Outros conceitos de orientação a objetos

Pilares da Orientação a Objeto?

O paradigma de programação Orientado a Objetos define 4 pilares considerados na definição de sua estrutura. São eles: Abstração, Encapsulamento, Herança e Polimorfismo.

Abstração: quando se fala em abstração na Orientação a Objetos busca-se um olhar isolado ao objeto de uma forma geral, definindo o que deve ser feito e não o como deve se fazer. A abstração é aplicada com as classes abstratas e interfaces, que definem um contrato com outras classes relacionadas.

Encapsulamento: os dados na Orientação a Objetos são protegidos através do acesso somente pelos seus atributos (identificadores) e métodos (comportamentos). À essa proteção denominamos Encapsulamento. Analogicamente podemos pensar em um ovo, onde a gema seria os dados e a clara os métodos e atributos que os cerca.

Em uma classe na linguagem Java por exemplo, usamos os métodos Getters and Setters que tem o objetivo de controlar o acesso a cada um dos atributos e operações de uma certa classe. Ou seja, tem a função de disponibilizar externamente os métodos que alteram e acessam (lê) os atributos de uma classe.

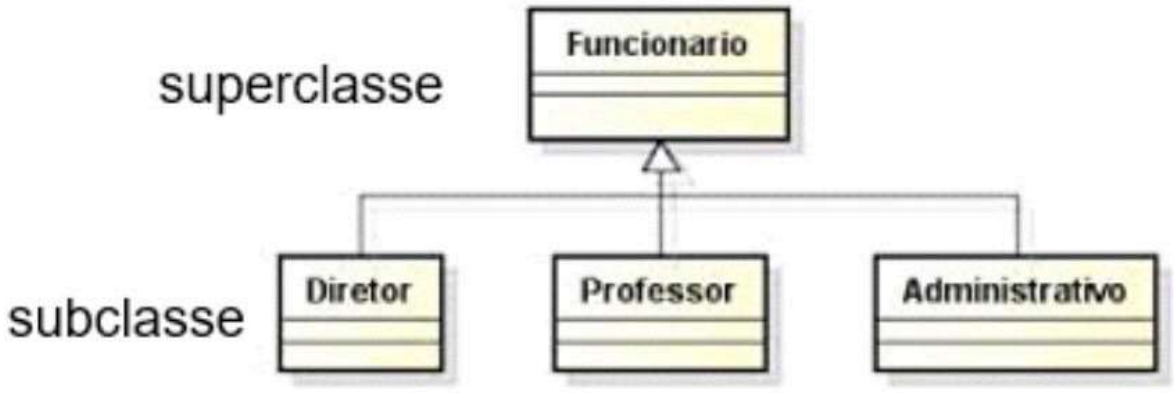
Métodos getters	Métodos setters

public String getNome() {
return nome;
}
 	
public double getSalario() {
return salario;
}
	public void setNome(String nome) {
	this.nome = nome;
	}
	
	public void setSalario(double salario) {
	this.salario = salario;
	}

Fonte: Autoral

Herança - Mecanismo que permite definir uma nova classe (subclasse) a partir de uma classe que já existe (superclasse). Uma classe "filha" herda comportamentos e atributos da classe "pai". É identificado como relacionamento de É-UM.

Exemplo: Diretor é um Funcionário; Professor é um funcionário; Administrativo é um funcionário.



