como números, strings, booleanos, entre outros.
- Se tivermos uma classe(Modelo) "Pessoa" com os atributos "nome", "idade" e "sexo",
- Podemos criar vários objetos dessa classe “Pessoa”, cada um com valores diferentes para esses atributos.
    - O objeto "João", por exemplo, pode ter o atributo "nome" definido como "João", "idade" como "30" e "sexo" como "masculino", enquanto

Métodos

- São “funções” que pertencem a uma classe ou objeto e são usadas para realizar ações ou operações em objetos dessa classe.
- Representam o comportamento do objeto e permitem que o objeto interaja com o mundo exterior.
- Por exemplo, se tivermos uma classe "**Pessoa**" com os métodos "andar", "falar" e "trabalhar", podemos chamar esses métodos em objetos dessa classe para realizar essas ações.
    - O método "andar" pode definir como o objeto se move, o método "falar" pode permitir que o objeto emita sons, e o método "trabalhar" pode definir como o objeto executa uma tarefa.

Instanciar

Instanciar significa criar um objeto a partir de uma classe.

Cada instância tem seus próprios dados (atributos) e pode executar os métodos definidos na classe.

Podemos criar várias instâncias de uma mesma classe, cada uma representando um elemento individual do modelo.

Instância

Uma instância é um objeto criado a partir de uma classe. Quando você instancia uma classe, está criando um objeto específico baseado no modelo definido pela classe.

Analogia: Pense em uma classe como uma receita de bolo. A receita define os ingredientes (atributos) e o modo de preparo (métodos).

A **instância** é um bolo real feito a partir da receita.

Você pode criar vários bolos (instâncias) a partir da mesma receita (classe), mas cada bolo pode ter características diferentes (exemplo: sabor, tamanho).

Classe vs Objeto

CLASSE	**OBJETO**
É um **modelo** ou **molde** para criar objetos.	É uma **instância** de uma classe.
Define **atributos** (dados) e **métodos** (comportamentos).	Possui valores concretos para os atributos e pode executar os métodos.
Não ocupa espaço na memória até que um objeto seja criado.	Ocupa espaço na memória ao ser instanciado.
Representa uma **ideia abstrata** (exemplo: "Carro").	Representa um **exemplo real** daquela ideia (exemplo: "Um carro vermelho da marca X").

Regras para nomear classes

```
class Carro {}
class ContaBancaria {}
class UsuarioSistema {}
```

- A nomenclatura utilizada para definir o nome de classes segue o padrão de PascalCase (também chamado de UpperCamelCase).

1. O nome da classe deve começar com letra maiúscula.
2. Se houver mais de uma palavra, cada palavra deve iniciar com letra maiúscula.
3. Não use caracteres especiais ou espaços (apenas letras e números, além de _ se necessário).
4. O nome da classe deve ser descritivo, representando o que a classe faz.

Regras para nomear métodos

- A nomenclatura utilizada para definir nomes de **métodos** segue o padrão **camelCase** (também chamado de **lowerCamelCase**).

1. O nome do método deve começar com letra minúscula.
2. Se houver mais de uma palavra, a partir da segunda palavra, cada uma deve iniciar com letra maiúscula.
3. O nome deve ser descritivo e representar a ação que o método executa.
4. Não use caracteres especiais e abreviações.
5. Métodos que retornam um valor booleano geralmente começam com "is" ou "has" (exemplo: isAtivo(), hasPermissao()).

Regras para nomear variáveis

- O nome da variável deve começar com uma letra minúscula.
- Se o nome tiver mais de uma palavra, a segunda palavra em diante deve começar com letra maiúscula.
- Não use caracteres especiais
- Não use palavras reservadas
- Use nomes descritivos que representem o propósito da variável.
- Evite nomes muito curtos ou muito longos
  - Exemplo ruim: a ou informacaoDoUsuarioSobreCadastroNoSistema
  - Exemplo bom: informacaoUsuario

Regras para o nomear arquivos de classes:

- Tipicamente um arquivo para cada classe.
- O nome do arquivo deve ser igual ao nome da classe.

Encapsulamento

- O encapsulamento é um dos pilares da Programação Orientada a Objetos (POO) e consiste em restringir o acesso direto aos atributos de uma classe, permitindo que eles sejam manipulados apenas através de métodos específicos (como get e set).
- O que é método?
- O que é atributo?
- O que é um método get?
- O que é um método set?

