

POO

Programação Orientada a Objetos

Material 001

Professor Maromo



Agenda

PILARES



Os pilares da Programação Orientada a Objetos (POO) referem-se aos quatro conceitos fundamentais que definem a abordagem da POO.

CONCEITOS

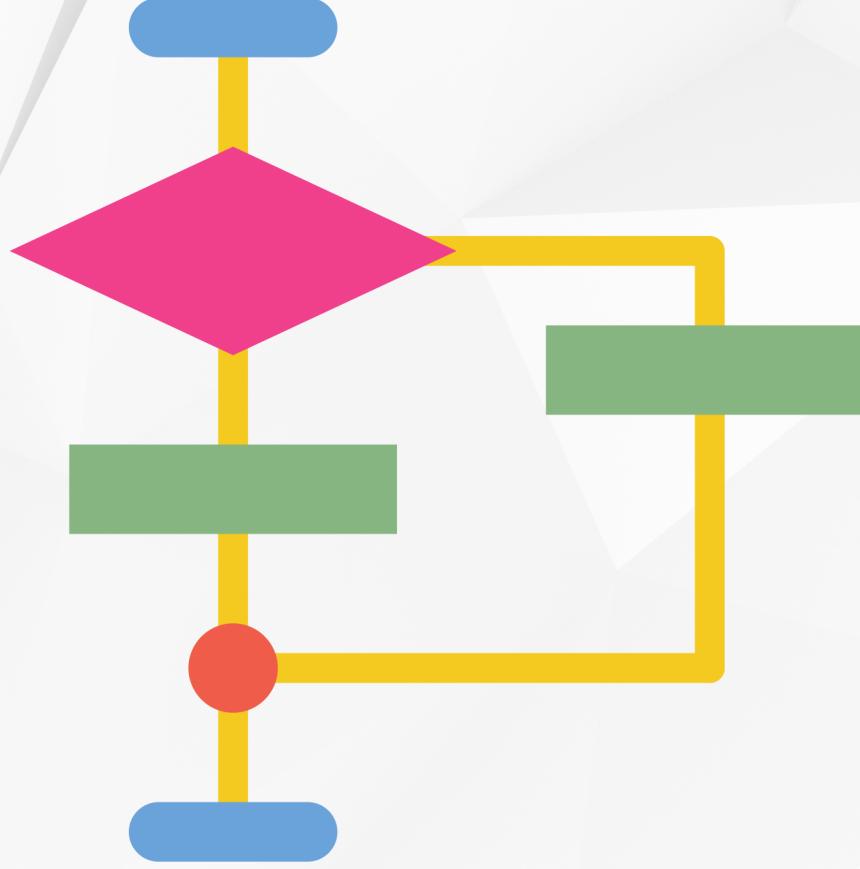


A Programação Orientada a Objetos (POO) é um paradigma de programação baseado em objetos, que podem conter dados e código para manipular esses dados.

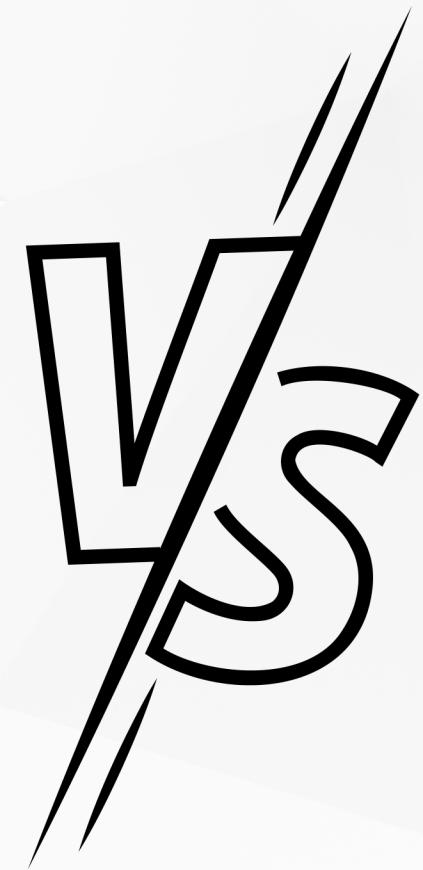
NOTA



“Apresenta-se de forma resumida os principais conceitos de Orientação a Objetos que será alvo de estudo ao longo do curso.