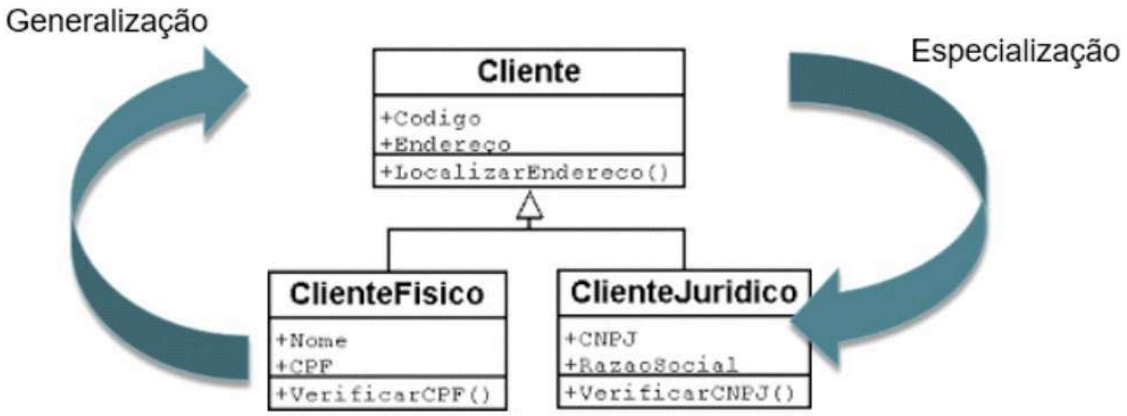
Fonte: Autoral

A herança é implementada a partir da representação da generalização e especificação de um objeto:

A generalização representa as superclasses e nela são definidos os atributos e comportamentos que estarão sendo definidos a todos os objetos das clas (especializações) ligadas a ela. As classes que representam a generalização (superclasse) são chamadas também de classes mãe.

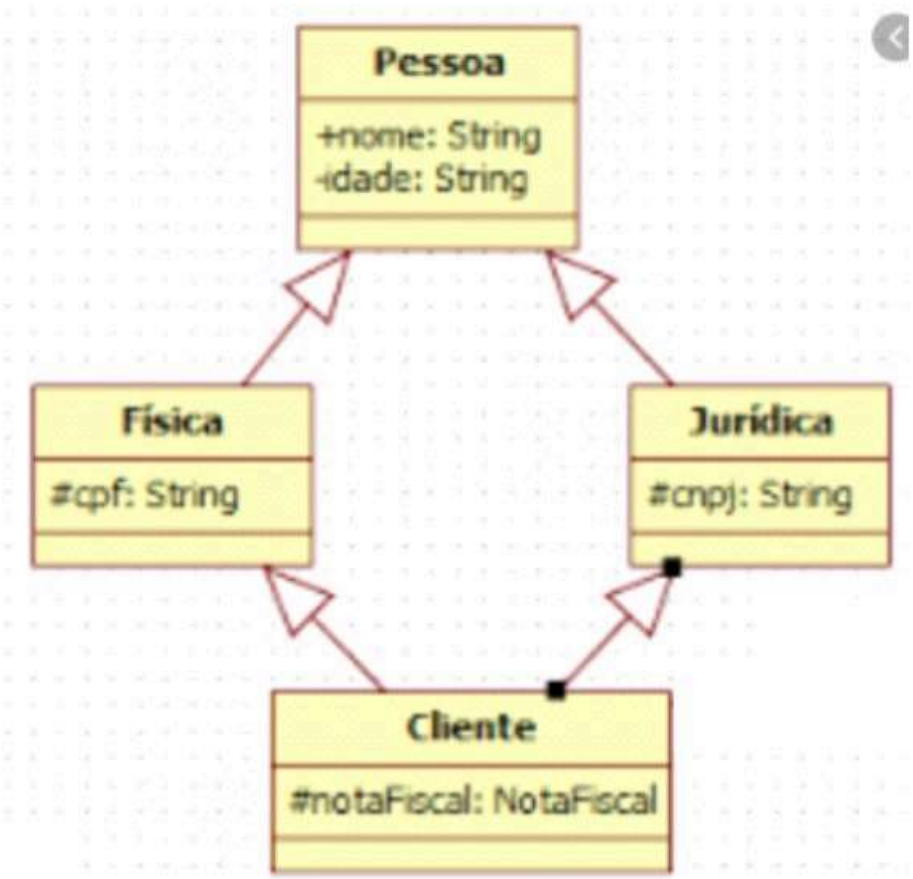
A especialização representa as subclasses e nela são definidos os atributos e comportamentos que estarão representando somente seus próprios objetos. As classes que representam a especialização (subclasse) são chamadas também de classes filhas.

