

Modelagem de Classes

Disciplina Engenharia de Software

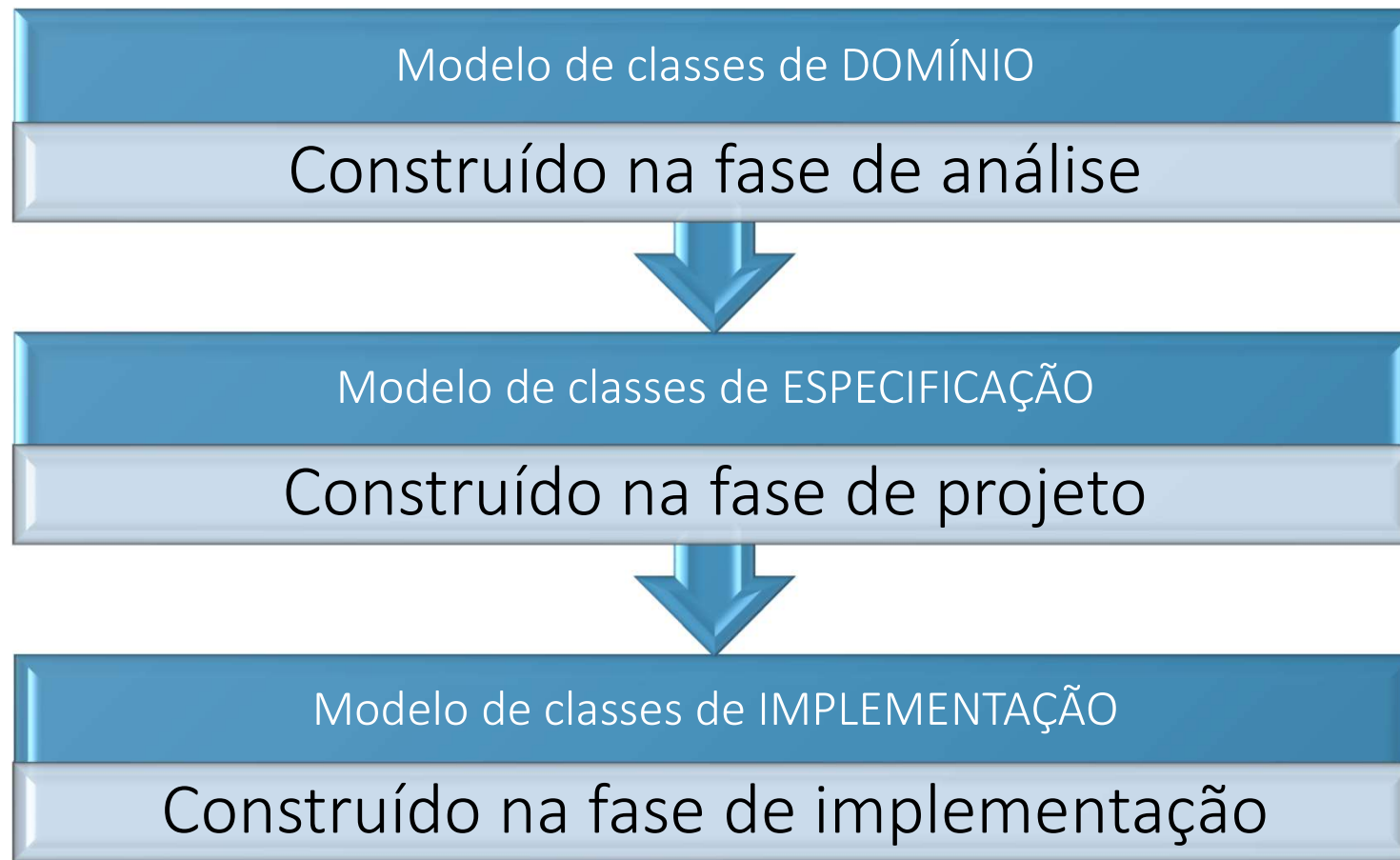
Professora Adriana Gomes Alves, Dra

adriana.alves@univali.br



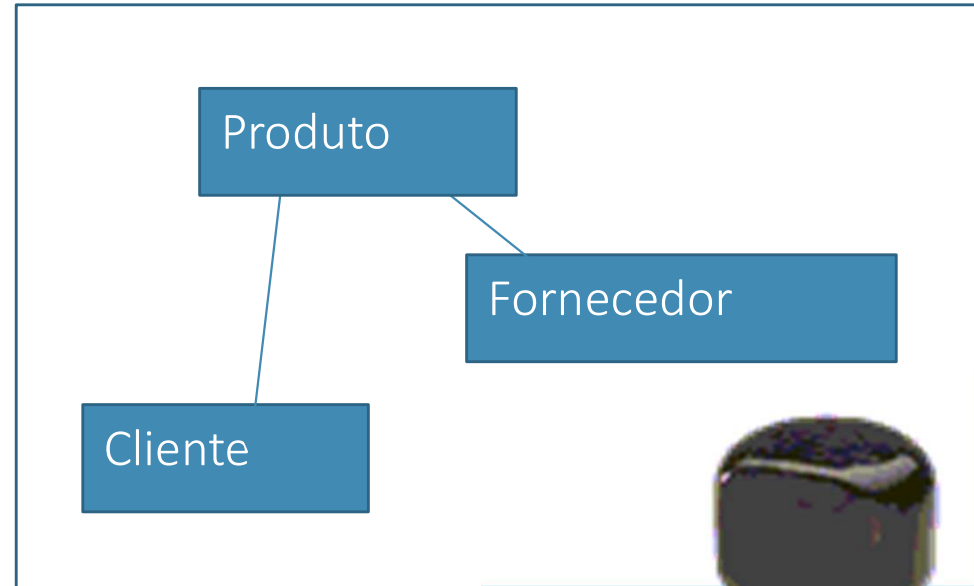
Represente
conceitos
relacionados a
uma
universidade...