Importante

O que é método?	São “funções” implementadas por uma classe.
O que é atributo?	São “variáveis” ou características que definem um objeto.
O que é um método get?	É uma “função” que **retorna** o conteúdo de um determinado atributo.
O que é um método set?	É uma “função” que **insere** o conteúdo de um determinado atributo.

Método set e get

- método set (ou setter) é usado para modificar o valor de um atributo privado.
- Vantagens do set e get
  - Permitem controle sobre os dados (ex.: validar entradas).
  - Impedem modificações indevidas dos atributos.
  - Melhoram a segurança e manutenção do código.
- Getters e Setters garantem o encapsulamento e segurança dos dados.
  - Evitam acesso direto aos atributos privados.
  - Permitem validações antes de alterar valores.
  - Boas práticas recomendam sempre encapsular atributos privados e expô-los por meio de métodos get e set!

Abstração

- A capacidade de se concentrar nos aspectos fundamentais de um contexto, desconsiderando características secundárias ou menos importantes, é conhecida como abstração.
- Na modelagem orientada a objetos, uma classe representa uma abstração de uma entidade presente no domínio do sistema de software. Algumas classes comuns incluem Pedido, Produto e Cliente.

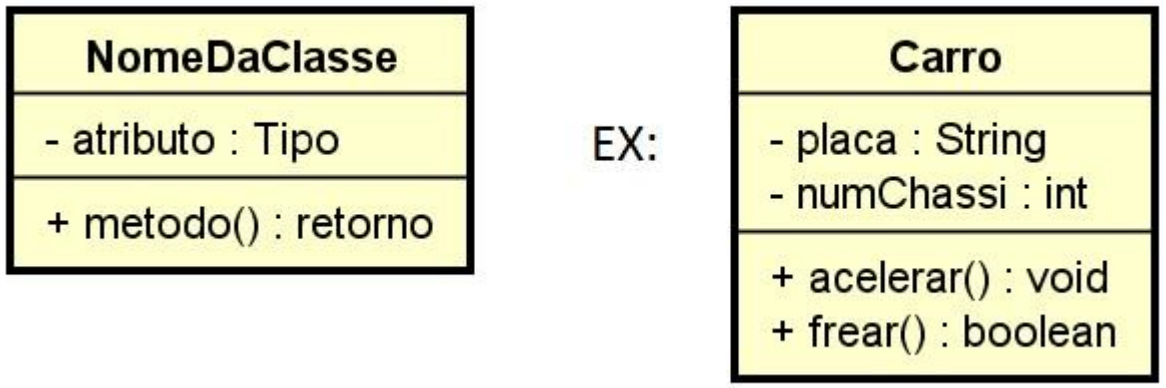
O processo de abstração.

- O resultado do processo de abstração em programação orientada a objetos é uma **classe**.

- As classes podem ser consideradas **MODELOS** (moldes ou formas) que podem dar origem a diversos objetos.

Diagrama de classes

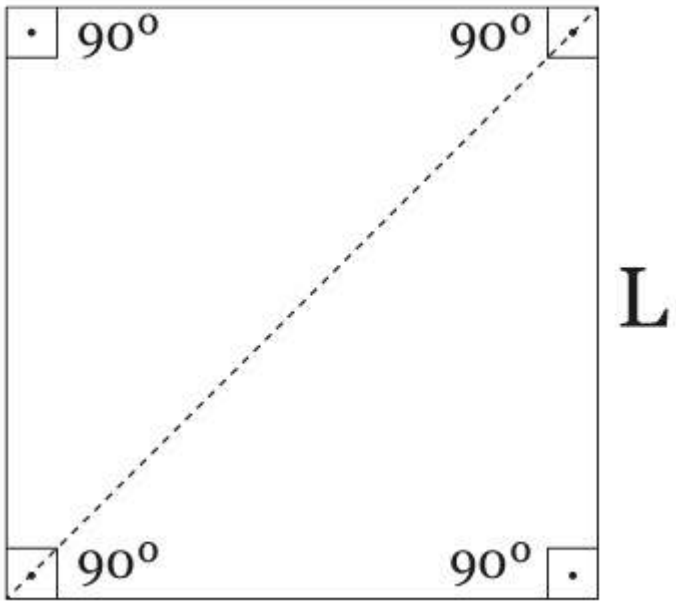
- Pode representar uma classe por meio de símbolos.
- (-) significa privado, pode ser utilizado apenas dentro da classe
- (+) significa publico, pode ser utilizado em todas as outras classes.

