**Curso: Programação Cliente Servidor**

1. Revisão OO (Orientação a Objeto)

É um paradigma de análise, projeto e programação de software baseada em composição e interação entre unidades de software chamadas Objetos. É uma forma de ver um problema, modelar uma solução e com isso desenvolver um sistema.

1. Abstração

É a capacidade de do ser humano concentrar-se apenas nos aspectos essenciais de um problema. Ignorando algumas outras que não são importantes.

No contexto da orientação a objetos, uma classe é uma abstração de conceitos do mundo real, sem que precisamos fazer uso de todas as características do conteúdo real, apenas usamos como modelo, as características que são fundamentais para o nosso sistema.

1. Classe

É uma representação genérica de um conjunto de indivíduo idênticos.

Teoricamente, uma classe é um modelo para a criação de **objetos**, então todos os objetos serão criados baseados a partir dessa classe, que possuirão os mesmos comportamentos(**classes**) e armazenarão as informações (**atributos**).

1. Atributos

Digamos que é quando se refere ao objeto, como características. Por exemplo, o **Aluno** Pedro cursa Análise e Desenvolvimento de Sistemas na Universidade Sr. Madruga e tem 20 anos de idade.

Nesse casso, a classe poderia se chamar **ALUNO**, que poderia ser formada pelos atributos **NOME DO ALUNO**, **CURSO**, **UNIVERSIDADE** e **IDADE**.

Poderíamos instanciar a classe aluno, fazendo uso de seus atributos, quantas vezes precisarmos para cadastrarmos novos aluno.

Exemplo:

|  |
| --- |
| **ALUNO** |
| **NOME**: Pedro **CURSO**: Análise e Desenvolvimento de Sistemas **IDADE**: 20 **Universidade**: Sr. Madruga |

Para os **atributos**, atribuímos **valores**, os atributos são abstratos, ou seja, a característica que abstraímos para formar um **objeto** **ALUNO**, já os valores podemos identifica-los (Pedro, Análise e Des. De Sistamas, 20, Sr. Madruga).

Obs.: O estado dos atributos, podem variar.

1. Métodos

São as **ações** que um Objeto pode fazer.

Por exemplo, a classe ALUNO pode fazer:

Estudar  
Ler  
Acordar  
Andar

Podemos atribuir PARÂMETROS aos métodos, que são as informações que ele precisa para realizar uma ação e retornar valores, que é o produto da ação realizada.

1. Encapsulamento

Os **dados ficam encapsulados** **no interior dos métodos** (ações) de um **objeto**, desta forma não saberemos o que há dentro dos métodos quando o estivermos manipulando. Impede acesso direto ao **Objeto.**

Exemplo:

Classe: TV  
Atributos: Cor, Polegadas, Modelo, Marca  
Métodos: Liga, Desliga, AumentaVolume, DiminuiVolume  
  
Cada método, possui a instrução que não poderá ser acessada pelo usuário enquanto estiver manipulando, mas podemos ver o que tem nos atributos como a cor da TV, a Polegada, o Modelo e a Marca, e podem ter seus valores alterados durante a manipulação.

1. Sobrecarga

Técnica que permite usar **vários métodos** que **tenham o mesmo identificador, variando somente a sua assinatura**.

1. Herança

É o comportamento pelo qual uma classe-filha ou subclasse, estende outra classe (classe-pai ou superclasse) herdando os seus comportamentos (métodos) e seus estados (atributos).

Exemplo:

A classe **Felino**, poderia estender uma classe **Animal** e consequentemente a classe animal herdaria seus atributos e estados.

Exemplo

CLASSE-PAI: **FELINO**ATRIBUTOS**: TIPO(char), IDADE(inteiro), NOME(char), TAMANHO(char), ORIGEM (char)**MÉTODOS: **MIA**, **ANDA**, **CAÇA**

CLASSE-FILHA: **ANINAL** (INSTANCIA A CLASSE-PAI FELINO)  
HERDA ATRIBUTO E RECEBE VALORES: **TIPO=ONÇA, IDADE=8, NOME=MINGAU, TAMANHO=4M, ORIGEM=ÁFRICA**HERDA AÇÕES: **MIA, ANDA, CAÇA**

A conclusão seria que o **Animal** que **Mia**, **Anda** e **Caça**, é um **Felino**.

1. Sobrescrita

Acontece quando precisamos reescrever um método herdado pela classe-pai. Nesse caso, o novo método deve ter a mesma assinatura da classe pai.

- Possuir o mesmo nome: **FELINO**  
- Retornar o mesmo tipo de dados: **Nome Atributo**(**char**), **Nome Atributo**(**inteiro**), **Nome Atributo**(**char**), **Nome Atributo**(**char**), **Nome Atributo**(**char**).  
- Recebe a mesma lista de parâmetros: **MÉTODOS(?)**.

1. Polimorfismo

É a capacidade de Objetos instanciados de classes diferentes usar um mesmo método, mas cada qual da sua maneira.

