Fundamentos da POO em PHP
Definição de Classes e Objetos

UNIVERSIDADE DO OESTE DE SANTA CATARINA - UNOESC

Ciência da Computação - Programação II

Prof Leandro Otavio Cordova Vieira

leandro.vieira@unoesc.edu.br | (49) 98825.4894 | @leo.vieira_tm

Objetivos da Aula

1

Entender o conceito de classes e objetos

Compreender a base teórica que fundamenta a Programação Orientada a Objetos e como ela se aplica em PHP.

2

Aprender a sintaxe básica de criação de classes em PHP

Familiarizar-se com a estrutura e sintaxe para definição de classes na linguagem PHP.

3

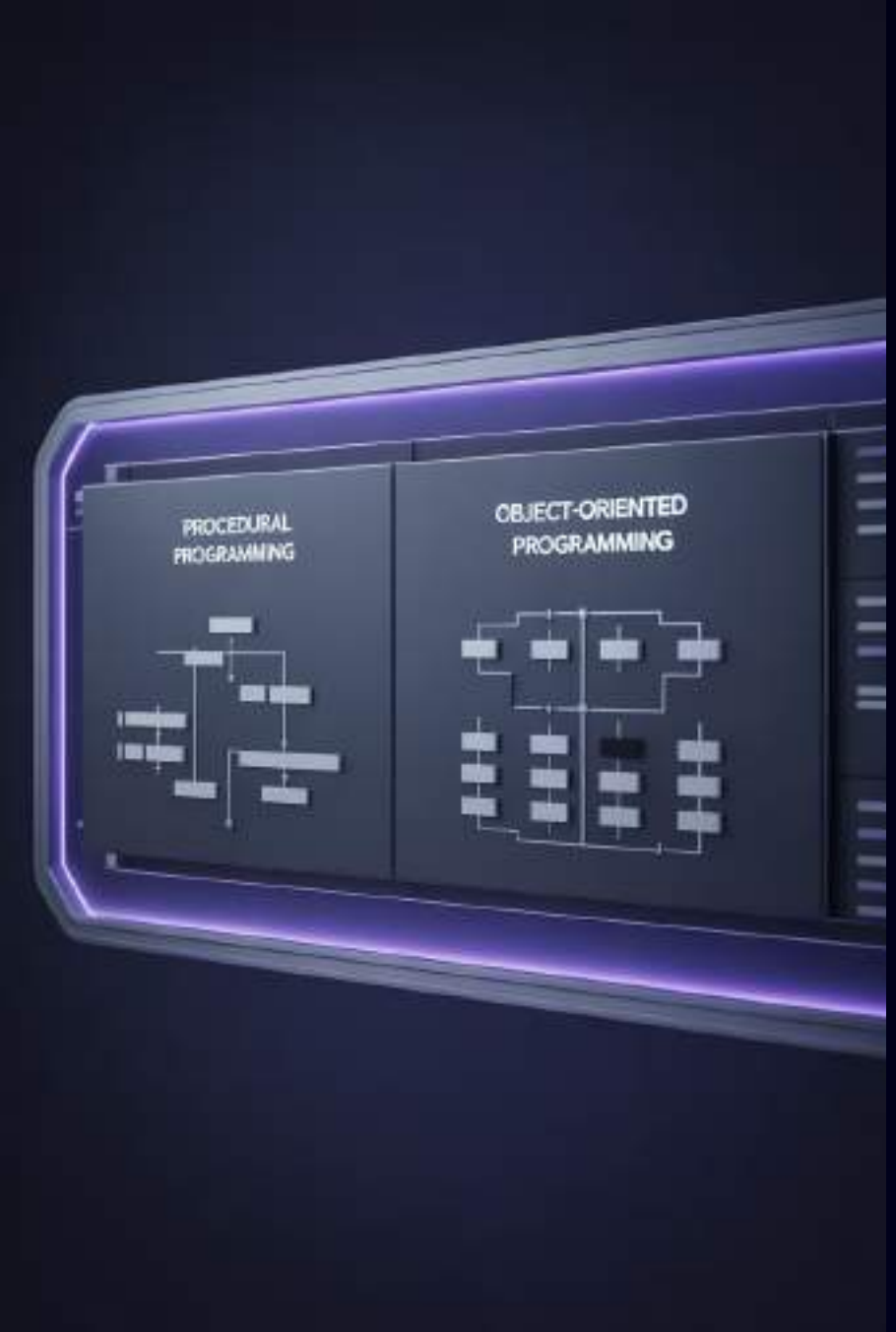
Criar e instanciar objetos

Entender como transformar definições de classes em objetos concretos que podem ser manipulados no código.