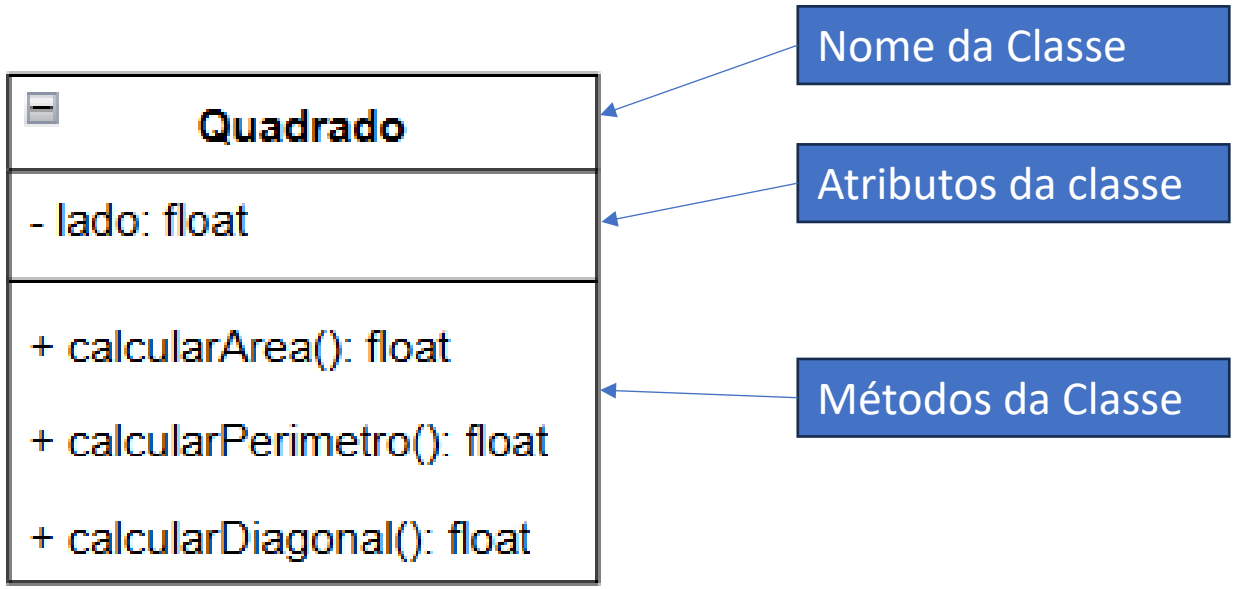


Abstraindo...

- **Quais as propriedades/Atributos de um Quadrado?**

Diagrama da classe Quadrado

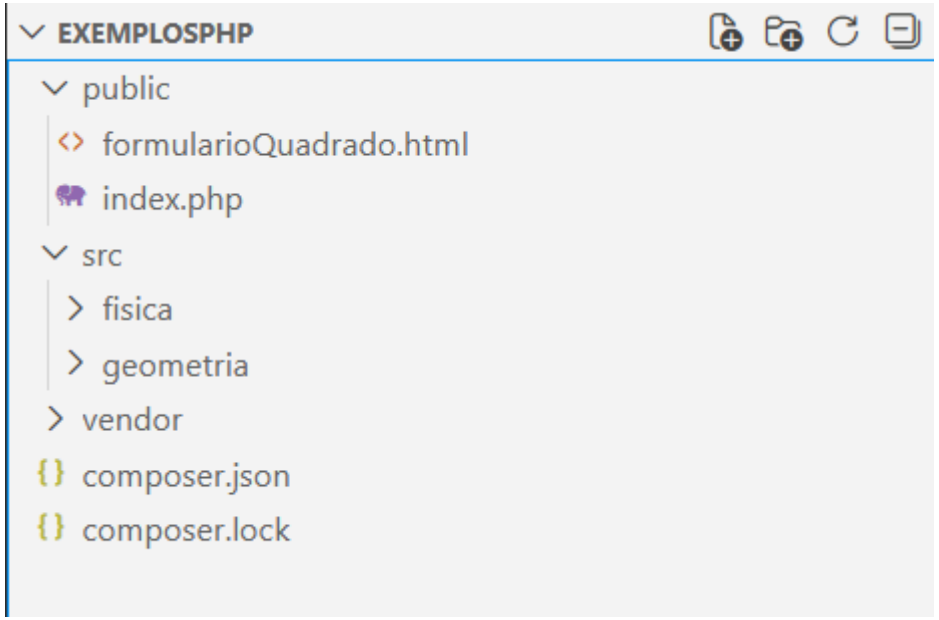




O que é um namespace no PHP?

