Herança

Conceitos

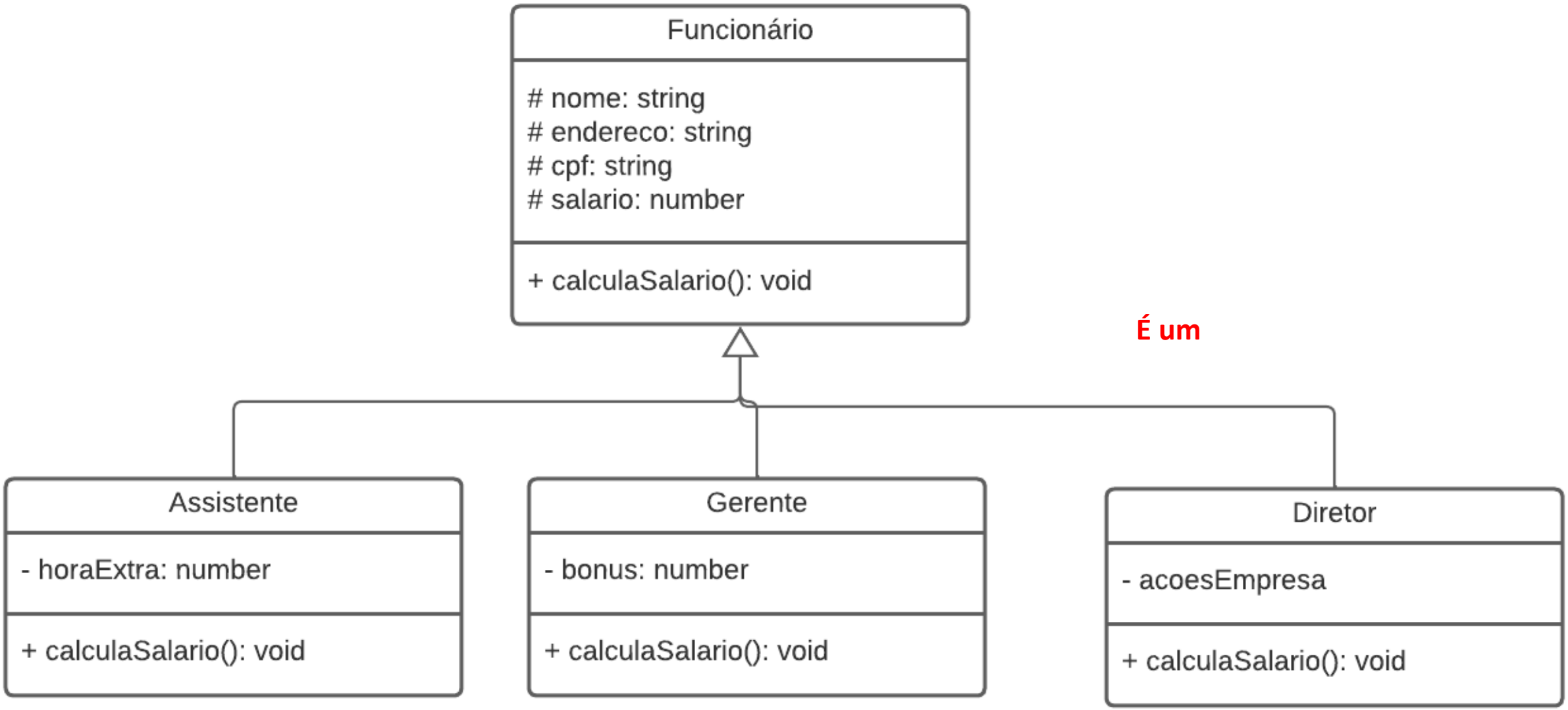
- Associação entre classes
  - Generalização e Especialização

- Classe mais genérica, chamada de classe pai ou superclasse

- Clsse mais específica, chamada de classe filha ou subclasse

Conceitos

- Recurso das linguagens de programação orientadas a objeto
- Recurso onde as classes filhas ou subclasses herdam variáveis e métodos da(s) classe(s) pai, ou superclasse(s)
- Isso promove a reutilização de código e facilita a organização hierárquica das classes.


