



# Diagrama de Classe

## **D**iagrama Estrutural – Diagrama de Classe

O diagrama de classes é um diagrama estático. Representa a visão estática de um aplicativo. O diagrama de classes não é usado apenas para visualizar, descrever e documentar diferentes aspectos de um sistema, mas também para construir código executável do aplicativo de software. O diagrama de classes descreve os atributos e operações de uma classe e também as restrições impostas ao sistema. Os diagramas de classes são amplamente usados na modelagem de sistemas orientados a objetos, pois é um dos únicos diagramas UML, que podem ser mapeados diretamente com linguagens orientadas a objetos. O diagrama de classes mostra uma coleção de classes, interfaces, associações, colaborações e restrições.

### **Objetivo dos diagramas de classes**

O objetivo do diagrama de classes é modelar a visão estática de um aplicativo. Os diagramas de classes são os únicos que podem ser mapeados diretamente com linguagens orientadas a objetos e, portanto, amplamente utilizados no momento da construção.

Diagramas UML, como diagrama de atividades, diagrama de sequência, podem fornecer apenas o fluxo de sequência do aplicativo; no entanto, o diagrama de classes é um pouco diferente. É o diagrama UML mais popular na comunidade de desenvolvedores.

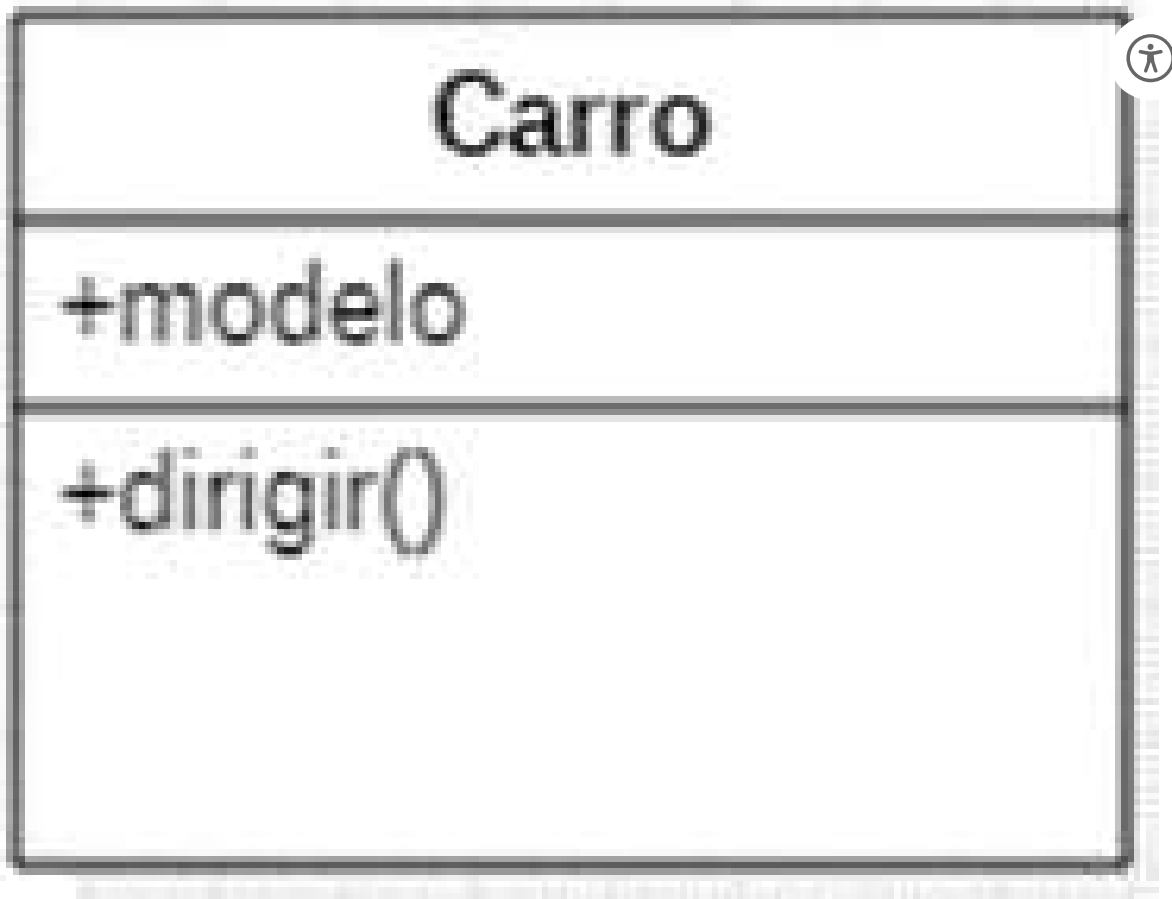
O objetivo do diagrama de classes pode ser resumido como:



