

Programação Orientada a Objetos – (POO0001)

UML

Alexandre Mendonça Fava
alexandre.fava@udesc.br

Universidade do Estado de Santa Catarina – UDESC
Programa de Pós-graduação em Computação Aplicada – PPGCA

Roteiro

- Definição
- História
- Diagrama
- Modelo
- Diagrama de Classes
- Diagrama de Objetos
- Diagrama de Pacotes
- Resumo
- Referências

Definição

UML

Definição

Linguagem de Modelagem Unificada



Definição

Linguagem

É um sistema responsável pela **comunicação** baseado em um conjunto de regras e símbolos

Modelagem

É um processo responsável pela elaboração e construção **representativa** de conceitos

BOOCH, Grady; RUMBAUGH, James; JACOBSON, Ivar. **UML**.

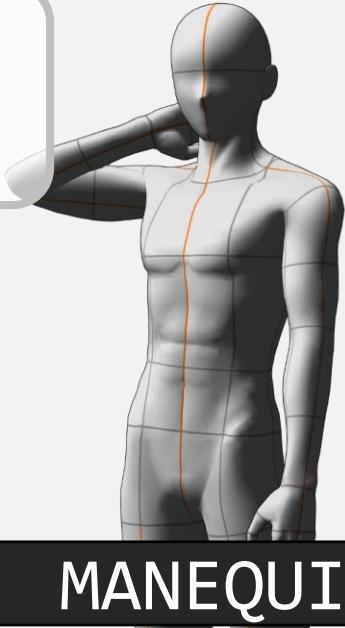
Afinal, o que é um modelo? Falando de uma maneira simples:

Um modelo é uma simplificação da realidade.

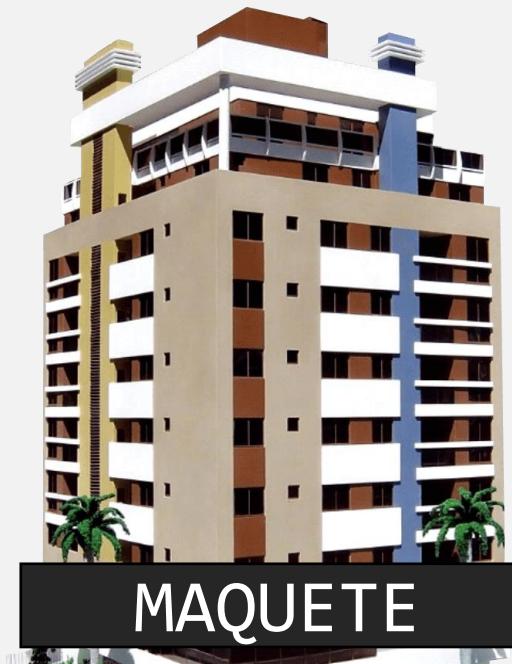
Definição

Linguagem de Modelagem

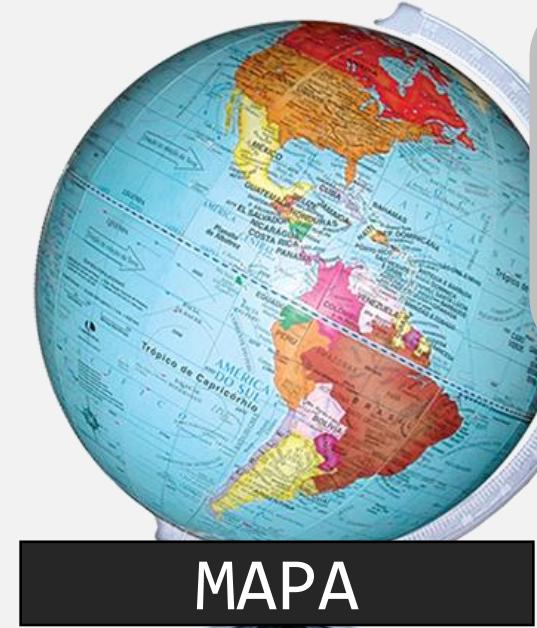
É uma forma de **representar** e **comunicar** as principais características e funcionalidades de determinando objeto



MANEQUIM



MAQUETE



MAPA



História

Antes de 1994 havia uma grande quantidade **de linguagens de modelagem orientadas a objetos**, porém três se destacavam...



Quero
um
padrão!

Tá caro!

Isso tá
uma
bagunça.

É como fazer uma
casa sem conseguir
unir seus projetos.

Essa grande quantidade gerava uma certa **incompatibilidade e interoperabilidade** nos serviços de algumas empresas, então elas estabeleceram um consórcio para tentar resolver isso...

OOSE

Diagrama de
Casos de Uso

Diagrama de
Interações

BOOCH

Diagrama de
Objetos

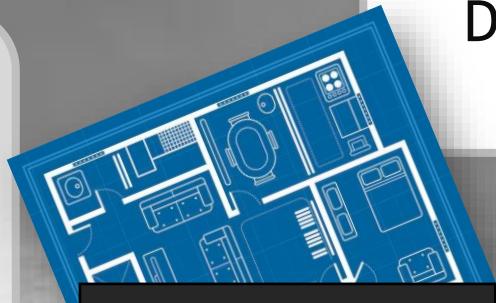
Diagrama de
Processos

Diagrama de
Módulos

OMT

Diagrama de
Estados

Diagrama de
Classes



ESTRUTURA



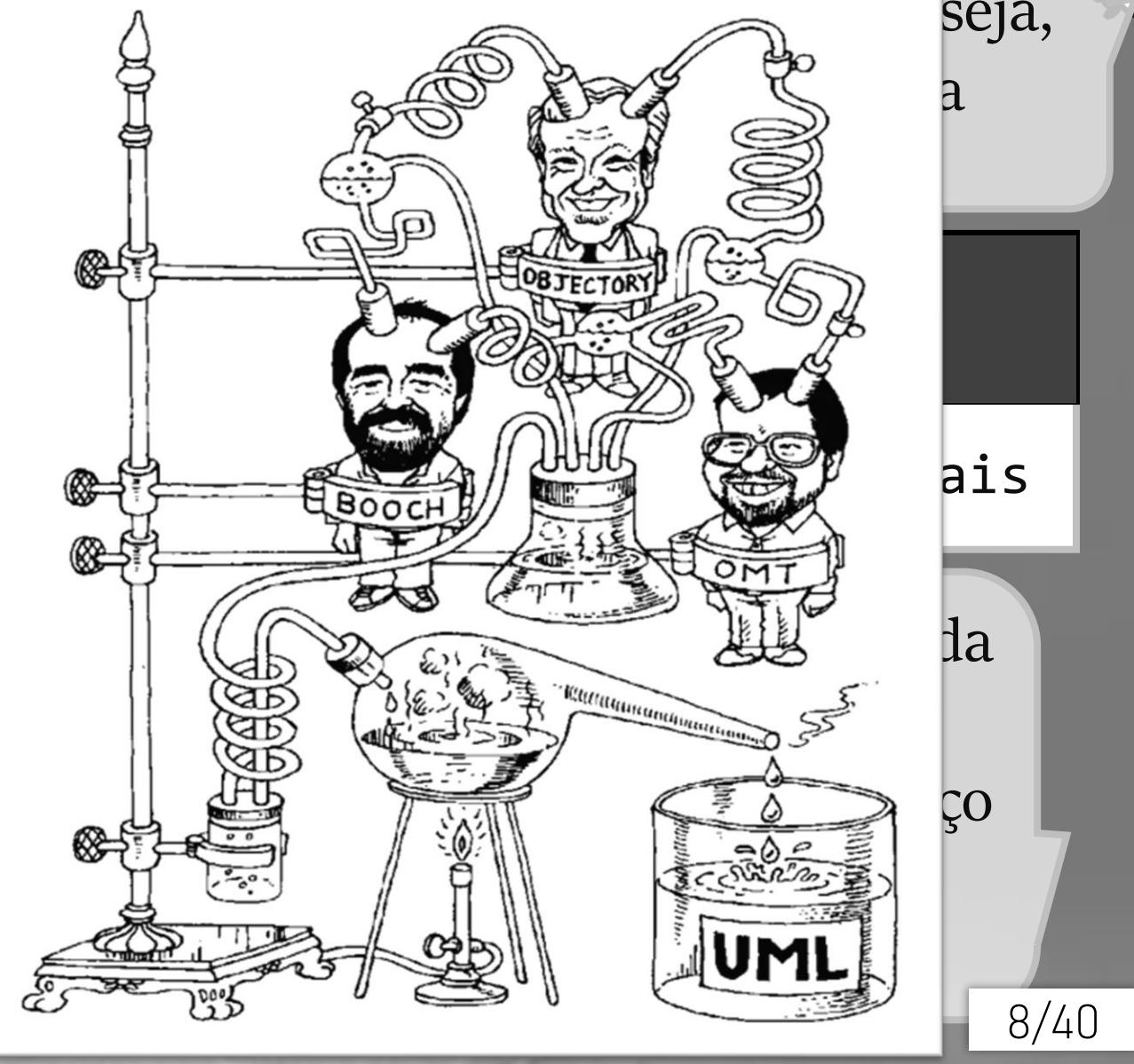
HIDRÁULICA



ELÉTRICA

História

Dá união das três nascia a UML, uma linguagem de modelagem de propósito geral capaz de modelar qualquer **coisa**.