4

Aplicar conceitos em exemplos práticos

Consolidar o aprendizado através de casos reais e exercícios práticos de implementação.



Revisão Rápida

O que é POO?

Paradigma de programação baseado no conceito de "objetos", que podem conter dados na forma de atributos e código na forma de métodos.

Vantagens da POO

- Reutilização de código
- Modularidade
- Manutenção facilitada
- Melhor organização

Diferença entre Procedural e Orientado a Objetos

No procedural, foco em funções que manipulam dados. Na POO, foco em objetos que encapsulam dados e comportamentos.

Definição:

Classe

Molde ou modelo que descreve as características e comportamentos de um objeto.

Uma classe define a estrutura e o comportamento que os objetos derivados dela terão, funcionando como um "projeto" para criação de instâncias.

Definição:

Objeto

Instância concreta de uma classe, que possui atributos e métodos definidos no molde.

Quando criamos um objeto, estamos criando uma "cópia funcional" baseada na definição da classe, com seus próprios valores para os atributos.

Sintaxe Básica de Classe em PHP

```php
class NomeDaClasse {
    // Atributos (propriedades)
    public $atributo1;
    public $atributo2;

    // Métodos (funções)
    public function metodo1() {
        // Código do método
    }

    public function metodo2() {
        // Código do método
    }
}
```

A palavra-chave `class` inicia a definição, seguida pelo nome da classe. Entre as chaves `{}` definimos os atributos e métodos que compõem a classe.

Criando uma Classe Pessoa

```
class Pessoa {
    // Atributos
    public $nome;
    public $idade;

    // Por enquanto sem métodos
}
```

Esta classe `Pessoa` possui dois atributos públicos:

- `$nome`: armazena o nome da pessoa
- `$idade`: armazena a idade da pessoa

O modificador de acesso `public` significa que estes atributos podem ser acessados diretamente de fora da classe.

Veremos outros modificadores de acesso em aulas futuras.

Instanciando um Objeto

```
// Criando um objeto da classe Pessoa
$p = new Pessoa();

// Definindo valores para os atributos
$p->nome = "Maria";
$p->idade = 25;

// Exibindo os valores
echo "Nome: " . $p->nome . "\n";
echo "Idade: " . $p->idade . " anos";
```

Usamos o operador `new` para criar uma instância da classe.

Depois, utilizamos o operador `->` (seta) para acessar e atribuir valores aos atributos do objeto.

Acessando Atributos e Métodos

Sintaxe de Acesso

Usamos o operador `->` (seta) para acessar atributos e métodos de um objeto em PHP.

```
// Acesso a atributos
$objeto->atributo = valor;
$valor = $objeto->atributo;

// Chamada de métodos
$objeto->metodo();
$resultado = $objeto->metodoComRetorno();
```

Diferença em Relação a Outras Linguagens

Em PHP, usamos `->` em vez do ponto `.` (usado em linguagens como Java e C#).

Também não usamos `$this.` (como em JavaScript), mas sim `$this->` dentro da classe.

O que NÃO fazer

```
// ERRADO:
$objeto.atributo = valor; // Errado!
$objeto->$atributo; // Errado! (exceto em casos específicos)
$this->variavel_local; // Errado! (variáveis locais não usam $this)
```

Exemplo com Método

```
class Pessoa {
    // Atributos
    public $nome;
    public $idade;

    // Método
    function apresentar() {
        echo "Olá, meu nome é $this->nome.";
    }
}
```

O método `apresentar()` exibe uma mensagem usando o valor do atributo `$nome`.

A palavra-chave `$this` refere-se ao próprio objeto e é usada para acessar seus atributos e métodos de dentro da classe.

É como se fosse uma "auto-referência" ao objeto atual.