## Os elementos essenciais do diagrama de classes UML

Abaixo são listados os três elementos fundamentais para entender o que é uma classe e que faz parte do diagrama de classe:

- Análise e design da visão estática de um aplicativo.
- Descreve as responsabilidades de um sistema.
- Base para diagramas de componentes e implantação.
- Engenharia avançada e reversa.
- Nome da Classe
- Atributos
- Operações



Nesta classe definimos o nome da classe como Carro, definimos um atributo chamado modelo e uma operação chamada dirigir.

O nome da classe é necessário apenas na representação gráfica da classe. Aparece no compartimento superior. Uma classe é o modelo de um objeto que pode compartilhar os mesmos relacionamentos, atributos, operações e semântica. A classe é renderizada como um retângulo, incluindo seu nome, atributos e operações em compartimentos separados.

As regras a seguir devem ser atendidas enquanto representam uma classe:

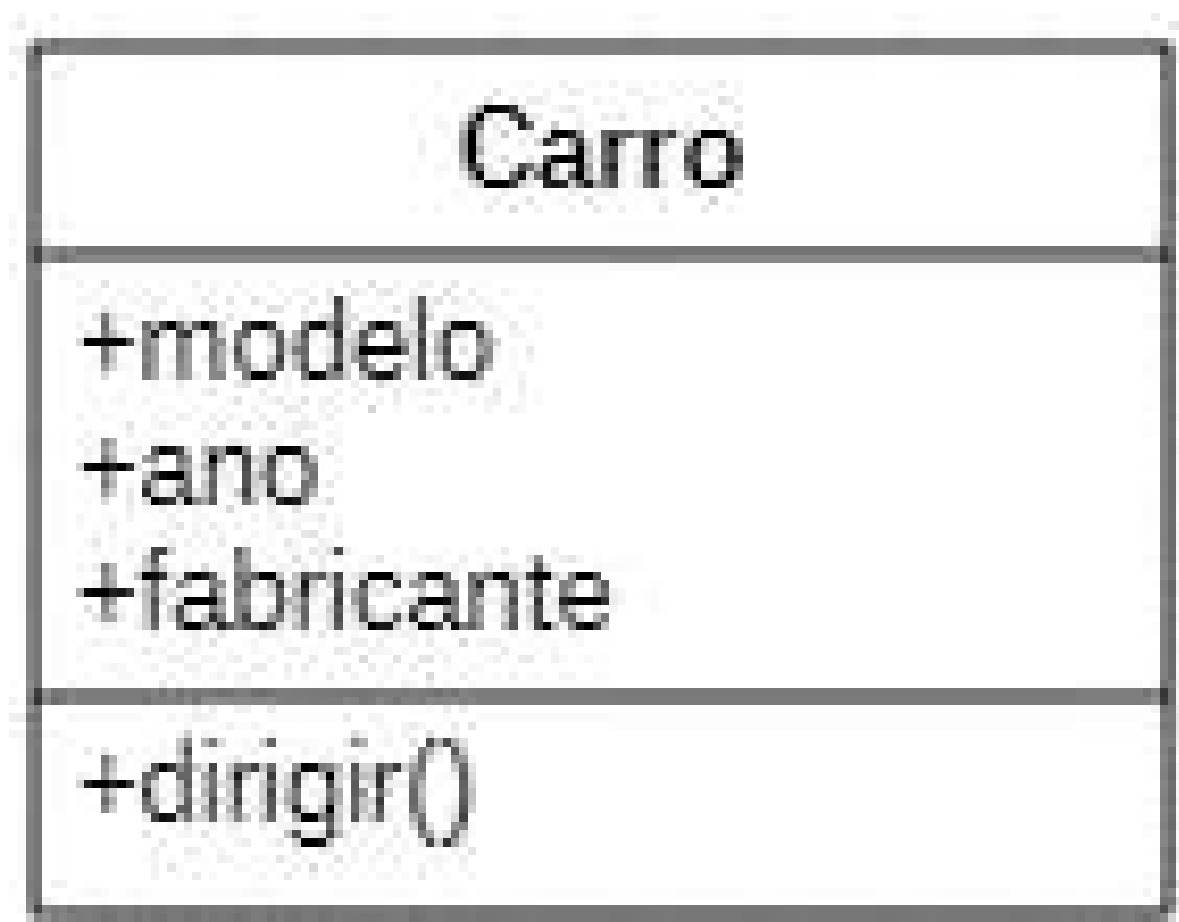
- Um nome de classe deve sempre começar com uma letra maiúscula.
- Um nome de classe sempre deve estar no centro do primeiro compartimento.
- Um nome de classe sempre deve ser escrito em formato negrito.