Diagrama

Relaxa, isso é assunto para Engenharia de Software. O que interessa aqui na disciplina de POO são esses [aqui](#).



Ferrou, vou ter que saber tudo isso?

UML

Diagramas Estruturais

Diagrama de Classes

Diagrama de Objetos

Diagrama de Pacotes

Diagrama de Perfil

Diagrama de Componentes

Diagrama de Implantação

Diagrama de Estruturas Compostas

Diagramas Comportamentais

Diagrama de Atividades

Diagrama de Casos de Uso

Diagrama de Interação

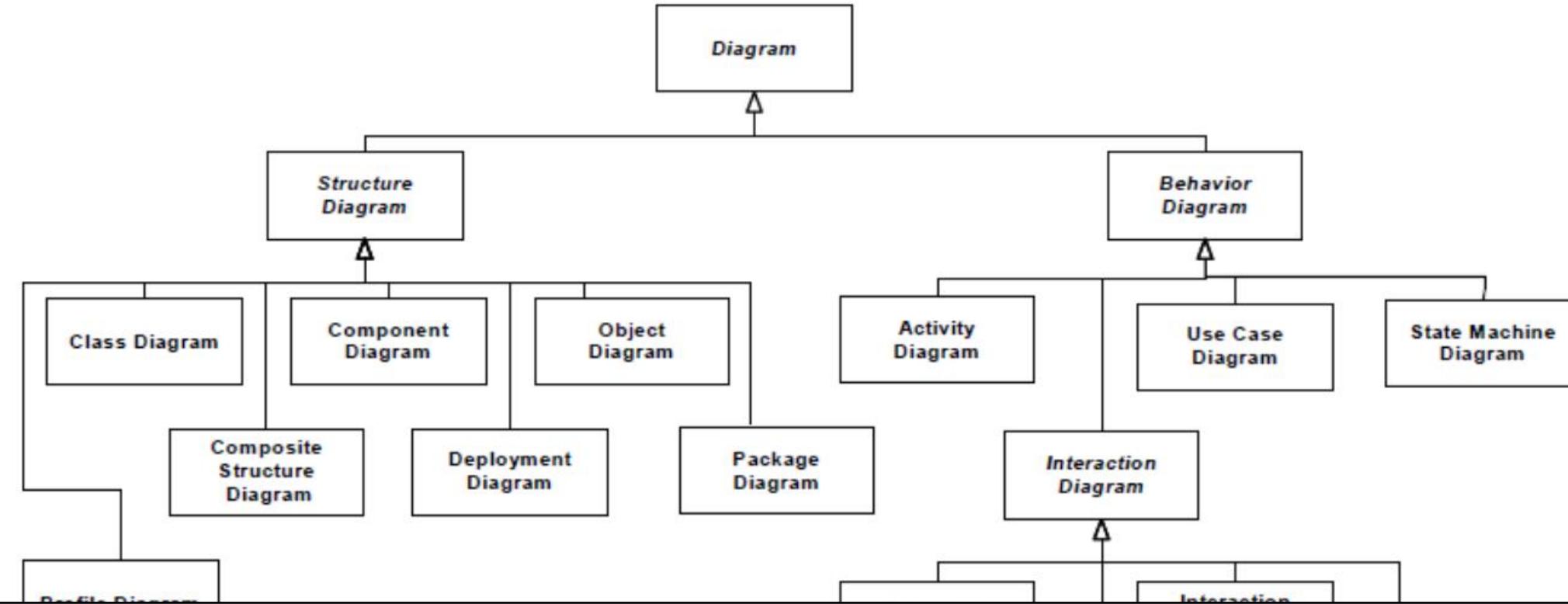
Diagrama de Sequência

Diagrama de Comunicação

Diagrama de Máquina de Estados

Diagrama de Tempo

Diagrama de Visão Geral



DEITEL, Paul. Java: como programar.

1.5.11 A UML (*unified modeling language*)

Embora existam muitos processos OOAD diferentes (*object-oriented analysis and design — OOAD*), uma única linguagem gráfica para comunicar os resultados de *qualquer* processo desse tipo veio a ser amplamente utilizada. A *unified modeling language* (UML) é agora o esquema gráfico mais utilizado para modelagem de sistemas orientados a objetos.

Diagrama

O que é um diagrama?

<https://pt.wikipedia.org/wiki/Diagrama>

Diagrama

文 A 48 línguas ▾

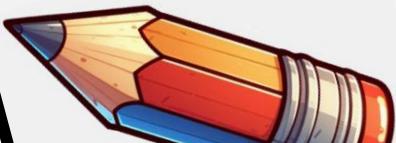
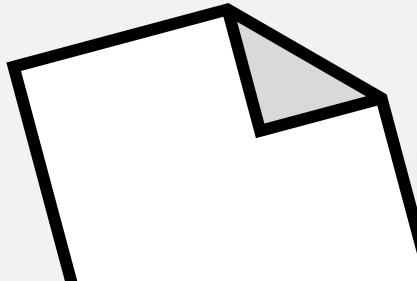
Artigo Discussão

Ler Editar Ver histórico Ferramentas ▾

Origem: Wikipédia, a enciclopédia livre.

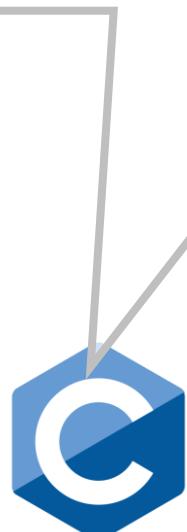
Um **diagrama** é uma representação visual simplificadamente estruturada de um determinado conceito ou ideia, um esquema.

Diagrama

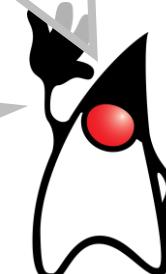


Representação visual?
Fazemos isso no NetBeans?

Não, não... Você pode fazer com
lápis e papel. Pode fazer no
GIMP. Pode fazer no PAINT.



Mas eu recomendo uma
ferramenta UML de diagramas.



Diagrama

https://en.wikipedia.org/wiki/List_of_Unified_Modeling_Language_tools

Ferramenta UML

文 A 5 línguas ▾

Artigo Discussão

Ferramentas ▾

Origem: Wikipédia, a enciclopédia livre.

Na **Engenharia de Software**, uma **ferramenta UML** ou **ferramenta de modelagem UML** é um **software aplicativo** que permite o uso de algumas ou todas as notações e semânticas associadas com a **Linguagem de Modelagem Unificada (UML)**, a qual é uma linguagem de modelagem de propósito geral.

Diagrama

Name	Open source	Software license	Programming language used
ArgoUML	Yes	EPL	Java, C++ (as module)
Astah	No	Commercial. Free education edition, subscription model	Java
ATL	Yes	EPL	Java

Oi, eu sou o **diagrams.net**, conhecido antes como **draw.io**, se quiser dou suporte a modelagem **UML**.



<https://app.diagrams.net/>

The screenshot shows the diagrams.net application interface. On the left is a sidebar with categories like Rascunho, Geral, Diversos, Avançado, Básico, Setas, Diagrama, Relação de entidade, and UML (which is highlighted in green). The main workspace contains three text boxes:

- A top-level text box: "Pensei que essa fosse a disciplina de **PROGRAMAÇÃO** Orientada a Objetos."
- An inner text box: "O paradigma orientado a objetos é um **filosofia de desenvolvimento**. Seu objetivo é **modelar, arquivar ou programar** o mundo real por meio da abstração de seus objetos."
- A right-side text box: "É por isso que a Orientação a Objetos está em: **Linguagens de Modelagem, Banco de Dados e Linguagens de Programação.**"

The interface includes a toolbar at the top with various icons, a status bar at the bottom showing "A4 (210 mm x 297 mm)", and a page number "15/40". A small penguin icon is visible in the bottom right corner.

Modelagem

Um mesmo sistema pode ser modelado (**UML**), programado (**Java**) ou armazenado de forma persistente em um banco de dados (**db4o**). A conversão entre **modelar, programar e armazenar** é mais intuitiva*.

se forem no mesmo paradigma

BOOCH, Grady; RUMBAUGH, James; JACOBSON, Ivar. UML .

A UML não é uma linguagem visual de programação, mas seus modelos podem ser diretamente conectados a várias linguagens de programação. Isso significa que é possível mapear os modelos da UML em linguagens de programação tais como Java, C++, Visual Basic ou até tabelas de bancos de dados relacionais ou o armazenamento de dados persistentes de um banco de dados orientado a objetos. A UML é capaz de representar tudo que possa ser melhor expresso em termos gráficos, enquanto as linguagens de programação representam o que é melhor expresso em termos textuais.

Modelagem

BOOCH, Grady; RUMBAUGH, James; JACOBSON, Ivar. UML.

Construímos modelos para compreender melhor o sistema que estamos desenvolvendo.

Quando eu disse
mais para a
DIREITA era a
MINHA direita.