Modelagem de classes

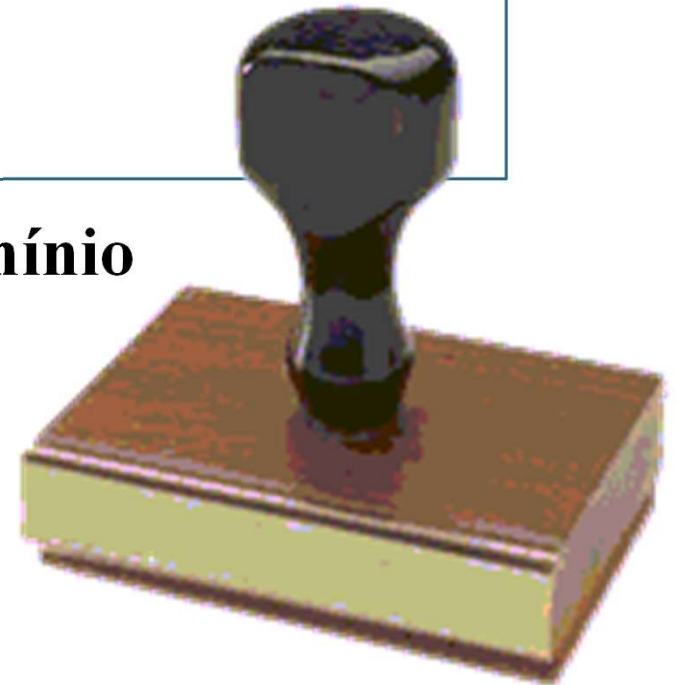




Negócio



Domínio

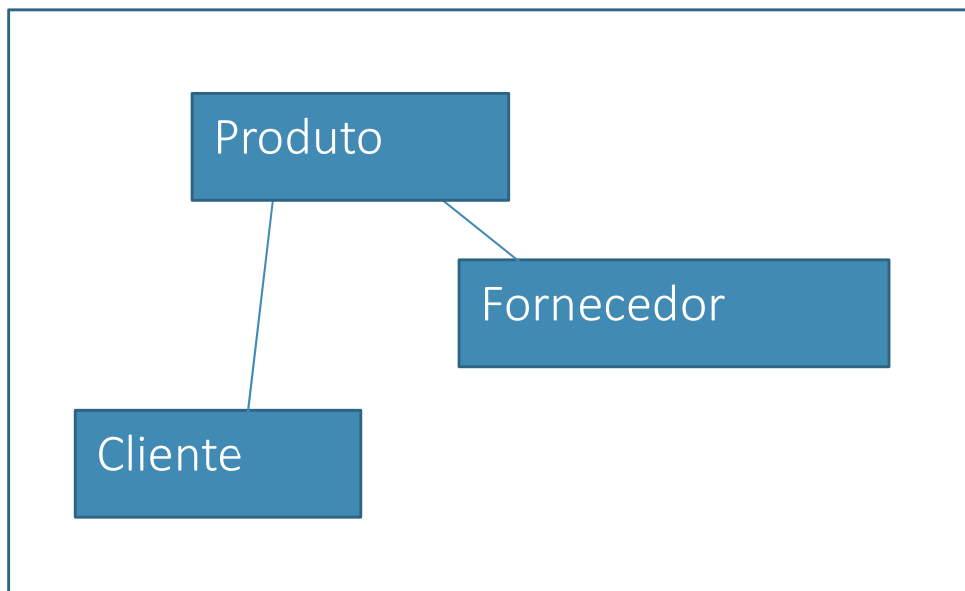


Modelo de Classes de Domínio

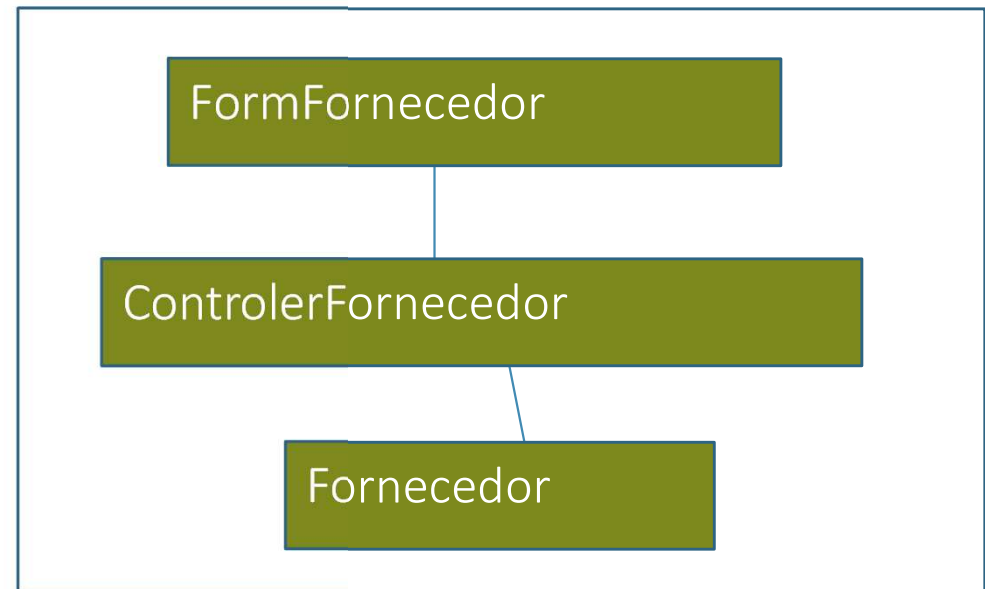
Construído na fase de análise de sistema

Representam o domínio do negócio em questão, de forma a compreender os objetos e relacionamentos envolvidos no sistema.

Nessa fase, não devemos nos preocupar com a tecnologia que será adotada.



Domínio



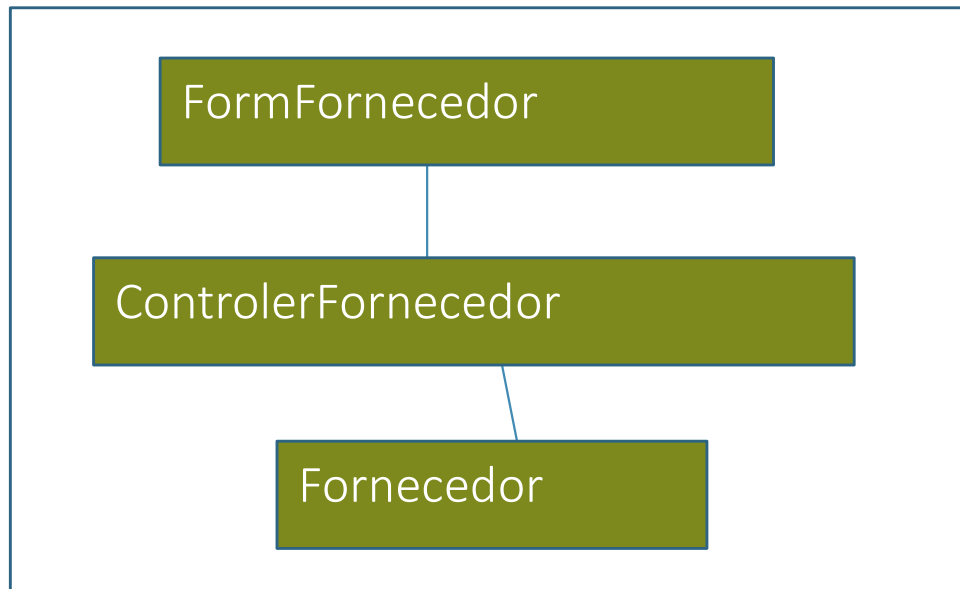
Especificação



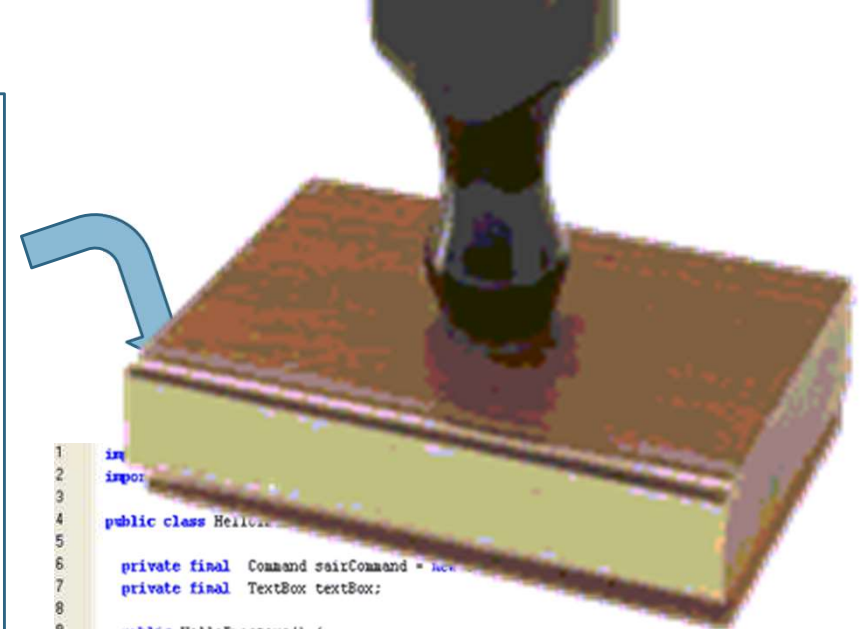
Modelo de Classes de Especificação

É uma extensão do modelo de classes de domínio, elaborado na fase de projeto do sistema.

Esta extensão é feita através da adição de novas classes e operações conforme a solução de software escolhida.



Especificação



```
1  import java.awt.*;
2  import java.awt.event.*;
3
4  public class HelloMasters {
5
6      private final Command sairCommand = new SairCommand();
7      private final TextBox textBox;
8
9      public HelloMasters() {
10         textBox = new TextBox("Titulo do TextBox", "HelloMasters!!!", 50, TextField.ANY);
11         textBox.addCommand(sairCommand);
12         textBox.setCommandListener(this);
13     }
14
15     public void startApp() {
16         Displayable telaAtual = Display.getDisplay(this).getCurrent();
17         if(telaAtual == null) {
18             Display.getDisplay(this).setCurrent(textBox);
19         }
20     }
21
22     public void pauseApp() { }
23
24     public void destroyApp(boolean b) { }
25
26     void sair() {
27         destroyApp(false);
28         notifyDestroyed();
29     }
30
31     public void commandAction(Command c, Displayable d) {
32         if (c == sairCommand) {
33             sair();
34         }
35     }
36 }
```

Implementação



Modelo de Classes de Implementação

É uma extensão do modelo de especificação, construído na fase de implementação do sistema.

Utiliza uma linguagem de programação orientada a objetos.

Diagrama de classes

Para que serve?

Um diagrama de classes:

- Dá uma visão estática do sistema, válida por todo o sistema

- Exibe um conjunto de classes, interfaces e seus relacionamentos

- As classes especificam tanto a estrutura como o comportamento dos objetos (que são instâncias de classes)

Classes, associações e atributos

Interfaces, incluindo métodos e constantes

Métodos

Informação de tipo de atributos

Navegabilidade

Dependências

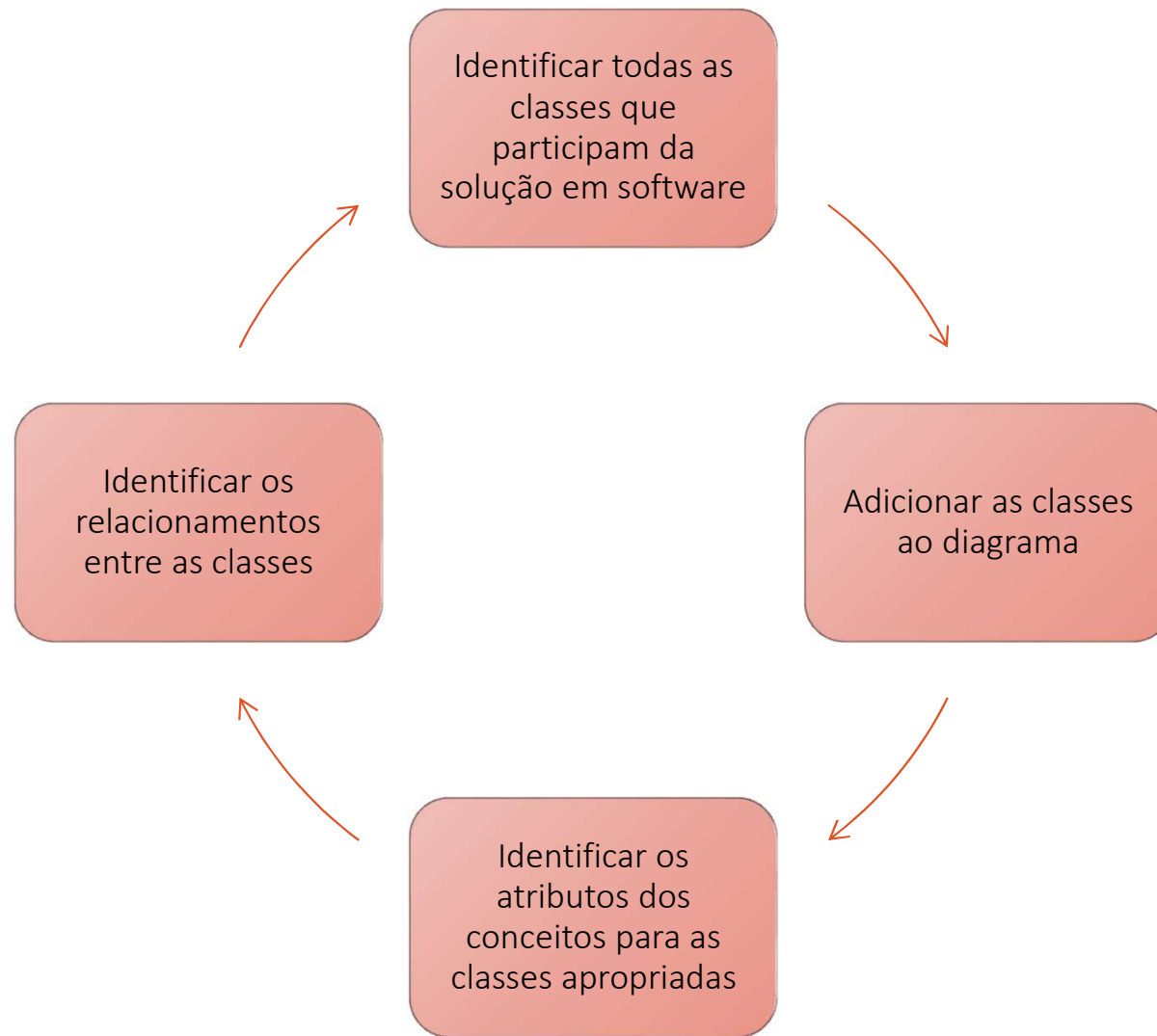
O que
encontramos
no diagrama
de classes