- Um nome de classe abstrata deve ser escrito em *itálico*.

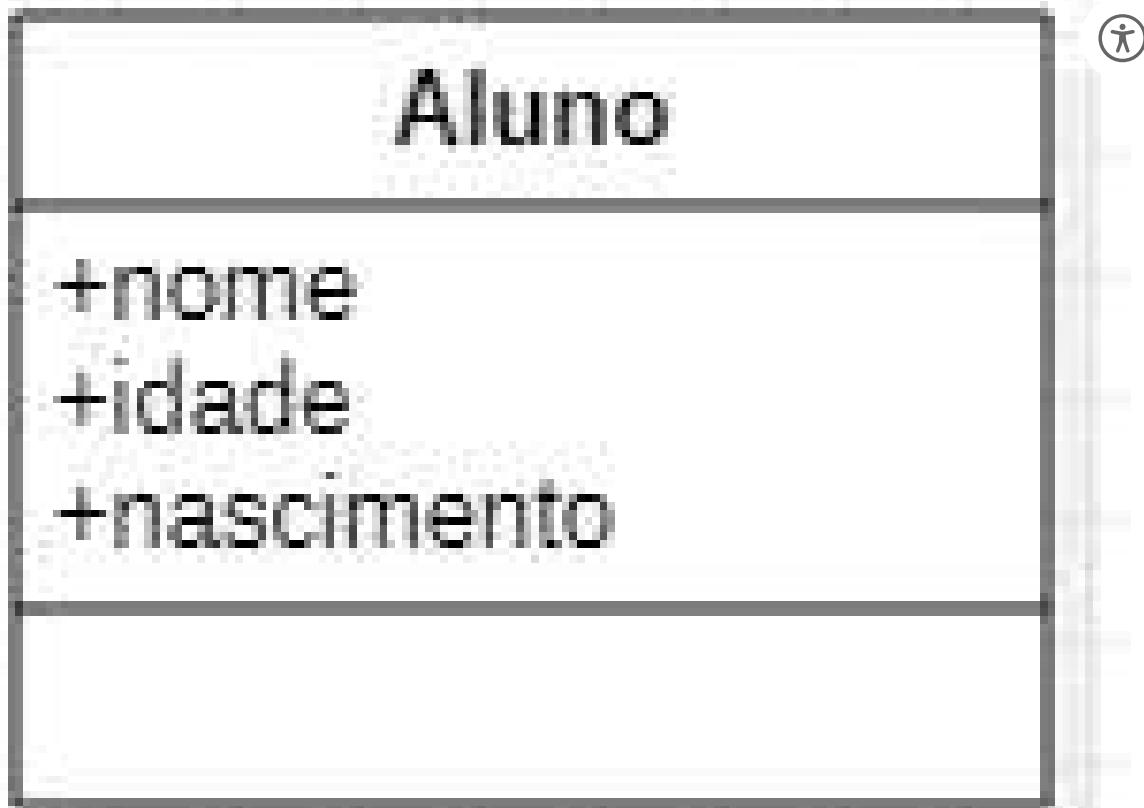


## Atributos

Um atributo é nomeado como a propriedade de uma classe que descreve o objeto que está sendo modelado. No diagrama de classes, os atributos são colocados logo abaixo do compartimento do nome da classe.



Um atributo derivado é calculado a partir de outros atributos. Por exemplo, a idade de aluno pode ser facilmente calculada a partir da data de nascimento.



### Características dos atributos

Os atributos são geralmente escritos junto com o fator de visibilidade. Público, privado, protegido e pacote são as quatro visibilidades denotadas por sinais +, -, # ou ~, respectivamente. Visibilidade descreve a acessibilidade de um atributo de uma classe. Os atributos devem ter um nome significativo que descreva o uso dele em uma classe.