Programação
Estruturada



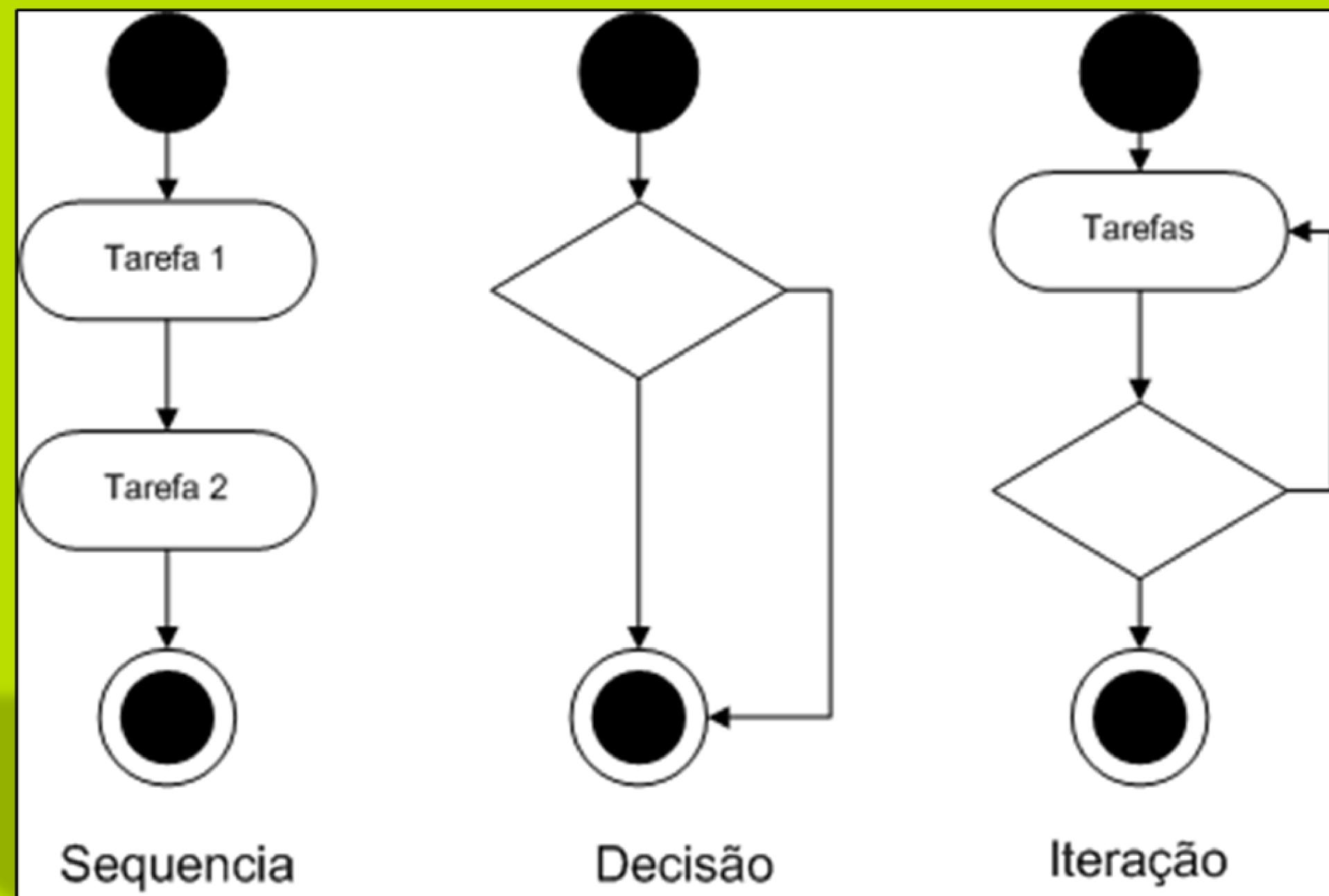
Programação
Orientada a Objetos

Programação Estruturada

- Fortemente baseada em modularização (dividir para conquistar):
 - Unidades menores são construídas para desempenhar uma tarefa específica.
 - Pode ser executada várias vezes.
 - Funções podem receber parâmetros e retornar valor.



Mecanismos de Interligação



Orientação a Objetos

- Representação por meio de abstrações

Ex: As crianças aprendem a reconhecer coisas simples como **pessoa, carro e casa**. Cada objeto é um exemplo de um determinado grupo.



Definição e Exemplo

- Definição: Uma classe é um modelo ou esquema usado para criar objetos. Ela define os **atributos** e **métodos** que seus objetos terão.
- Exemplo: A classe "Carro" define os atributos e métodos comuns a todos os carros.



