

# Tópico 3 – UML - Diagrama de Classes



Luiz Antônio M. Pereira

[lpereira@uninet.com.br](mailto:lpereira@uninet.com.br)

[lpereira@luizantoniopereira.com.br](mailto:lpereira@luizantoniopereira.com.br)

# Diagrama de Classes – Motivação

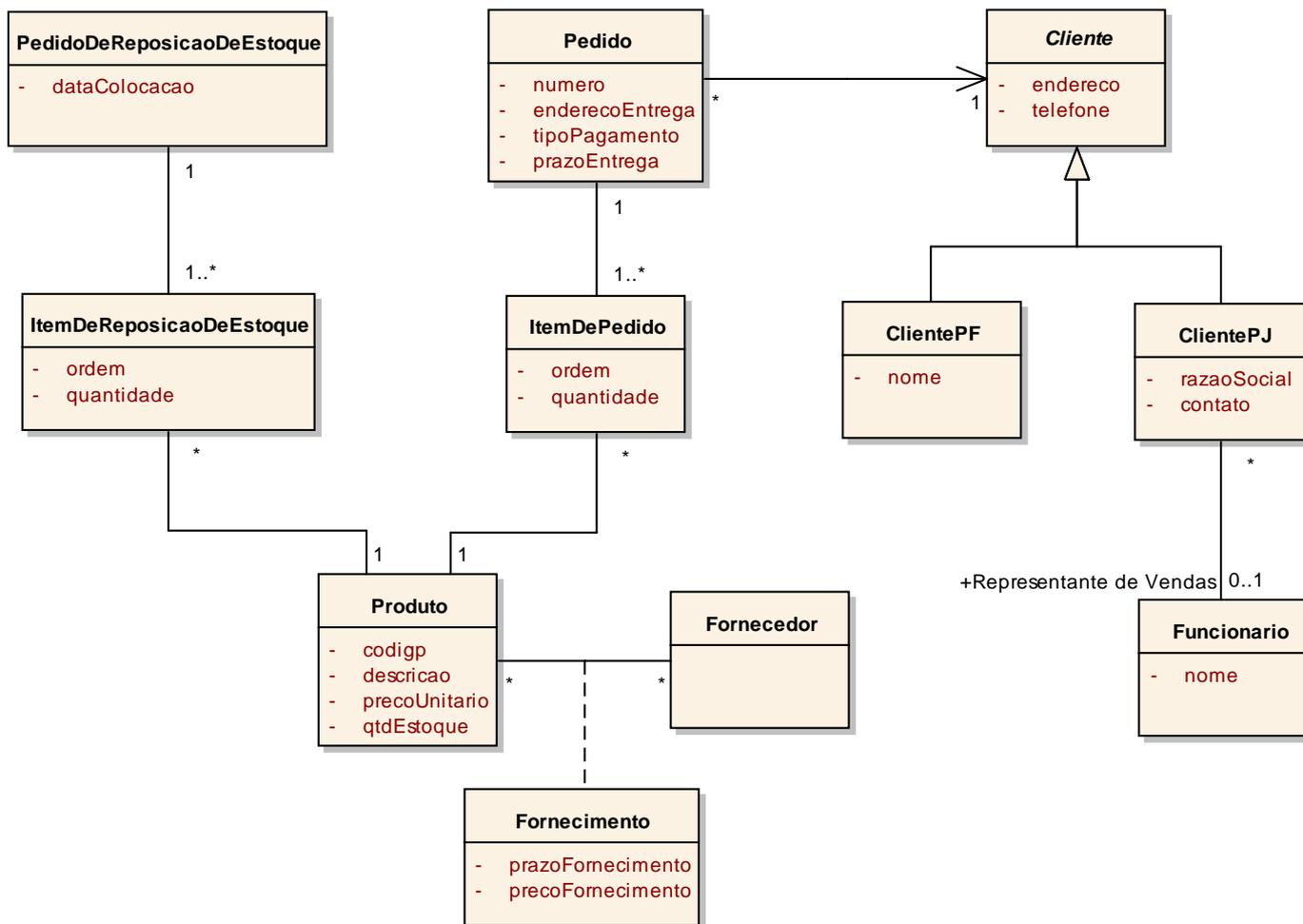
- Durante as nossas conversas com o cliente a respeito do novo sistema, relacionamos, além dos requisitos:
  - As informações que tratam de "coisas" ou conceitos com os quais os colaboradores em uma organização lidam no dia a dia;
  - Os relacionamentos que os conceitos mantêm entre si;
  - Restrições/regras diversas...

# Diagrama de Classes – Motivação

- Uma descrição textual não é apropriada porque
  - Os conceitos tipicamente possuem muitos detalhes;
  - Os conceitos usualmente se relacionam de formas intrincadas e obedecendo regras ou restrições complexas.
- Diagramas de classes são bons para essas descrições.

# Diagrama de Classes – Conceitos

## Sistema de Controle de Fornecimento e Vendas



# Diagrama de Classes – Conceitos

- É uma visão estática do sistema.
- Descreve relações atemporais entre elementos do domínio.
- Compõe-se de classes, relacionamentos entre elas, restrições, etc.

# Diagrama de Classes – Conceitos

- Também podem modelar o domínio sob três perspectivas:
  - Conceitual.
  - Especificação.
  - Implementação.
- Cada perspectiva representa o domínio com graus diferentes de abstração



# Diagrama de Classes – Conceitos

- Na perspectiva conceitual queremos:
  - Representar abstrações;
  - Independência de implementações;
  - Representar informações do mini-mundo;
  - Facilidade de comunicação.

# Diagrama de Classes – Conceitos

---

Na medida em que se caminha em  
direção à implementação ...

## Diagrama de Classes – Conceitos

... dotamos nosso modelo de mais detalhes (perspectiva de especificação). Então

- Representamos as navegabilidades;
- Definimos os tipos e visibilidades dos atributos.
- ...

# Diagrama de Classes – Conceitos

---

Antes de iniciarmos a implementação devemos ter todos os detalhes definidos

# Diagrama de Classes – Conceitos

- Na perspectiva de implementação, onde representamos:
  - As operações get/set;
  - As operações/atributos “*protected*” e “*private*” e demais detalhes necessários à implementação na LP escolhida, etc.

# Diagrama de Classes

---

- Referência Básica:
  - UML 2.0 - Superestrutura



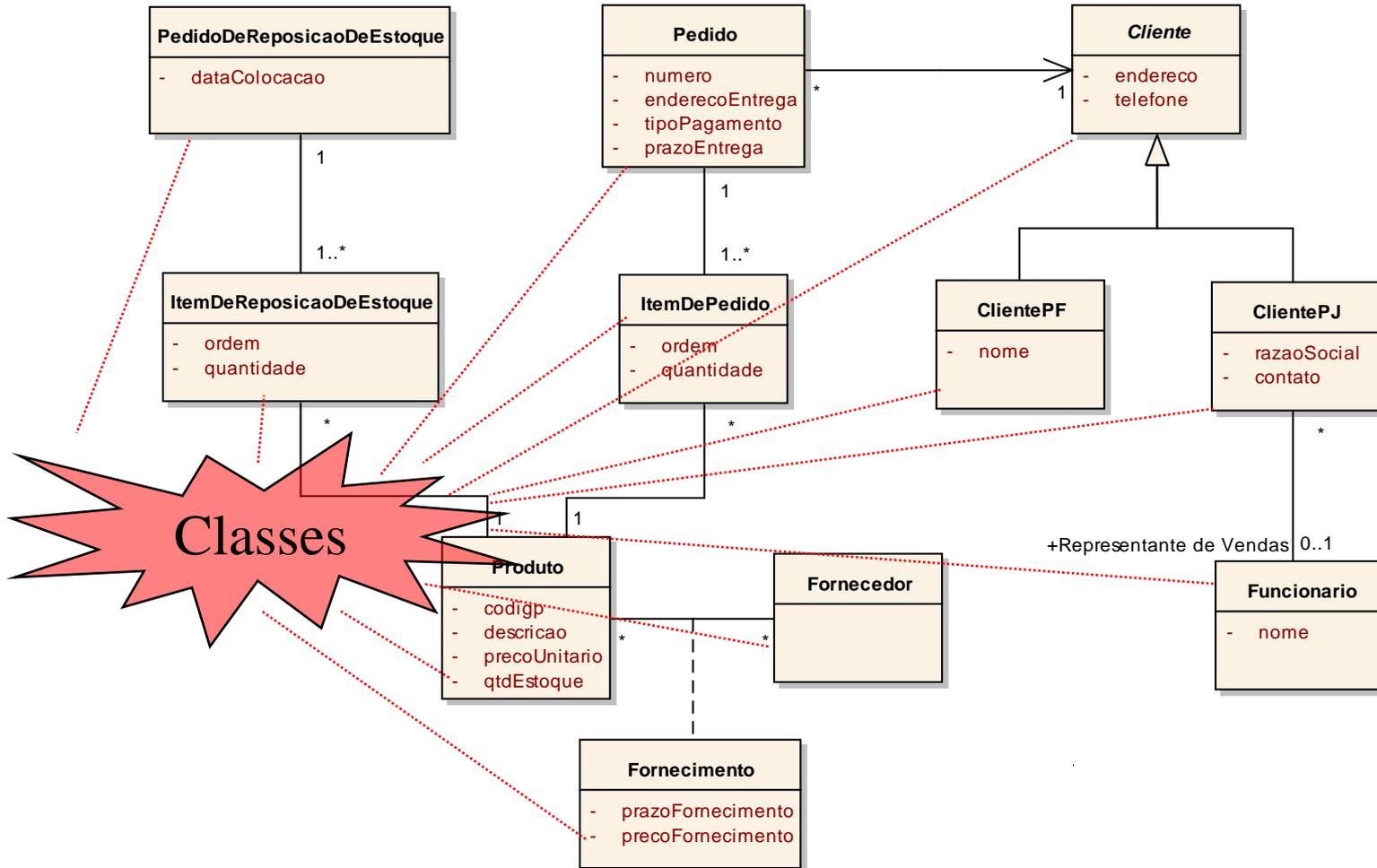
# Diagrama de Classes

---

## Elementos



# Diagrama de Classes – Classes





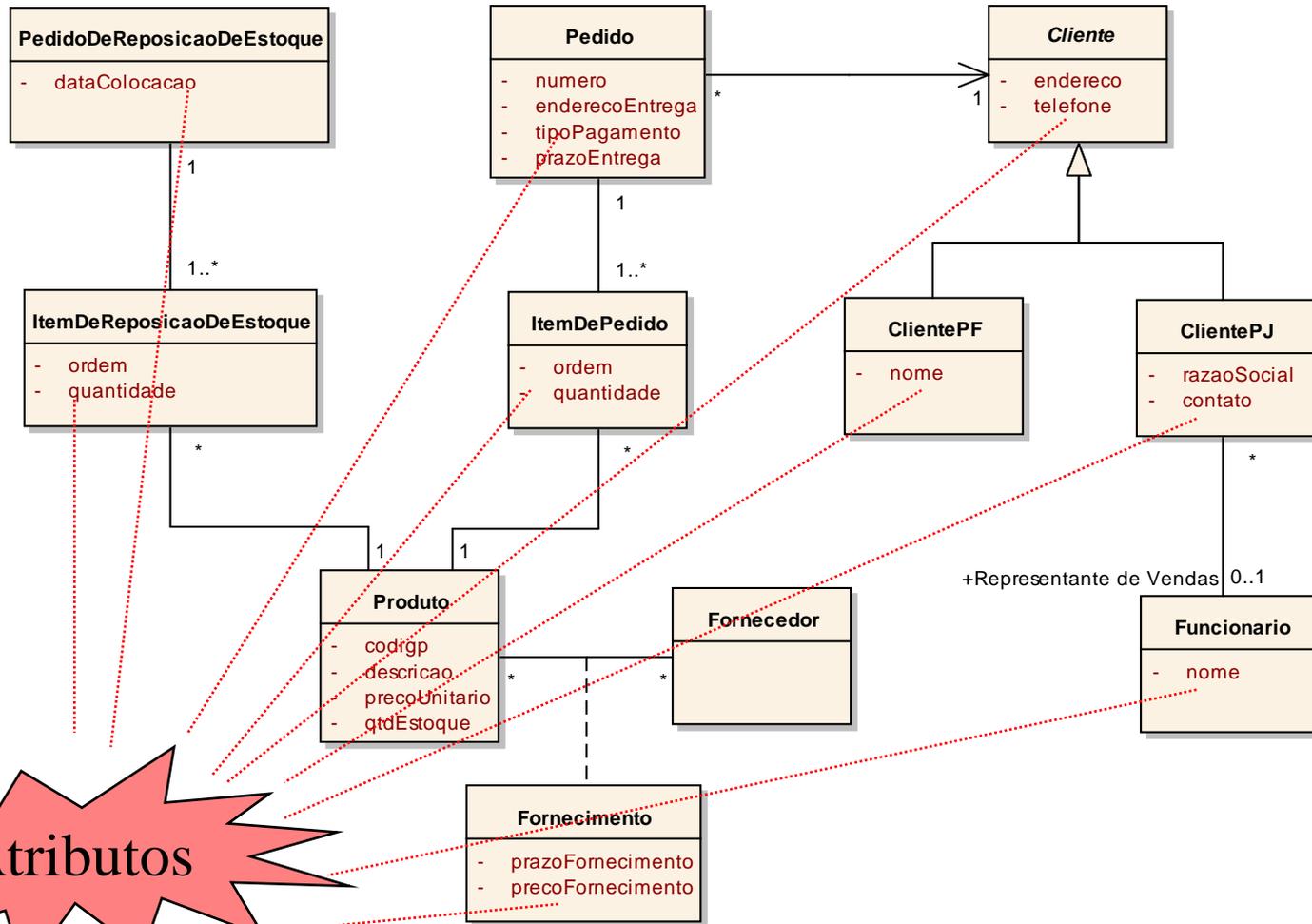
# Diagrama de Classes – Classes

## Classes:

- Cada classe deve ter um nome (centralizado, iniciando por maiúscula e em **negrito**) que a diferencie de outras classes (substantivos ou expressões breves);
- Retângulos compartimentados ou não, dependendo da perspectiva;
- Mais compartimentos podem ser criados para acomodação de regras de negócios, restrições, responsabilidades, etc.