E eu ia saber que
era a sua direita?

Quero ver falarmos
isso para o chefe:



DESENVOLVIMENTO EQUIPE

Modelar? Eu não preciso disso!
Já **compreendo** o sistema.
O sistema já está todo modelado na
minha cabeça, falta só programar.



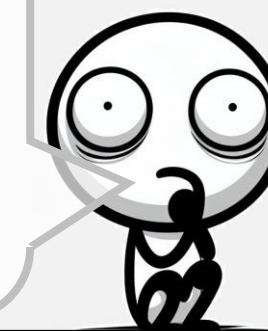
DESENVOLVIMENTO SOLO

Modelagem

BOOCH, Grady; RUMBAUGH, James; JACOBSON, Ivar. UML.

Quanto mais complexo for o sistema, maior será a probabilidade de ocorrência de erros ou de construção de itens errados, caso não haja qualquer modelagem. Existem limites para a capacidade humana de compreender complexidades. Com a ajuda da modelagem, delimitamos o problema que estamos estudando, restringindo nosso foco a um único aspecto por vez.

Fiz todo meu programa no sistema métrico, como eu ia saber que a empresa queria no sistema imperial? *O que eu vou fazer agora? Se eu deixar assim não vai dar ruim né?*



Modelagem

BOOCH, Grady; RUMBAUGH, James; JACOBSON, Ivar. UML.

Quanto mais complexo for o sistema, maior será a probabilidade de ocorrência de erros ou de construção de itens errados, caso não haja qualquer modelagem. Existem limites para a capacidade humana de compreender complexidades. Com a ajuda da modelagem, delimitamos o problema que estamos estudando, restringindo nosso foco a um único aspecto por vez.

<https://www1.folha.uol.com.br/>

AFOGANDO EM NÚMEROS (prejuízo de US\$ 125 mi)

A Mars Climate Orbiter foi destruída ao tentar entrar na órbita de Marte. Ao se aproximar do planeta, a sonda da Nasa recebeu duas informações conflitantes. Uma, no Sistema Métrico, outra, em unidades britânicas.

Modelagem

Montar um “quebra-cabeça” é muito mais fácil quando temos um modelo a ser seguido.



- Auxilia a visualização do sistema
- Ajuda na especificação de estruturas e/ou comportamentos
- Fornece um guia para a construção do sistema
- Trás documentação as decisões tomadas

BOOCH, Grady; RUMBAUGH, James; JACOBSON, Ivar. **UML**.

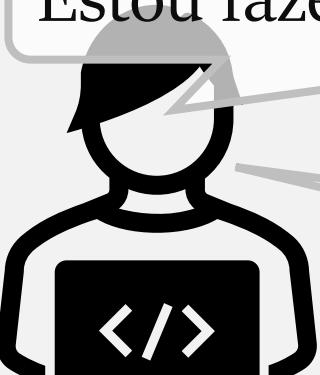
O desenvolvedor poderá rabiscar uma idéia em um quadro-negro ou em uma folha de papel, com a finalidade de visualizar parte do sistema, ou a equipe poderá usar cartões CRC para trabalhar em um cenário específico ou estruturar determinado mecanismo. Não há nada errado nesses modelos. Se funcionarem, sem dúvida deverão ser utilizados.

Modelagem

VANTAGENS

- Diminuição de tempo na implementação
- Problemas já foram encontrados (maioria)
- Dúvidas já foram tiradas antes (maioria)

Estou fazendo a lista de coisas para fazer!



DESVANTAGENS

- Aumento de tempo na modelagem (mais tempo “pensando” em fazer o projeto)

O que você está fazendo corpo?

Sabe que fazer a lista não é fazer as coisas, né?

Depois eu faço!

Depois nunca faz...

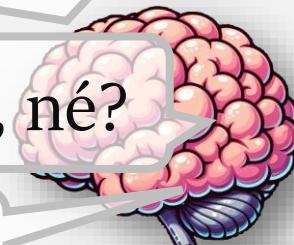


Diagrama de Classes

Objetivo: Modelar as **classes** de um sistema e seus **relacionamentos**

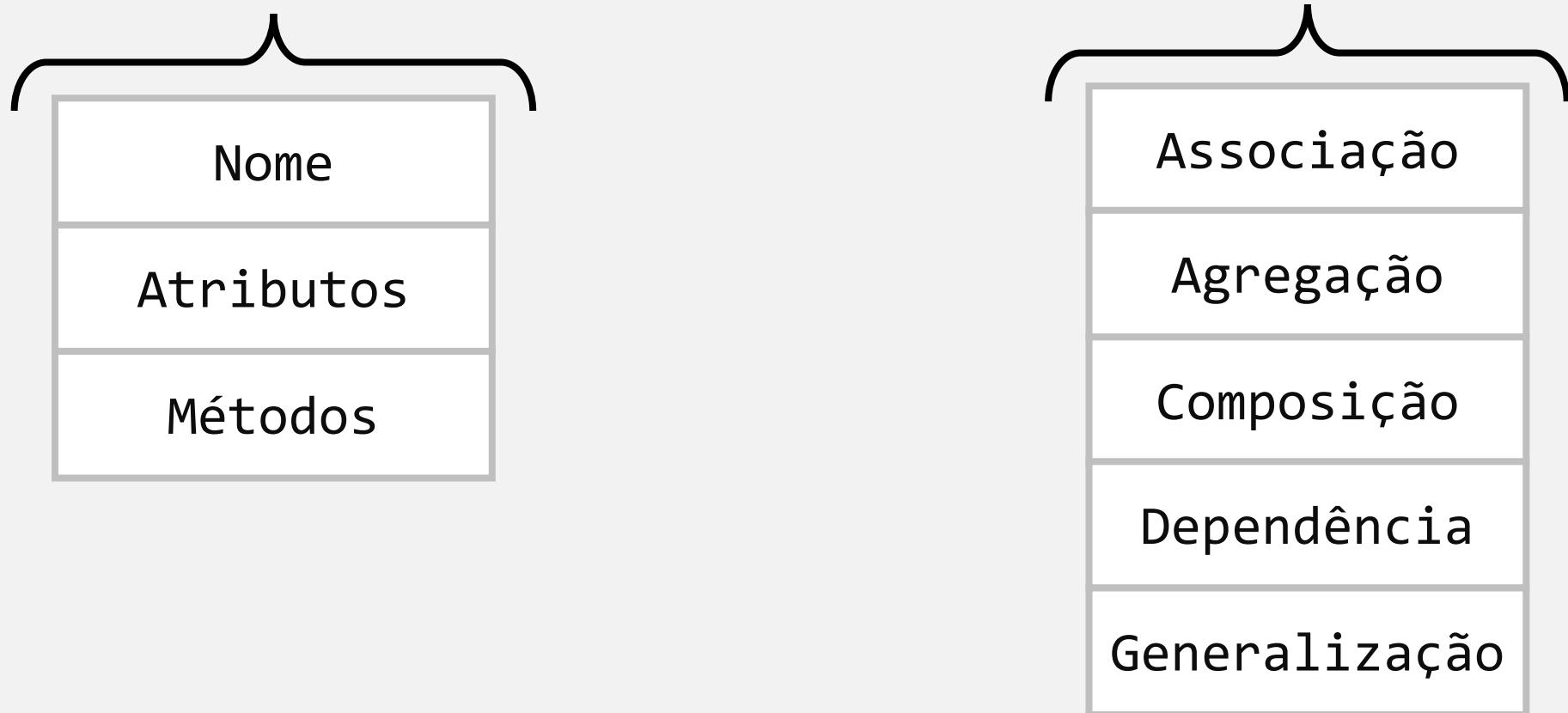


Diagrama de Classes

Uma **classe** é representada na notação de um retângulo dividido em três seções horizontalmente.

A seção superior comporta o nome.
A seção central comporta os **atributos**.
A seção inferior comporta os **métodos**.