Funcionário
nome: string
endereco: string
cpf: string
salario: number
+ calculaSalario(): void
É um
Assistente
- horaExtra: number
+ calculaSalario(): void
Gerente
- bonus: number
+ calculaSalario(): void
Diretor
- acoesEmpresa
+ calculaSalario(): void

- **Classe Base (ou Superclasse)**: Também conhecida como classe pai ou classe mãe, é a classe da qual outra classe herda. Ela contém os atributos e métodos comuns que são compartilhados pelas subclasses.
- **Classe Derivada (ou Subclasse)**: Também chamada de classe filha, é a classe que herda de outra classe. Ela pode adicionar novos atributos e métodos ou modificar os existentes, além de herdar os atributos e métodos da classe base.

- **Relação "é-um"**: A herança estabelece uma relação "é-um" entre a classe base e suas subclasses. Por exemplo, se temos uma classe "Animal" como classe base e uma classe "Cachorro" como uma de suas subclasses, podemos dizer que um cachorro é um tipo de animal.
- **Método de Herança**: É o processo pelo qual uma classe deriva características e comportamentos de uma classe base. Isso é feito através da declaração da relação de herança na definição da classe.

- **Membros Privados, Protegidos e Públicos**: Na herança, membros privados da classe base não são diretamente acessíveis pelas subclasses. Membros protegidos são acessíveis pelas subclasses, enquanto membros públicos são acessíveis tanto pelas subclasses quanto por classes externas.
- **Sobrescrita de Métodos (ou Overriding)**: As subclasses podem fornecer implementações específicas para métodos que foram definidos na classe base. Isso permite que as subclasses personalizem o comportamento dos métodos herdados.

- **Chamada ao Método da Classe Base**: As subclasses podem chamar métodos da classe base usando a sintaxe apropriada, permitindo a reutilização de funcionalidades existentes.
- **Herança Múltipla (em algumas linguagens)**: Algumas linguagens de programação orientada a objetos suportam herança múltipla, permitindo que uma classe tenha mais de uma classe base. Isso pode levar a complexidades de design e ambiguidades e nem todas as linguagens oferecem suporte a essa característica.

Visibilidade protegida (#)

- Variável protegida é:
  - pública para as classes filhas
  - privada para as classes que não pertencem à hierarquia de herança