# Diagrama de Classes – Atributos



Atributos



# Diagrama de Classes – Atributos

## Atributos:

- Alinhados à esquerda, no segundo compartimento (e se não houver atributos?);
- Sintaxe UML: *visibilidade nome: tipo = valor\_default {propriedade}*
- Visibilidade = "+" (*public*), "#" (*protected*), "-" (*private*)
- Visibilidade, tipos e valores *default* podem ser omitidos (perspectiva!);



# Diagrama de Classes – Atributos

## Atributos (cont~):

- Multiplicidade (qdo. for o caso) entra depois do nome. Ex:

...

+contato[0..1]:string

...

- Propriedade: { *changeable* | *addOnly* | *frozen* };
  - *changeable*: não há restrições p/ modificação do atributo;
  - *addOnly*: novos valores podem ser adicionados quando multiplicidade > 1;
  - *frozen*: atributo não pode ser modificado após iniciação do objeto (*const* em C);
  - Default = *changeable*



# Diagrama de Classes – Operações

- Colocadas no terceiro compartimento, na perspectiva de especificação em diante (diagramas conceituais de classes usualmente não apresentam operações)





# Diagrama de Classes – Operações

## Operações:

- São processos que a classe *sabe* realizar;
- Alinhados à esquerda, no terceiro compartimento (e se não houver operações?);
- Sintaxe UML: *visibilidade nome (lista\_parâmetros): expressão\_de\_tipo\_de\_retorno {propriedade}*



# Diagrama de Classes – Operações

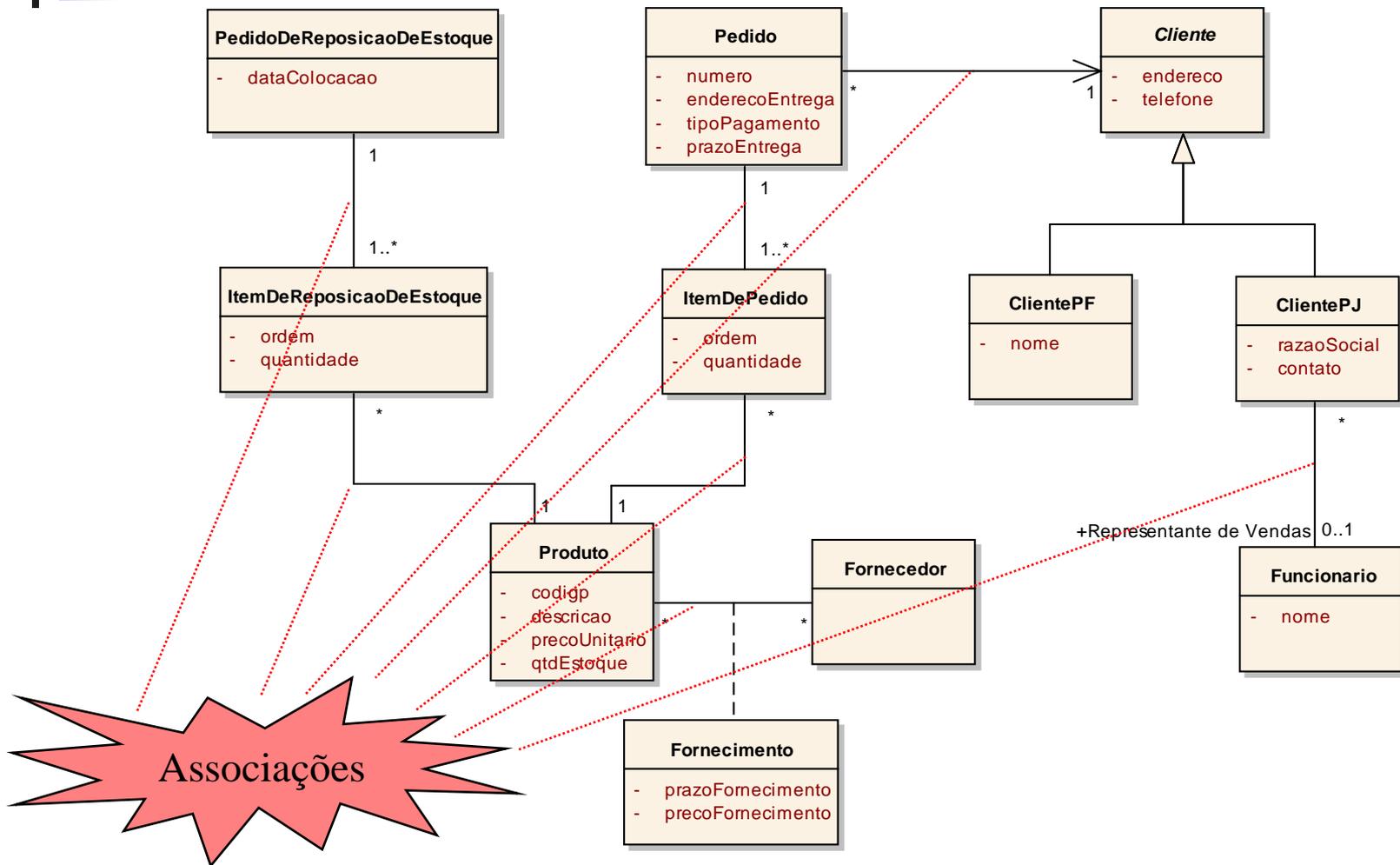
## Operações (cont~):

- Visibilidade: (vide atributos);
- Lista\_de\_parâmetros: separados por “,”, com a seguinte sintaxe:

*{in/out/inout} nome : tipo = valor\_default*

- Expressão\_de\_tipo\_de\_retorno: lista de tipos de retorno separados por “,” (são permitidos vários tipos de retorno);

# Diagrama de Classes – Associações





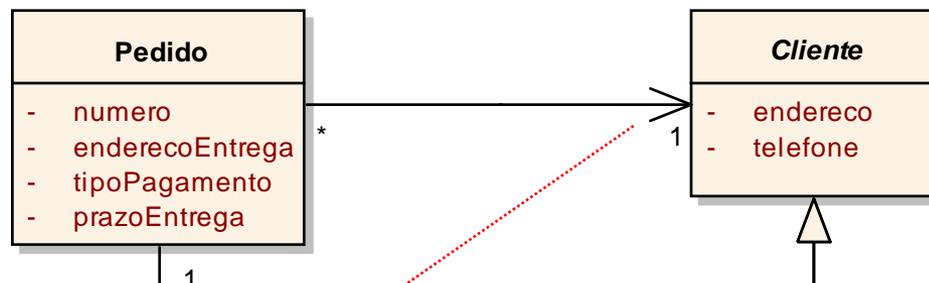
# Diagrama de Classes – Associações

## Associações:

- Representam relações entre ocorrências de classes;
- Quando a navegabilidade é representada significa que os objetos da classe origem têm a responsabilidade de determinar os objetos da classe destino aos quais estão relacionados;
  - Navegabilidades são normalmente associadas a questões de performance e facilidade de implementação;
  - São usualmente deixadas para a perspectiva de especificação.

Ver a seguir:

# Diagrama de Classes – Associações



**Navegabilidade**

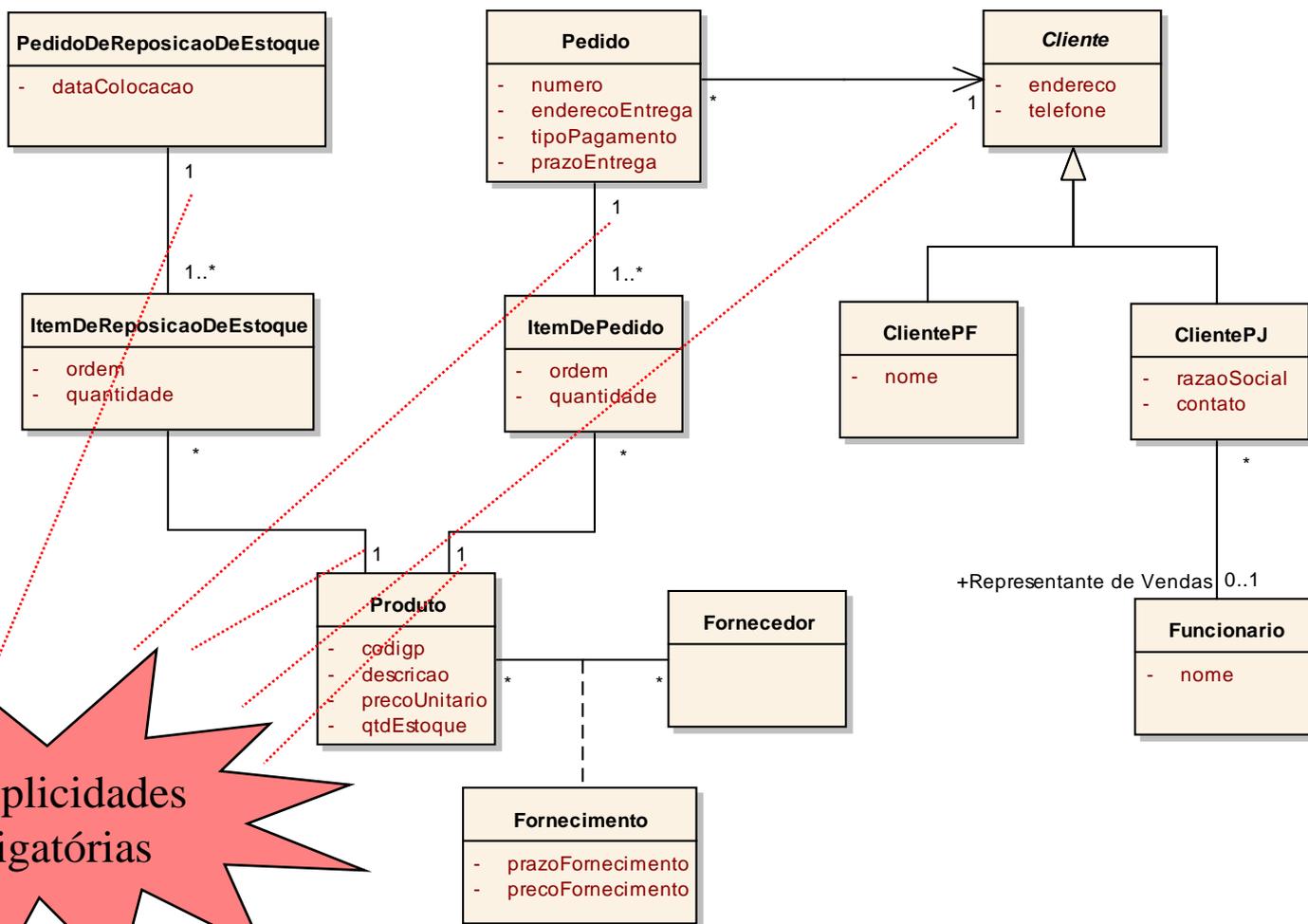
Se quem diz isso é o cliente, podemos representar isso já na perspectiva conceitual.

A partir de um determinado **Pedido**, PRECISO poder obter o(s) **Cliente(s)** a ele relacionado(s)  $\Rightarrow$  Devo dispor de operações na classe Pedido que me possibilitem isso. Sob a perspectiva de implementação, devo representar os atributos necessários para tal.