São as características
da classe (adjetivo)

São as atividades
da classe (verbo)

Diagrama de Classes

Código (Carro.java)

```
1 public class Carro{  
2     char cor;  
3     int tanque;  
4     int velocidade;  
5  
6     void acelerar(){  
7         velocidade++;  
8     }  
9     void freiar(){  
10        velocidade--;  
11    }  
12}  
13}
```

Convenções:

- ✓ Nome: Negrito
- ✓ Nome: Maiúscula
- ✓ Atributo: Minúscula
- ✓ Método: Minúscula
- ✓ Método: Parêntese

Carro

cor
tanque
velocidade

acelerar()
freiar()

Diagrama de Classes

visibilidade

(modificador de acesso)

(operadores)

Um modificador de visibilidade são palavras-chaves que sinalizam o nível de acesso aos atributos e métodos internos de uma determinada classe

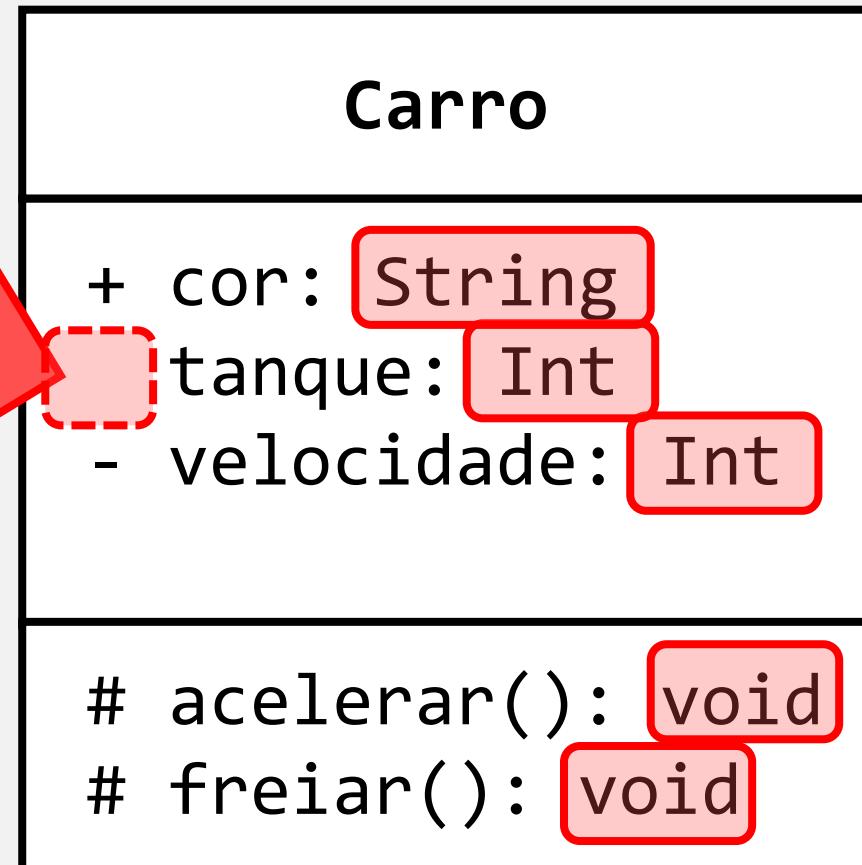
UML	JAVA	Visibilidade
-	private	Permite acesso apenas dentro da própria classe
+	public	Permite acesso a partir de qualquer classe
#	protected	Permite acesso da própria classe e suas subclasses
~	package	Permite acesso apenas pelas classes no mesmo pacote

Diagrama de Classes

Código (Carro.java)

```
1 public class Carro{  
2     public char cor;  
3     private int tanque;  
4     private int velocidade;  
5  
6     protected void acelerar(){  
7         velocidade++;  
8     }  
9     protected void freiar(){  
10        velocidade--;
```

*Na ausência, assume-se
com visibilidade privada*



A UML não define um conjunto específico de tipos primitivos, mas alguns dos utilizados são: Boolean, Int, String, Real, Date, Time

Diagrama de Classes

Um **relacionamento** representa a relação entre classes

Associação



Agregação



Composição



Dependência



Generalização



Chamamos a ponta da seta de “**navegabilidade**”, serve para identificar o sentido que as informações são transmitidas entre os objetos das classes relacionadas.

A classe 1 pode executar métodos da classe 2, mas não o contrário.

Classe 1

...

...



Classe 2

...

...

Diagrama de Classes

Um **relacionamento** representa a relação entre classes

Associação: Objetos de uma classe estão vinculados (associados) a objetos de outra classe

Agregação: Objetos agregados a uma classe que podem existir independente dela

Composição: Objetos que compõe uma classe e não podem existir sem ela

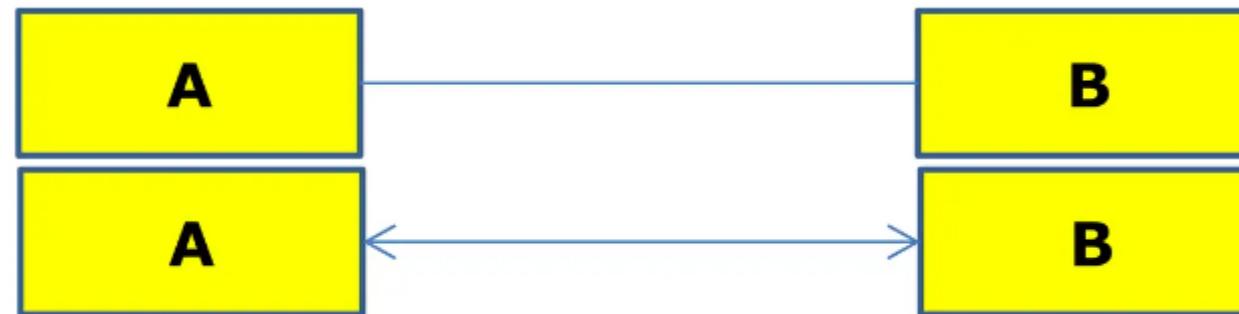
Dependência: Objetos de uma classe dependem de outra classe para funcionarem

Especialização: Objetos de uma classe herdam características de outra classe

Navegabilidade

Navegabilidade indica a direcionalidade com que as classes se relacionam

- A e B se conhecem mutuamente



- B não sabe da existência de A



- A não sabe da existência de B



Diagrama de Classes

multiplicidade

(cardinalidade)

A multiplicidade especifica o número mínimo e máximo de objetos que podem estar envolvidos em um relacionamento

UML	Multiplicidade
0..1	Zero ou um
0..*	Zero ou mais (indeterminado)
1	Apenas um
1..*	Um ou mais (indeterminado)
*	Zero ou mais (indeterminado)
n	Único valor (determinado)
m..n	Faixa de valores (determinado)

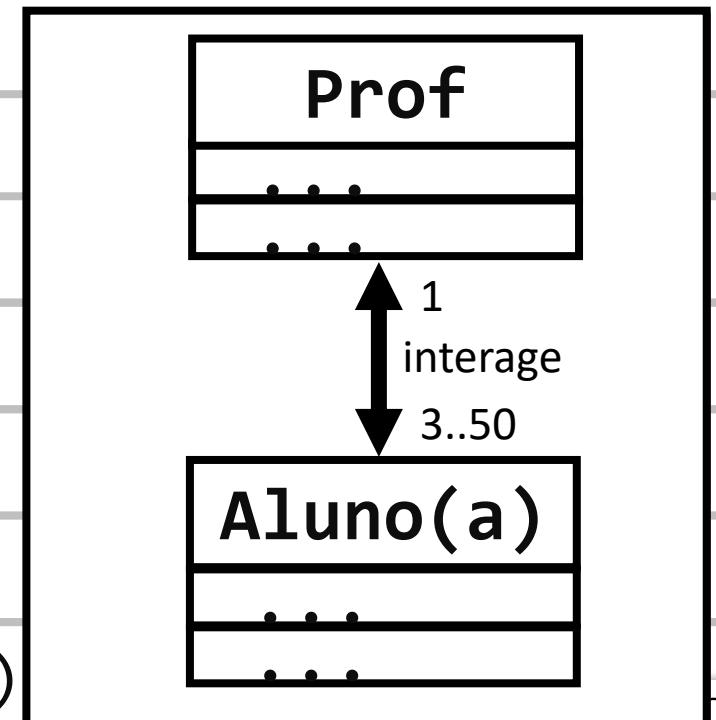


Diagrama de Classes

multiplicidade

0..1	No máximo um. Indica que os objetos da classe associada não precisam obrigatoriamente estar relacionados.
1..1	Um e somente um. Indica que apenas um objeto da classe se relaciona com os objetos da outra classe.
0..*	Muitos. Indica que podem haver muitos objetos da classe envolvidos no relacionamento
1..*	Um ou muitos. Indica que há pelo menos um objeto envolvido no relacionamento.
3..5	Valores específicos.

- Utilizando as seguintes notações a
- **0..1**: nenhuma ou uma;
- **0..***: nenhuma ou muitas;
- **1**: apenas uma;
- **1..***: uma ou muitas;
- *****: muitas (indeterminado);
- **n**: muitas (determinado);

- **1** Exatamente um
- **1..*** Um ou mais
- **0..*** Zero ou mais (muitos)
- ***** Zero ou mais (muitos)
- **0..1** Zero ou um
- **m..n** Faixa de valores (por exemplo: 4..7)

Diagrama de Classes

multiplicidade

Multiplicidade (UML)	Exemplos
$*..1$	Uma mesa de restaurante pode ter vários ou nenhum pedido
$1..*$	Uma cotação pode incluir no mínimo 1 e até muitos itens cotados
$0..3$	Um time pode ter presente 0 a 3 funcionários

Diagrama de Classes

associação

A associação indica que uma classe mantém uma referência a outra classe ao longo do tempo

Classe A tem uma classe B



A relação pode ser nomeada auxiliando na semântica da relação

Uma pessoa pode estar inscrita em nenhum ou muitos canais. Um canal pode ter nenhuma ou muitas pessoas inscritas. Ambas as classes existem independentemente, porém as “inscrição” precisa de uma pessoa e de um canal.