Instanciando e Usando o Método

Depois de instanciar o objeto e definir seus atributos, podemos chamar seus métodos usando o operador `->`.

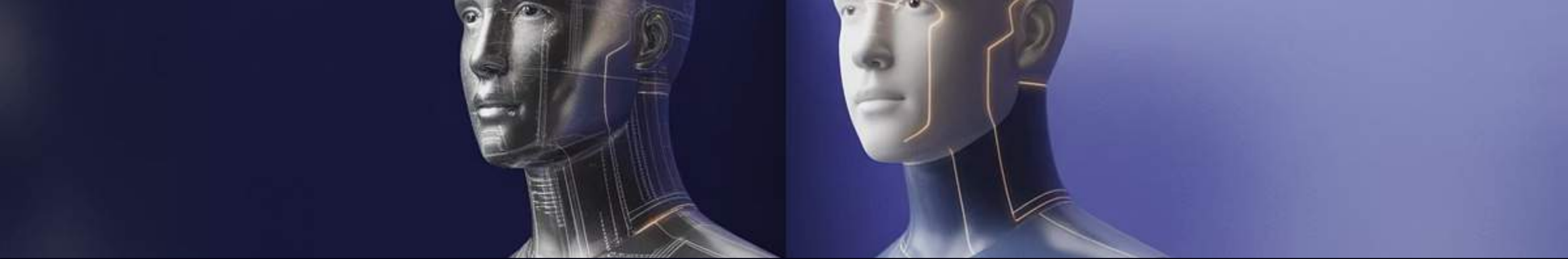
```
// Criando um objeto da classe Pessoa
$p = new Pessoa();

// Definindo o atributo nome
$p->nome = "João";

// Chamando o método apresentar
$p->apresentar();

// Saída: "Olá, meu nome é João."
```

O método `apresentar()` usa o valor do atributo `$nome` que definimos para construir a mensagem.



Exemplo Prático

Definição da Classe

```
class Pessoa {
    public $nome;
    public $idade;

    function apresentar() {
        echo "Olá, sou $this->nome e tenho
$this->idade anos.";
    }
}
```

Criação de Objetos

```
// Primeiro objeto
$pessoa1 = new Pessoa();
$pessoa1->nome = "Maria";
$pessoa1->idade = 25;

// Segundo objeto
$pessoa2 = new Pessoa();
$pessoa2->nome = "João";
$pessoa2->idade = 30;
```

Resultado

```
$pessoa1->apresentar();
// Saída: "Olá, sou Maria e tenho 25
anos."

$pessoa2->apresentar();
// Saída: "Olá, sou João e tenho 30
anos."
```

Observe que ambos os objetos compartilham a mesma estrutura (definida pela classe), mas possuem valores diferentes para seus atributos.

Exemplo Classe Produto

```
class Produto {
    // Atributos
    public $nome;
    public $preco;
    public $quantidade;

    // Método que calcula o valor total
    function valorTotal() {
        return $this->preco * $this->quantidade;
    }
}
```

Esta classe representa um produto com nome, preço e quantidade.

O método `valorTotal()` calcula o valor total multiplicando o preço pela quantidade.

Note que usamos `return` para devolver o resultado do cálculo, em vez de apenas exibi-lo.

Instanciando Produto

A instanciação segue o mesmo padrão que vimos anteriormente, mas agora com atributos diferentes.

```
// Criando um objeto da classe Produto
$prod = new Produto();

// Definindo valores para os atributos
$prod->nome = "Caneta";
$prod->preco = 2.5;
$prod->quantidade = 10;

// Calculando e exibindo o valor total
echo "Produto: " . $prod->nome . "\n";
echo "Valor total: R$ " . $prod->valorTotal();
// Saída: "Valor total: R$ 25"
```

O método `valorTotal()` retorna o resultado do cálculo, que então pode ser usado em outras operações ou exibido.