Diagrama de classes



Modelo Conceitual x Diagrama de Classes

Os modelos conceituais são feitos usando diagramas de classes, onde as classes são conceitos do domínio do problema e não classes de software



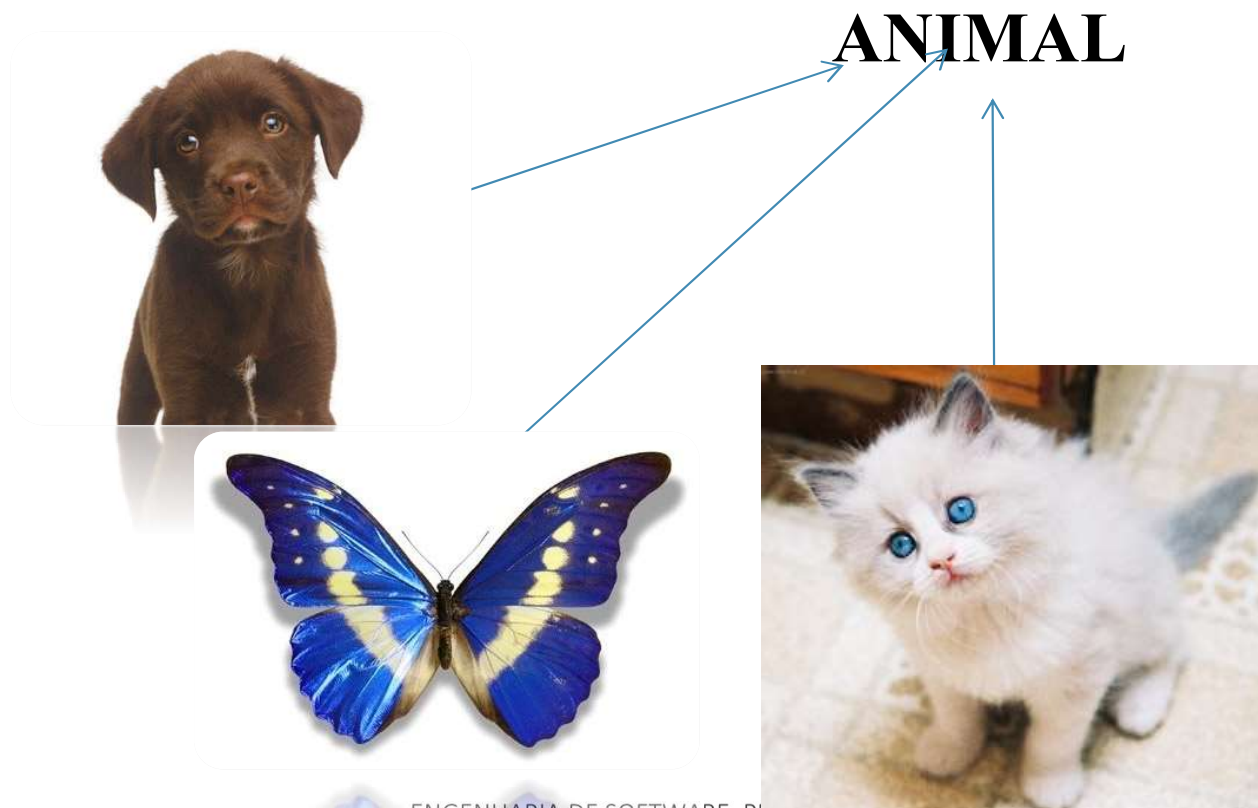
Construindo um Diagrama de Classes

Descobrendo as classes

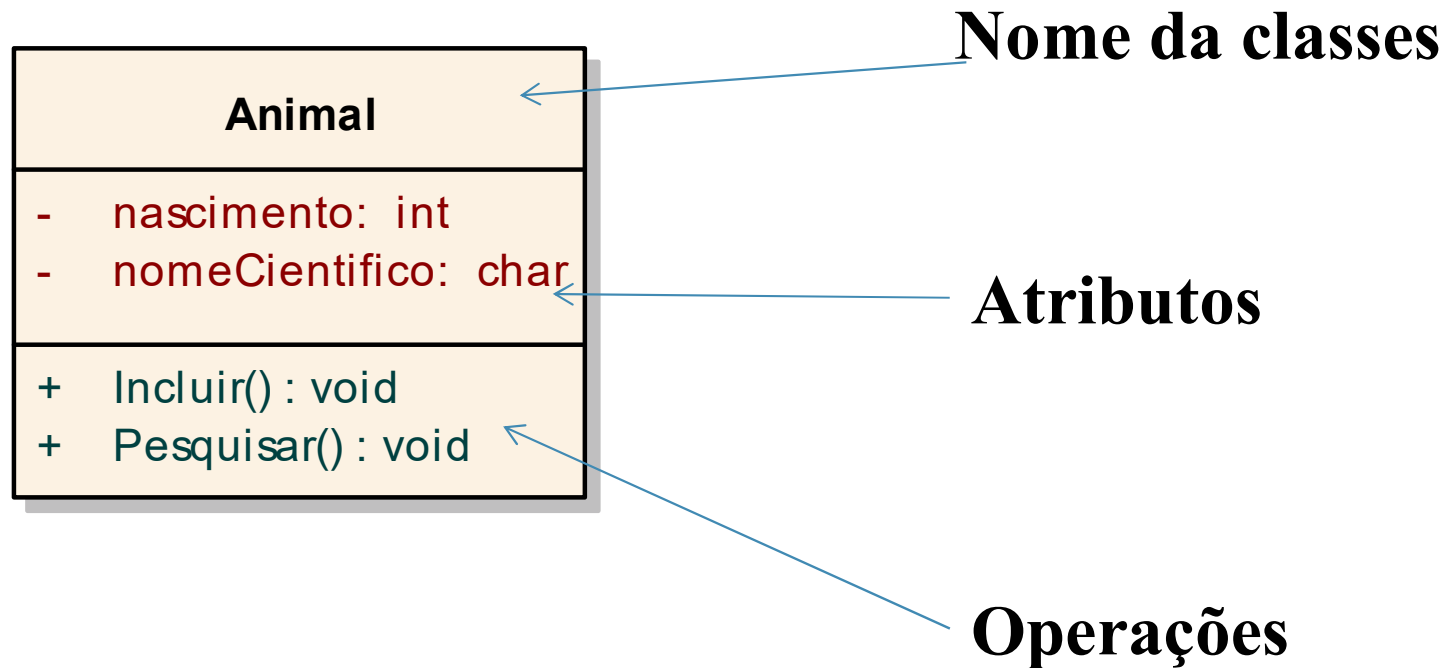
- Existem informações que devem ser armazenadas ou analisadas? Se existir alguma informação que tenha que ser guardada, transformada ou analisada de alguma forma, então é uma possível candidata para ser uma classe.
- Existem sistemas externos ao modelado? Se existir, eles deverão ser vistos como classes pelo sistema para que possa interagir com outros externos.
- Existem classes de bibliotecas, componentes ou modelos externos a serem utilizados pelo sistema modelado? Se sim, normalmente essas classes, componentes e modelos conterão classes candidatas ao nosso sistema.
- Qual o papel dos atores dentro do sistema? Talvez o papel deles possa ser visto como classes, por exemplo, usuário, operador, cliente e daí por diante.
- Etc

O que é uma classe ?