Como exemplo, observando a figura a seguir, CLIENTE possui os atributos código e endereço e o método localizarEndereço(), que serão também atributos e métodos de ClienteFísico e ClienteJurídico. Desta forma, ClienteFisico terá como atributo: codigo, endereço, nome e cpf e, métodos: LocalizaEndereço() e VerificarCPF(), assim como ClienteJuridico terá como atributo: codigo, endereço, cnpj e RazaoSocial e, métodos: LocalizaEndereço() e VerificarCNPJ(),



Fonte: Autoral

Herança múltipla - Cada classe é declarada como uma subclasse de uma ou mais superclasses. Quando existe mais de uma superclasse, a relação de herança é denominada herança múltipla.



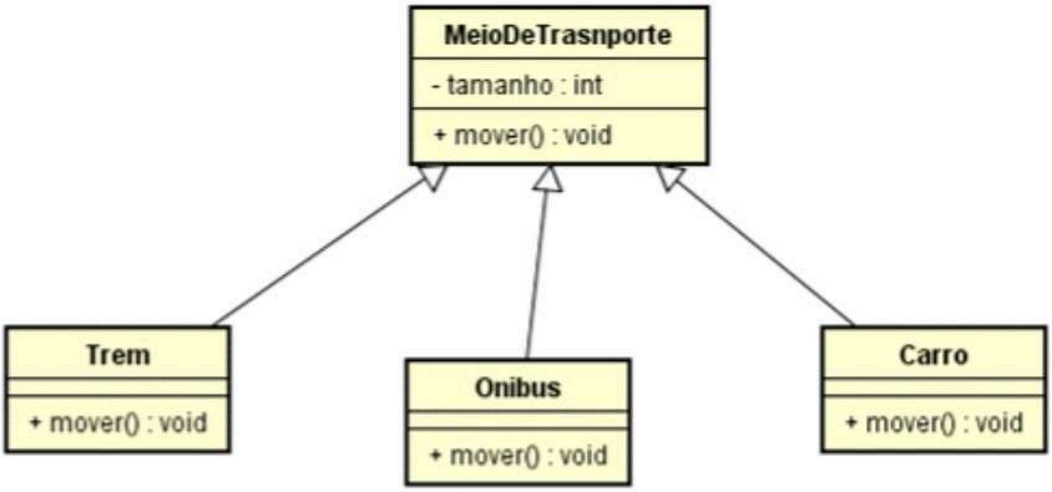
Fonte: Autoral

Pilares da Orientação a Objeto - Polimorfismo

Polimorfismo: pela própria formação da palavra podemos entender de muitas formas! Na Orientação a Objetos este pilar é aplicado para prover ao objeto comportamentos diferentes dentre os vários tipos de um mesmo grupo, representados na generalização / especialização a partir do mecanismo de herança.

Perceba que usamos o conceito de herança para termos o efeito do polimorfismo. Observe a figura a seguir, que temos o meio de transporte como classe Pai e a classe Trem, Ônibus, Carro como filha. As três classes são meios de transporte, só que se movem de formas diferentes (método mover() está presente na classe pai e filhas). Isso caracteriza o polimorfismo.

Quando o método mover() for executado para objetos das classes filhas, estará utilizando a definição dentro da própria classe.