# Diagrama de Classes – Associações

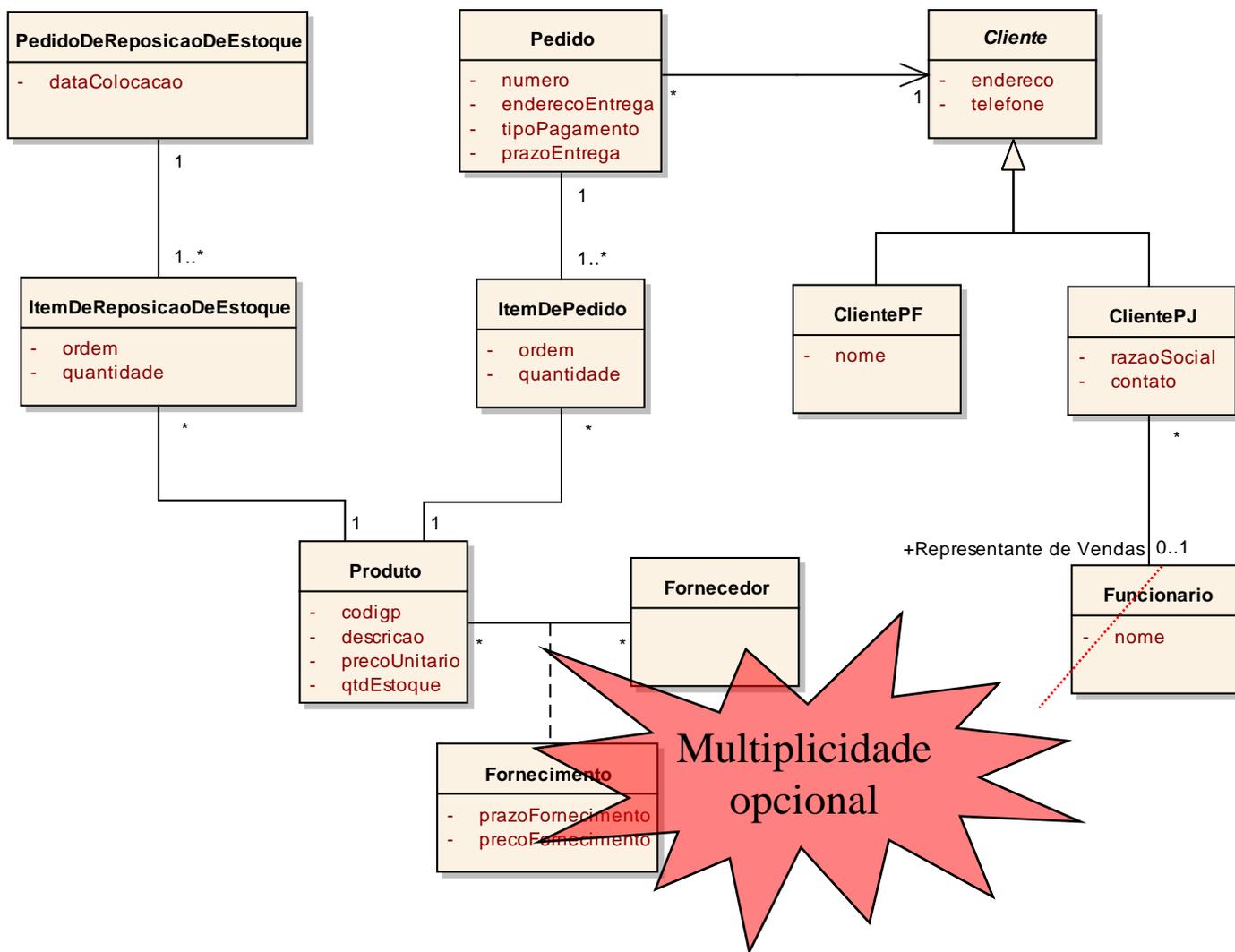
- Navegabilidades podem ser:
  - Uni-direcionais;
  - Bidirecionais;
  - Indeterminadas.
- Notação:
  - Uni-direcional: uma seta;
  - Bidirecional: \_ \_ \_ \_ setas ... ou nenhuma seta (convenciona-se para o projeto).

# Diagrama de Classes – Associações



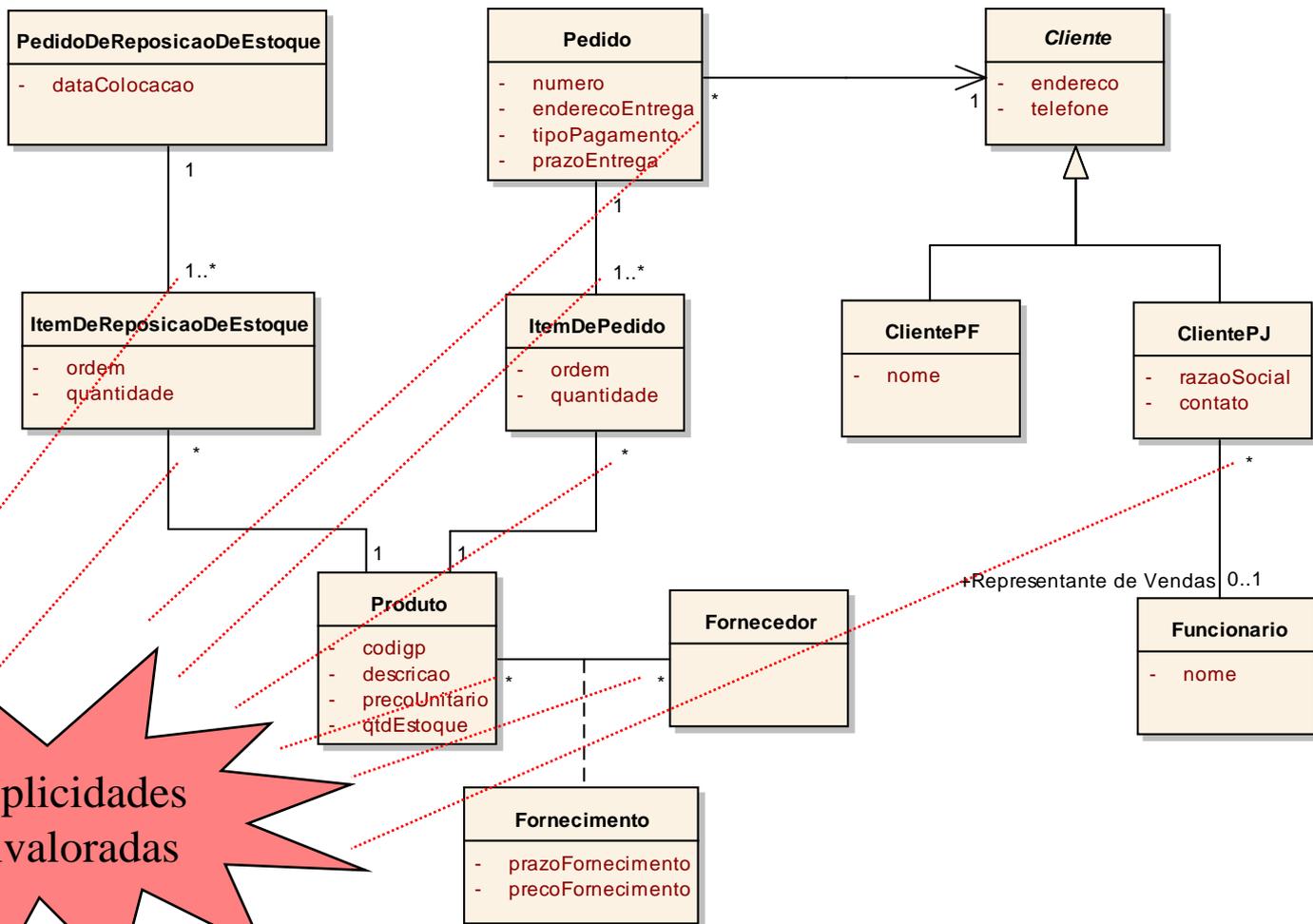
Multiplicidades obrigatórias

# Diagrama de Classes – Associações





# Diagrama de Classes – Associações

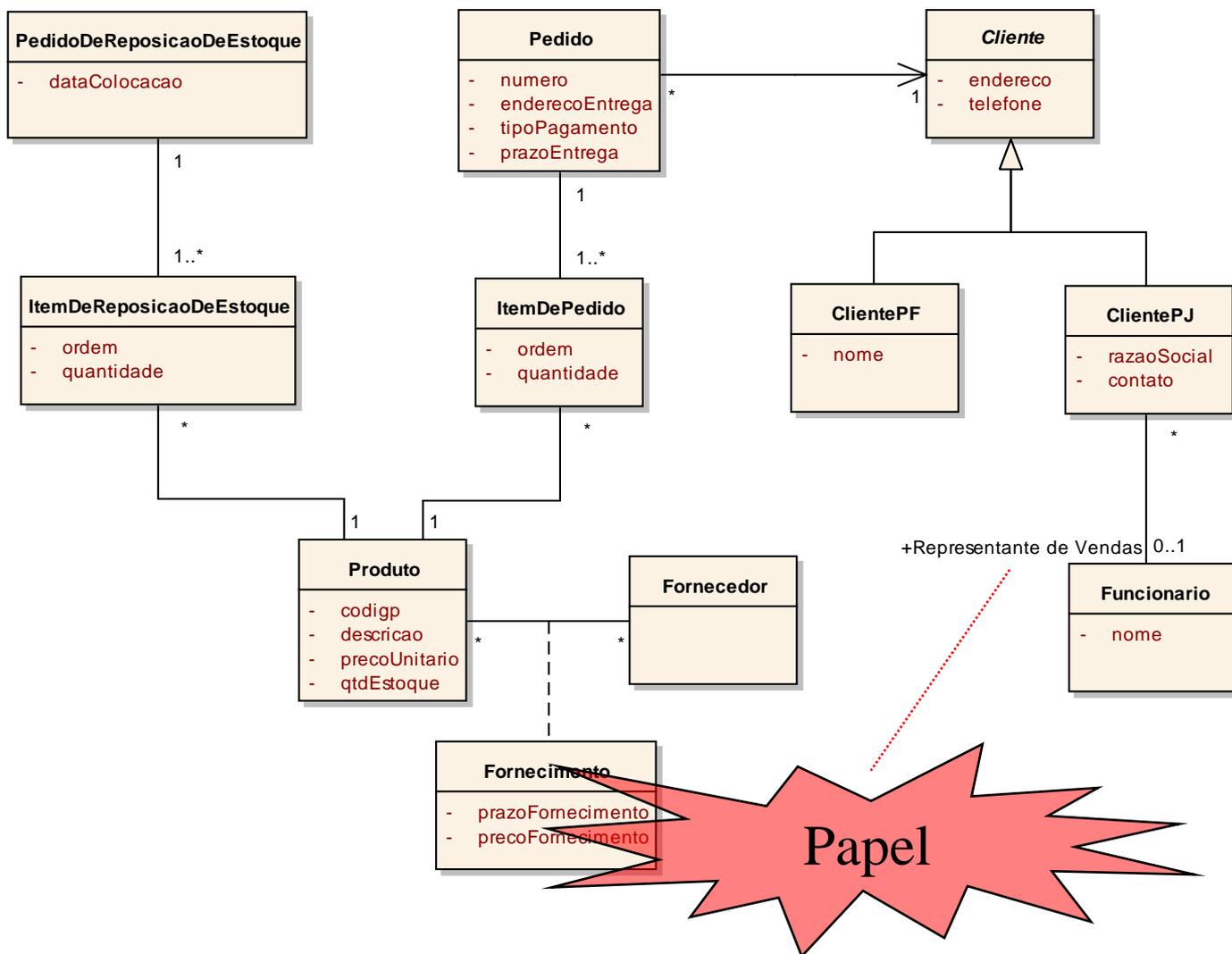


Multiplicidades multivaloradas

# Diagrama de Classes – Associações

- Semântica do "\*" sozinho: 0 a  $\infty$
- Possibilidades:
  - 1..\*
  - 2..3
  - 2..3, 5..7, 15..20

# Diagrama de Classes – Associações

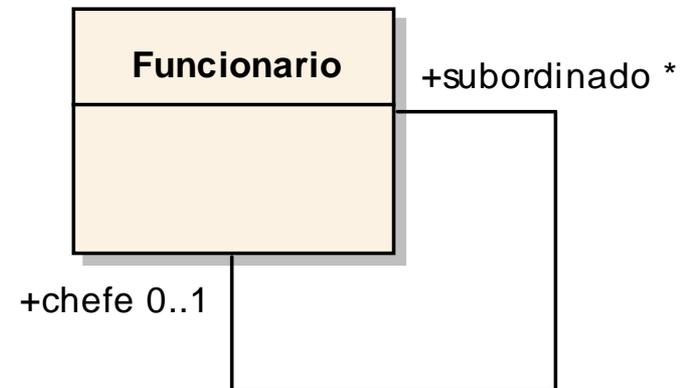
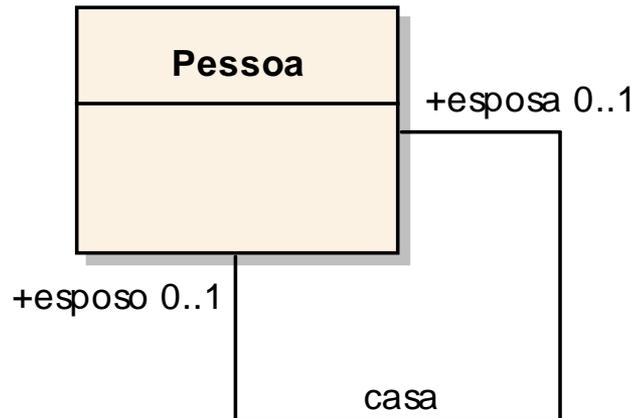