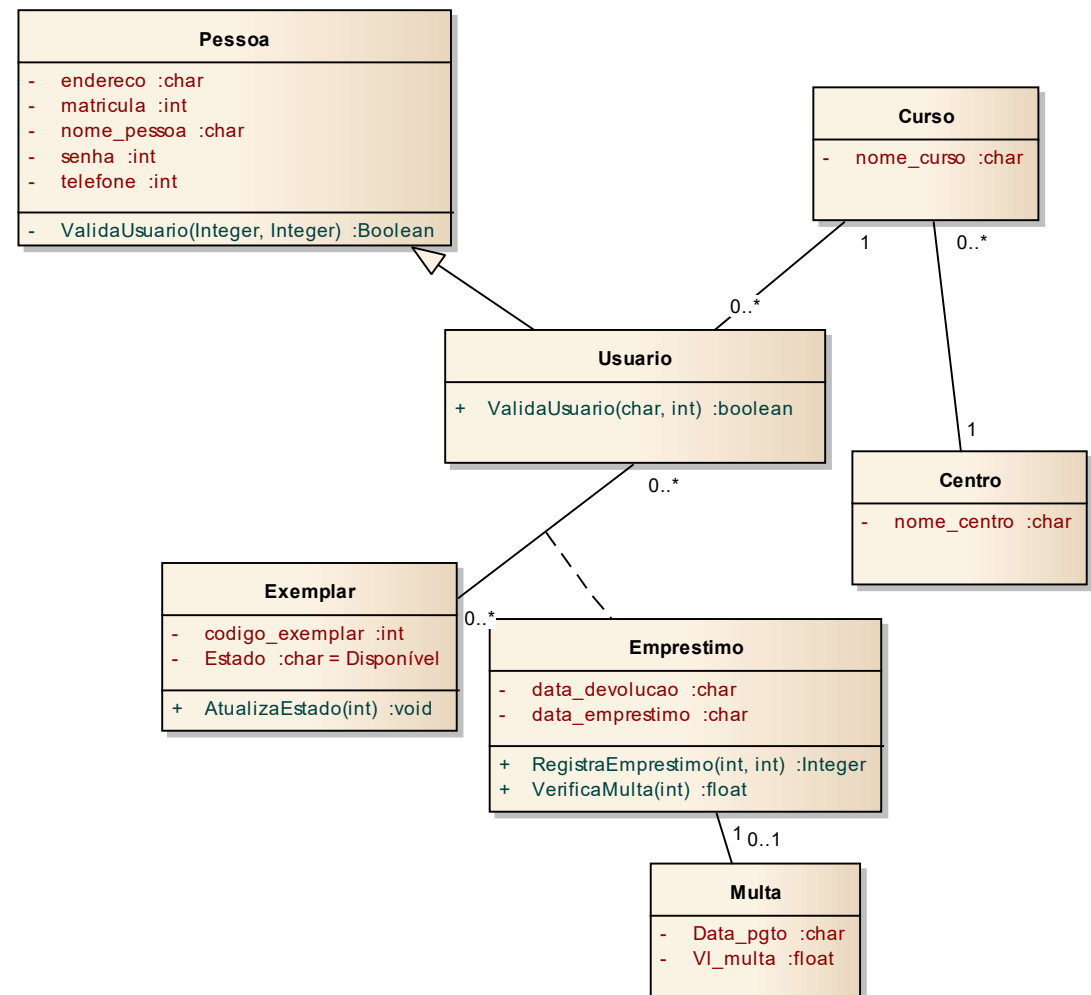
Uma classe é a descrição de um tipo de objeto.



Representação de classe



Exemplo de Diagrama de Classes



Atributos

01

Um atributo representa alguma propriedade que é compartilhada por todos os objetos de uma classe

02

Descrevem os dados contidos nas instâncias de uma classe

03

Servem para manter o estado dos objetos.

- Cada objeto possui valores independentes para os mesmos atributos



Sintaxe de Atributos



Sintaxe

- [visibilidade] nome
[[multiplicidade]]
[:tipo]
[= valor inicial]
[{propriedades}]



Exemplos

- CPF: Integer {frozen}
- Nome: String =
"Alberto"
- Endereço [0..2] : String

Sintaxe de Atributos

Visibilidade

público (+)

O atributo é acessível a qualquer outro objeto ou classe

protegido (#)

O atributo é acessível apenas nas subclasses

privado (-)

O atributo só é acessível dentro da classe onde foi definido

Operações

Uma operação é a implementação de um serviço que pode ser requisitado a qualquer objeto ou classe, possivelmente afetando o seu estado

A execução de uma operação pode resultar na alteração do valor de seus atributos

Operações de instância

Atuam sobre uma instância (objeto) de uma classe

Operações de classe

Atuam sobre a classe, criando e/ou modificando atributos de classe

Sintaxe para Operações

Sintaxe

[visibilidade] nome [(lista-de-parâmetros)]
[:tipo-de-retorno] [{propriedades}]

Exemplos


LerTemperatura () : Number

Área (Lado : Integer) : Integer


ValorPadrão () : Integer {query}

Relacionamentos

Poucas classes têm
sentido sozinhas



Os relacionamentos ligam
classes/objetos entre si
criando relações lógicas
entre eles



Os relacionamentos podem
ser dos seguintes tipos:

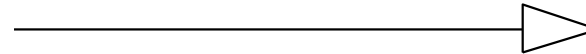
- Associação
- Composição e Agregação
- Dependência
- Generalização

Notação para Relacionamentos

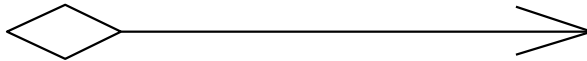
Associação



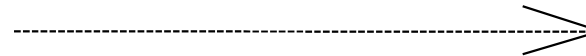
Herança



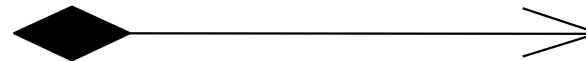
Agregação



Dependência



Composição



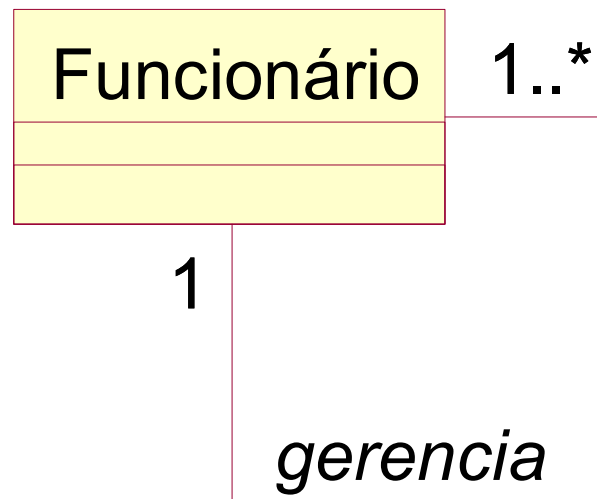
Associação

- É um relacionamento estrutural que especifica que objetos de um elemento estão conectados a objetos de outro elemento