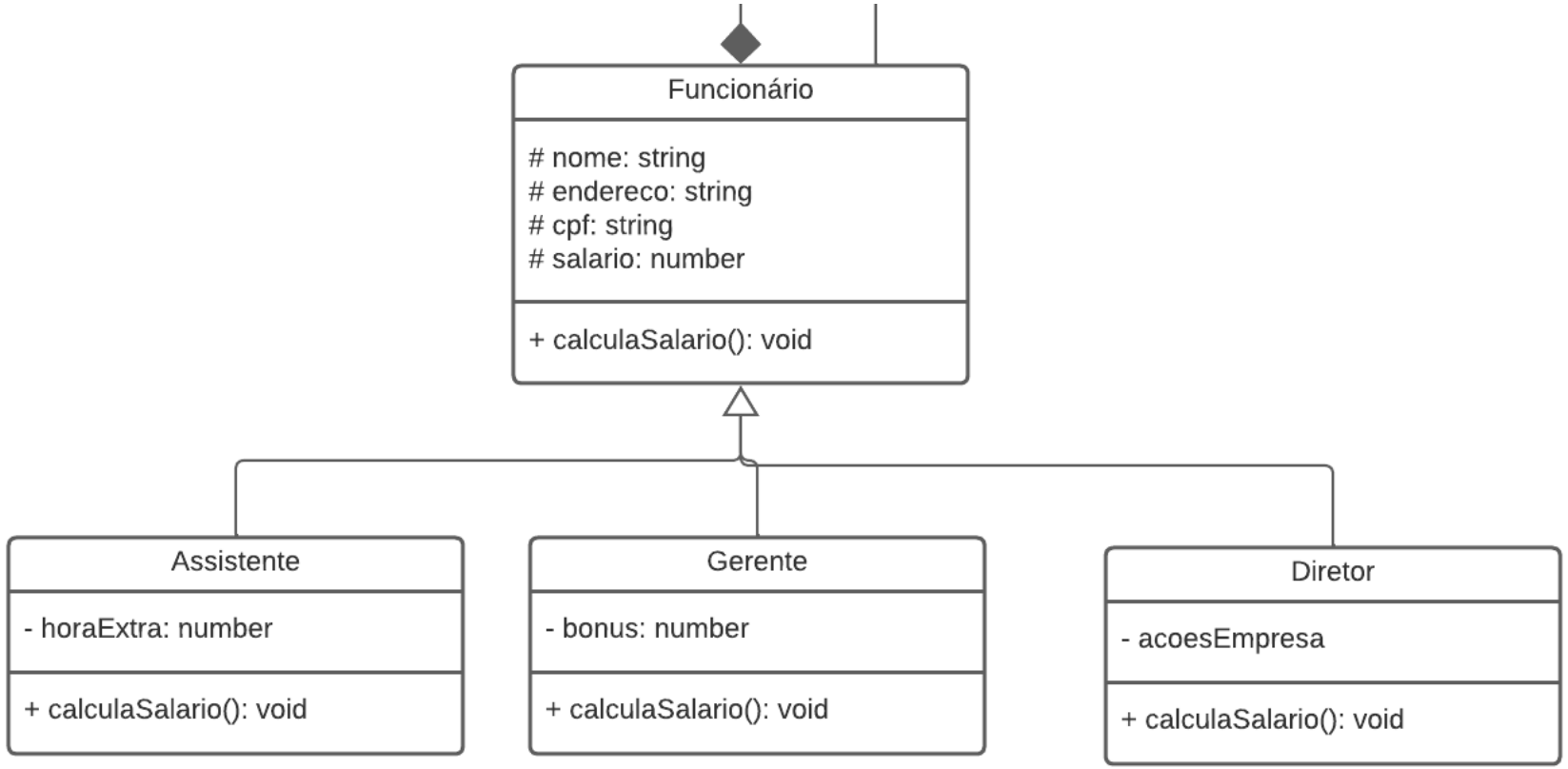
2 tipos de herança

- Herança simples
  - Classe filha ou subclasse tem apenas um único pai ou superclasse