### Relacionamentos

Existem principalmente três tipos de relacionamentos na UML:

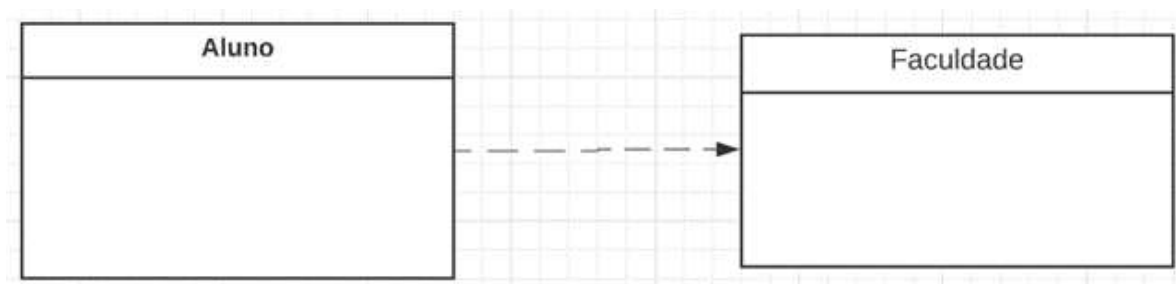
- Dependências
- Generalizações / Especializações



## ***Dependência***

Uma dependência significa a relação entre duas ou mais classes em que uma mudança em uma, pode forçar mudanças na outra. No entanto, sempre criará um relacionamento mais fraco. A Dependência indica que uma classe depende de outra.


No exemplo a seguir, o Aluno depende da faculdade.

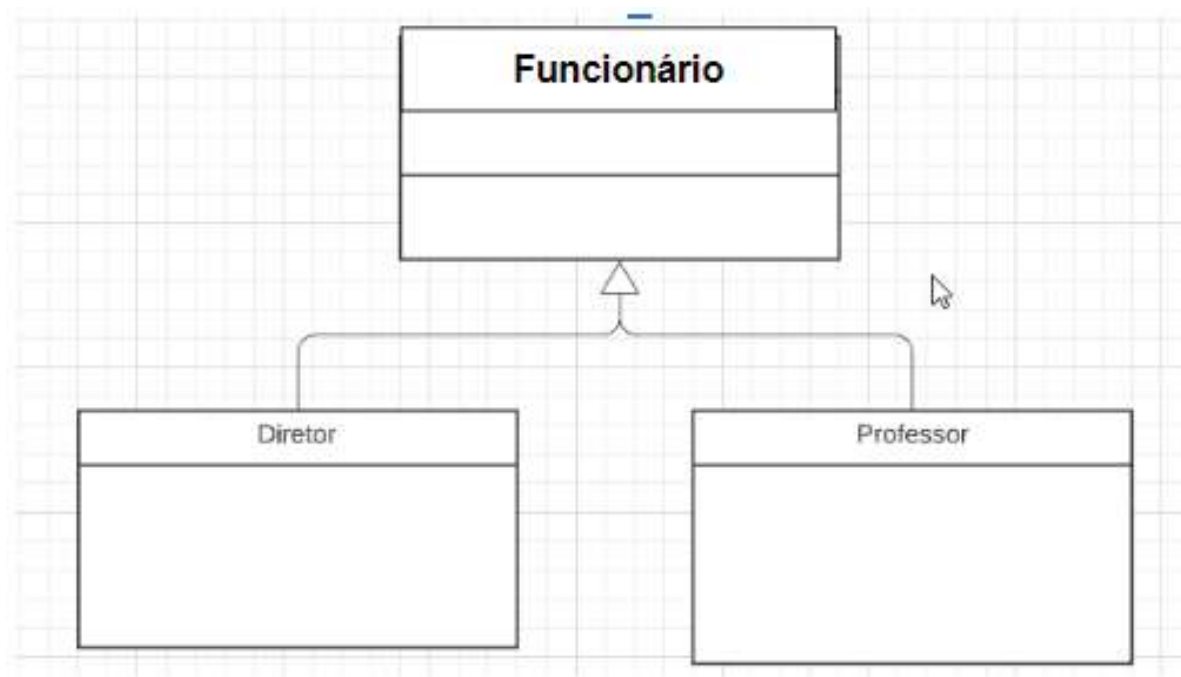


## ***Generalização / Especialização***

A generalização é a representação que se dá a conjuntos de objetos que podem ser classificados em tipos diferentes. Geralmente existem semelhanças entre essas diferentes classes, o que faz com que criemos especializações.