Associação Unária

Quando há um relacionamento de uma classe para ela mesma



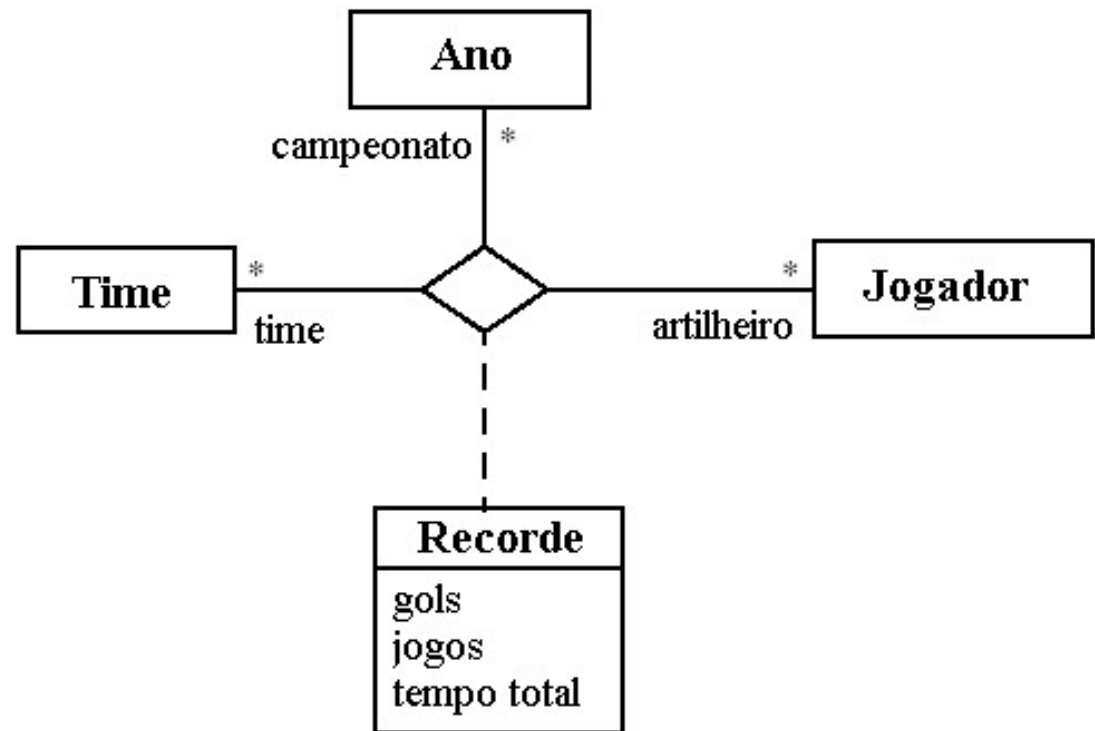
Associação Binária

Quando há duas classe envolvidas na forma direta de uma para a outra



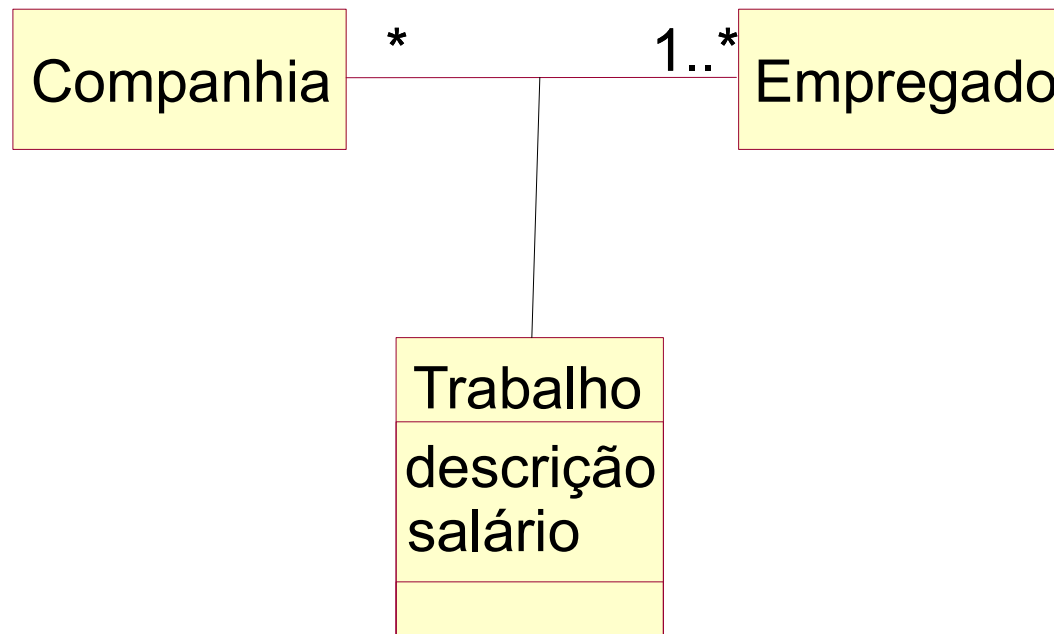
Associação N-ária

As linhas das associações são conectadas por um losango



Classe Associativa

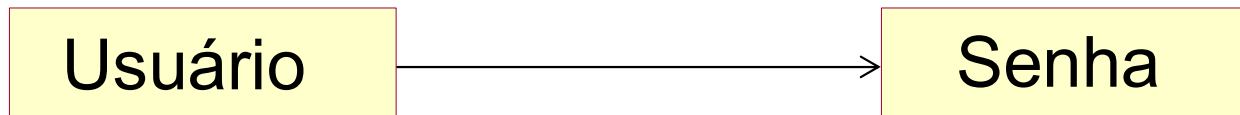
É introduzida quando uma associação tem propriedades associadas



Navegabilidade

Em geral a navegação entre as classes de uma associação é bi-direcional.

Porém, podemos limitá-la a apenas uma direção

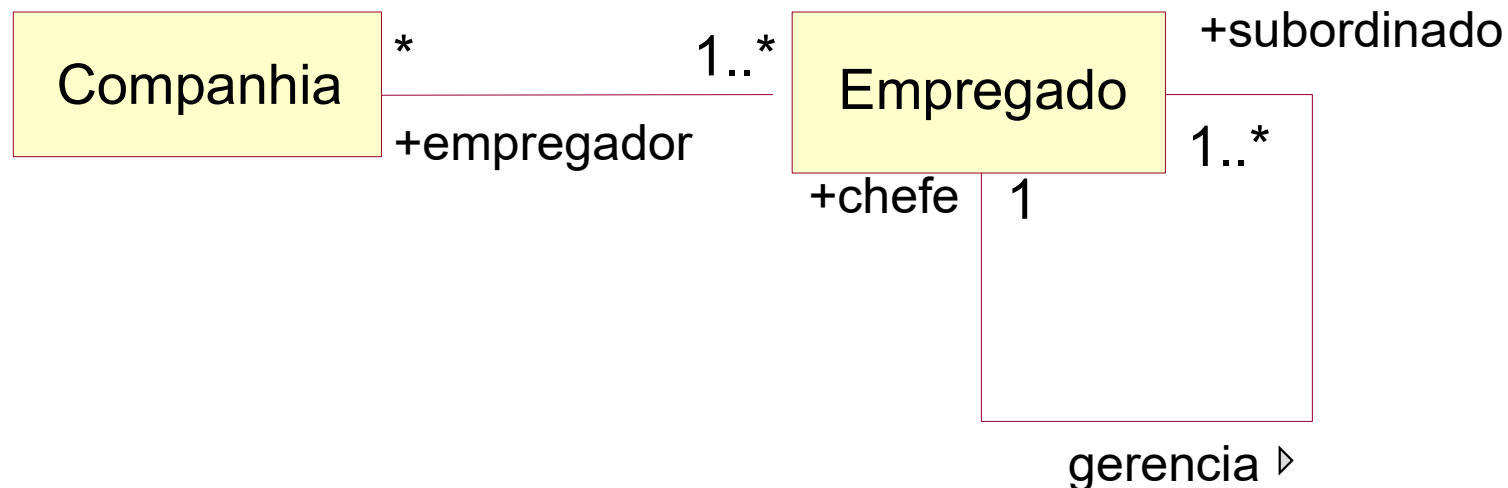


Papéis em Associações

Papéis normalmente são colocados nas extremidades de uma associação quando:

A associação relaciona dois objetos da mesma classe

Há mais de uma associação entre as classes



Agregação e Composição

Em uma Agregação a vida das “Partes” não está relacionada à vida do “Todo”

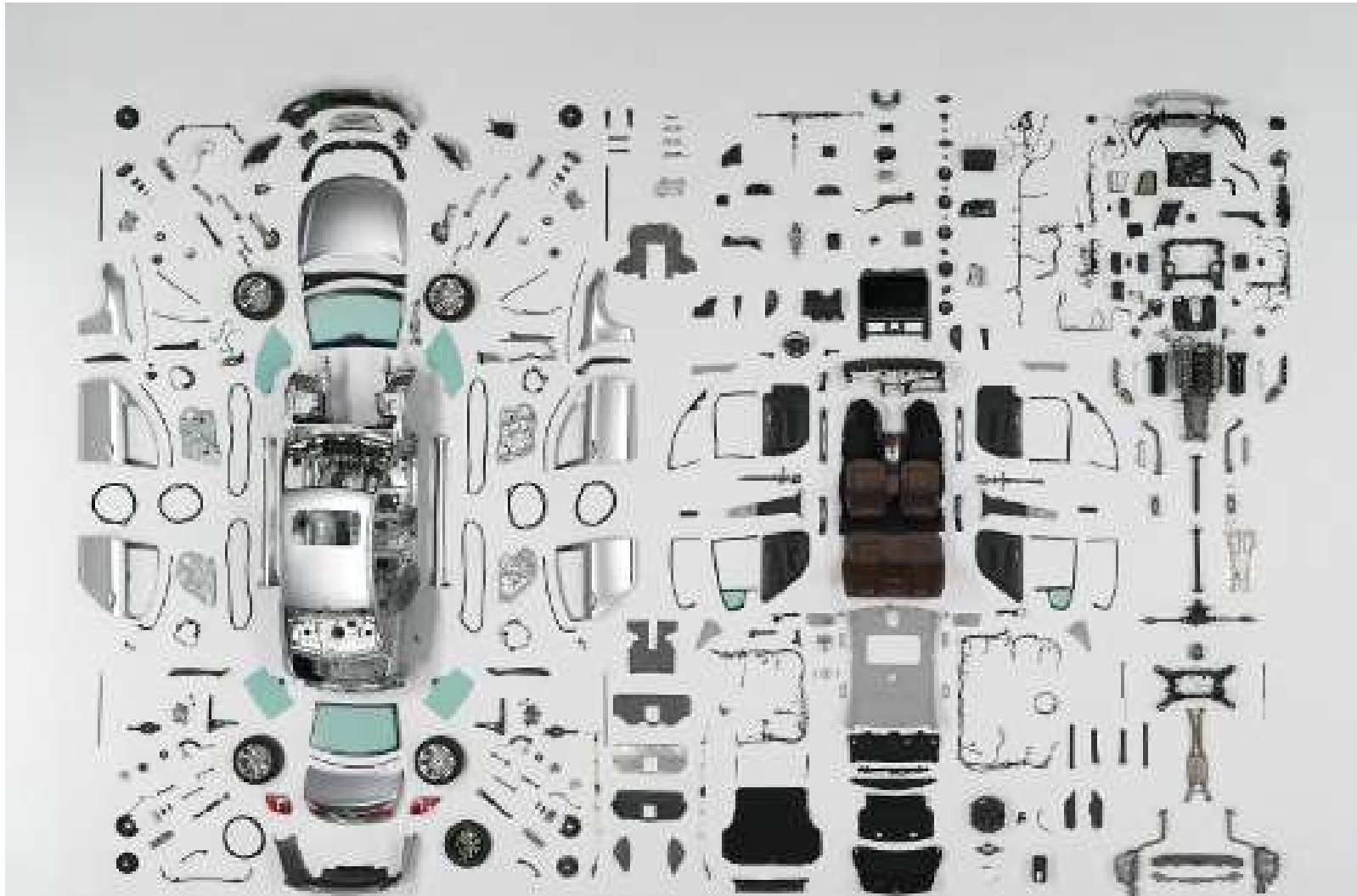
A Composição é uma forma mais forte de Agregação