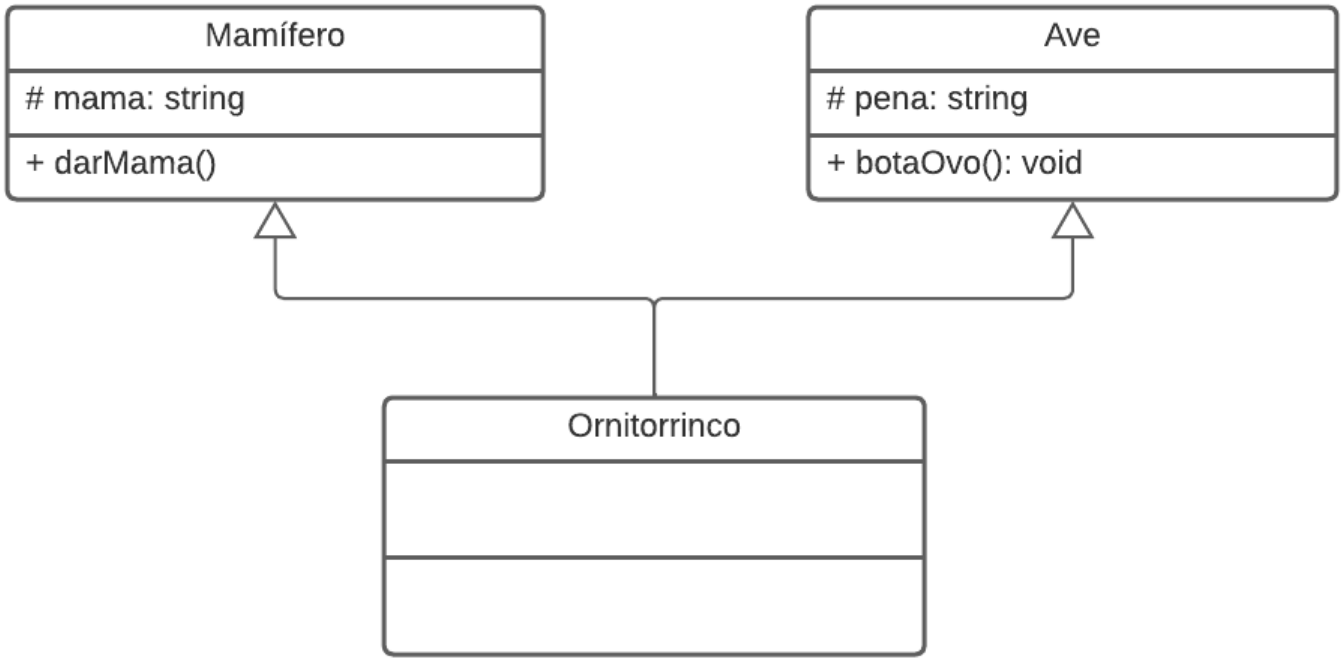
- Herança múltipla
  - Classe filha ou subclasse tem dois ou mais pais ou superclasses