Dentre as especializações, duas ou mais classes compartilham os mesmos atributos e / ou os mesmos métodos. Como você não precisa escrever o mesmo código repetidamente, deseja um mecanismo que aproveite essas semelhanças. Quando A herda (usa) de B, dizemos que A é a subclasse de B e B é a superclasse de A. Além disso, dizemos que temos “herança pura” quando A herda todos os atributos e métodos de B.

A notação de modelagem UML para herança é uma linha com uma ponta de seta fechada apontando da subclasse para a superclasse, conforme a figura a seguir. 

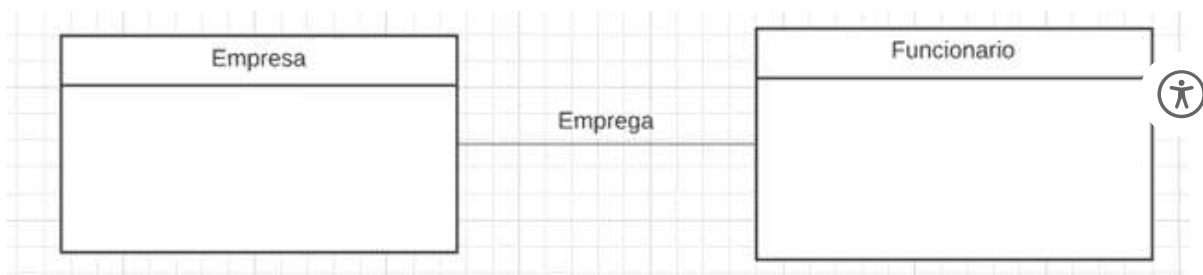


## ***Associação***

Esse tipo de relacionamento representa relacionamentos estáticos entre as classes A e B. Por exemplo: um funcionário trabalha para uma organização.

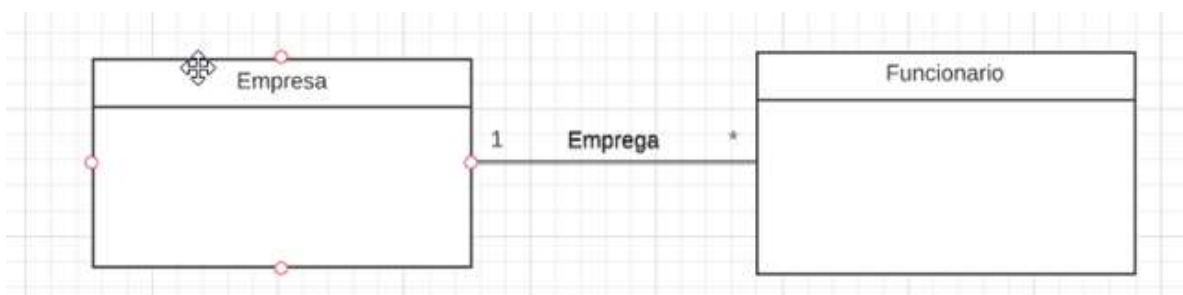
Aqui estão algumas regras para associação:

- Associação é principalmente verbo ou frase verbal ou substantivo ou frase substantivo.
- Deve ser nomeado para indicar o papel desempenhado pela classe anexada no final do caminho da associação.
- Obrigatório para associações reflexivas



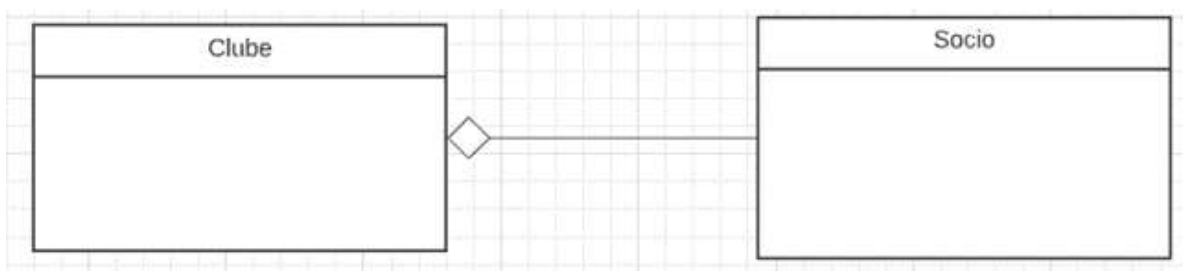
### ***Associação - Multiplicidade***

Uma multiplicidade é um fator associado a um atributo. Ele especifica quantas instâncias de atributos são criadas quando uma classe é inicializada. Se uma multiplicidade não for especificada, por padrão, uma será considerada como uma multiplicidade padrão.