Nota: na Linguagem UML (Linguagem de Modelagem Unificada) uma Classe é identificada por um retângulo. A figura acima apresenta uma divisão. Entretanto, normalmente é representada com três divisões.

Por padrão na linguagem JAVA a classe é definida com a primeira letra maiúscula.

Atributos

- Atributos são as características ou propriedades que um objeto possui.
- A **marca**, o **modelo** e a **cor** são atributos de um objeto da classe "Carro".

Carro

marca

modelo

cor

Métodos

- Métodos ou comportamentos: Representam uma atividade que um objeto de uma classe pode realizar.
- Exemplo: Os métodos `ligar()` e `desligar()` da classe "Carro" controlam o estado do carro.

Carro

marca

modelo

cor

ligar

desligar

Visibilidade

Refere-se ao controle de acesso aos membros de uma classe (ou seja, atributos e métodos). É um aspecto essencial do encapsulamento, pois permite que os desenvolvedores restrinjam o acesso a certas partes de um objeto, protegendo assim sua integridade e ocultando detalhes de implementação.

Carro

+marca

+modelo

+cor

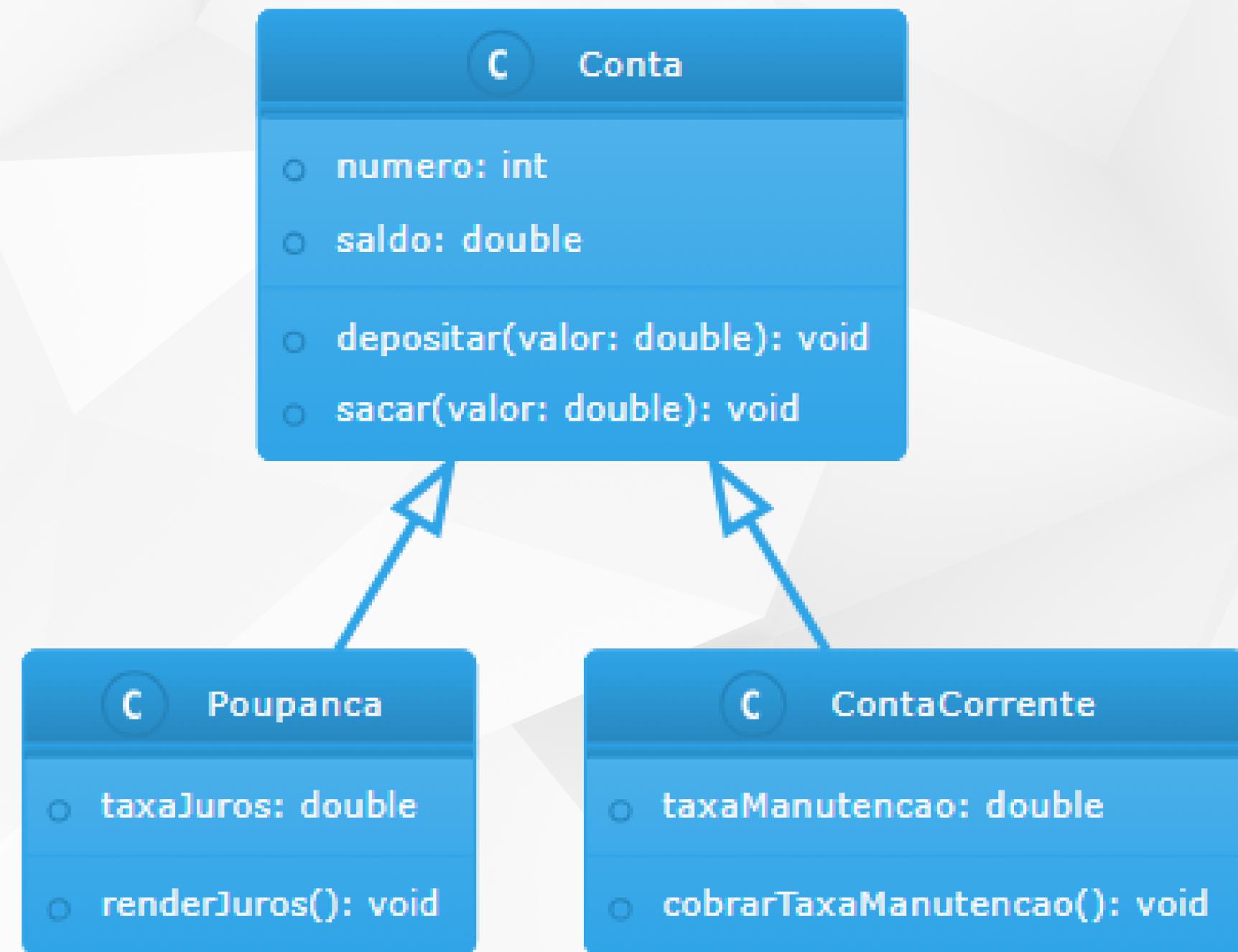
-ligar

#desligar

Visibilidade

	Definição	Exemplo
Public [+]	Acessível a qualquer parte do código, dentro ou fora da classe.	Um método público pode ser chamado a partir de qualquer parte do programa.
Private [-]	Acessível somente dentro da própria classe onde foram declarados.	Um atributo privado só pode ser modificado pelos métodos dentro da mesma classe.
Protected [#]	Acessível dentro da mesma classe ou em subclasses.	A classe derivada pode acessar membros protegidos da classe base, mas outros códigos fora dessas classes não podem.

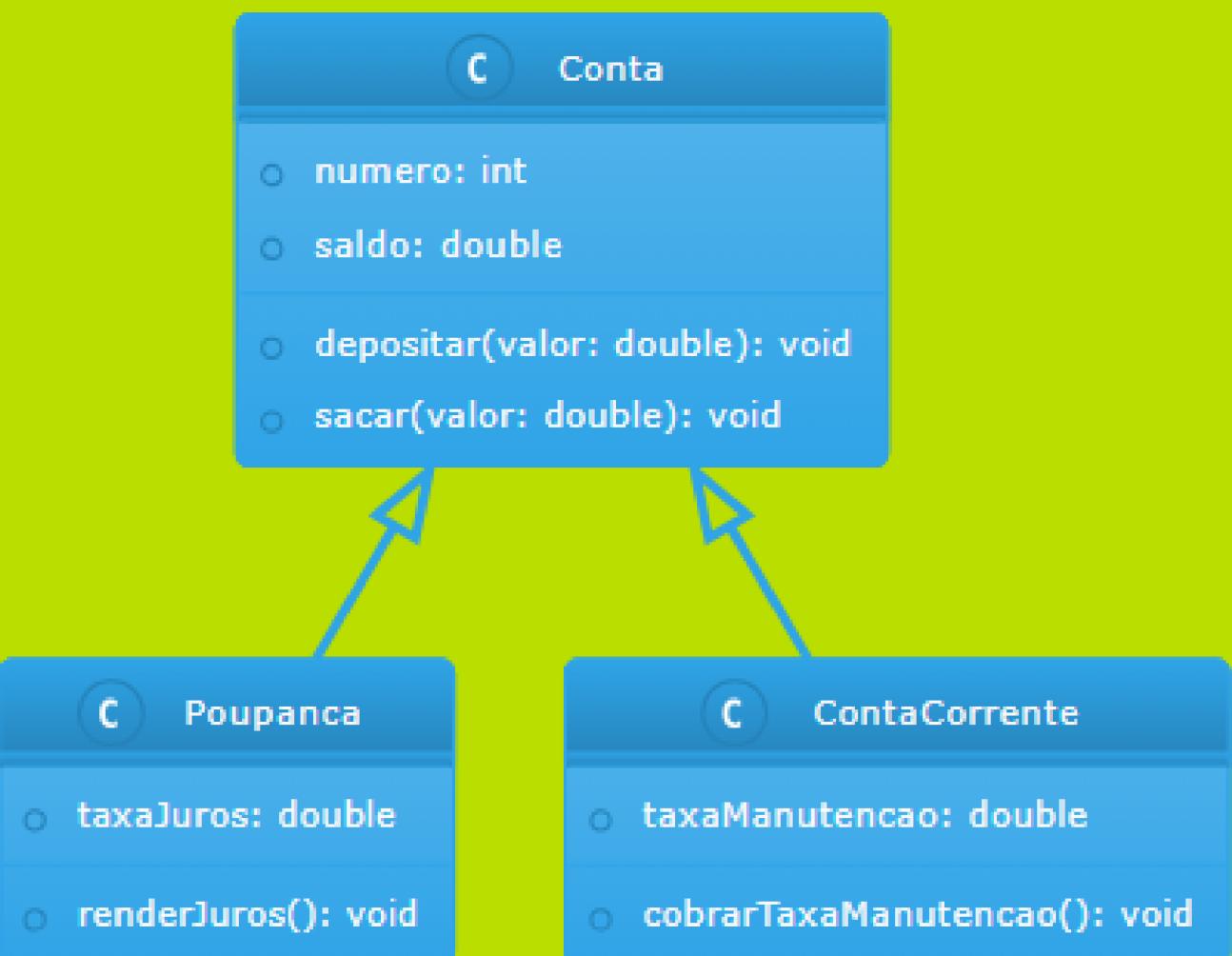
Herança



Herança

Para entender o conceito de herança, primeiro precisa-se saber primeiro o que são: superclasses e subclasses.