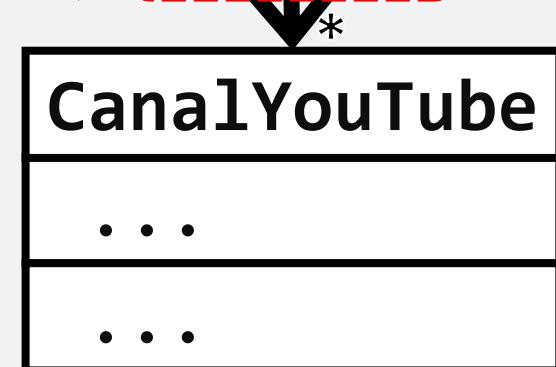


Diagrama de Classes

associação^{ternária}

A associação ternária conecta três ou vários objetos estabelecendo relações entre eles

Classe A, classe B e classe C tem uma relação em conjunto

Um professor leciona para um ou vários alunos em uma ou várias salas.

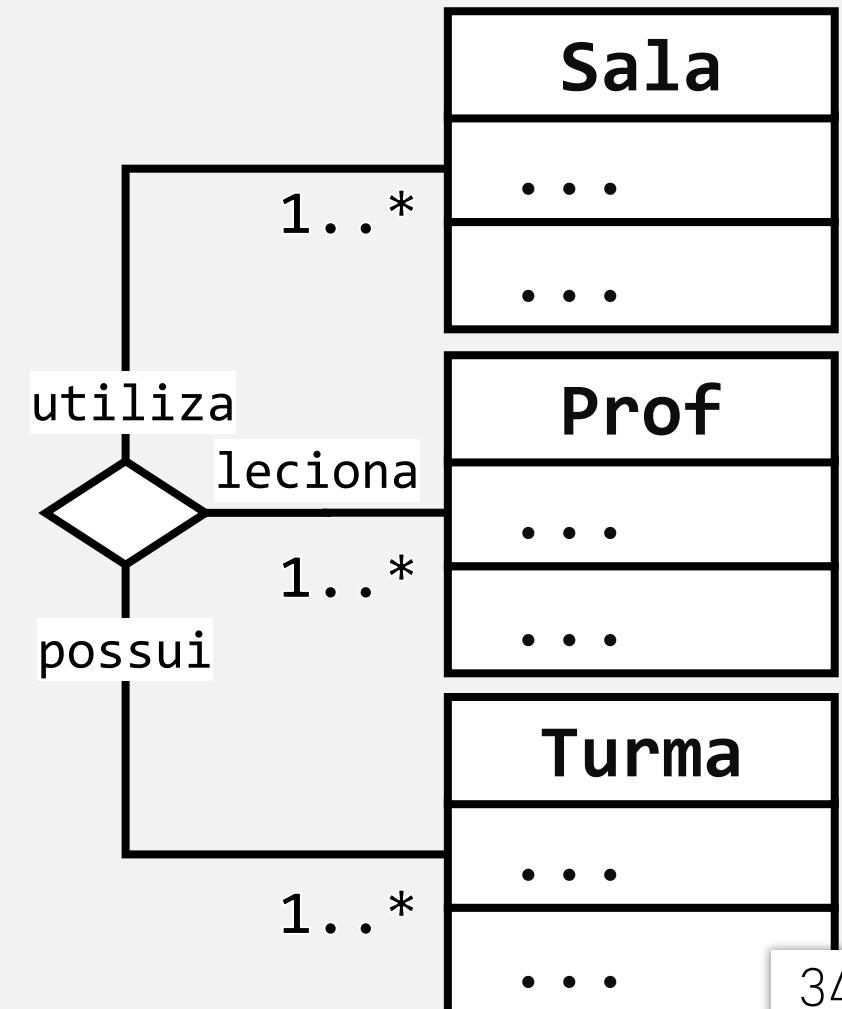


Diagrama de Classes

dependência

A dependência indica que uma classe usa informações (atributos) ou serviços (métodos) de outra classe em um determinado momento

Classe A depende da classe B



Um carro pode existir sem roda, mas para que ele consiga exercer todas as funções (métodos) ele precisa da roda

classe usa informações de outra classe em um

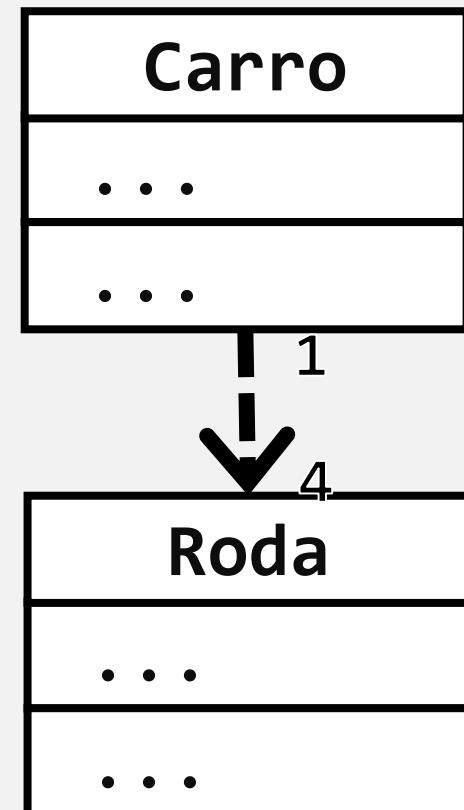


Diagrama de Classes

agregação

A agregação é um tipo especial de associação no qual uma classe é criada por outra classe.

Classe A possui uma classe B



Um pedido possui um ou vários itens e um item DEVE pertencer a um pedido

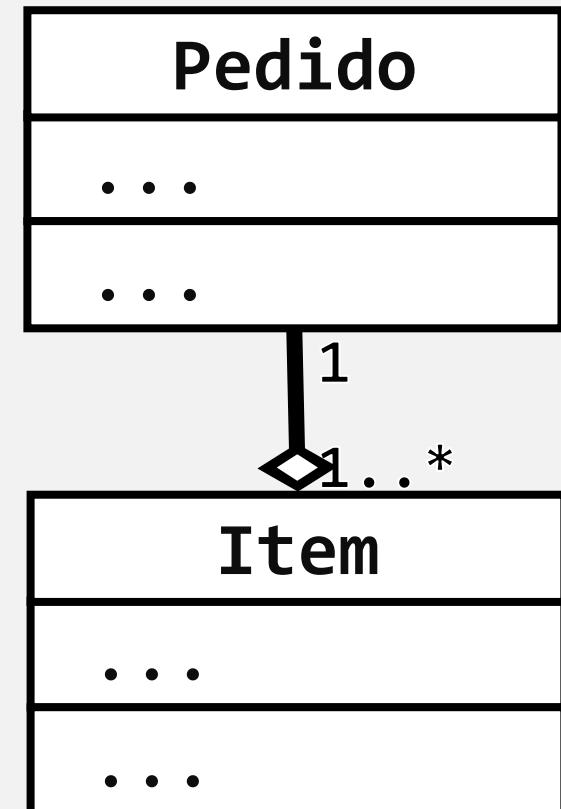


Diagrama de Classes

composição

A **composição** indica uma dependência de uma classe com outra classe.

Classe A é parte da classe B



Se eu fechar meu navegador a aba também fecha. A aba não existe sem o navegador, ela tem uma forte dependência com ele.

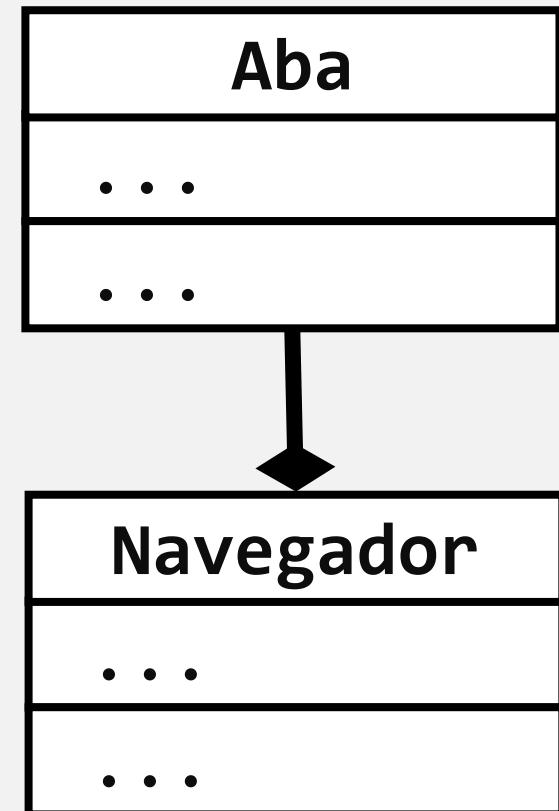


Diagrama de Classes

generalização

É um relacionamento entre instâncias gerais (superclasses) instâncias mais específicas (subclasses).

Classe A herda da classe B



Todo carro é um veículo, mas nem todo o veículo é um carro.
O carro herda atributos e métodos da classe veículo.