### ***Agregação***

A agregação é um tipo especial de associação que modela um relacionamento de parte inteira entre agregado e suas partes.





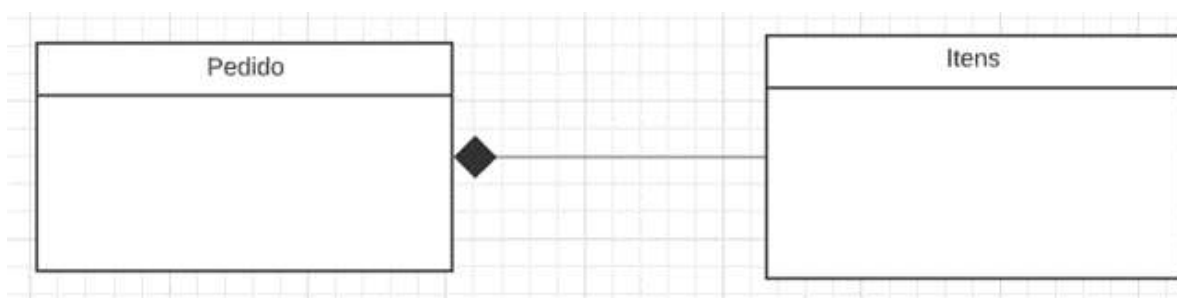
Um Sócio é parte de um clube. Mas se ele deixar de existir o clube continua existindo.



## **Composição**

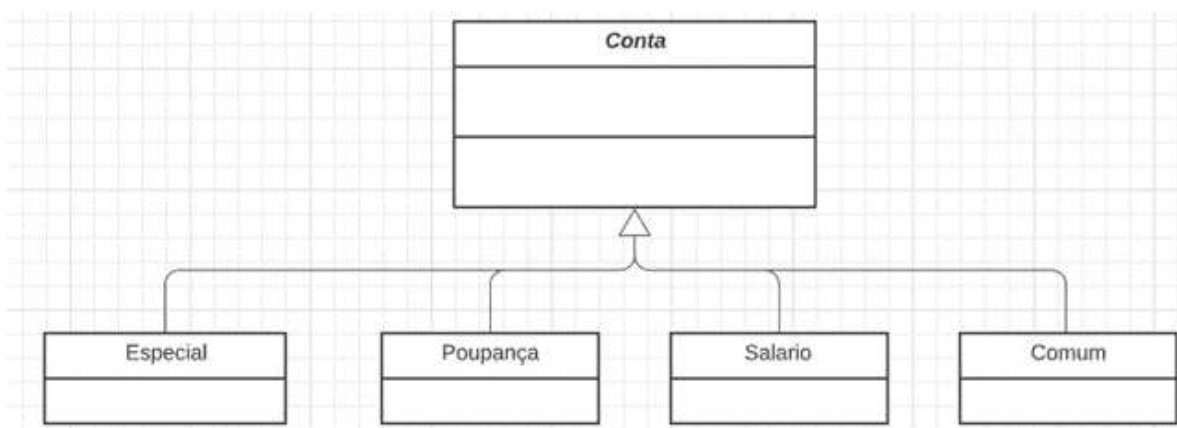
A composição é um tipo especial de agregação que denota forte propriedade entre duas classes quando uma classe faz parte de outra classe.

Por exemplo, se um pedido é composto de itens de pedidos. O pedido pode conter muitos itens. Portanto, se não tiver pedido, todos os itens de pedido também serão removidos.



## **Classes abstratas**

É uma classe com um protótipo de operação, mas não a implementação. Também é possível ter uma classe abstrata sem operações declaradas dentro dela. Um resumo é útil para identificar as funcionalidades entre as classes.





## **Atividade extra**

Leia este texto interessante: “Orientações básicas na elaboração de um diagrama de classes” no site da DEVMEDIA. Nele, vemos um exemplo de como construir um diagrama.

## **Referências Bibliográficas**

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

Grandy 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**