- **Superclasse:** ou classe-mãe, é uma classe que possui classes derivadas a partir dela.
- **Subclasse:** ou classe-filha, é uma classe derivada de uma superclasse. As classes-filhas (subclasses) herdam características de uma classe-mãe (atributos e métodos).



Herança

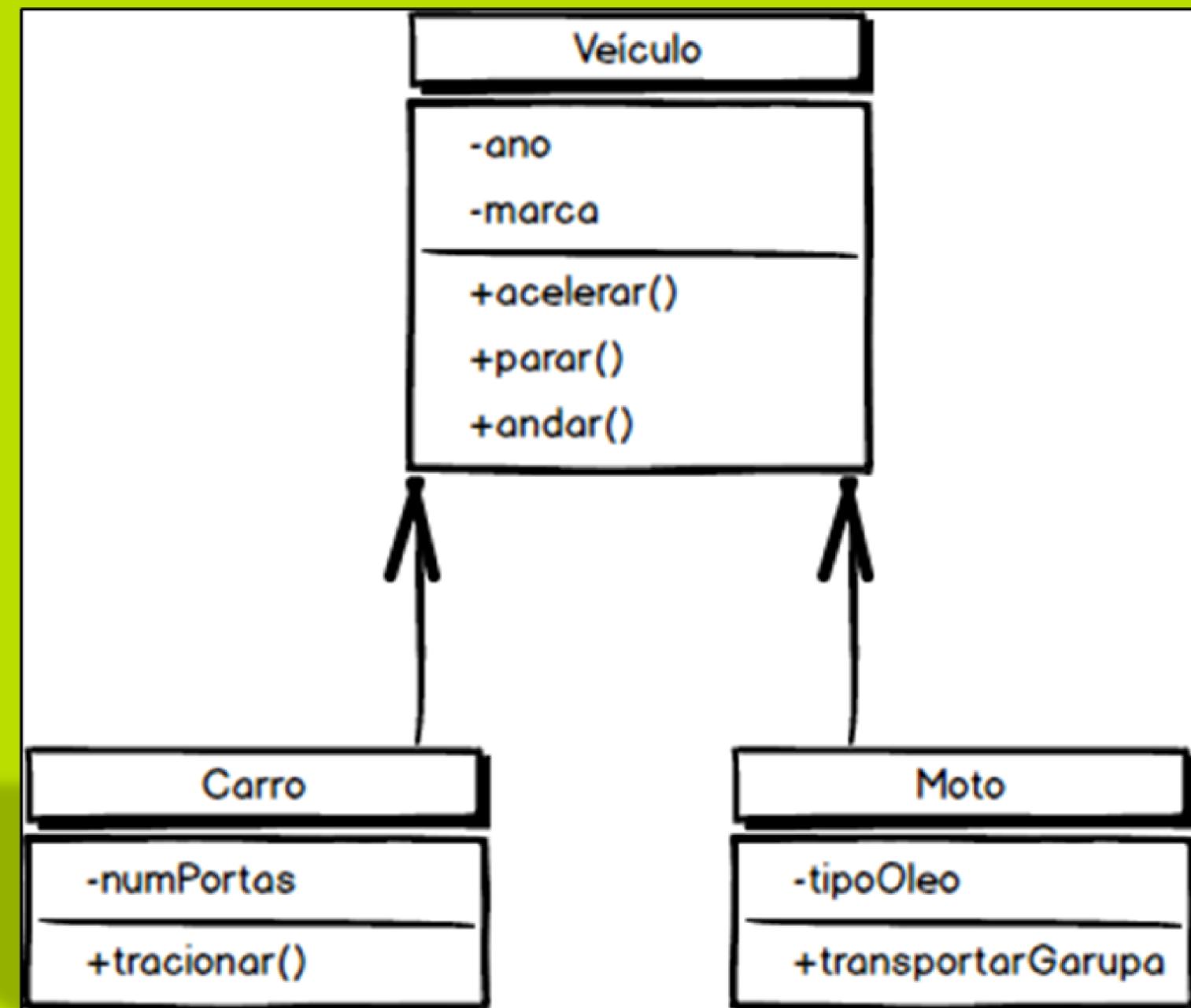
A principal vantagem é o compartilhamento de atributos e comportamentos entre classes de uma mesma hierarquia.

