UML: Diagramas de Classes

Desenho de Bases de Dados Relacionais com UML

Fundamentos de Bases de Dados
(FBD)

Licenciatura em

Engenharia de Telecomunicações e Informática (ETI)

Autoria:
Pedro Ramos, José Farinha

Docente:
José Farinha

Diagramas de Classes UML
Índice

- Conceitos Básicos
- Associações
- Classes Associativas
- Agregações
- Composições
- Generalizações
- Atributos vs. Associações *N-para-1*
- Associações n-árias
- Associações singulares (uma classe)
- Relações de Dependência
- *Roles*
- Navegação
- *Packages*

Objectos

- Objecto: Qualquer coisa ou acontecimento do universo que queremos registar e que tem:
  - Uma identificação
    - Valor que permite diferenciar o objecto de todos os outros
  - Um estado
    - Conjunto de valores que nos dão informação acerca das características do objecto
  - Comportamento
    - Conjunto de acções que o objecto sabe realizar

Objecto: Cliente José Silva

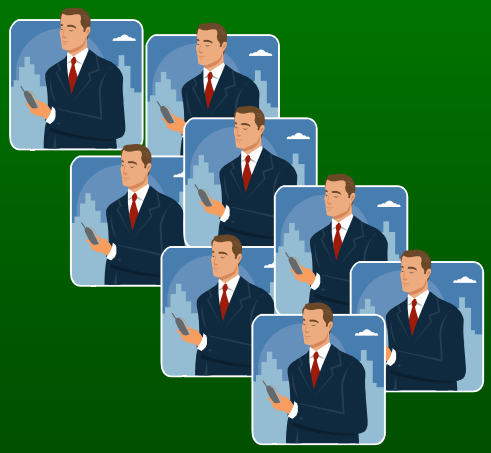
- É distinto de todos os outros clientes da empresa pois tem o número 484848
- Tem o nome “José Silva”, morada “R. de cima…”, nº contribuinte “8242424”, ...
- Operações: encomendar produto, pagar factura, alterar morada, ...

Objectos

- Não têm necessariamente que corresponder a entidades com representação física
- Um **conceito abstracto** (p.ex, um departamento) pode ser um objecto, desde que seja relevante para o domínio em causa.

Classes

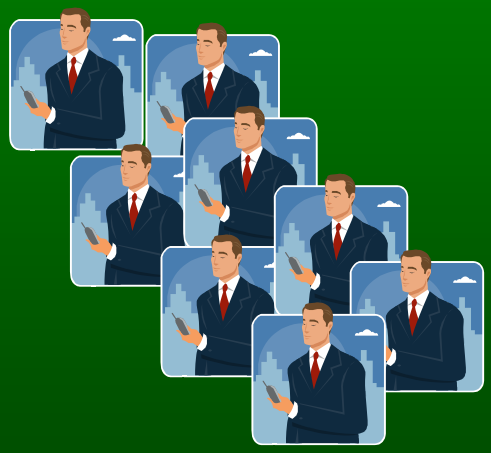
Classe: conjunto de objectos que partilham o mesmo meio de identificação, propriedades de estado, comportamento, relações e semântica.



Classe dos clientes

- Todos distintos uns dos outros
- Todos têm nome, morada, nº contribuinte, ...
- Todos estão aptos para realizar as mesmas acções: encomendar produto, pagar factura, alterar morada, ...
- Todos se relacionam com os mesmos tipos de objectos (p.ex, com os produtos que adquirem).
- Representam a realidades da mesma natureza (tem a mesma semântica)

Classe: Representação gráfica

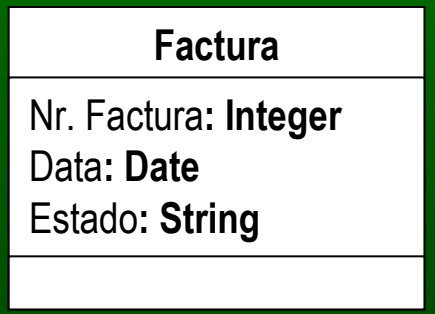


Classe dos clientes

Cliente	Nome (único no domínio)
Nr. contribuinte Nome Morada	Atributos
Encomendar produto ( ) Pagar factura ( ) Mudar de morada ( )	Operações

Atributos

- Um atributo numa classe representa uma característica típica dos objectos dessa classe
- Pode assumir qualquer valor, se não houver mais nenhuma especificação relativamente ao atributo
- Pode especificar-se um tipo de dados para um atributo
  - Neste caso, os valores que podem ser atribuídos ao atributo estão condicionados à compatibilidade com o tipo