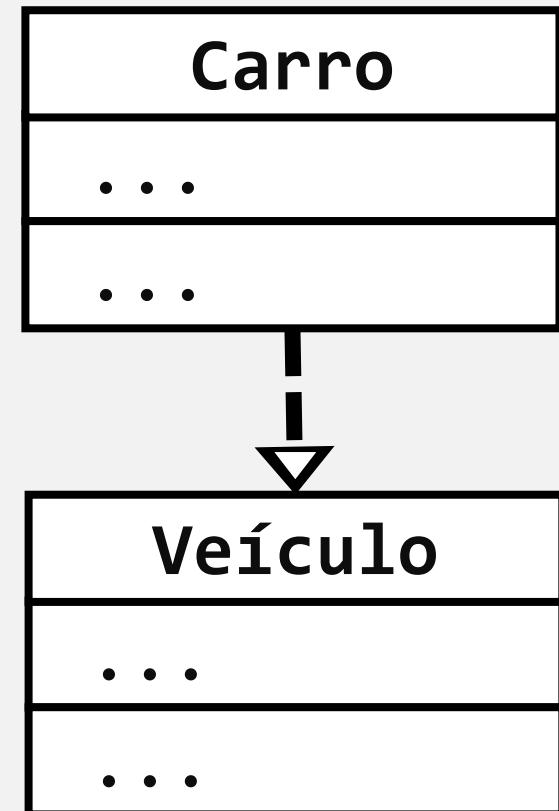
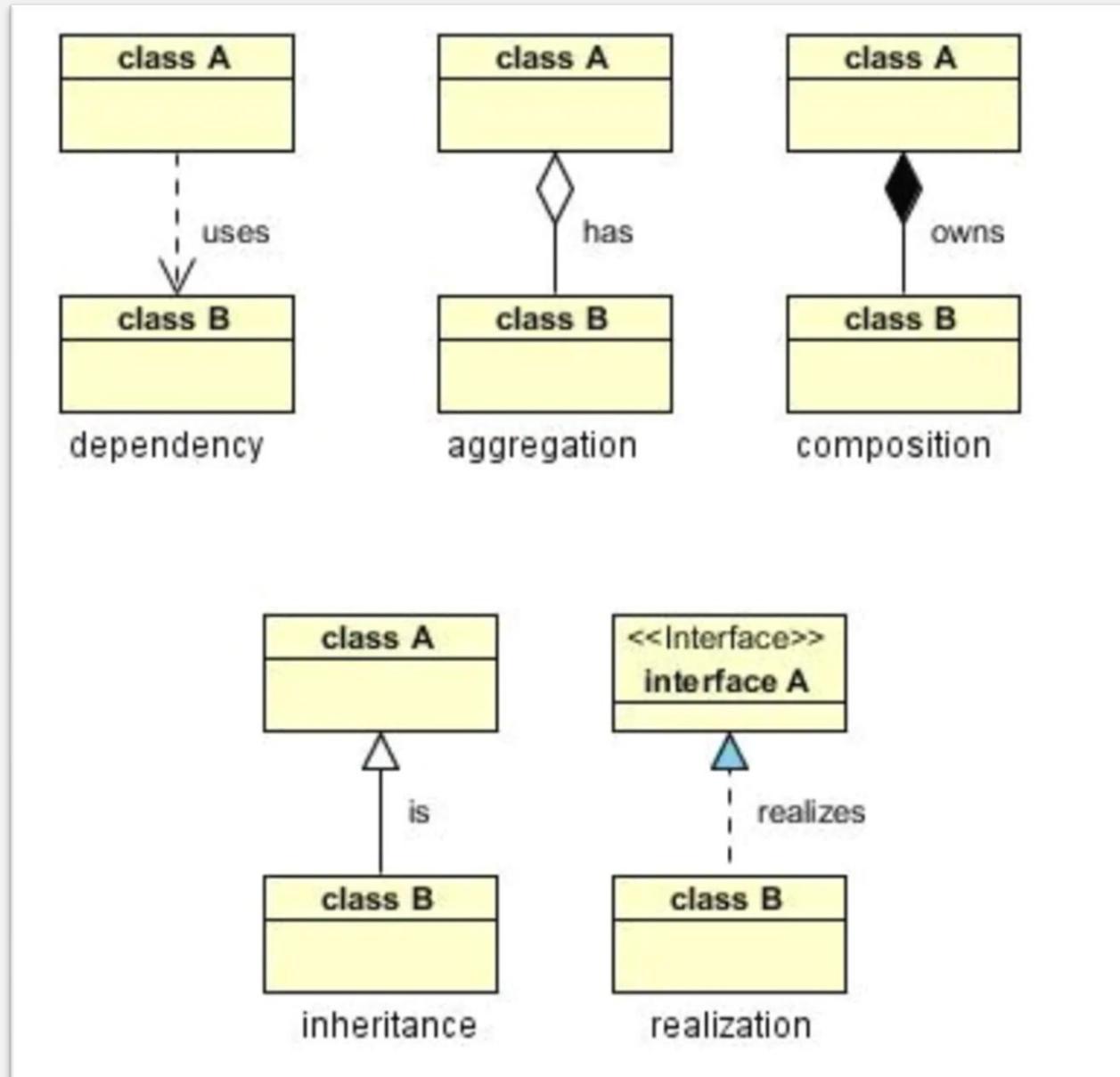


Diagrama de Classes

RESUMO



Resumo

- **Modelo:** Representação abstrata de um sistema, considerando suas características e relações.
- **Diagrama:** Representação gráfica de um sistema, considerando suas características e relações.
- **UML:** Linguagem padrão para modelagem de sistemas orientados a objetos.
- **Visibilidade:** Remete o nível de acesso que um elemento tem em relação a outros elementos, podendo ser privado, público, derivado, estático, etc...
- **Multiplicidade:** Indica quantos objetos podem participar de uma relação entre classes.
- **Relacionamento:** Descreve a conexão entre classes do sistema, podendo ser associação, dependência, agregação, composição, etc...

Referências

SCHILDT, Herbert. **Java para iniciantes: crie, compile e execute.** 6. ed. Porto Alegre: Bookman, 2015. Disponível em: <<https://app.minhabiblioteca.com.br/reader/books/9788582603376>>.

SILVA, Fabricio; LEITE, Márcia; OLIVEIRA, Diego. **Paradigmas de Programação.** Porto Alegre: SAGAH, 2019. Disponível em: <<https://app.minhabiblioteca.com.br/reader/books/9788533500426>>.

DEITEL, Paul. **Java: como programar.** 4. ed. Porto Alegre: Bookman, 2003.

DEITEL, Paul. **Java: como programar.** 10. ed. São Paulo: Pearson Education do Brasil, 2017.

Programação Orientada a Objetos – (POO0001)

UML

Alexandre Mendonça Fava
alexandre.fava@udesc.br

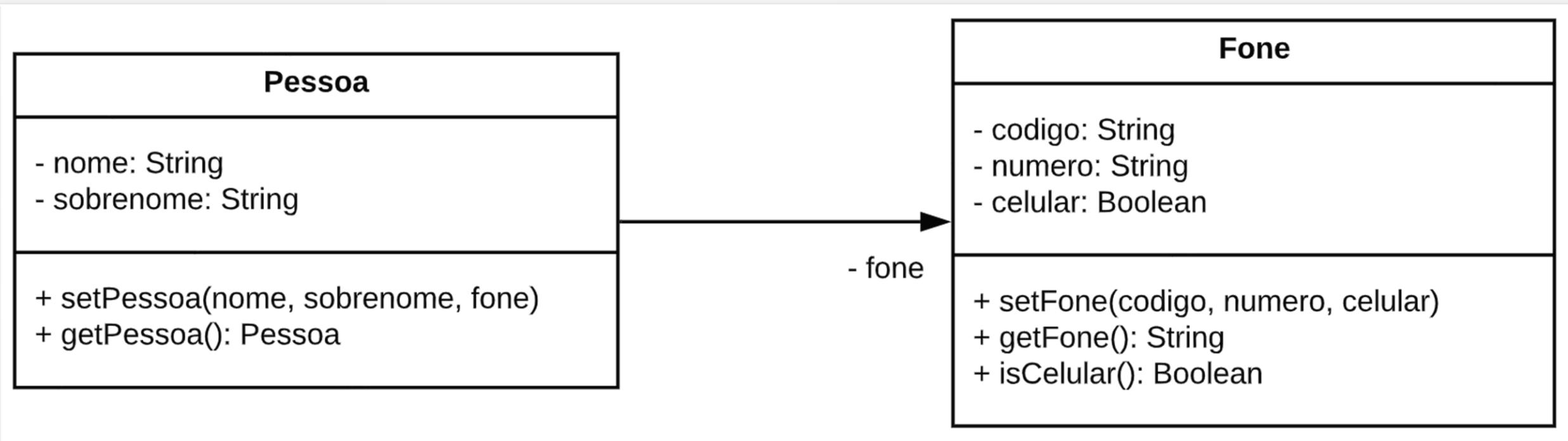
Universidade do Estado de Santa Catarina – UDESC
Programa de Pós-graduação em Computação Aplicada – PPGCA

Outros Exemplos (Atributos e Métodos)