1. Classes e Métodos Abstratos

* **Classe Abstrata**: É toda classe que **não possui a capacidade de gerar objetos**, servindo apenas como **modelo para uma classe concreta**.
* **Métodos Abstratos**: Devem ser **implementados na classe concreta** que **estenda a classe na qual foi declarado na qual pode ter estendida**. Um método abstrato, somente podem ser definidos em classes abstratas.

1. Classe Final

O **modificador final**, ***pode ser estendido como o oposto do******modificador abstrato***.

Uma **classe final** é uma ***classe que não pode ser estendida***, ***podendo apenas ser instanciada***.

1. Método Final

Seguindo a mesma lógica, o **método final** é um ***método concreto que não podem ser sobrescritos***.

1. Atributos Finais

Nesse caso, eles se **comportarão como constantes**, ***não sendo permitida a ação dos valores definidos na sua inicialização***.

1. Interfaces

**Podem ser estendidas como classes totalmente abstratas** sendo assim, ***não podem possuir métodos concretos*** assim como também ***não podem possuir um construtor***.

Caso tenha algum **atributo**, esses **devem ser finais**.

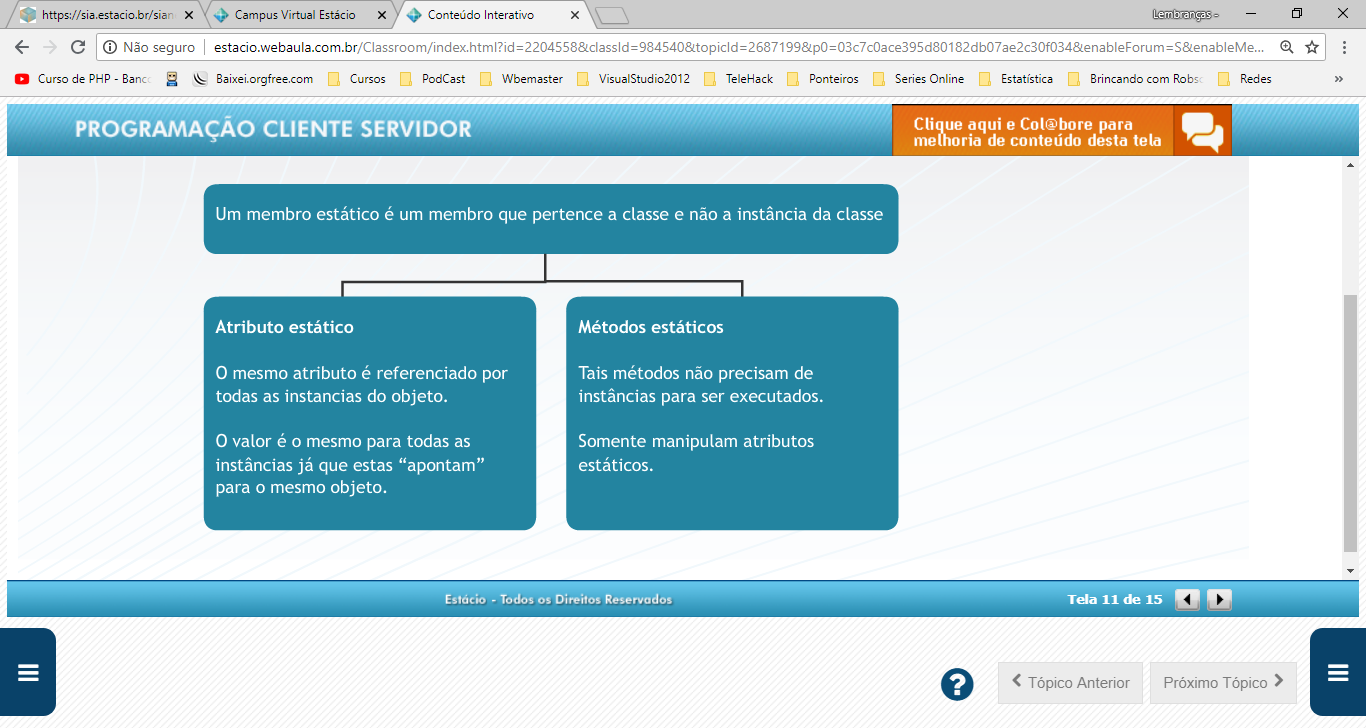
São utilizadas para ***definir padrões para especificação do comportamento de classes***.

1. Visibilidade de Membros de Classes (atributos)

* **Pública**: Podem ser *acessadas por qualquer objeto*.
* **Privada**: Podem ser *acessadas somente dentro da classe onde foi declarada*.
* **Protegida**: Podem ser *acessadas por qualquer objeto dentro do pacote no qual foi declarada* ou *fora desse pacote pelas classes que estendam a essa classe*.
* **Padrão**(**pacote**): Podem ser *acessadas somente por objetos que estejam no mesmo pacote em que a classe foi declarada*.

1. Membros de classe estáticos

Além dos *modificadores de visibilidade*, os *membros de uma classe* podem ainda ser *definidos como estáticos*.



1. A Linguagem JAVA