Há uma coincidência da vidas das partes

Uma vez criada a parte, ela irá viver e morrer com o todo

O “Todo” é responsável pelo gerenciamento da criação e destruição das partes

Agregação



Composição

O Todo

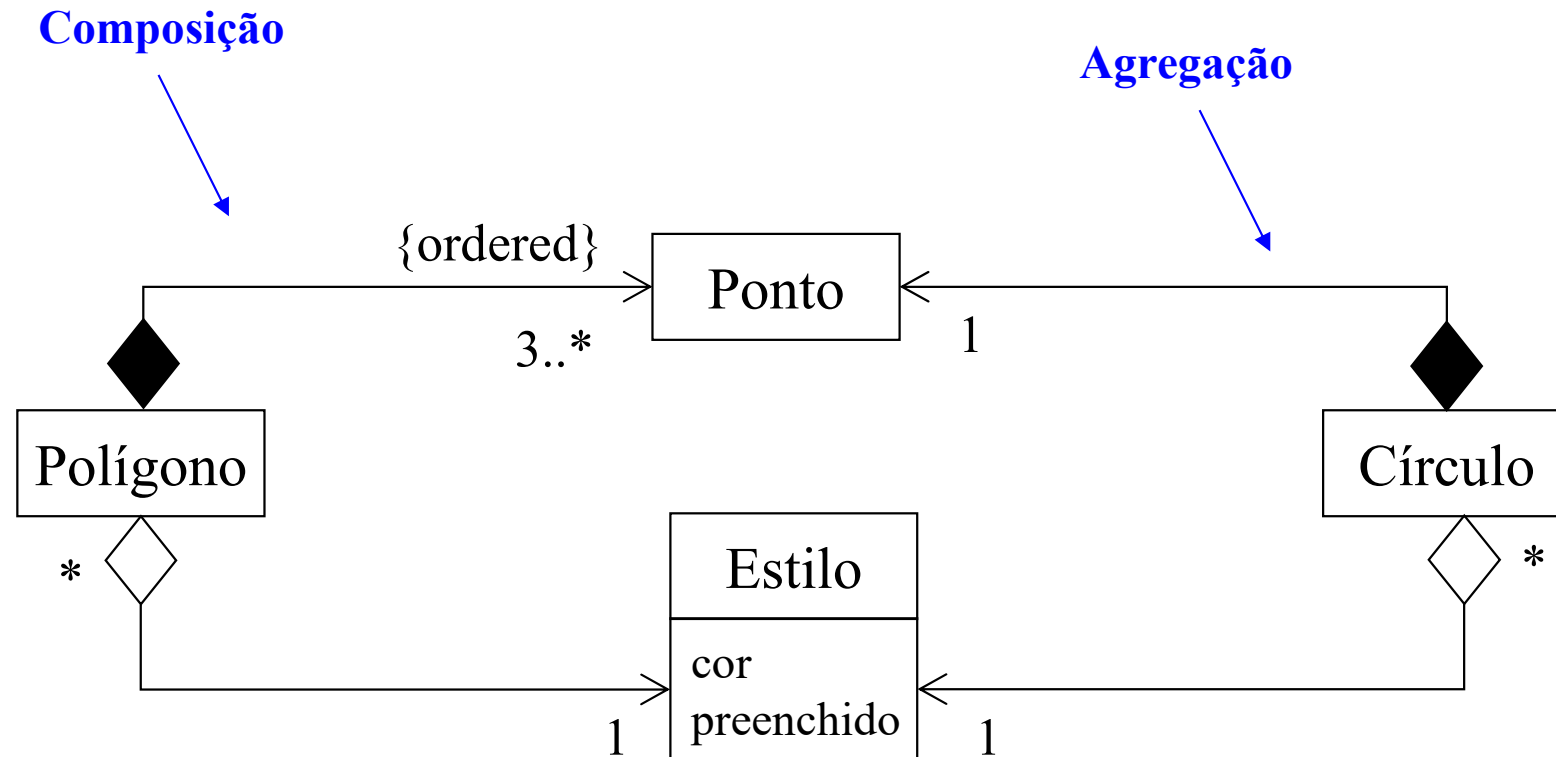


Partes



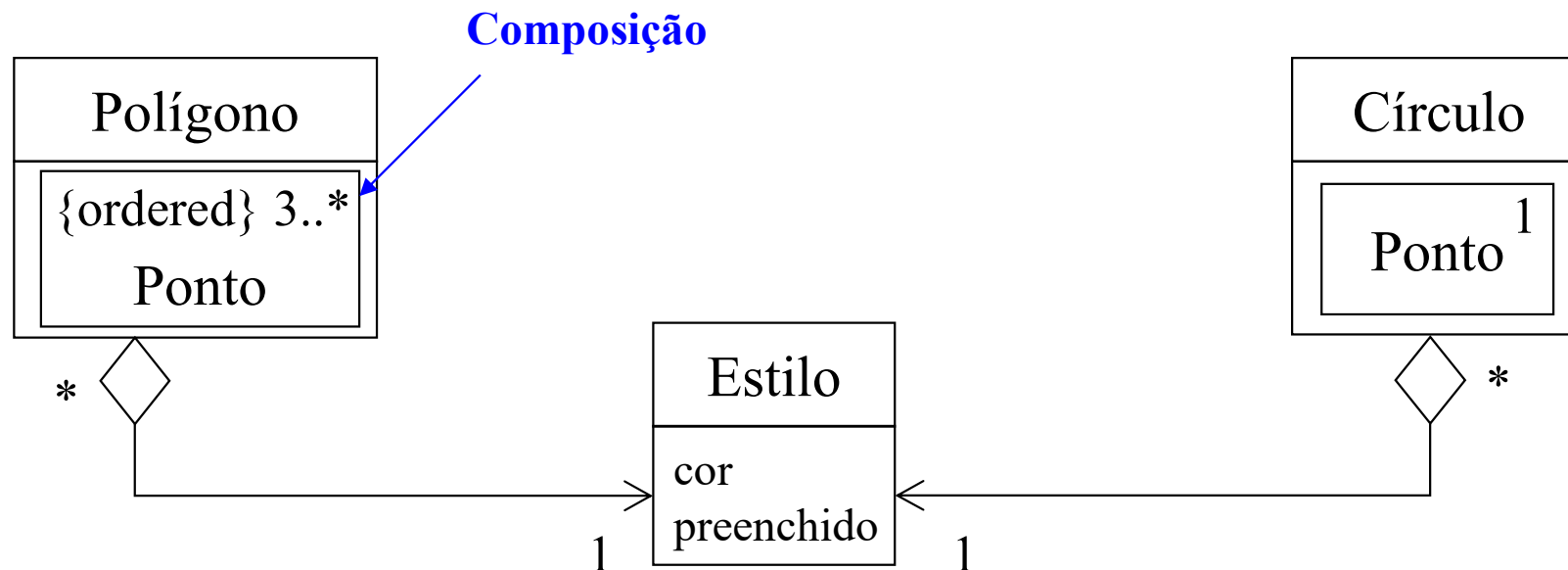
Aggregação e Composição

Notação



Agregação e Composição

Notação Alternativa



Dependências

Dependências são relações de uso

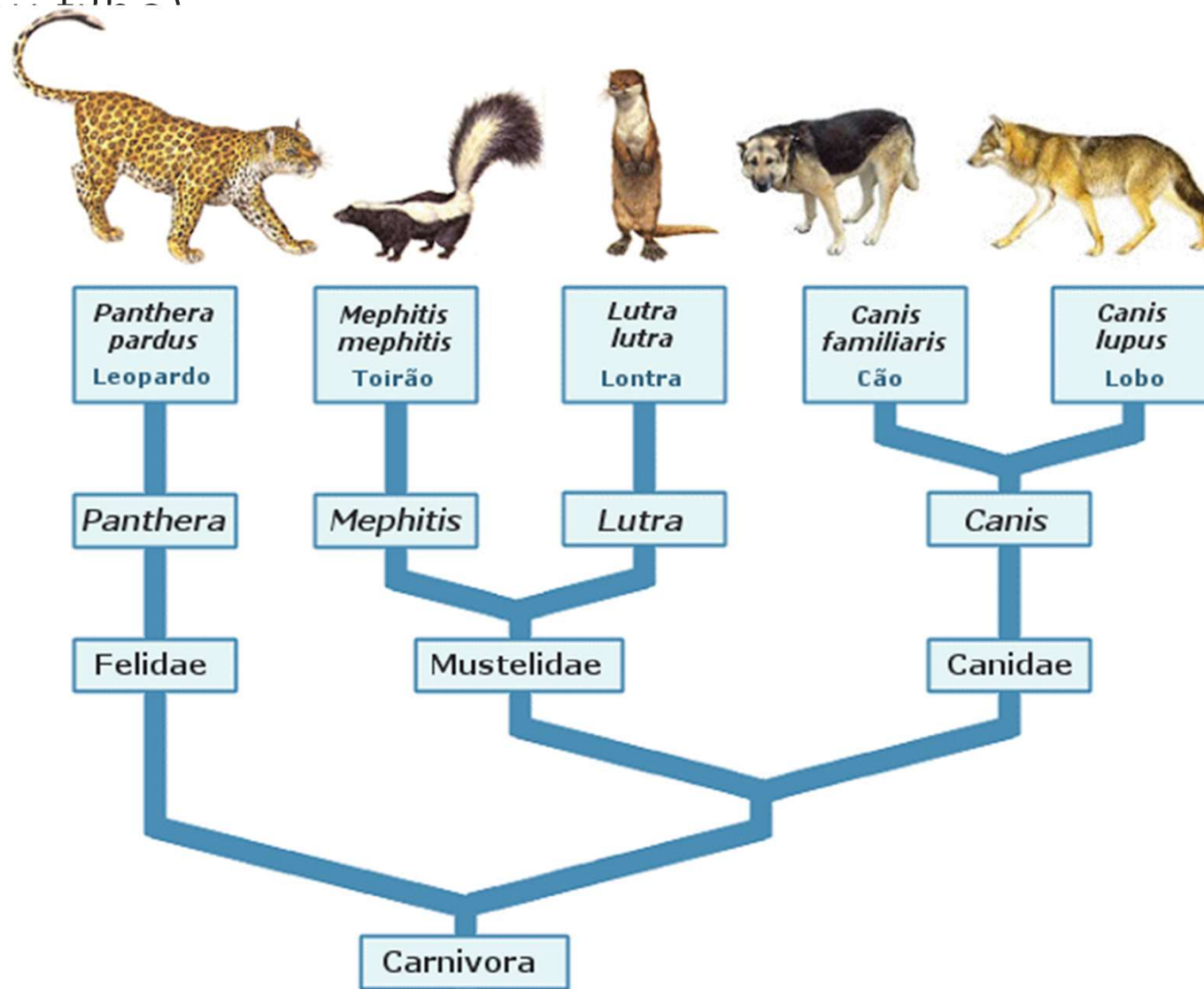
Uma dependência indica que mudanças em um elemento (o “servidor”) podem afetar outro elemento (o “cliente”)