Podemos reaproveitar uma estrutura já existente que nos forneça uma base abstrata para o desenvolvimento de software.



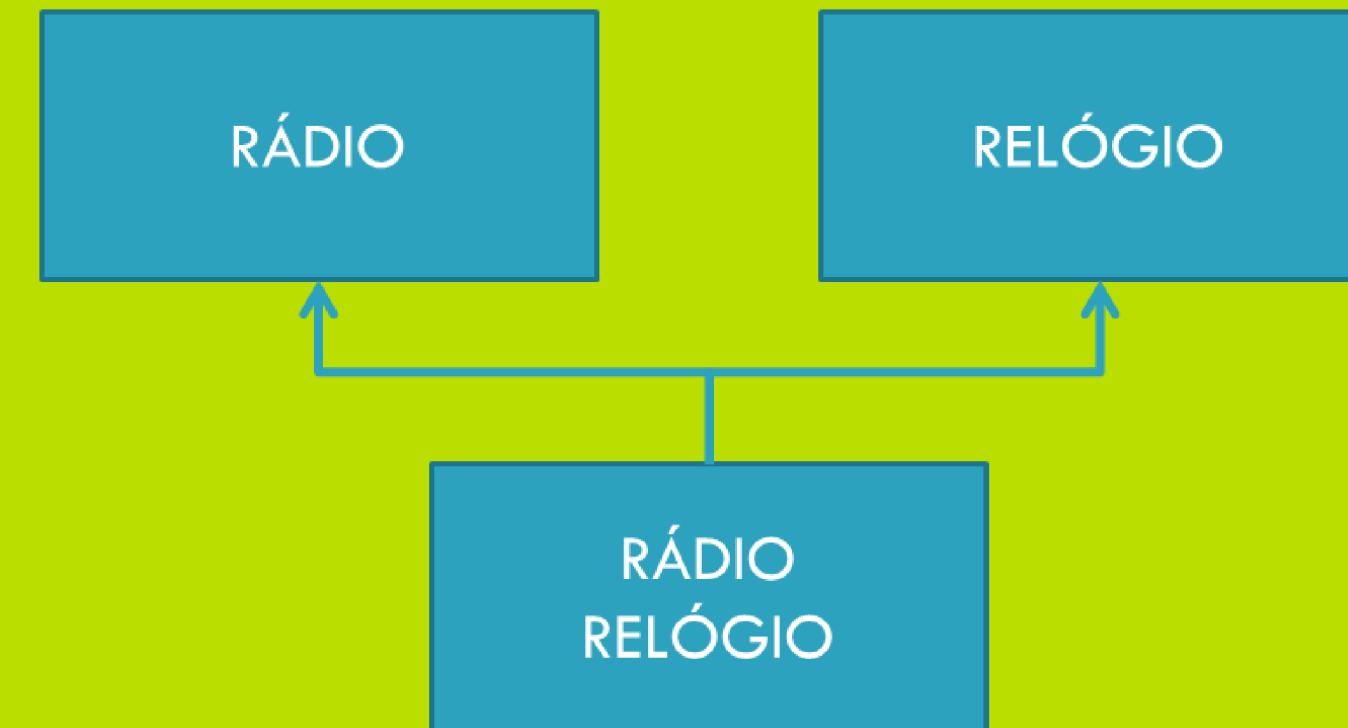
Generalização e especialização

Podemos criar classes gerais, com características compartilhadas por muitas classes (Veiculo), mas que possuem diferenças pequenas entre si (Carros possuem portas, motos não) – consideramos carro e moto como especializações de veículos.



Herança Múltipla

Ocorre quando uma subclasse herda características de duas ou mais superclasses.



