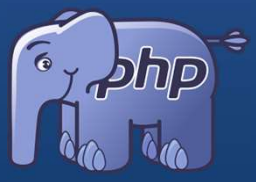
Treinamento Fundamentos de PHP

Turma 2023/01 – Aula 5/8



1

Semana 2

Aula 5 – Orientação a Objetos

- Encapsulamento
- Herança
- Polimorfismo
- Abstração
- Atributos e métodos estáticos
- Namespaces

2

Recapitulando...

Na última aula vimos sobre:

- PHP na web: pegando dados do formulário
- Pilha de execução (Call Stack)
- Erros e Exceções
- Tratamento de exceções
- Lançamento de exceções personalizada
- Atividade 3: Calculadora WEB

3

Programação Orientada a Objetos

- Paradigma de programação que organiza o código em torno de objetos, que representam entidades do mundo real ou conceitos abstratos
- Em vez de construir sistemas com um conjunto de procedimentos e variáveis, assim como se faz em linguagens estruturais, em OO utilizamos uma lógica bem próxima do mundo real, lidando com objetos, estruturas que já conhecemos e sobre as quais possuímos uma grande compreensão.
- Pilares: encapsulamento, herança, polimorfismo e abstração

4

Programação Orientada a Objetos

Classe x Objeto

- **Classe**: é um modelo ou uma definição para criar objetos. Ela especifica quais atributos e métodos um objeto terá quando for instanciado a partir dela. As classes são usadas para criar vários objetos semelhantes e fornecem uma estrutura para organizar o código
- **Objeto**: é uma instância de uma classe. Quando uma classe é instanciada, é criado um objeto específico com seus próprios atributos e métodos. Os objetos são criados com base na definição da classe e podem interagir uns com os outros.

5

Programação Orientada a Objetos

Classe x Objeto

```php
<?php

class Carro {
    public string $marca;
    public string $modelo;
    public int $ano;

    public function ligar() {
        echo 'O carro foi ligado...';
    }

    public function acelerar() {
        echo 'O carro está acelerando...';
    }

    public function info() {
        echo "Carro {$this->marca} {$this->modelo}, ano {$this->ano}";
    }
}
```

```php

// Instanciando objeto da classe Carro
$carro1 = new Carro();

// Difinindo propriedades (atributos)
$carro1->marca = 'Ford';
$carro1->modelo = 'KA';
$carro1->ano = 2002;

// Chamando métodos (comportamento)
$carro1->acelerar();
$carro1->info();

```


6

Programação Orientada a Objetos

Encapsulamento

- O encapsulamento permite ocultar certos detalhes internos dos objetos e expor apenas uma interface pública para interagir com eles, garantindo a segurança e integridade dos dados

- É implementado através da visibilidade dos métodos e propriedades
  - **public** (padrão) – O método ou atributo em questão pode ser acessado por todas as outras classes e métodos, sem quaisquer restrições.
  - **protected** – Pode ser acessado apenas por métodos da própria classe e pelas classes-filhas.
  - **private** – Modificador que não permite o acesso por classes descendentes (classes-filhas), e só pode ser acessado dentro da própria classe.

7

Programação Orientada a Objetos

Herança

- A herança é um mecanismo que permite criar novas classes com base em classes existentes.
- É o compartilhamento de atributos e comportamentos entre as classes de uma mesma hierarquia (Árvore).
- A classe derivada (ou classe filha) herda as características da classe pai e pode adicionar novos atributos e métodos, além de sobrescrever ou estender os existentes.
- É implementada utilizando a palavra reservada **extends**.

8

Programação Orientada a Objetos

Polimorfismo

- O polimorfismo permite que objetos de diferentes classes sejam tratados de maneira semelhante, mesmo que possuam comportamentos diferentes.
- Isso é alcançado através do uso de **herança**, onde os métodos da classe pai, pode ser modificados nas classes-filhas
- O polimorfismo permite escrever código genérico que pode ser aplicado a vários tipos de objetos.