Uma dependência entre classes indica que os objetos de uma classe usam serviços dos objetos de outra classe

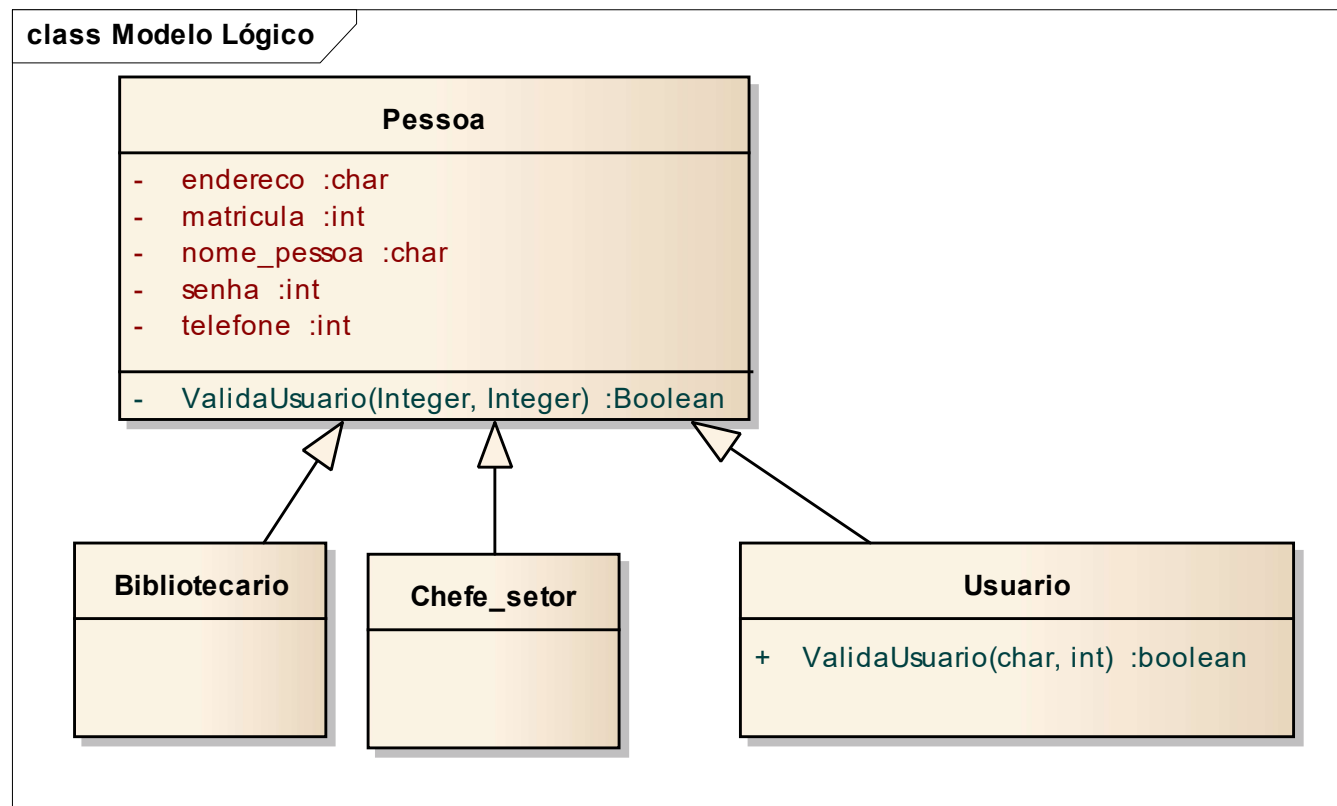


Generalização ou herança

Relacionamento entre um elemento mais geral (chamado de superclasse ou pai) e um mais específico (chamado de subclasse ou filho)