- Namespace é um "espaço de nomes" usado para organizar o código e evitar conflitos entre classes, funções e constantes que tenham o mesmo nome.
- **Classes são agrupadas em Namespaces** para organizar melhor o código e evitar conflitos de nomes.
- Um **namespace** funciona como um “pacote” ou “pasta lógica” que agrupa classes relacionadas.

Estrutura de diretórios



composer.json

```
{
    "require": {
        "slim/slim": "^4.15",
        "slim/psr7": "^1.8"
    },
    "autoload": {
        "psr-4": {
            "src\\": "src/"
        }
    }
}
```

Ao alterar o arquivo composer.json
rode no terminal:
composer dump-autoload

```php
<?php
declare(strict_types=1); //ativa tipagem restrita
namespace src\geometria;
class Quadrado{
    public function __construct(
        private float $lado
    ) {}
    public function calcularArea(): float
    {
        return $this->lado * $this->lado;
    }
    public function calcularPerimetro(): float
    {
        return 4 * $this->lado;
    }
    public function calcularDiagonal(): float
    {
         return $this->lado * sqrt(2); // d = l * √2
    }
    public function setLado(float $novoLado): void
    {
        $this->lado = $novoLado;
    }
    public function getLado(): float
    {
        return $this->lado;
    }
}
```

Classe: Quadrado.php

Para cada atributo deve existir um get e um set

Caso uma classe possua 10 atributos, somando os métodos
Get e set obtemos um total de?

`$this:` Faz referência a recursos do objeto

Os objetos da classe Quadrado possuem os recuros: $lado, CalcularArea(), calcularPerimetro(), etc....

INVARIANTE

- Uma invariante é uma regra que **nunca pode ser violada durante a vida do objeto**
- Exemplo no Quadrado
  - lado > 0
- Se essa regra for quebrada, o objeto deixa de representar um quadrado válido.

Validação de dados

- A **validação de dados no método set** (no seu caso, setLado) é uma prática essencial para garantir a **integridade da regra de domínio** e evitar que o objeto fique em um estado inválido.
- No seu exemplo da classe Quadrado, o método ***setLado*** é o ponto responsável por alterar o valor do lado.
    - Portanto, ele deve garantir que esse valor seja válido antes de aplicá-lo.
- **O que a validação protege?**
    - No contexto geométrico, um quadrado:
        - Não pode ter lado negativo
        - Não pode ter lado zero (dependendo da regra)
        - Deve ter lado maior que zero

Exemplo com validação

- **O que está acontecendo aqui?**
  - O método recebe o valor.
  - Antes de alterar o estado do objeto, verifica a regra.
  - Se for inválido, lança uma exceção.
  - O objeto nunca entra em estado inconsistente.

```php
public function setLado(float $novoLado): void{
    if ($novoLado < 0) {
        throw new \InvalidArgumentException(
            'O lado do quadrado não pode ser negativo.'
        );
    }
        $this->lado = $novoLado;
}
```