# Diagrama de Classes – Associações

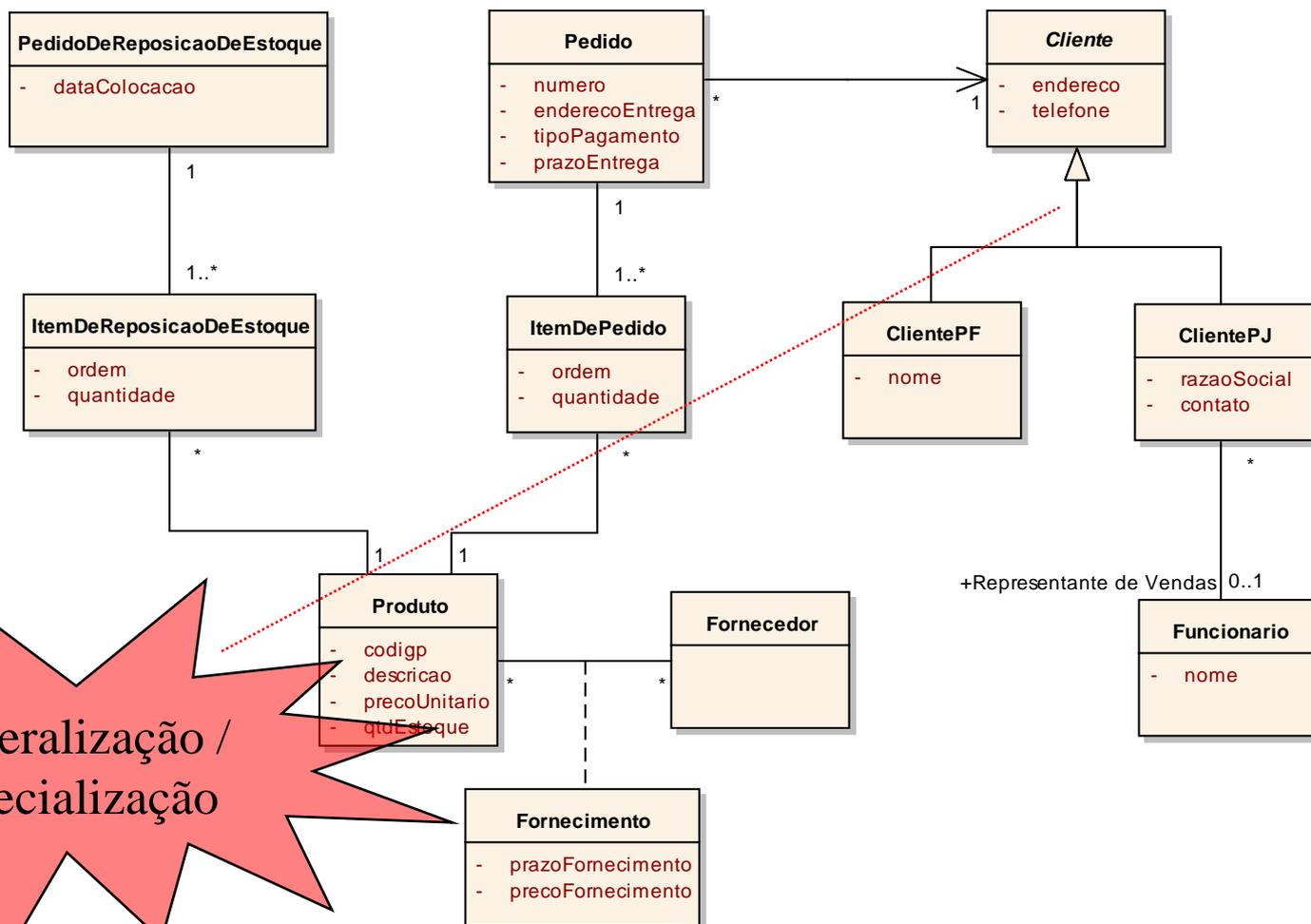
- Pontas das associações chamam-se “papéis”;
- Pontas podem ser rotuladas.
- Quando rótulos existem estes dão nomes aos papéis. Quando não existentes, os papéis levam o nome das *classes alvo*.

# Diagrama de Classes – Associações

Pode haver auto-associação. Exemplos:



# Diagrama de Classes – Generalização



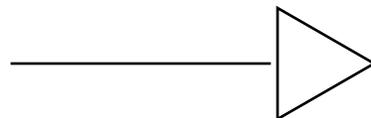
Generalização /  
especialização



# Diagrama de Classes – Generalização

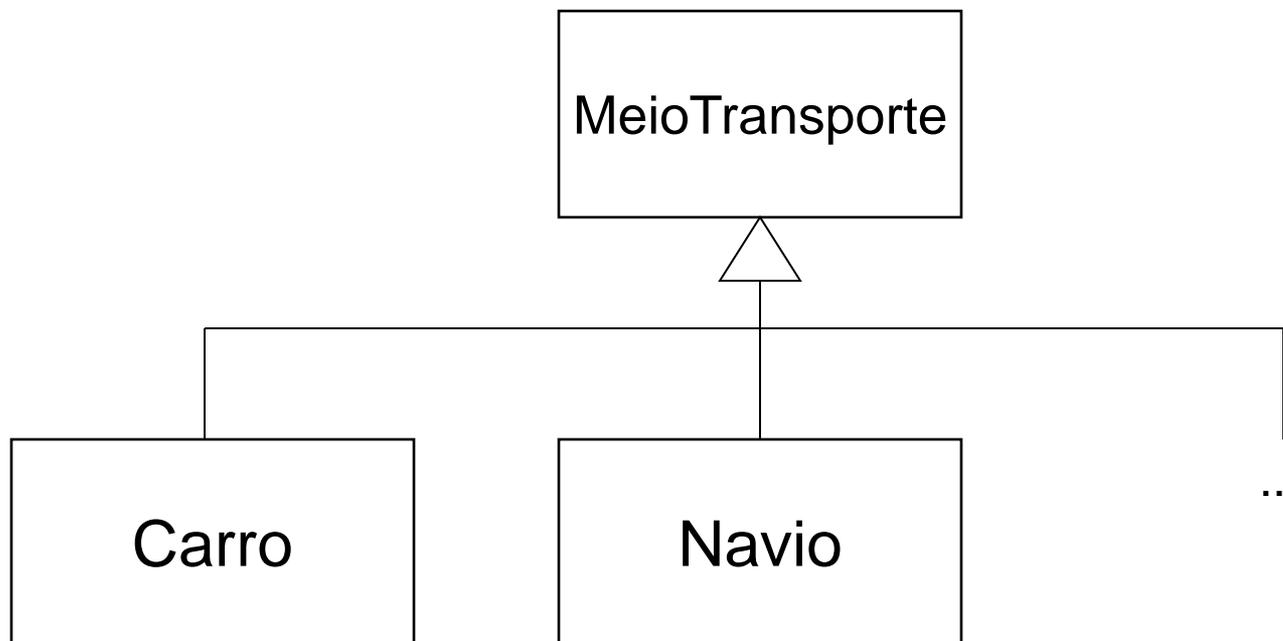
## Generalização:

- Atributos e operações comuns ficam na super-classe;
- Diferenças vão para as sub-classes que herdam da super-classe atributos e operações comuns;
- Lê-se “é um tipo de”



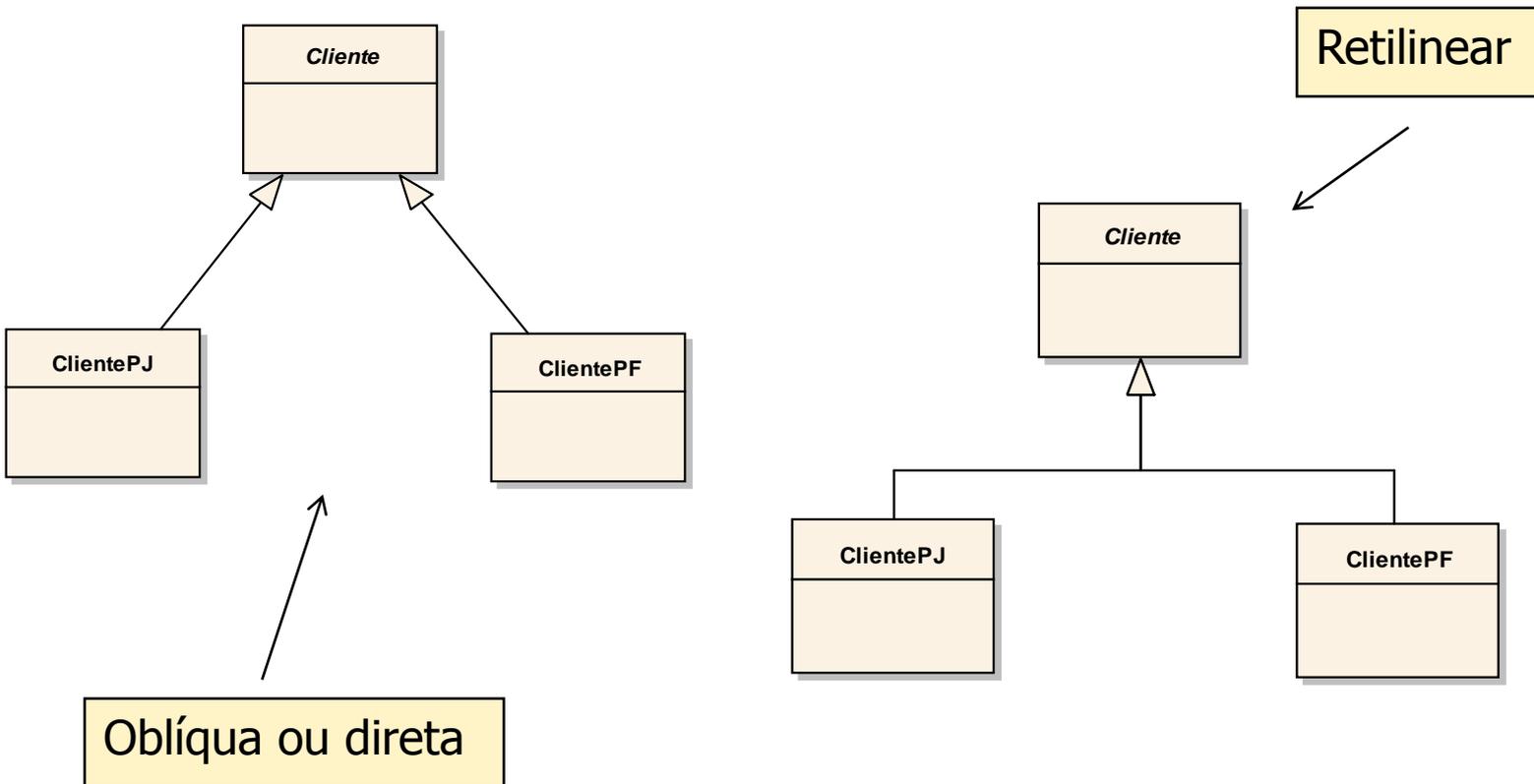
# Diagrama de Classes – Generalização

Outro exemplo:



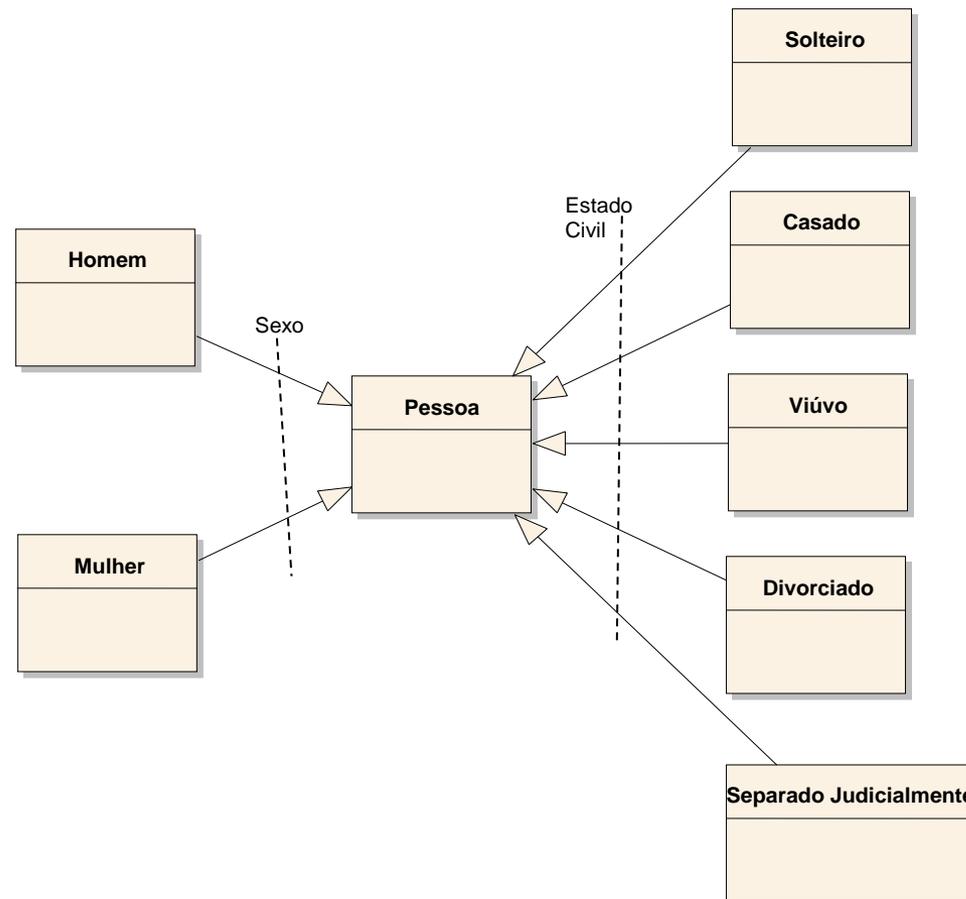
# Diagrama de Classes – Generalização

- Formas de Apresentação (semanticamente idênticas, segundo a UML)