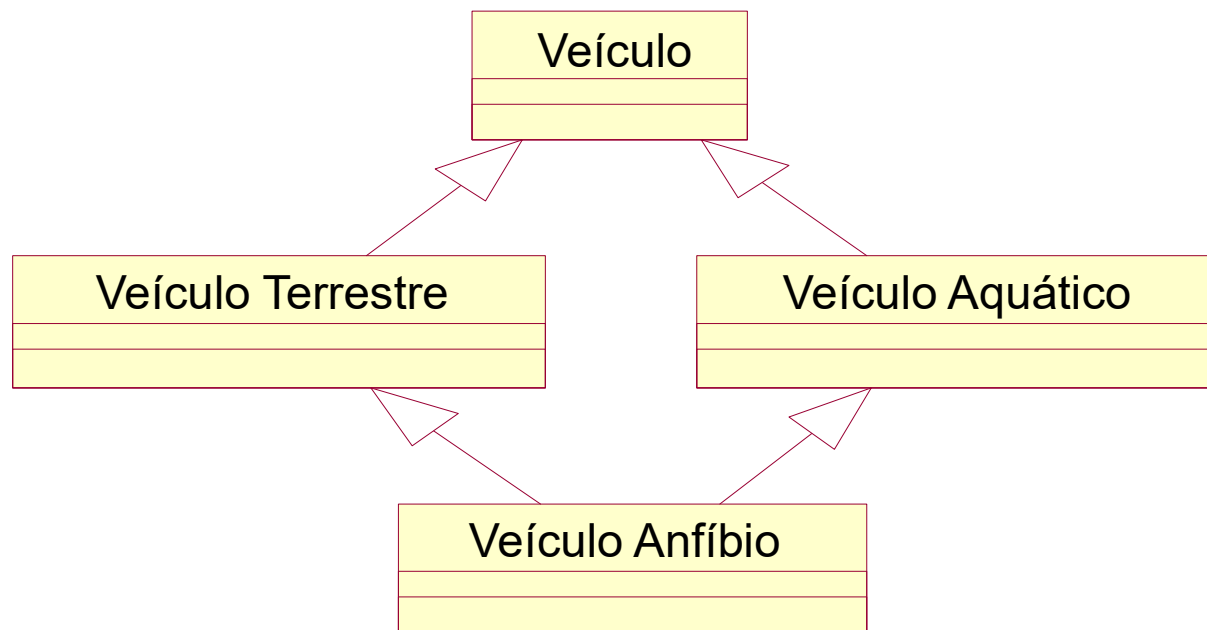


Generalização ou herança



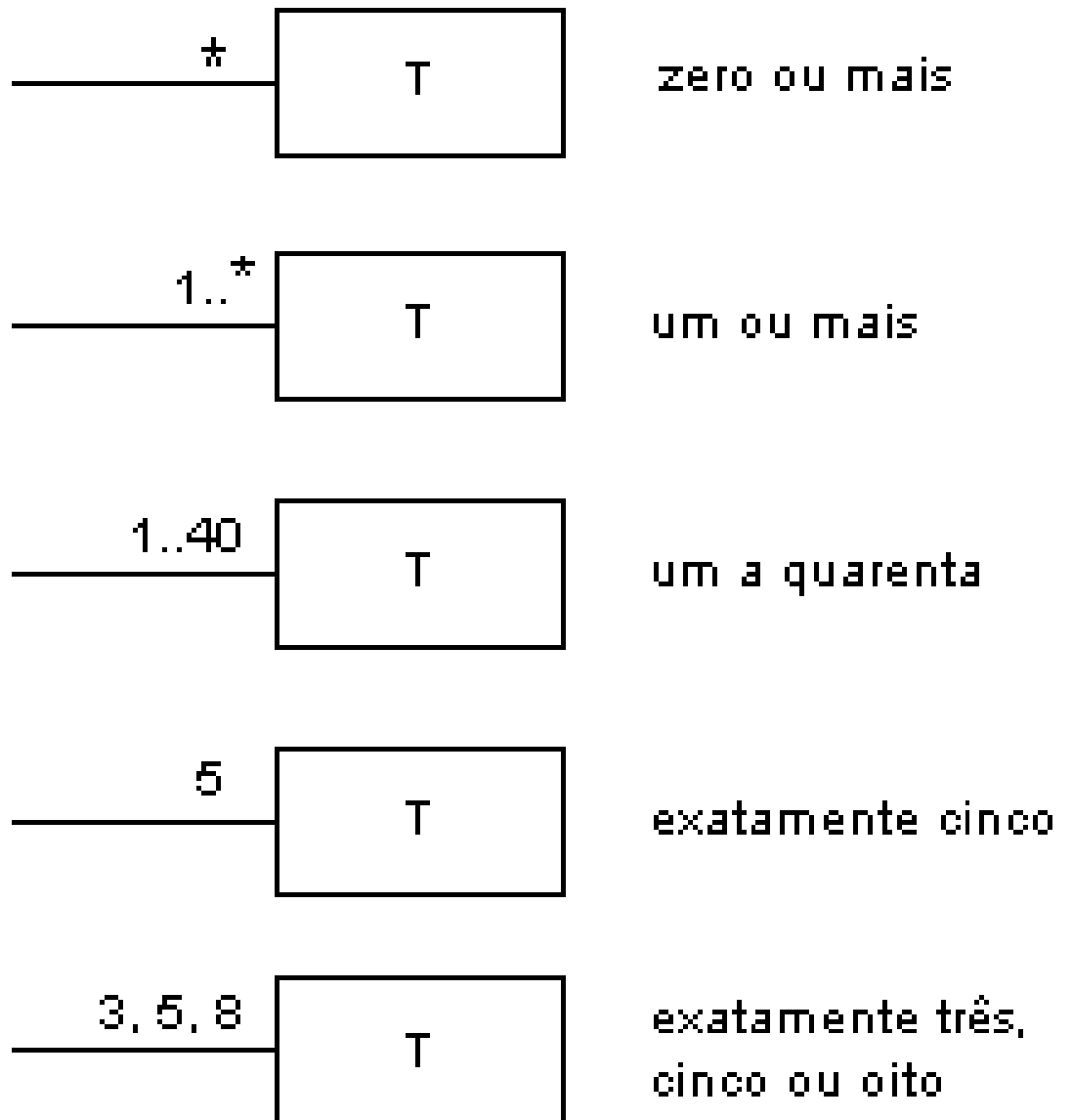
Herança Múltipla

Quando uma classe tem múltiplas superclasses



Multiplicidade

É a cardinalidade de uma associação



Atributos Derivados

São atributos que podem ser derivados (calculados) a partir de outros atributos

Normalmente servem como *cache*

Exemplo:

Data
Valor : String /Dia : Number /Mês : Number /Ano : Number
getDia() : Number getMes() : Number getAno() : Number

Modelando classes de domínio

Analisar o caso de uso

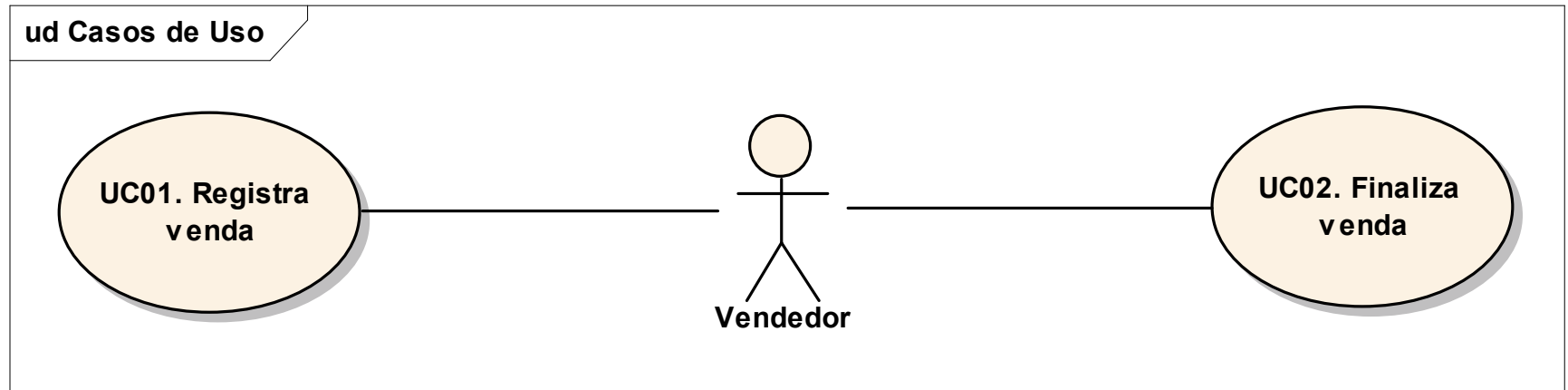
Analisar os protótipos de telas do sistema

Identificar as classes

Definir atributos

Definir relacionamentos

Exemplo



Exemplo

UC01. Registra venda

public UseCase:

Constraints

- *Approved Pre-condition* . O vendedor deve estar logado no sistema.
- *Approved Post-condition* . Uma venda foi registrada.

Connections

- Association link to actor *Vendedor*

Scenarios

1. Registra venda {Principal}.

- 1.1 O vendedor informa o número do cliente (TEL004).
- 1.2 O sistema apresenta na tela o nome do cliente.
- 1.3 O vendedor clica em confirmar.
- 1.4 O sistema abre a tela para registro do pedido (TEL005).
- 1.5 O vendedor informa os dados do pedido e seus itens, registrando o pedido (TELA005).
- 1.6 O sistema salva o pedido registrando como "Em aberto".

A1. Cliente inexistente {Alternativo}.

No passo 1.2, caso o cliente informado não exista, o sistema emite mensagem de erro "Cliente inexistente" (TEL003).


Exemplo

TEL004 - Registra venda

Número do cliente:

Nome cliente:

TEL003 - Mensagens

 Texto da mensagem

TEL005 - Registro do Pedido

Número pedido:

Nome cliente:

Data pedido:

Local entrega:

Situação: Valor total:

Produto	Preço unitário	Quantidade	Total

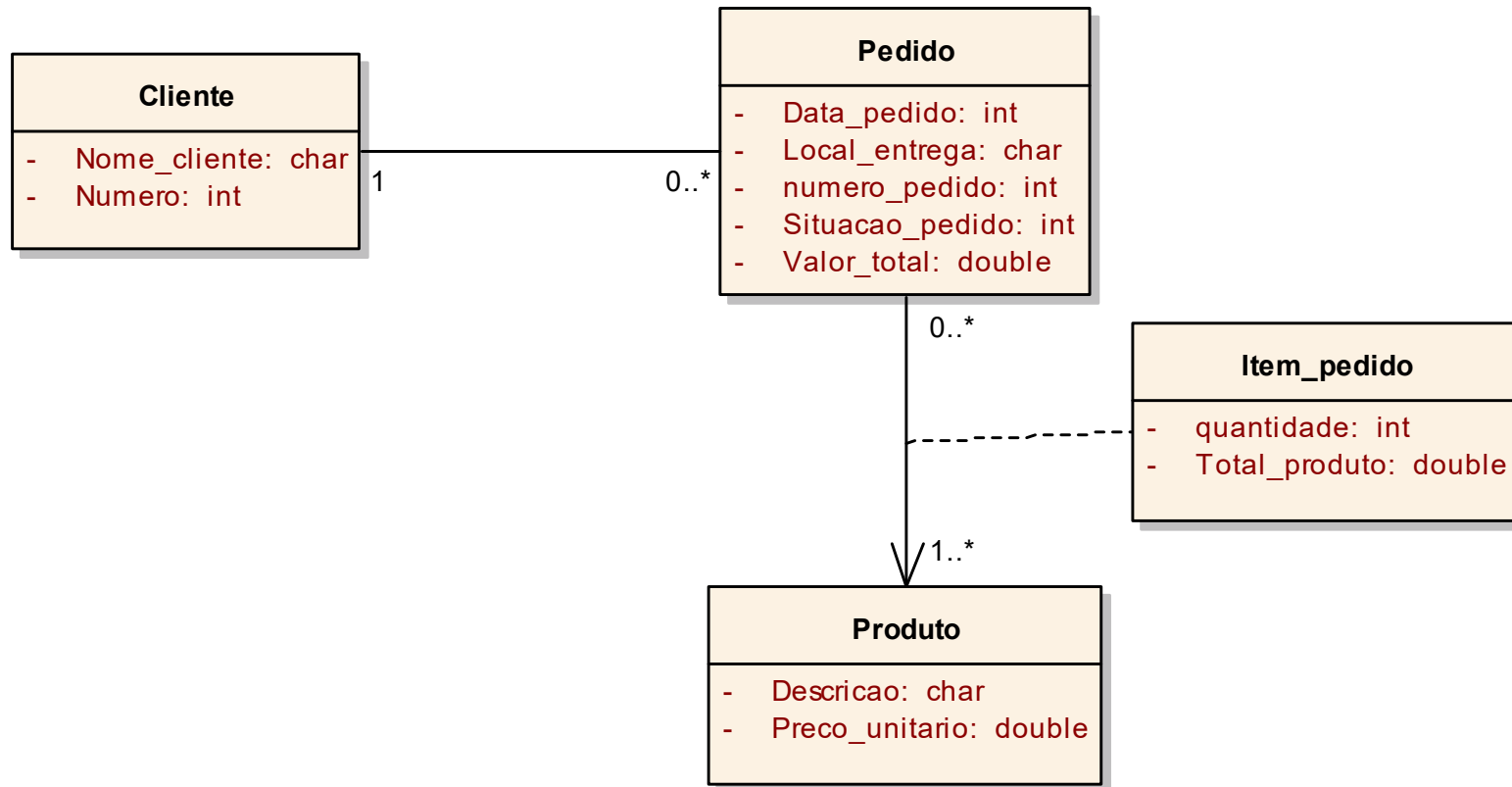


Diagrama de classes

Exercício

Nossa Frota:



Elabore um diagrama de classes para uma locadora de veículo, onde temos um cliente que aluga automóveis. Cada automóvel pertence a uma única loja da locadora, que ficam localizadas em diferentes cidades.

Os clientes podem ser pessoas físicas ou jurídicas.

Sobre o cliente deseja-se saber seu nome, endereço, telefone, CNH, e CPF se for pessoa física ou CNPJ se for pessoa jurídica. Também ser for pessoa jurídica deve ser informado o nome do condutor do automóvel.

Cada automóvel tem uma marca, modelo e placa.

Cada loja tem o nome do gerente, telefone e endereço.