Conceito importante

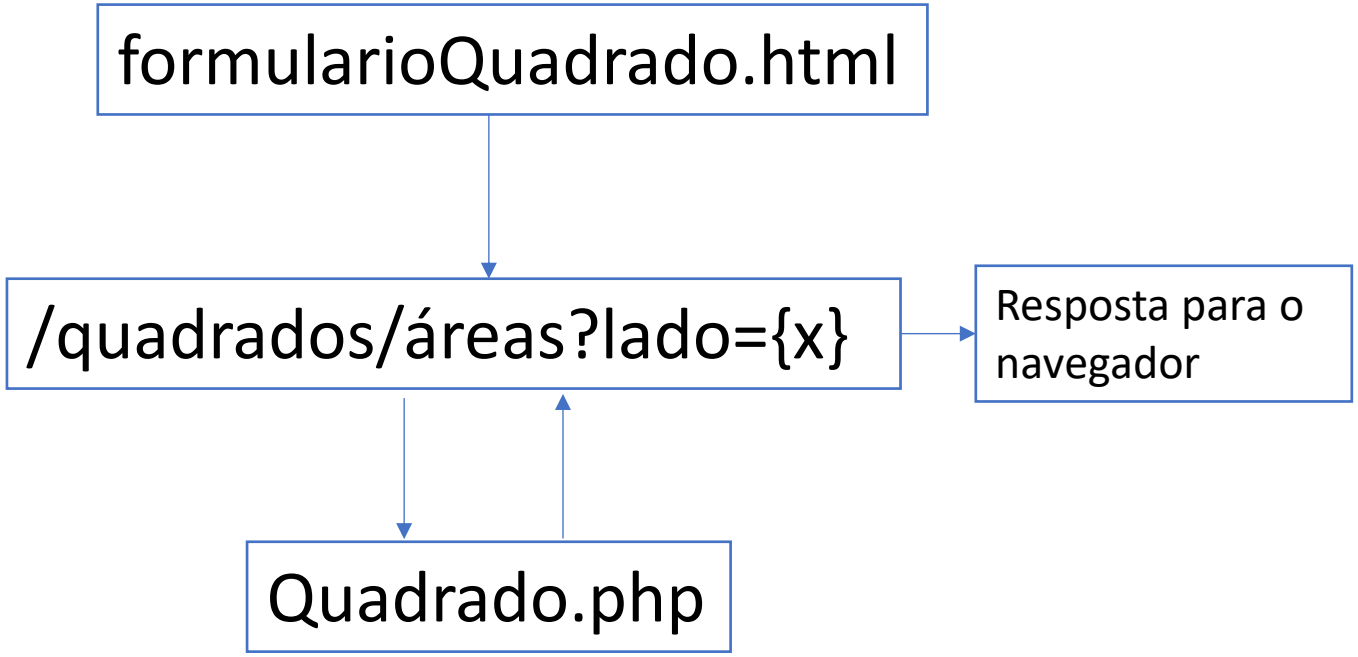
- Validar no set garante:
  - Encapsulamento
  - Proteção do estado interno do objeto
  - Confiabilidade dos cálculos (calcularArea, calcularPerimetro, etc.)
  - Código mais seguro e previsível

Erro comum

- Validar apenas no ***formulário, roteador*** ou ***controller*** e não na entidade.
  - Isso é perigoso, porque alguém pode instanciar a classe diretamente no código e passar valores inválidos.
- A entidade deve ser **auto-protegida**.

Arquivos / Arquitetura

- Formulário:
  - formularioQuadrado.html
- "/quadrados/areas"
- Classe:
  - Quadrado.php



Estrutura de hierarquia.

- htdocs
  - formularioQuadrado.html
  - controle_Quadrado.php
  - Quadrado.php

Utilizando a classe

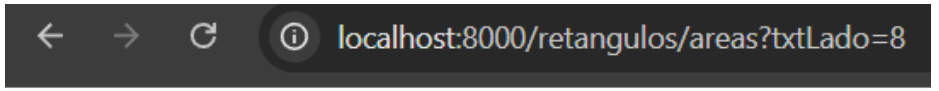
formularioQuadrado.html

```
<!DOCTYPE html>
<html>
<head>
    <title>Formulários</title>
</head>
<body>
    <div>
        <form action="/retangulos/areas" method="get">
            <input type="text" name="txtLado"><br>
            <input type="submit" value="Calcular área">
        </form>
    </div>
    <br>
</body>
</html>
```

```php
<?php
// Carregar autoload do Composer
require __DIR__ . '/../vendor/autoload.php';
use src\geometria\Quadrado;
use Slim\Factory\AppFactory;
use Psr\Http\Message\ResponseInterface as Response;
use Psr\Http\Message\ServerRequestInterface as Request;
use Psr\Http\Message\ResponseInterface as ResponseInterface;
// Criar aplicação Slim
$app = AppFactory::create();

$app->get(
    '/retangulos/areas',
    function (Request $request, Response $response){
        $dados = $request->getQueryParams();
        $lado = $dados["txtLado"] ?? 0;
        $quadrado = new Quadrado();
        $quadrado->setLado($lado);
        $area = $quadrado->calcularArea();
        $resultado = "area = $area";
        $response->getBody()->write($resultado);
        return $response;
    }
);
// Executar a aplicação
$app->run();
```

Controller para o endpoint
GET: `'/retangulos/areas`



area = 64

Abstraindo. ..

- Quais as propriedades/Atributos de um Retângulo?
- Quais os métodos ou ações poderiam ser realizadas por uma classe retângulo?

Classe (Retangulo)

- Atributos (Características)
  - base
  - altura

- Métodos
  - CalcularArea()
  - CalcularDiagonal()
  - CalcularPerimetro()