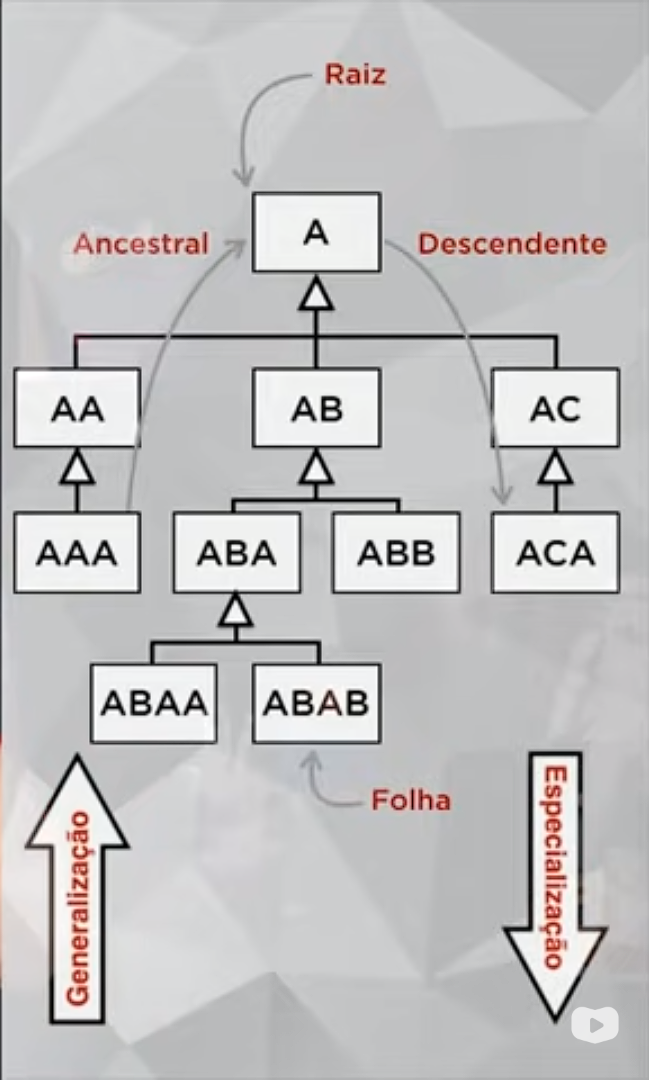
POO aula 11: Herança(parte 2):

Como funciona a herança na teoria?

Os conceitos base da árvore hierárquica estão na imagem, onde há uma explicação para cada um.

Primeiro: **Nem toda super classe ou subclasse terão esse cargo para sempre**, ou seja, **pode existir classes que são tanto super classes ou subclasses**, um exemplo disso é a classe AB(que é subclasse de A, porém superclasse de ABA e ABB).

**Raiz:** Toda superclasse que não herda nenhuma classe, ou seja a que dá mas não recebe, é considerada a classe raiz da árvore hierárquica.

**Folha:** Toda subclasse que não tem progenitores, ou seja a que recebe, mas não dá, é considerada a classe folha da árvore hierárquica.

**Descendente:** Toda subclasse que tem uma classe avó, por exemplo, ABA é descendente de A, ou ABAA é descendente de AB e A.

**Ancestral:** Toda superclasse que tem uma classe neto, por exemplo, A é ancestral de ABA ou A/AB são ancestrais de ABAA, sendo o inverso de Descendente.

Quando se lê ou processa a árvore da raiz até a folha(de cima para baixo), este ato se chama **Especialização**.

Quando se lê ou processa a árvore folha até a raiz(de baixo para cima), este ato se chama **Generalização**.

Tipos de herança:

* Herança de Implementação(Herança pobre): É quando se implementa uma classe abstrata, sendo a subclasse tendo a função de herdar os atributos e métodos da superclasse, porém não criar nenhum atributo ou método próprio.

Exemplo:

Classe Pesssoa tem 3 atributos(nome, idade e sexo) e 1 método(fazerAniv())

Cria-se uma subclasse para a Classe Pessoa, chamada visitante(indo com base no exemplo da aula 10), porém a classe visitante não cria nenhum atributo e métodos próprios só herda os da classe Pessoa.

isso é Herança por Implementação

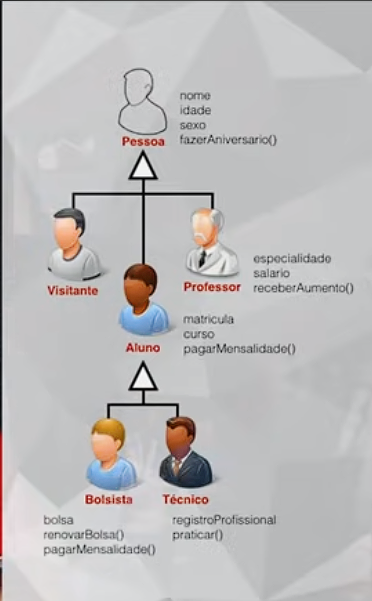
* Herança para Diferença(Mais completa): É o tipo de herança utilizado nas outras aulas de herança, onde a subclasse herda os atributos e métodos da superclasse, porém também cria seus próprios atributos e métodos.

Exemplo: Ao pegar a Classe Pessoa do exemplo anterior, vamos estender essa classe para uma nova subclasse, no caso aluno.

Class Aluno estende Pessoa, porém cria seus atributos e métodos próprio(Novos atributos, curso e matrícula) e método (cancelarMatricu())

Sobrescrição de Métodos:

Exemplo da Aula:



Um ponto interessante a se ressaltar é que o aluno é uma generalização de bolsista e técnico, enquanto Bolsista e Técnico são uma especialização de aluno.

**Conceitos de Abstrato e Final: tanto classes e métodos podem utilizar este conceito.**

**Classe Abstrata:** Não pode ser instanciada. Só pode servir como progenitora.

**Método Abstrato:** Declarado, mas não implementado na progenitora.

**Classe Final:** Não pode ser herdada(não pode ter filhos[Classe Estéril]) por outra classe. Obrigatoriamente folha.

**Método Final:** Não pode ser sobrescrito pelas suas subclasses. Obrigatoriamente Herdado.

Algoritmo do Exemplo da Aula:

Classe Abstrata Pessoa

privado nome: Caractere

privado idade: Inteiro

privado sexo: Caractere

publico método final fazerAniv()

FimMétodo

FimClasse

Classe Visitante estende Pessoa //Herança de Implementação

FimClasse

Classe Aluno estende Pessoa //Herança por Diferença

privado matricula: inteiro

privado curso: caractere

publico método PagarMensalidade()

FimMétodo

FimClasse

Classe Bolsista estende Aluno //Especialização de Aluno

privado bolsa: Inteiro

publico metodo RenovarBolsa()

FimMétodo

@Sobrepor

publico metodo PagarMensalidade()// Esse se encaixando no conceito do 3 pilar de POO: Polimofismo

FimMétodo

FimClasse

Polimorfismo conceito básico: Fazer a mesma coisa, de maneiras diferentes.