# Diagrama de Classes – Generalização

## ■ Conjuntos de Generalização





# Diagrama de Classes – Generalização

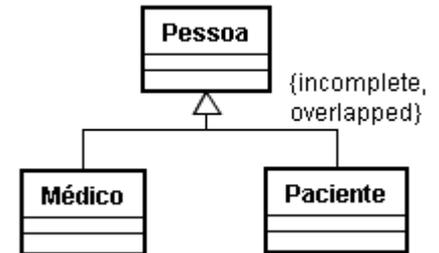
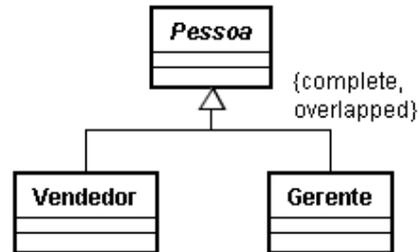
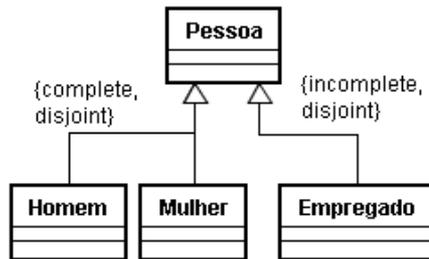
## ■ Partições

- Completo (*complete*): as especializações geram TODAS as instâncias dos objetos (cobertura total).
- Incompleto (*incomplete*): objetos podem ser instâncias das especializações ou da generalização (cobertura parcial).
- Disjunto (*disjoint*): instâncias são de um tipo OU (exclusivo) de outro.
- Sobreposto (*overlapping*): instâncias podem ser de um tipo E de outro (s).

Exemplos a seguir:

# Diagrama de Classes – Generalização

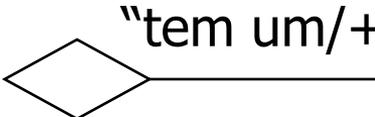
## ■ Partições (exemplo)



# Diagrama de Classes – Agregação

## Agregação:

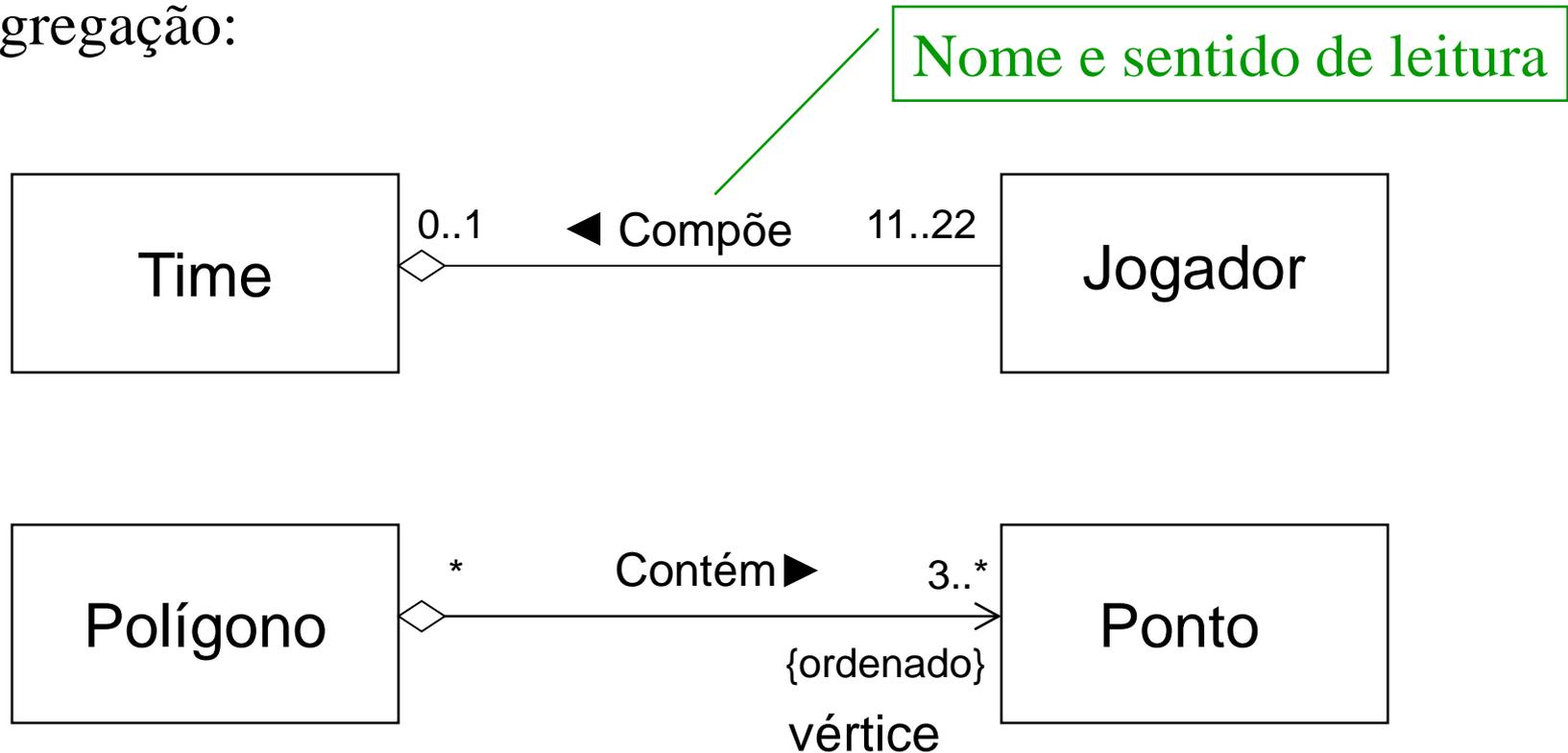
- É o relacionamento “todo-parte”;
- Provida de quase nenhuma semântica (“placebo de modelagem” - Rumbaugh);

- Lê-se  “tem um/+”

- Exemplos:

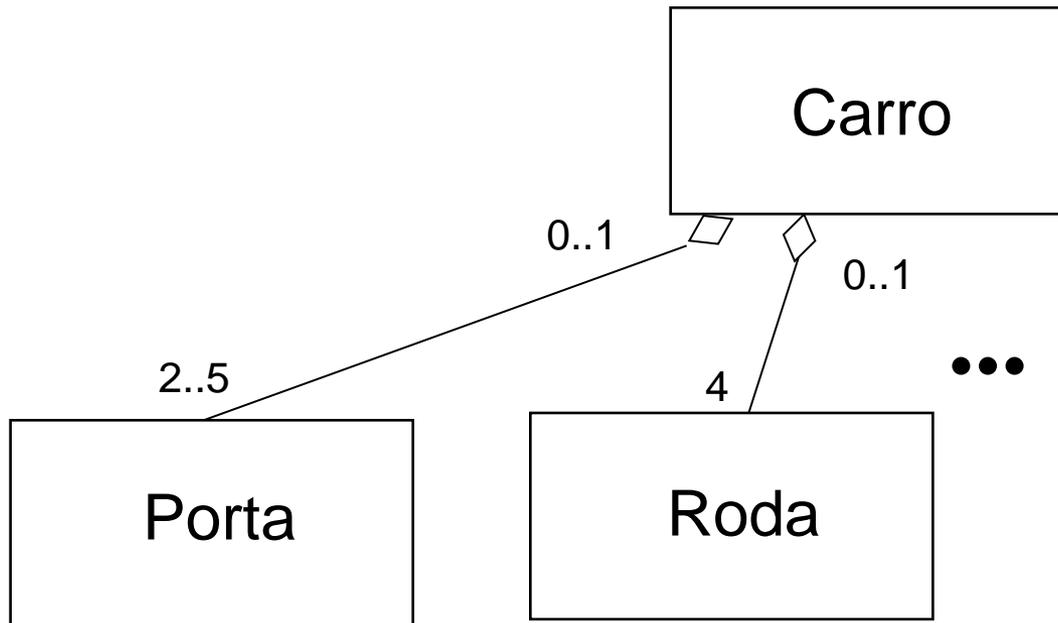
# Diagrama de Classes – Agregação

Agregação:



# Diagrama de Classes – Agregação

Agregação:





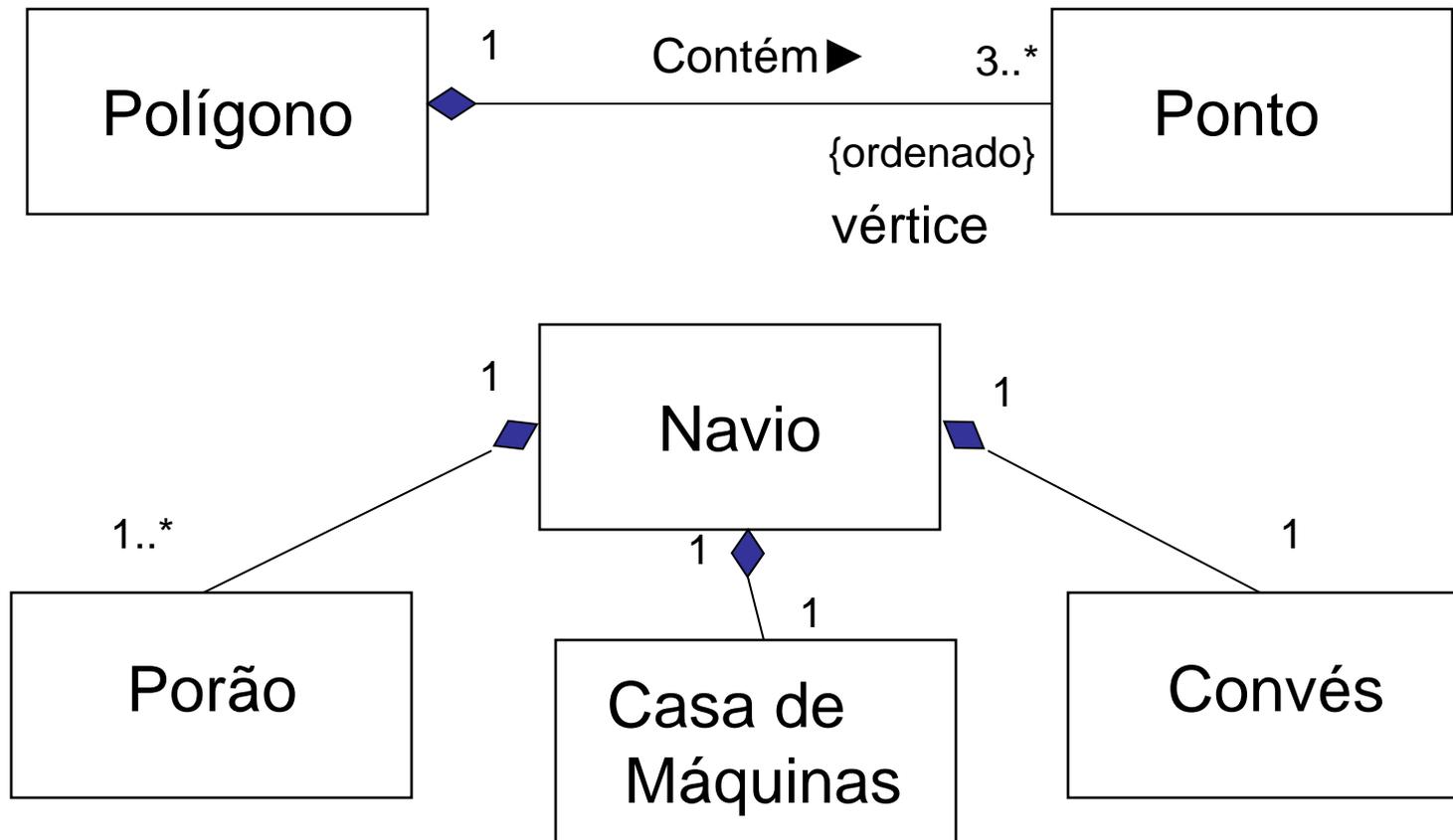
# Diagrama de Classes – Composição

## Composição (ou agregação composta):

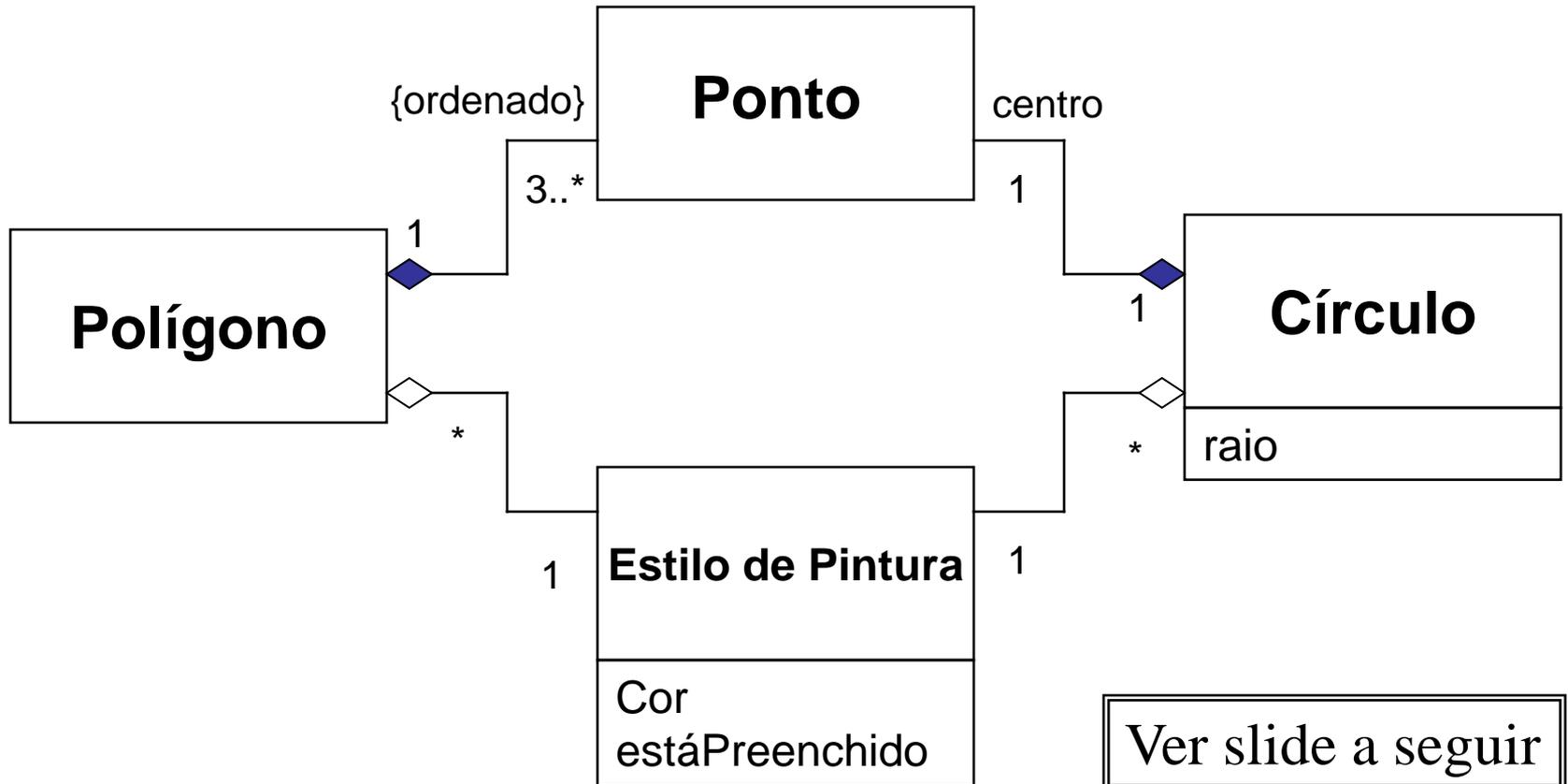
- Uma variedade mais forte de agregação (também é relacionamento todo-parte) ;
- Um objeto pode pertencer a, no máximo, um todo (0..1);
- As partes morrem com o todo (remoção do todo implica na remoção, em cascata, das partes);
- Exemplos:

# Diagrama de Classes – Composição

Composição:



# Diagrama de Classes – Composição



Ver slide a seguir



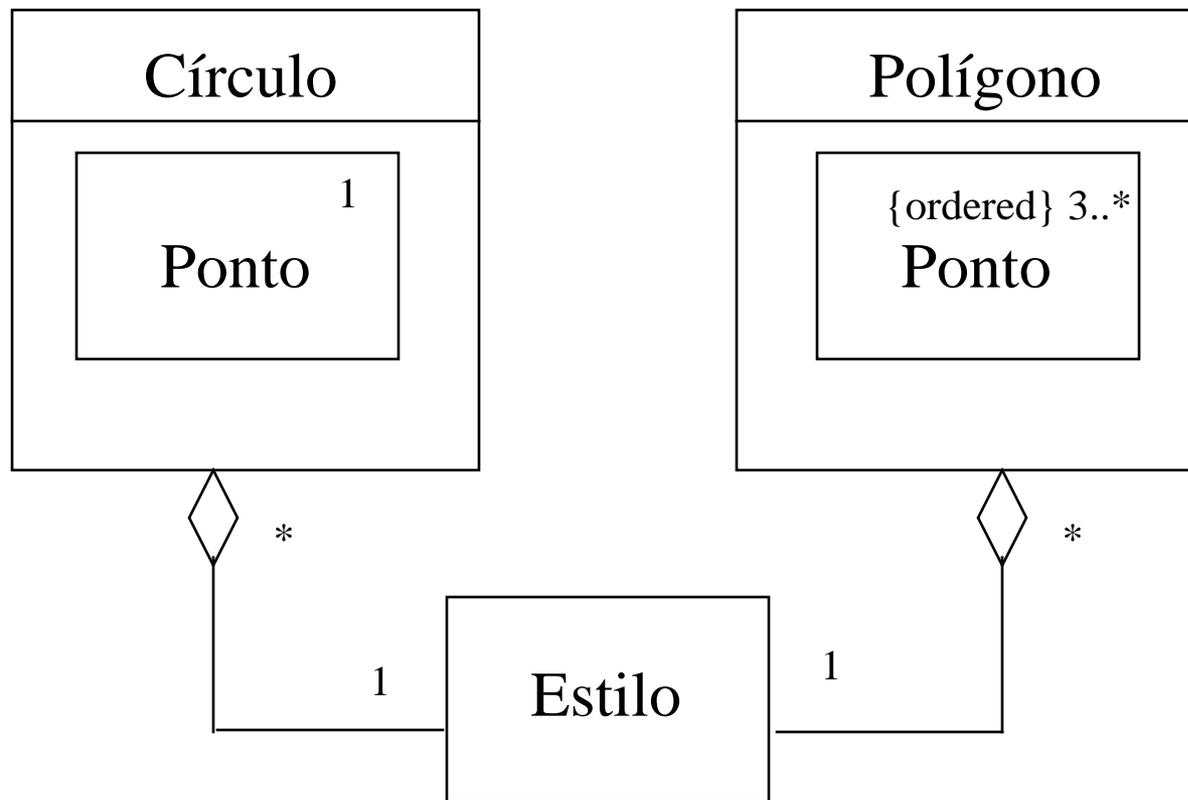
# Diagrama de Classes – Composição

No slide anterior podemos observar:

- Um ponto pode ser vértice de um polígono mas não pode ser (ao mesmo tempo) centro de um círculo;
- Um mesmo estilo pode estar associado (ao mesmo tempo) a nenhum ou vários polígonos e a nenhum ou vários círculos;
- A remoção de um polígono provoca a remoção de seus vértices mas não provoca a remoção do estilo associado. Analogamente, o mesmo para o círculo e seu centro.

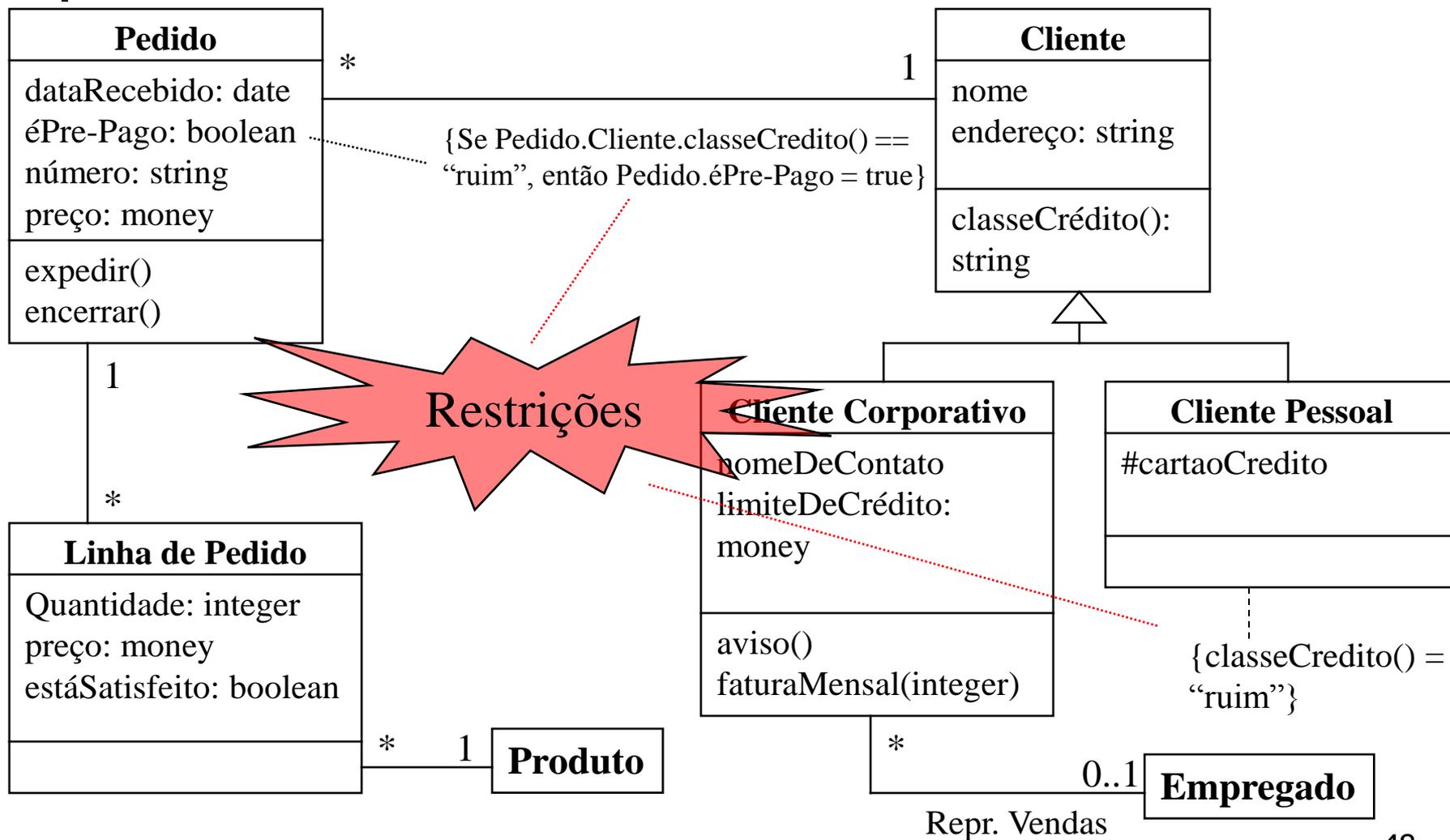
# Diagrama de Classes – Composição

Uma notação alternativa:





# Diagrama de Classes – Restrições





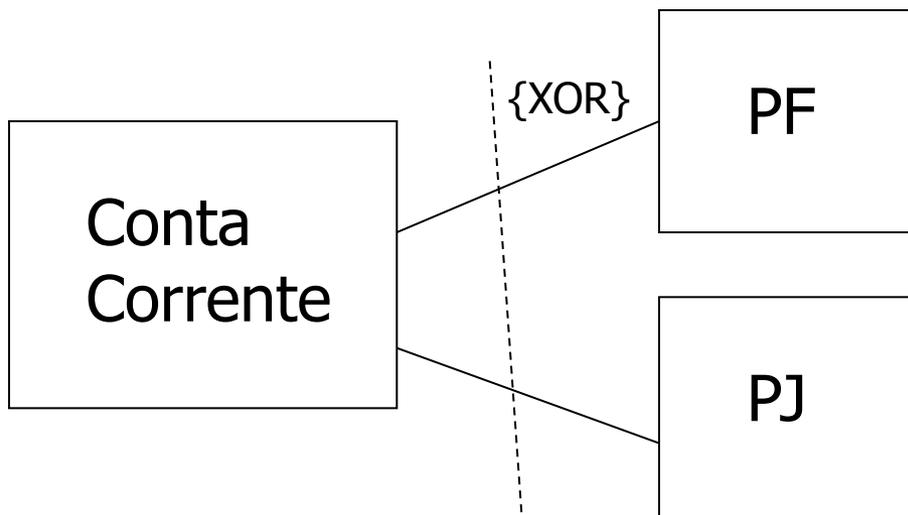
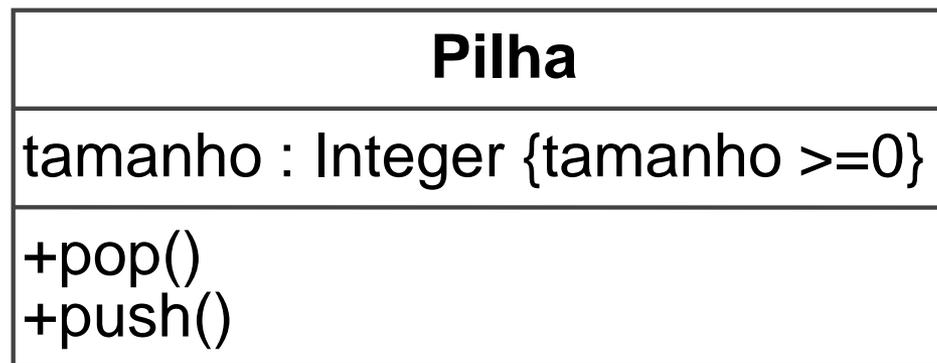
# Diagrama de Classes – Restrições

## Restrições:

- Precisam ser capturadas e o diagrama de classes é um bom lugar para isso;
- Única regra: colocação da restrição entre {};

# Diagrama de Classes – Restrições

- Exemplos:



# Diagrama de Classes – Restrições

- Outra possibilidade: uso de OCL (*Object Constraint Language* – Linguagem de Restrição de Objeto – volume a parte na UML 2.X).