Discussão Orientada

Semelhanças entre Pessoa e Produto

- Ambas são classes com atributos públicos
- Ambas possuem métodos que utilizam seus próprios atributos
- Seguem o mesmo padrão de instanciação com `new`
- Representam entidades do mundo real

Diferenças nos Métodos

- `apresentar()` da classe Pessoa usa `echo` para exibir uma mensagem
- `valorTotal()` da classe Produto usa `return` para devolver um valor calculado
- Diferentes propósitos: apresentação vs. cálculo

Para Refletir

Quais outras entidades poderiam ser modeladas como classes? Que atributos e métodos elas teriam?

Como você decidiria entre usar `echo` ou `return` em um método?

Boas Práticas Iniciais

Nomes de classes com letra maiúscula

Sempre inicie o nome da classe com letra maiúscula (ex: `Pessoa`, `Produto`, `ContaBancaria`). Esta é uma convenção amplamente adotada que ajuda a diferenciar classes de variáveis e funções.

Métodos e atributos com nomes claros

Use nomes descritivos que expliquem claramente o propósito (ex: `calcularJuros()` em vez de `calc()`). Nomes bem escolhidos tornam o código mais legível e auto-explicativo.

Código organizado

Mantenha uma indentação consistente, agrupe atributos e métodos logicamente, e adicione comentários quando necessário para explicar a lógica complexa ou o propósito de uma classe ou método.

Um arquivo por classe

À medida que seu projeto cresce, é recomendável manter cada classe em seu próprio arquivo, com o nome do arquivo igual ao nome da classe (ex: `Pessoa.php`).

Erros Comuns

Esquecer o `new` ao instanciar

```
// ERRADO
$pessoa = Pessoa();  // Tenta chamar função

// CORRETO
$pessoa = new Pessoa();  // Instancia objeto
```

Confundir atributos públicos e privados

```
// Tentando acessar atributo privado
$pessoa->nome_privado = "João";  // Erro!
```

Atributos privados só podem ser acessados dentro da própria classe.

Erros de sintaxe com `->`

```
// ERRADO
$pessoa.nome = "Maria"; // Usa ponto em vez de ->
$pessoa->$nome = "Maria"; // $ extra antes de nome
```

Atividade 1

Crie uma classe Carro com atributos marca, modelo e ano. Adicione um método que exiba as informações.

```
class Carro {
    // Seus atributos aqui
    // ...

    // Seu método aqui
    // ...
}

// Código para testar sua classe
$meuCarro = new Carro();
$meuCarro->marca = "Volkswagen";
$meuCarro->modelo = "Gol";
$meuCarro->ano = 2020;
$meuCarro->exibirInformacoes();  // Deve mostrar os dados do carro
```

Implemente a classe `Carro` com os atributos solicitados e um método chamado `exibirInformacoes()` que mostre todos os dados do carro de forma organizada.

Atividade 2

Crie uma classe Aluno com nome e média. Crie método que retorne se o aluno está aprovado (média >= 7).

```
class Aluno {
    // Seus atributos aqui
    // ...

    // Seu método aqui
    // ...
}

// Código para testar sua classe
$aluno1 = new Aluno();
$aluno1->nome = "Ana";
$aluno1->media = 8.5;

$aluno2 = new Aluno();
$aluno2->nome = "Pedro";
$aluno2->media = 6.0;

echo $aluno1->nome . ": " .
     ($aluno1->verificarAprovacao() ?
      "Aprovado" : "Reprovado");

echo $aluno2->nome . ": " .
     ($aluno2->verificarAprovacao() ?
      "Aprovado" : "Reprovado");
```

Dicas:

- Use um método chamado `verificarAprovacao()` que retorne um valor booleano (`true` ou `false`)
- A aprovação ocorre quando a média é maior ou igual a 7.0
- Teste com diferentes valores de média para confirmar o funcionamento

Resumo e Próximos Passos

1 O que Aprendemos

- Definição de classes como moldes para objetos
- Criação de atributos e métodos
- Instanciação de objetos com `new`
- Acesso a atributos e métodos com `->`
- Uso de `$this` dentro da classe

2 Próxima Aula

Métodos e Atributos em profundidade:

- Modificadores de acesso (public, private, protected)
- Métodos construtores e destrutores
- Métodos getters e setters
- Atributos e métodos estáticos

3 Preparação

Para a próxima aula, recomendo:

- Praticar os exercícios desta aula
- Tentar criar outras classes simples
- Revisar o código dos exemplos