**Java abandona a ideia de herança múltipla.
Em seu lugar usa-se interfaces.**

Conceito

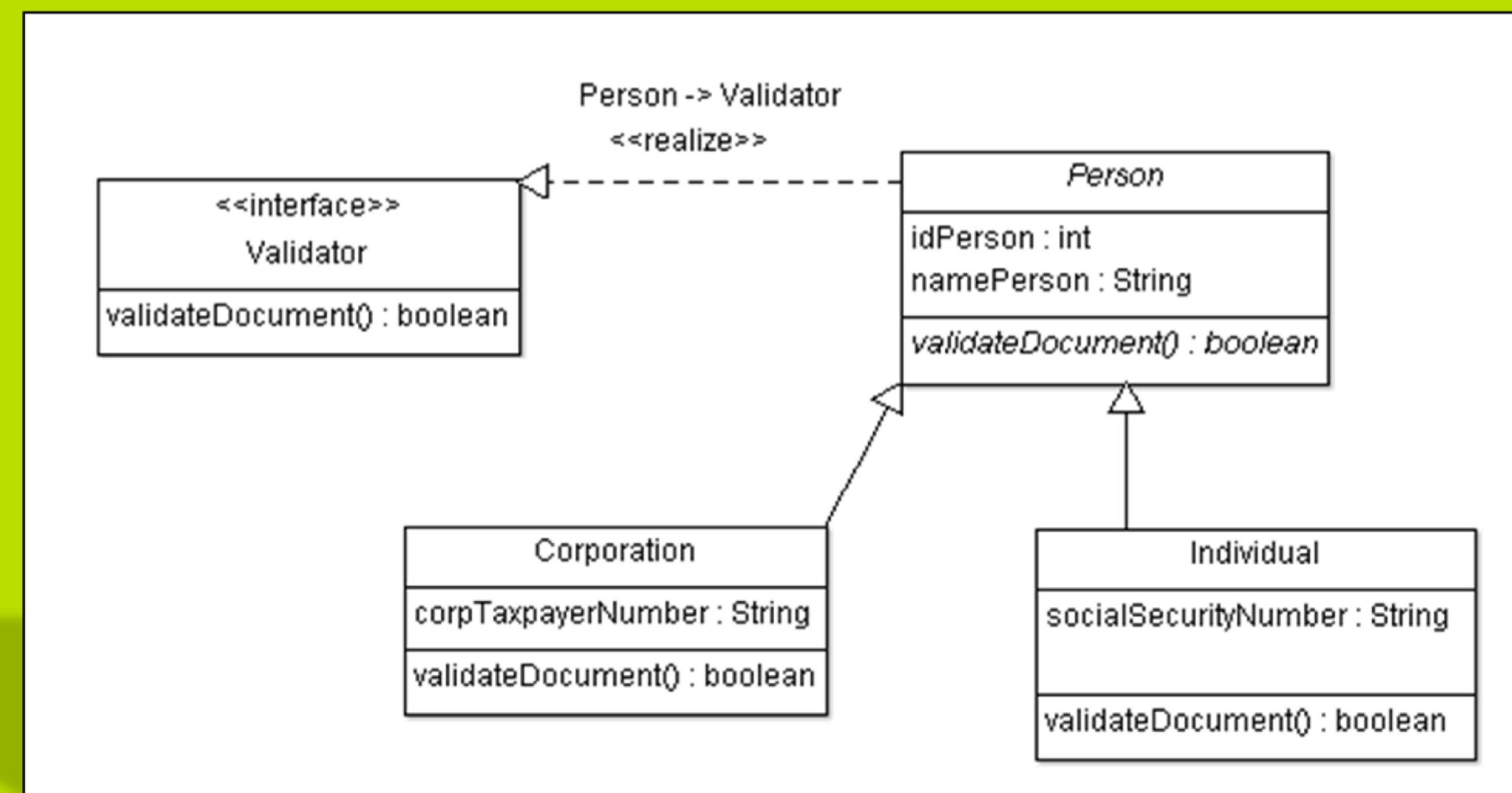
O polimorfismo é um dos pilares fundamentais da Programação Orientada a Objetos (POO). Refere-se à capacidade de uma classe ter múltiplas formas.

Em outras palavras, diferentes objetos podem ser tratados como instâncias da mesma classe, permitindo que eles sejam usados de maneira intercambiável.

O polimorfismo aumenta a flexibilidade e a reutilização do código.



Polimorfismo



Abstração

Definição: Abstração envolve focar nas características essenciais de um objeto e ignorar os detalhes desnecessários.

Exemplo: Ao dirigir um carro, você não precisa entender como o motor funciona.

Capacidade de considerar de forma isolada, simplificar, determinar o problema genericamente, dando importância aos aspectos mais relevantes.

Na prática, construir “peças” (classes) bem definidas que possam ser reaproveitadas, formando uma estrutura hierárquica.



Pesquisa



Investigação sobre Paradigmas de Programação, com Ênfase na Programação Orientada a Objetos:

Objetivo: Compreender os diferentes paradigmas de programação, focando na Programação Orientada a Objetos (OOP), e compará-los entre si.