9

Programação Orientada a Objetos

Abstração

- Capacidade de separarmos mentalmente o sistema em módulos ou partes lógicas, para posteriormente, aplicarmos este conceito na prática ao desenvolvermos em OO
- Não podemos confundir abstração com classes abstratas.
- Abstração é o conceito de abstrairmos um problema e resolvê-lo.
- Classes abstratas são **uma das formas** de aplicarmos o conceito de abstração na prática

10

Programação Orientada a Objetos

Classes abstratas

- Classes abstratas serve de base para outras classes
- Não podem ser instanciadas, somente sua descendência
- Características de um método abstrato:
  - Ele só pode ser definido numa classe abstrata.
  - Ele deve ter apenas a sua assinatura como método, ele não pode conter nenhuma implementação na classe abstrata.
  - Obrigatoriamente deve ser implementado na classe filha que o estender.

11

Atributos e métodos estáticos

- Elementos de uma classe que pertencem à própria classe, e não a instância (objeto) dessa classe
- São definidos com a palavra-chave **static**.
- São acessados diretamente pela classe usando o operador de resolução de escopo **::** e o nome do elemento (atributo/método) estático
- Os atributos estáticos são inicializados apenas uma vez, na primeira vez que são referenciados.
- Os métodos estáticos não têm acesso aos atributos e métodos não estáticos da classe, pois não estão vinculados a uma instância específica.

12

Namespaces

- Os Namespaces do PHP fornecem uma maneira de agrupar classes, interfaces, funções e constantes relacionadas.
- Seu conceito abstrato é semelhante aos diretórios de qualquer sistema operacional.
- Um namespace é declarado usando a palavra-chave **namespace**, que deve ser a primeira instrução de um arquivo PHP.
- Um mesmo namespace pode ser usado em múltiplos arquivos, que podem definir classes ou funções. Isso implica que funções com mesmo nome também são possíveis, desde que em namespaces distintos

https://www.php.net/manual/pt_BR/language.namespaces.rationale.php

Namespaces

- Quando não fazemos uso do namespace, o PHP entende que todas as classes estão num namespace global
- Simulando um problema:

index.php > ...
```php
<?php

require_once './minhas-classes/Mensagem.php';
require_once './classes-terceiros/Mensagem.php';

echo '<h3>Estudando sobre namespace</h3>';
echo '<hr>';

$minha = new Mensagem();

$terceiros = new Mensagem();
```

minhas-classes > Mensagem.php > Mensagem
```php
<?php

class Mensagem {
    public function __construct() {
        echo '<p>Mensagem de minhas-classes</p>';
    }
}
```

classes-terceiros > Mensagem.php > Mensagem
```php
<?php

class Mensagem {
    public function __construct() {
        echo '<p>Mensagem de classes-terceiros</p>';
    }
}
```

Namespaces

- Solucionando o problema:

index.php > ...

```php
<?php

require_once './minhas-classes/Mensagem.php';
require_once './classes-terceiros/Mensagem.php';

echo '<h3>Estudando sobre namespace</h3>';
echo '<hr>';

$minha = new MinhasClasses\Mensagem();

$terceiros = new ClassesDeTerceiros\Mensagem();
```

minhas-classes > Mensagem.php > Mensagem

```php
<?php

namespace MinhasClasses;

class Mensagem {
    public function __construct() {
        echo '<p>Mensagem de minhas-classes</p>';
    }
}
```

classes-terceiros > Mensagem.php > Mensagem

```php
<?php

namespace ClassesDeTerceiros;

class Mensagem {
    public function __construct() {
        echo '<p>Mensagem de classes-terceiros</p>';
    }
}
```


15

Autoload de classes

- Função: spl_autoload_register()

index.php > ...

```php
<?php

// require_once './MinhasClasses/Mensagem.php';
// require_once './ClassesTerceiros/Mensagem.php';

spl_autoload_register(function ($className) {
    $path = "{$className}.php";
    require_once $path;
});

echo '<h3>Estudando sobre namespace</h3>';
echo '<hr>';

$minha = new MinhasClasses\Mensagem();

$terceiros = new ClassesDeTerceiros\Mensagem();
```


16

Atividade

Refatorar Hangman (Jogo da Forca)

- Converter o jogo para o paradigma POO