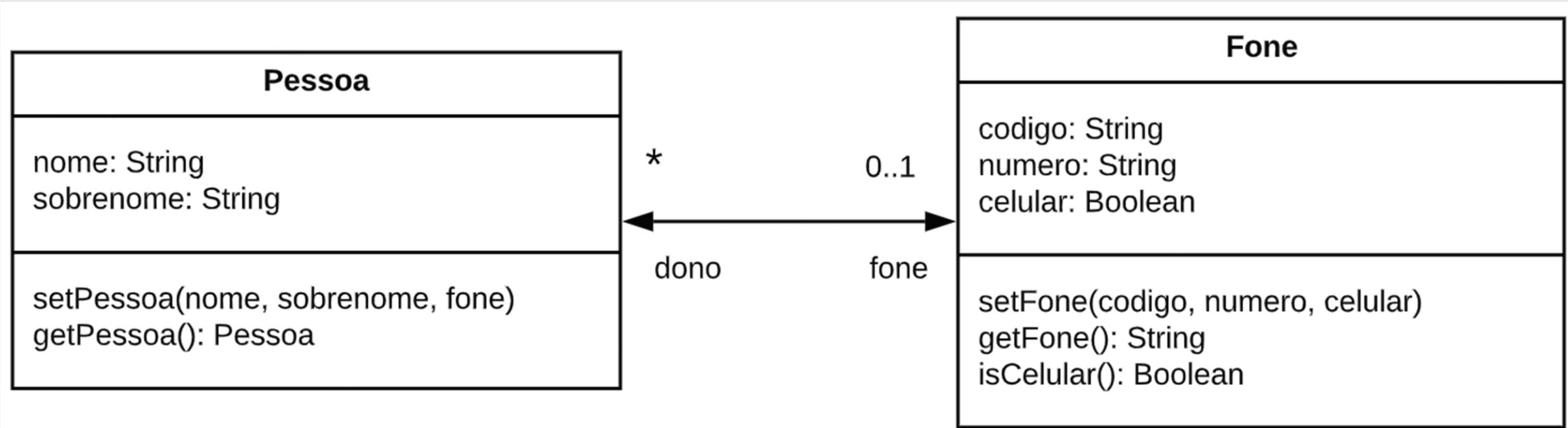
Pessoa
- nome: String - sobrenome: String - fone: Fone
+ setPessoa(nome, sobrenome, fone) + getPessoa(): Pessoa

Fone
- codigo: String - numero: String - celular: Boolean
+ setFone(codigo, numero, celular) + getFone(): String + isCelular(): Boolean

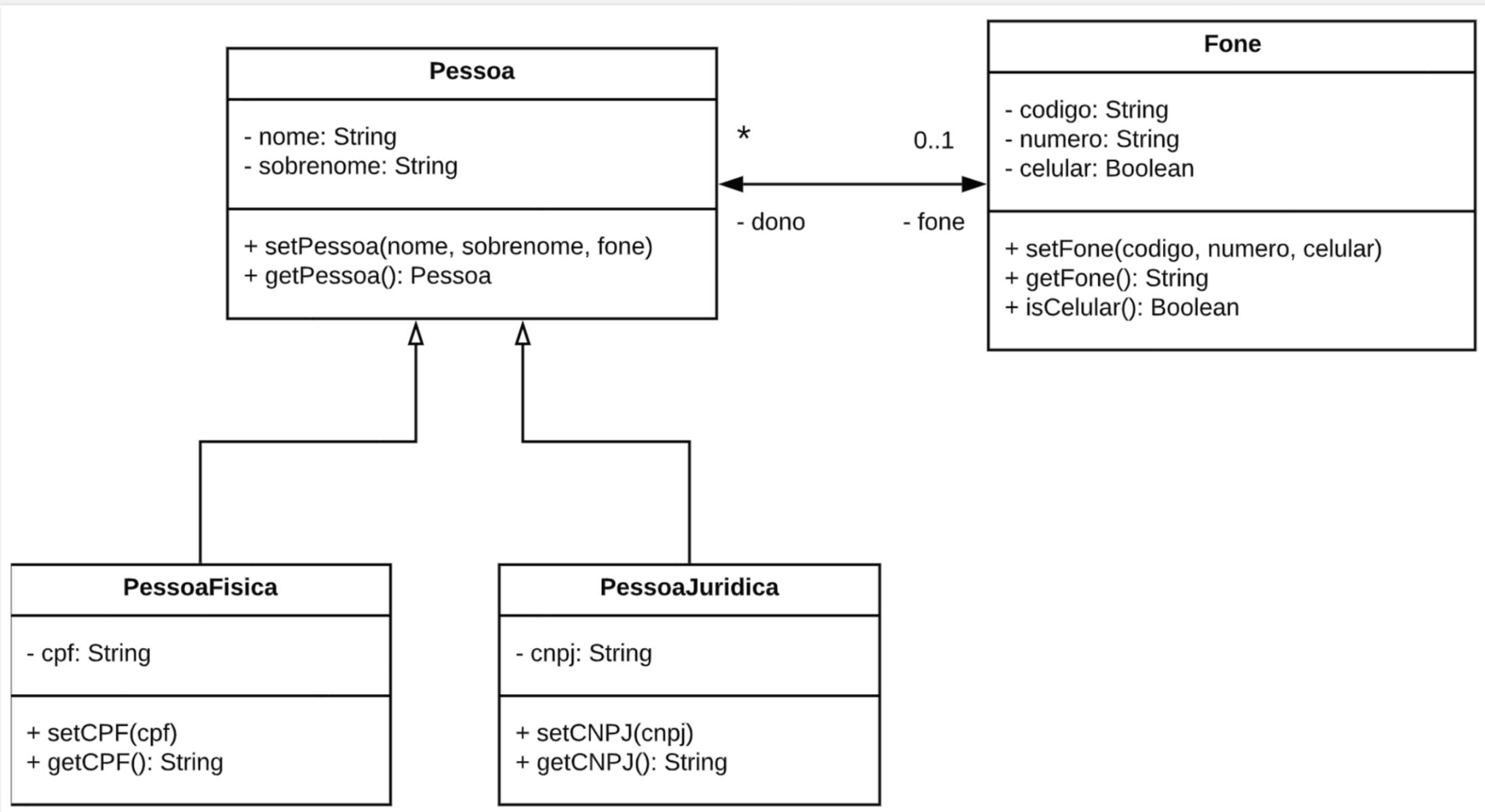
Outros Exemplos (Associação)



Outros Exemplos (Associação Bidirecional)



Outros Exemplos (Herança)



Outros Exemplos (Associação)