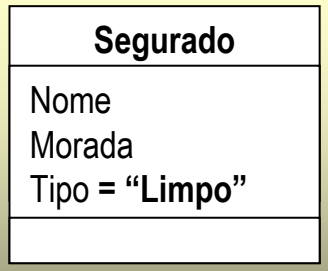


Atributos: obrigatoriedade de preenchimento

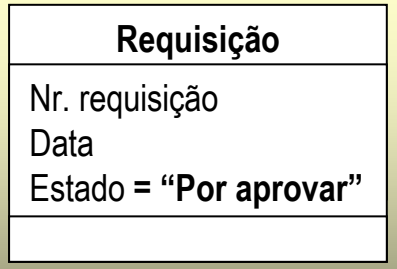
- Em UML, um atributo é de preenchimento opcional:
  - Em cada novo objecto que seja criado, o dito atributo poderá ser ou não preenchido.
- Em desenho de base de dados é importante identificar a obrigatoriedade de preenchimento
  - Normalmente feito apenas sobre o modelo relacional
  - Se for considerado muito relevante colocar essa informação no diagrama de classes UML, indicar em caixa de comentário UML

Atributos: valor por omissão

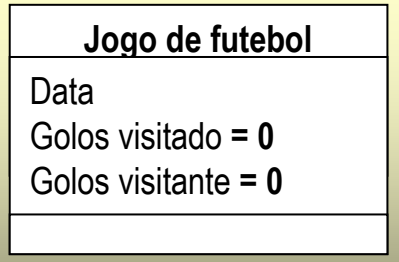
- Em UML pode especificar-se um valor por omissão (*default value*) para um atributo
- Mais adequado para aplicar em situações em que existe um valor inicial



Os novos segurados começam por não ter participações.



Uma requisição é criada por um subordinado e mais tarde aprovada pela chefia



Neste caso, torna-se não necessário fornecer o número de golos quando se cria o jogo.

- Não é muito adequado para modelar o conceito de *valor mais frequente*

Atributos de identificação 1

- Ao definir os atributos de uma classe, deverá incluir-se sempre um (ou mais) atributo(s) que possa(m) funcionar como mecanismo de identificação dos objectos dessa classe.
- Isto é, um (ou mais) atributo(s) para o(s) qual(is):
  - todos os objectos têm valor;
  - o valor nesse atributo (ou conjunto de atributos) garantidamente não se repete em quaisquer dois objectos.

Atributos de identificação 2

- Em certas classes, não se conseguem apurar atributos naturais para este efeito.
- Especificamos um atributo adicional, cujos valores serão "artificialmente" atribuídos a cada objecto, sem causar repetições;
- Este atributo diz-se um **Identificador Interno**, **Identificador Único** ou **Identificador de Objecto** (***OI, Object Identifier***) e é frequente receber nomes do género:

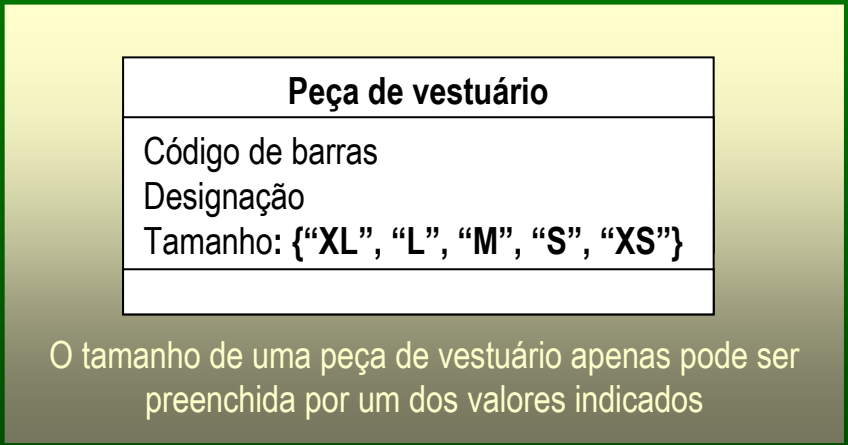
  Número [de ...]

  Código [de ...]