Herança simples

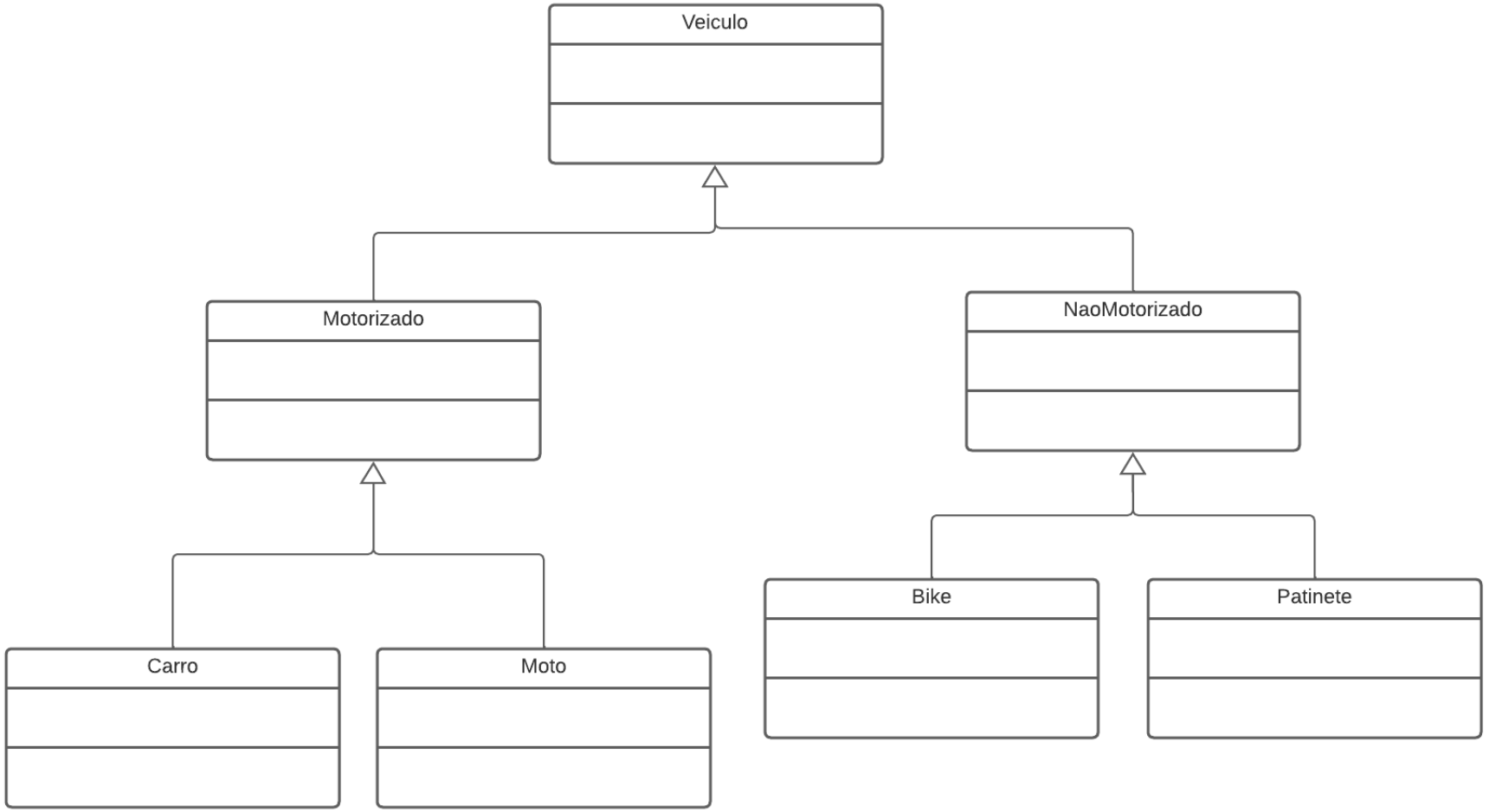
- Exemplo



Herança Múltipla



Herança com vários níveis hierárquicos



Hands On

Anulação de métodos

- herdamos toString(), mas ele não é suficiente
- vamos anular o método toString() herdado

```
toString(): string {
    return `${super.toString()} Poisonous: ${this.poisonous}`
}
```

Anulação de métodos

- herdamos move(), mas não o queremos
- vamos anular o método move()

```
move(): string {
        return `snake crawling`
    }
```

Polimorfismo de herança

- Baseado em herança e substituição de métodos.
- O polimorfismo de subtipo ocorre quando uma classe filha (subclasse) herda de uma classe pai (superclasse) e substitui os métodos da classe pai com sua própria implementação.
- Permite que você trate objetos de subclasses como objetos de sua superclasse, usando a referência da superclasse para manipular os objetos.

Polimorfismo - exemplo

```
function exemploPolimorfismo(cameleao: Animal): void{
    // se a função for chamada passando uma cobra, camaleao vai representar uma cobra
    // e portanto, será executado o toString() e o move() da cobra
    // se a função for chamada passando um cavalo, camaleao vai representar um cavalo
    // e portanto, será executado o toString() e o move() do cavalo
    console.log(cameleao.toString()) // polimorfismo
    console.log(cameleao.move()) // polimorfismo
}
```

Classe Abstrata

- É uma classe que não pode ser instanciada diretamente, mas pode ser usada como uma superclasse para outras classes.
- Ela serve como um esqueleto para outras classes que estendem dela, fornecendo implementações parciais ou completas de métodos, enquanto ainda permitindo que subclasses forneçam suas próprias implementações.

Classe Abstrata

- Uma classe abstrata é declarada usando a palavra-chave abstract antes da palavra-chave class.
- Uma classe abstrata pode conter métodos abstratos, que são métodos declarados sem uma implementação.
- Métodos abstratos são definidos usando a palavra-chave abstract e terminam com um ponto e vírgula, sem corpo.
- Subclasses devem fornecer implementações para todos os métodos abstratos da classe abstrata.

Classe Abstrata

- Além de métodos abstratos, uma classe abstrata pode conter métodos concretos (ou normais) com implementações.
- Os métodos concretos podem ser herdados pelas subclasses ou sobrescritos, se necessário.
- Subclasses devem fornecer implementações para todos os métodos abstratos herdados da classe abstrata.