Proprietário

nome

endereço

telefone

consultar

incluir

possui

Automóvel

proprietário

marca

placa

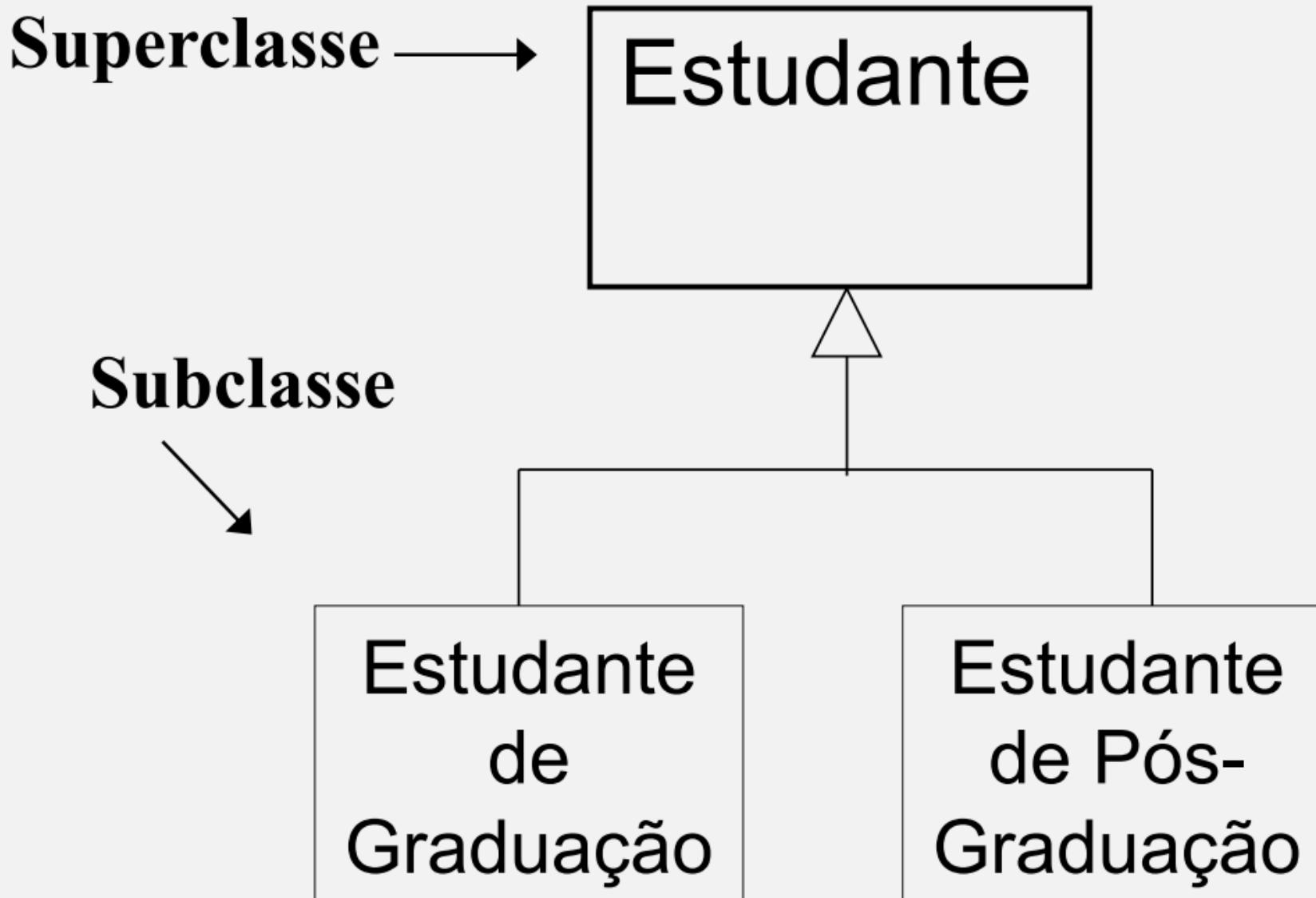
ano

registrar

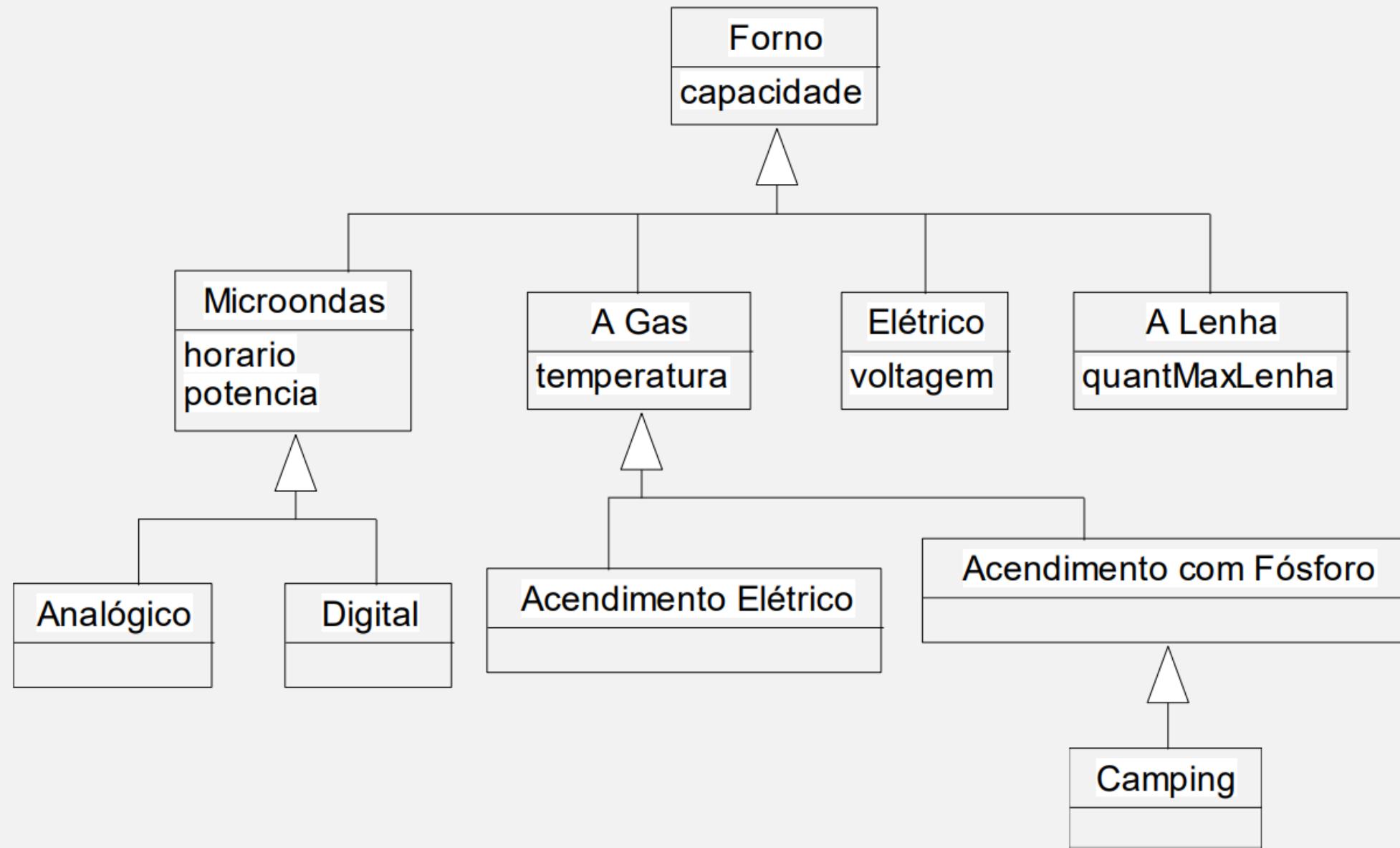
transferir_Proprietário

mudar_Placa

Outros Exemplos (Herança)

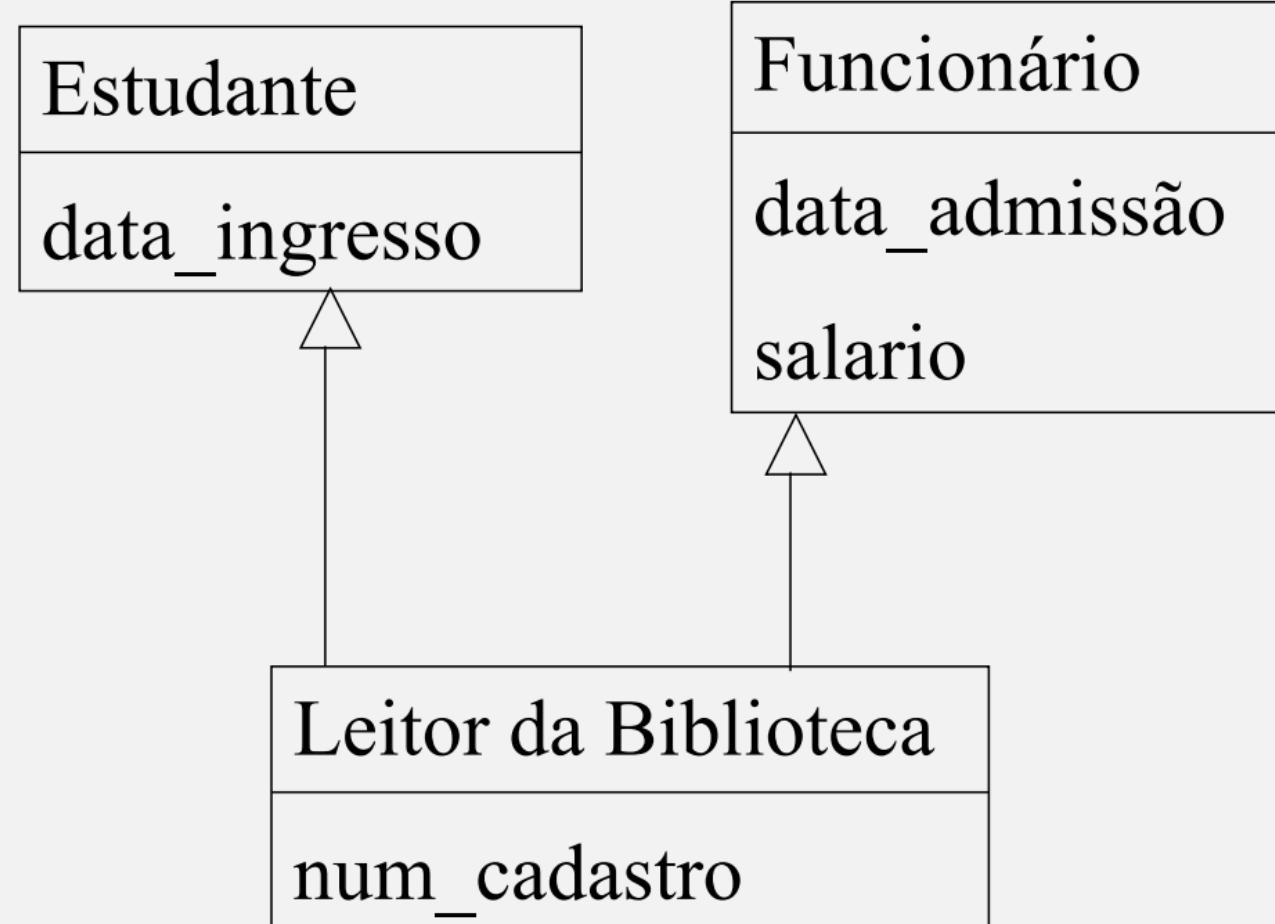


Outros Exemplos (Herança)



Outros Exemplos (Herança Múltipla)

Na herança múltipla existe mais de uma superclasse, ou seja, uma classe é declarada como uma subclasse de uma ou mais superclasses.



Exercícios