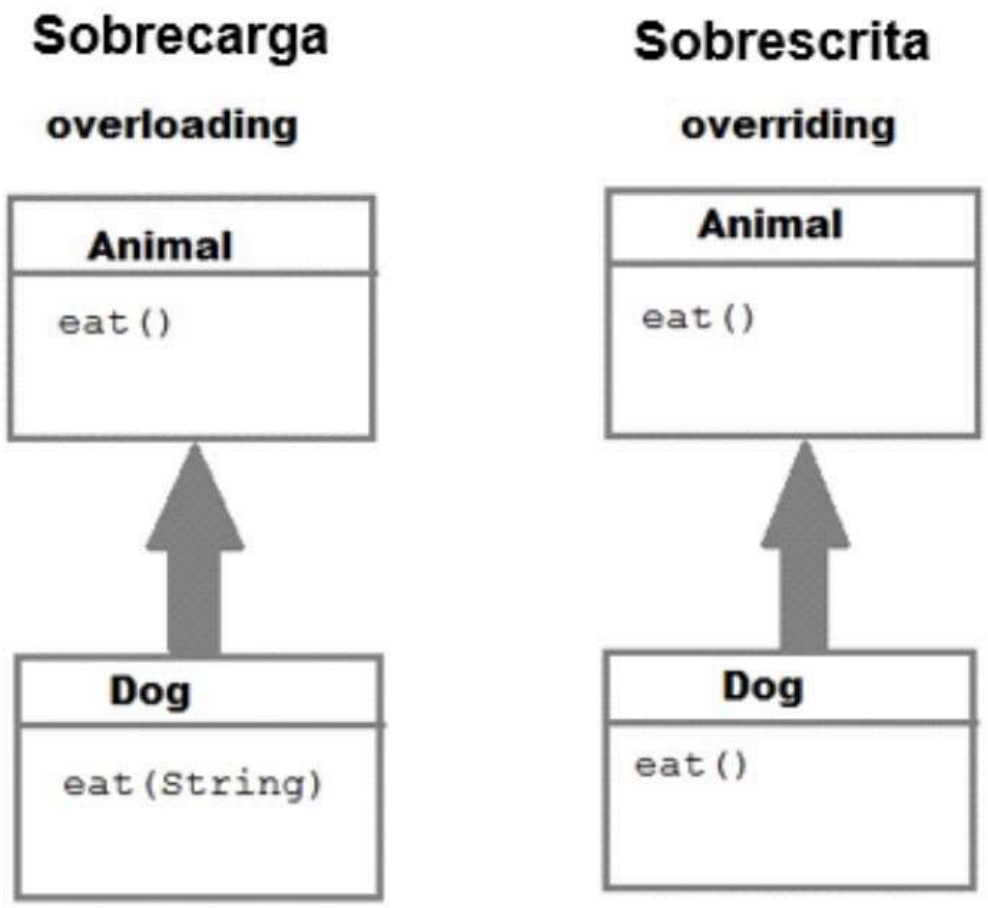


Fonte: Autoral

Existem 2 tipos de polimorfismo:

- **Dinâmico –** identificado com sobrescrita (override, binding dinâmico). Com a sobrescrita, conseguimos especializar os métodos herdados das superclasses, alterando (redefinindo) o seu comportamento nas subclasses por um mais específico. Mesmo nome de método e argumentos iguais. Um exemplo de sobrescrita é o método calculoSalario() nas classes Coordenador e Professor. Ambas as classes sobrescrevem o método calculoSalario(). Uma outra característica da sobrescrita é que a escolha do método a executar é realizada em tempo de execução para o objeto instanciado.

- **Estático** identificado como sobrecarga (overloading). Está diretamente relacionada com o polimorfismo. Mesmo nome de método e argumentos diferentes (tipo de retorno ou número de parâmetros). A ideia é construir um novo comportamento e não redefinir o comportamento (Sobrescrita). Neste caso, a escolha do método a ser executado ocorre em tempo de execução.



Fonte: Autoral

Atividade extra

Para saber mais sobre Herança e Composição leia o artigo “Conceitos e Exemplos – Polimorfismo: Programação Orientada a Objetos” da DEVMEDIA.

Referências Bibliográficas

Gilleanes T. A. Guedes. **UML 2 - Uma Abordagem Prática**. São Paulo: NovaTec, 2018.

Grady Booch. **Uml - Guia do Usuário**. Rio de Janeiro: Editora Campus, 2018.

Ian Sommerville. **Engenharia de software**. São Paulo: Pearson, 2015.

Roger Pressman, Bruce Maxim. **Engenharia de Software**. Porto Alegre: Bookman, 2010.

Ir para exercício