# Diagrama de Classes

## Atributos derivados:

- Podem ser calculados de outros atributos (ex.: *idade = **now** - dataNascimento; saldo = saldoInicial - debitos + creditos*);
- Notação: com uma "/" na frente do atributo derivado;
- "/" indica, apenas, que o atributo é derivado na perspectiva de especificação. A relação entre os atributos vem especificada como uma restrição;
- O programador pode decidir implementar a derivação de outra maneira (Ex.: *duração = fim - início* ou *fim = início + duração*).



# Diagrama de Classes – Classes Abstratas/Interfaces

Interfaces e classes abstratas:

- Interfaces são (apenas as) declarações de operações;
- Denotadas com o nome em ***Itálico*** ou através de um “{abstract}”
- As classes que declaram as operações (apresentam a interface) são classes abstratas;

# Diagrama de Classes – Classes Abstratas/Interfaces

---

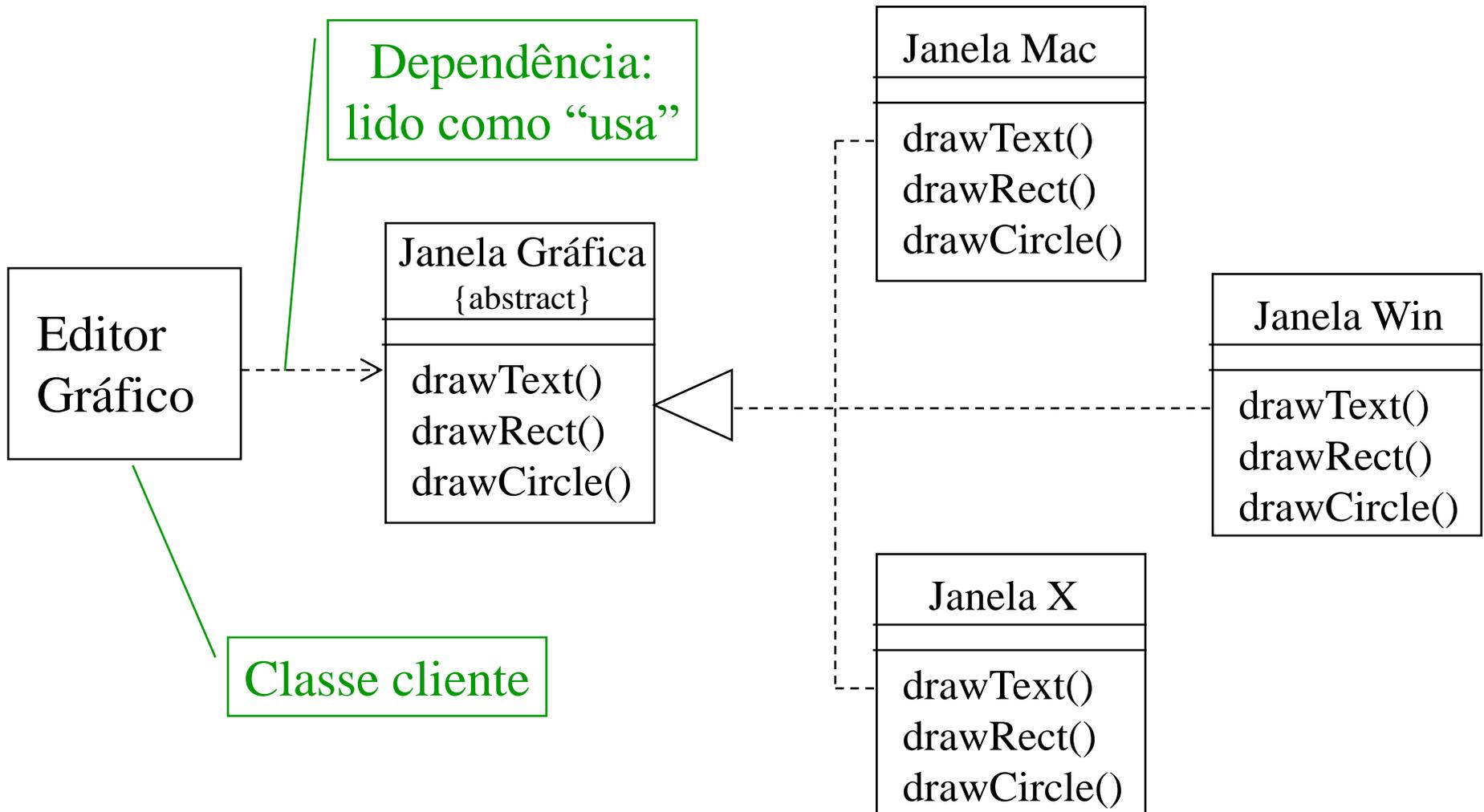
- As implementações ficam por conta das subclasses;
- Principal objetivo: isolar o “o que fazer” do “como fazer”. Exemplo:



# Diagrama de Classes – Classes Abstratas/Interfaces

Preciso desenvolver um editor gráfico para vários ambientes (Windows/Intel, Sun(X), Mac, etc.). Quero isolar os aspectos conceituais da aplicação dos aspectos de implementação (aspectos de h/w e do ambiente operacional) de cada ambiente. Suponhamos que meu editor gráfico utilize as primitivas gráficas `drawText(p: ponto, texto: string)`, `drawRect(p1, p2: ponto)` e `drawCircle(c: ponto, r: integer)`.

# Diagrama de Classes – Classes Abstratas/Interfaces



# Diagrama de Classes – Classes Abstratas/Interfaces

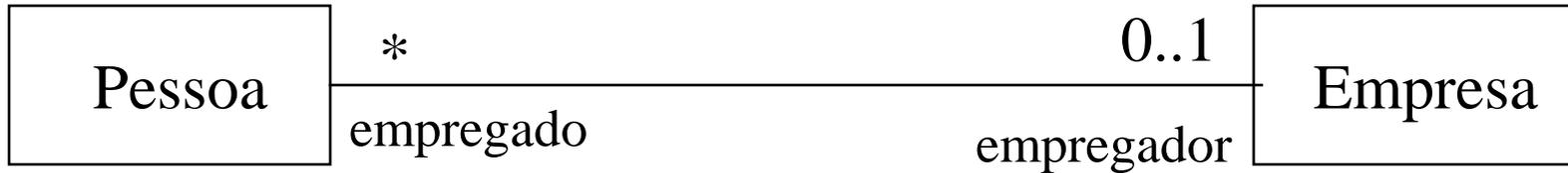
## Observações:

- A dependência indica que as funcionalidades que podem ser utilizadas pela classe cliente dependem das funcionalidades disponibilizadas pela classe abstrata.
- Classes abstratas podem implementar alguns métodos. Interfaces (puras) não.



# Diagrama de Classes – Classes de Associação

Um problema:



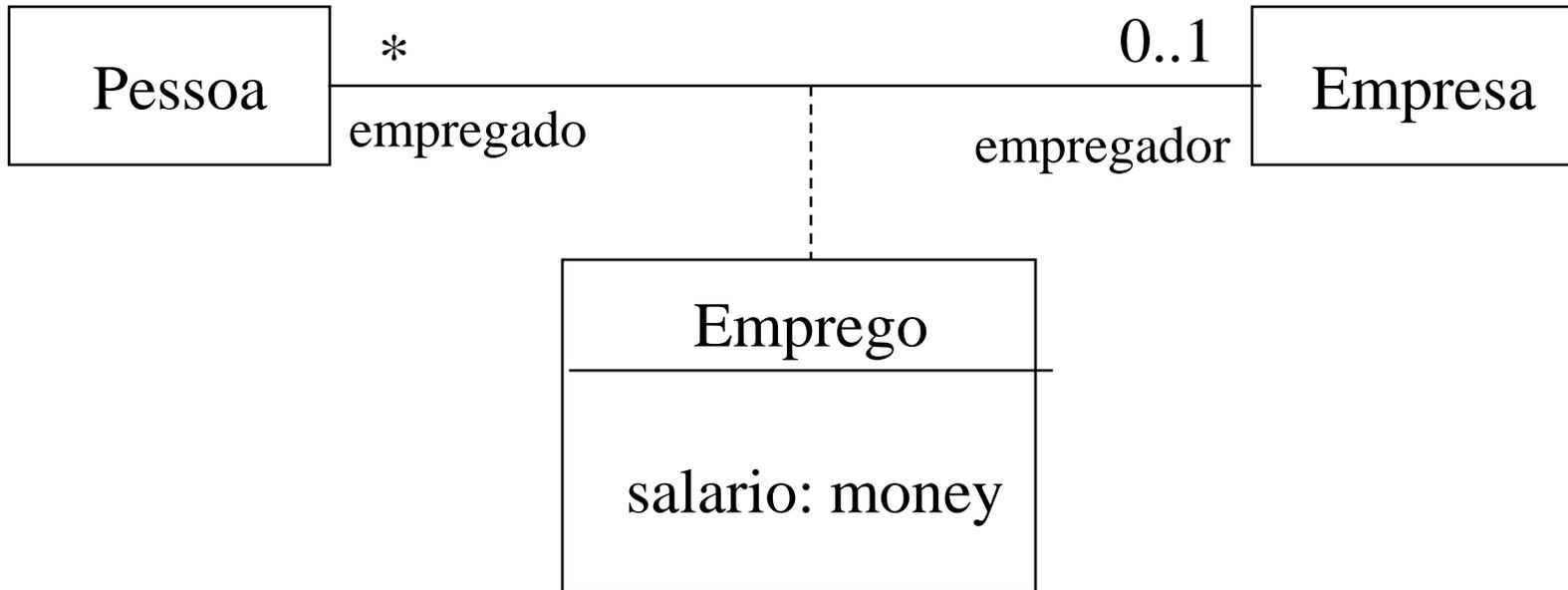
## Problema:

Onde guardo as informações do empregado que dizem respeito à empresa para a qual trabalha? Como atributo de Pessoa? Como atributo de Empresa?



# Diagrama de Classes – Classes de Associação

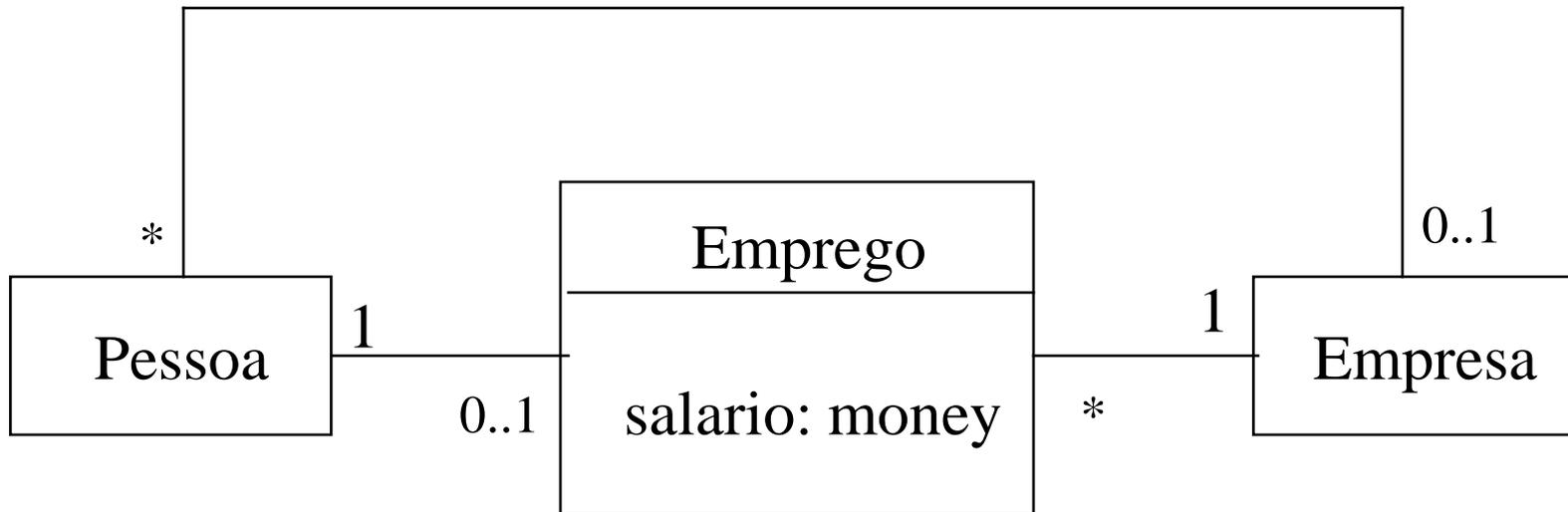
## Solução 1: Classes de associação





# Diagrama de Classes – Classes de Associação

Solução 2:





# Diagrama de Classes – Classes de Associação

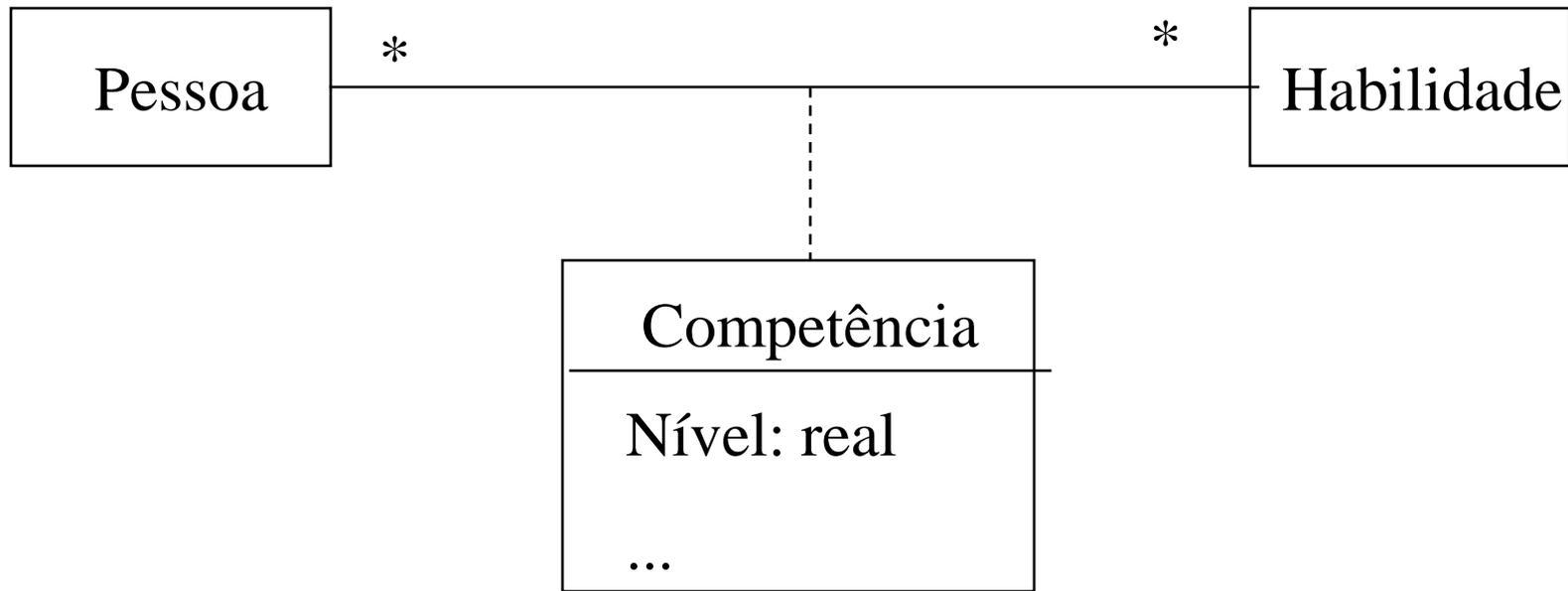
Obs. importante:

Na solução 1 não é possível ter mais de uma ocorrência do objeto emprego para as mesmas instâncias de pessoa e empresa.

Um exemplo onde uma classe de associação é bem empregada ...



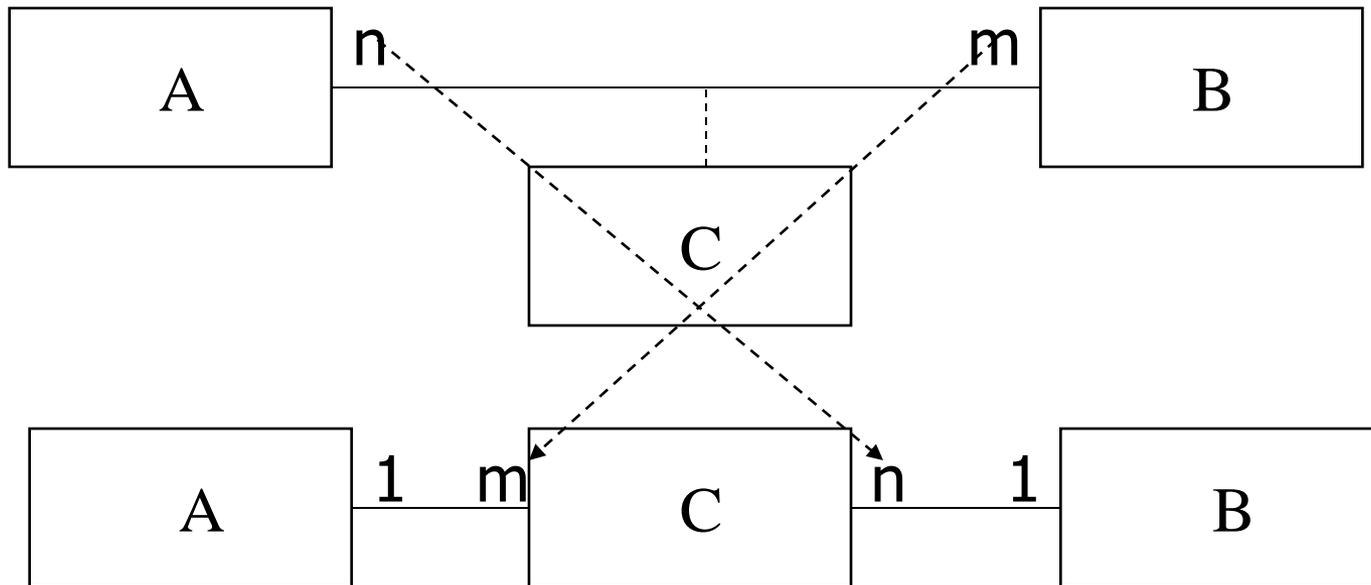
# Diagrama de Classes – Classes de Associação



Não existe mais de um nível de competência de uma mesma pessoa em relação a uma mesma habilidade.

# Diagrama de Classes – Classes de Associação

- Transformando Classe de Associação em Classe “Cheia” (o que pode ser feito sempre):





# Diagrama de Classes

## Palavras-Chave

### Palavras-chave.

- A UML fornece uma linguagem para itens estruturais (classes, interfaces, casos de uso, etc.), comportamentais (interações e máquinas de estados), de agrupamento (pacotes), itens anotacionais (notas/comentários), os blocos relacionais básicos (associações diversas), etc., etc.

# Diagrama de Classes

## Palavras-Chave

- Esses itens permitem a modelagem de quase a totalidade de aplicações. Há casos, entretanto, que é necessário estender a UML;
- Palavras-chave são o núcleo do mecanismo de extensão da UML;
- São usadas quando se necessita de uma construção de modelagem que não existe na UML, mas que é semelhante a algo que já existe (na UML).



# Diagrama de Classes

## Palavras-Chave

- Podem ser de classes, associações ou generalizações;
- São mostradas em texto entre "<<" e ">>". Ex.: <<estende>>, <<inclui>>, <<bind>>
- Podemos pensar em palavras-chave como sub-tipos dos tipos Classe, Associação e Generalização do metamodelo.



# Diagrama de Classes

---

E por fim ...

# Diagrama de Classes – Reconhecendo classes

---

Classes farão parte do diagrama se:

- Forem entidades sobre as quais nos interessa conhecer seu funcionamento e estrutura internos;
- Possuem atributos (armazenam estados);
- Possuem comportamento;
- Possuem responsabilidades;
- Executam operações.



# Diagrama de Classes – Quando usar

Quando usar diagramas de classes:

- O tempo todo, pois são (semanticamente) ricos e são a base de qualquer metodologia OO!

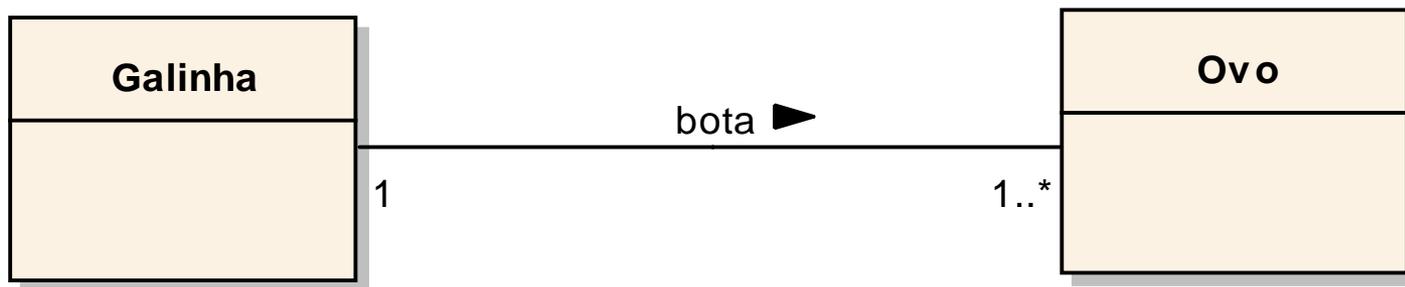
# Diagrama de Classes – Como usar

Como usar (algumas dicas - Fowler):

- 90% das necessidades de notação são atendidas com uma pequena parte da notação. Tente usar o mínimo de recursos;
- Não se perca nas perspectivas; procure separar aspectos conceituais, de especificação e de implementação;
- Não se atenha a detalhes (atributos, operações e visibilidades) fora de hora;

# Diagrama de Classes – Como usar

- Por exemplo, estando no nível conceitual, não se preocupe com questões de
  - Normalização de tabelas (associações muitos-para-muitos);
  - Integridade referencial - a clássica dúvida de quem vai ser instanciado primeiro num sistema de controle de ovos e galinhas poedeiras : o ovo ou a galinha?



## Diagrama de Classes – Mais dicas

- Não modele classes muito grandes (com muitas responsabilidades) nem muito pequenas;
- Classes muito grandes são mais difíceis de serem reutilizadas;
- Considerando um conjunto de classes, procure responsabilidades, atributos e operações que sejam comuns a duas ou mais classes. Pense na possibilidade de colocá-los em uma classe mãe;
- Redes de heranças não deverão ser muito profundas (> 5 níveis).



# Diagrama de Classes

---

## Exercícios



# Diagrama de Classes – Exercícios

## Exercício 1: Ambiente acadêmico

Cada escola da comunidade Alfa é dividida em um ou mais departamentos (letras, matemática, etc.). Um departamento é chefiado por um de seus professores, mas há casos em que esse cargo está vago. Não há acúmulo de chefia. Professores podem estar alocados em um ou mais departamentos. Um departamento pode ser criado sem que haja professores a ele alocados. Um aluno pode estar matriculado em mais de uma escola e pode frequentar mais de um curso na mesma escola. Escolas podem não ter alunos matriculados. Cada departamento tem seu conjunto específico de cursos (pelo menos um). Cada curso pode ser ministrado por um ou mais professores. Cada professor pode ministrar qualquer número de cursos.

# Diagrama de Classes – Exercícios

## Exercício 2: Sistema de Bibliotecas

Cada escola da comunidade Alfa possui pelo menos uma biblioteca. O sistema de controle da biblioteca deverá possuir usuários que se classificam em usuários comuns e usuários funcionários. Para todo e qualquer usuário é necessário que se tenha o seu nome no cadastro. Qualquer usuário é capaz de fazer consultas por autor, por título e por assunto. Usuários comuns têm um número de registro que é alfanumérico.

# Diagrama de Classes – Exercícios

(Sistema de Bibliotecas – cont.):

Usuários funcionários possuem um nome de *login* e uma senha de acesso ao sistema. Usuários comuns podem solicitar renovação de empréstimo diretamente via sistema. Os empréstimos, devoluções, reservas e cobrança de multa por atraso são atribuições exclusivas dos funcionários da biblioteca. O mesmo acontece para as operações de inclusão de uma nova obra ou exemplar e funções típicas de consulta a empréstimos e elaboração de estatísticas diversas.

# Diagrama de Classes – Exercícios

(Sistema de Bibliotecas – cont.):

Para cada obra armazena-se o ISBN, o(s) autor(es), o título, a editora, o assunto e a edição. Os exemplares possuem a data de inclusão no acervo e uma marcação de disponibilidade (as bibliotecas só cadastram obras que constam do acervo).

As bibliotecas emprestam livros aos alunos e aos membros da comunidade. Alunos podem retirar até 2 títulos; membros da comunidade podem retirar apenas um título por vez.

# Diagrama de Classes – Exercícios

---

Outros Exercícios: Q-Sereia/ManutAir