Instruções:

Programação Orientada a Objetos (OOP):



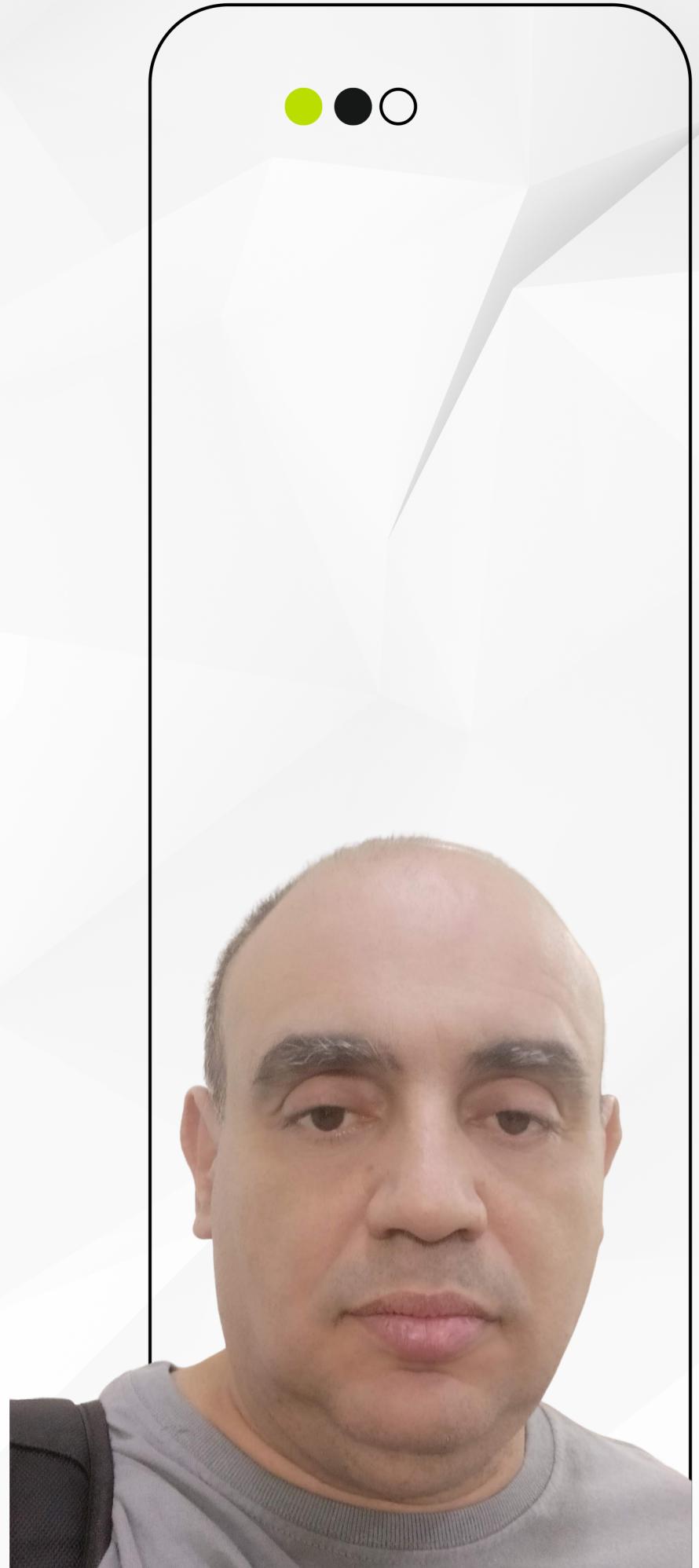
- Descreva o que é OOP e explique suas principais características, como encapsulamento, herança, polimorfismo e abstração.
- Discuta a importância do OOP no desenvolvimento de software moderno e como ele facilita a modelagem de entidades do mundo real.

Outros Paradigmas de Programação:

- Identifique e explique pelo menos três outros paradigmas de programação, como programação procedural, funcional e lógica.
- Compare cada um desses paradigmas com o OOP, destacando suas diferenças e semelhanças, vantagens e desvantagens.

Conclusão:

- Resuma suas descobertas, destacando o papel do OOP em relação a outros paradigmas.
- Reflita sobre as situações em que cada paradigma pode ser mais adequado e como a escolha do paradigma pode influenciar o desenvolvimento de software.



Obrigado

fim



Até a próxima aula



Referências Bibliográficas



- Mendes; **Java com Ênfase em Orientação a Objetos**, Novatec.
- Deitel; **Java, como programar** – 10º edição. Java SE 7 e 8
- Arnold, Gosling, Holmes; **A linguagem de programação Java** – 4º edição.
- Apostilas da Caelum.