**Aula 02 – Prog II**

Simula e Smaltalk foram as primeiras linguagens POO, na década de 60-70;

PHP - Hypertext Preprocessor;

PHP4 – 1994, 1998...;

Classe: Molde; Objeto: Instância concreta da classe com atributos e métodos;

O Java incialmente surgiu como uma linguagem mais de Business;

Em 1995 foi lançado o PHP juntamente do Java, que foram as linguagens que ajudaram a consolidar o paradigma de Orientação a Objetos no desenvolvimento de softwares.

Definição de POO: Paradigma que organiza o código em torno de **objetos**.

Objetos são entidades que combinam dados e comportamentos, sendo os mesmos consecutivamente Atributos e Métodos.

**Atributo/dado** podemos assimilar a **características**, são o que definem o objeto, por exemplo, o objeto carro tem portas, quatro roda, espelho.

**Métodos/comportamentos** podemos relacionar a **ações**, por exemplo, o carro se move para frente, para trás, para os lados, acelera, freia...

Podemos atribuir os mesmos métodos a diferentes objetos, pois uma moto e um carro podem acelerar tanto um quanto o outro.

**Programação Procedural:**

* Foco em funções e procedimentos;
* Dados e comportamentos separados;
* Variáveis globais e frequentes;
* Código sequencial linear;
* Um procedimento leva a outro;

**Programação Orientada a Objetos:**

* Foco em objetos que encapsulam dados e comportamentos;
* Métodos associados a objetos específicos;
* Atributos encapsulados em classes;
* Estrutura modular e hierárquica.

**Vantagens de usar POO:**

**- Organização**

Estrutura o código em unidades lógicas e coesas, facilitando a compreensão e manutenção do sistema.

**-Reutilização e manutenção**

Permite aproveitar componentes em diferentes partes do sistema ou em outros projetos, reduzindo duplicação, juntamente da facilidade de realizar manutenções.

**- Escalabilidade**

Proporciona base sólida para crescimento do sistema, com adição de novas funcionalidades de forma modular.

Útil para projetos que podem escalonar. Partes do código são independentes, assegurando a estrutura do código para manutenções, pois ele não depende totalmente das linhas anteriores para que funcione, por exemplo.

**Pilares da POO**

**Encapsulamento**

Oculta detalhes internos e expõe apenas o necessário

Protege os dados e farante que sejam acessados de forma controlada;

**Herança**

Permite que classes derivem de outras classes

Promove reutilização de código e estabelece hierarquias;

**Polimorfismo**

Permite que objetos de diferentes classes respondam à mesma mensagem

Possibilita tratar objetos de tipos diferentes de maneira uniforme.

Objeto -> Instância de uma classe

Representa uma entidade do mundo real com características e comportamentos específicos.

Para pensar em uma classe/ Objeto, sempre siga esses 3 passos:

**Instanciação**

$p = new Produto();

Criamos uma nova instância da classe produto e a armazenamos na variável $p.

**Definição de Atributos**

$p->nome = “Caneta”;

$p->preco = 3.5;

Atribuímos valores específicos as características do objeto.

**Chamada de Método**

$p->exibir()

Executamos o método que mostrará:  
“Caneta: R$3.5” na tela.

**Converta o código Procedural a seguir em uma versão orientada a objetos.**

**Instruções:**

Como ficaria este código usando uma classe Usuario?

* Defina os atributos apropriados
* Crie os métodos necessários
* Inclua um construtor
* Demonstre a instanciação e uso

Código Procedural

//Dados do usuário

$nome = “João”;

$email = [joao@exemplo.com](mailto:joao@exemplo.com);

$idade = 25;

//Função para exibir dados

function exibirDados($n, $e, $i){

echo “Nome: $n, Email: $e, idade: $i”;

}

//Função para verificar maioridade

Function ehMaiorDeIdade($i){

Return $i >= 19;

}

**Versão Orientada a Objetos**

public class Usuario{

$nome; $email; $idade

};

$usuario = new Usuario()  
{

$nome = Shewbaitola;

$email = [shewbai@tola.org.br](mailto:shewbai@tola.org.br);

$idade = 34.

}

Discussão Orientada

Semelhança 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 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 elas teriam:  
  
Como você decidiria entre usar echo ou return em um método?

**Boas Práticas Inciais**

Nomes de classes com letra maiúscula

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

**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.

**Métodos e atributos com nomes claros**

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

**Um arquivo por classe**

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

Em PHP, o “.” É usado para concatenar algo, se quiser acessar algo, use a flechinha “->”.

Atividade 1

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