  Id
- Por razões de performance, no modelo lógico e/ou físico da base de dados poderá introduzir-se um atributo destes mesmo que exista uma forma de identificação natural na classe
- Num diagrama de classes UML um atributo deste género apenas deverá ser indicado se não existir uma forma de identificação natural na classe

Atributos enumerados

- Atributos que apenas podem assumir valores entre um certo conjunto de opções
- Exº de especificação na própria classe



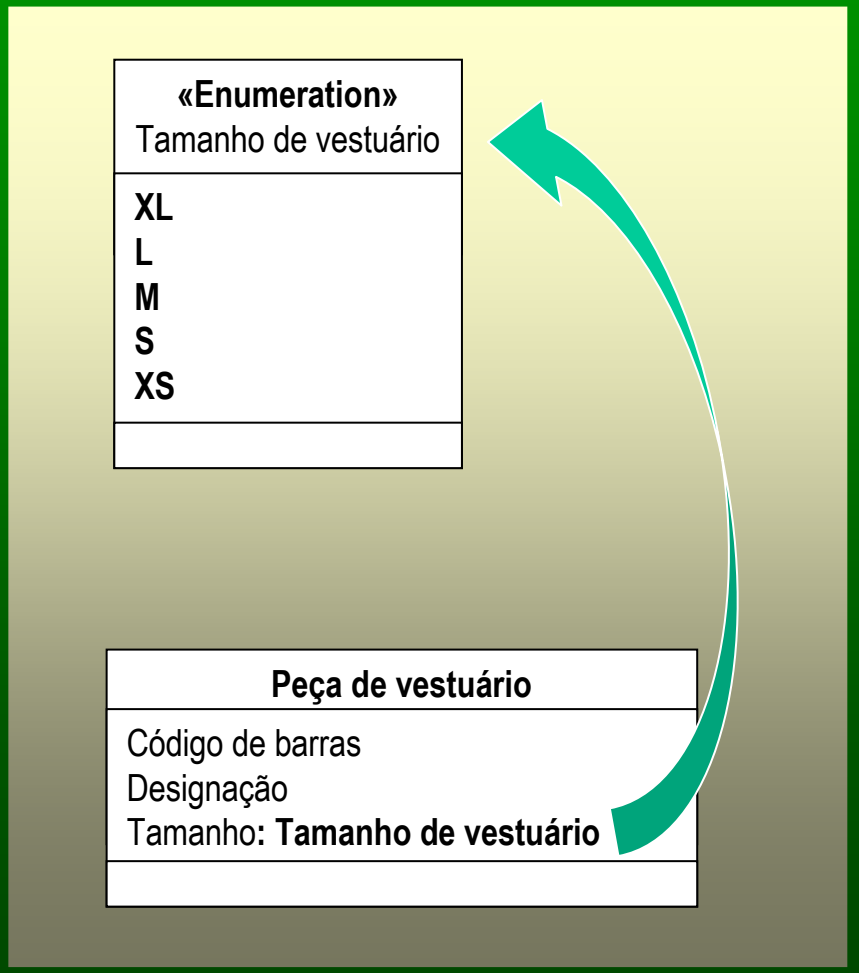
Atributos enumerados

- Se o conjunto de opções é usado em mais que uma classe

  ...ou mesmo se o conjunto de opções é muito extenso, tornando a representação gráfica da classe muito larga

- Pode definir-se um tipo de dados enumerado, que pode ser partilhado por vários atributos

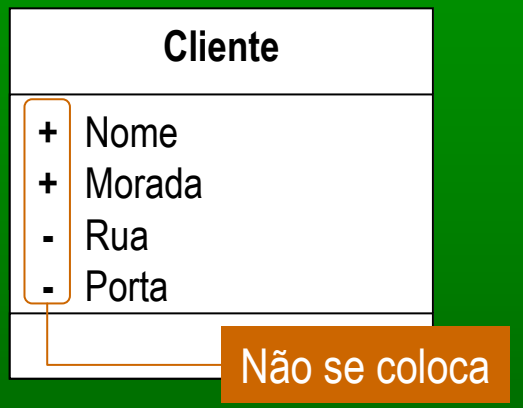
Representação gráfica:

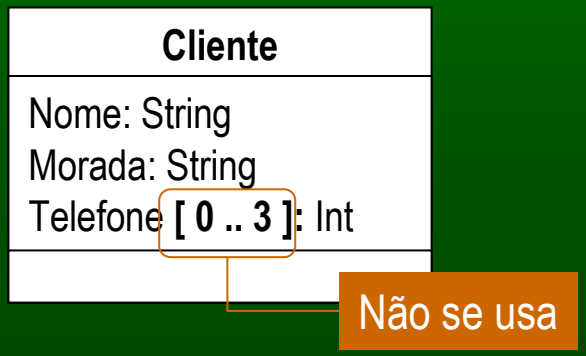


Desusos de UML para desenho de bases de dados relacionais

Para efeitos de desenho de base de dados relacional:

- Não se especificam as visibilidades – público/protegido/privado – dos atributos: todos se assumem públicos;
  - Se pretendêssemos desenhar uma base de dados orientada por objectos, tal já seria especificado;
- Relativamente a um atributo, não se faz especificação de multiplicidades superiores a 1, pois:
  - O modelo relacional não permite que, num registo, um atributo possua mais do que um valor
  - Não se pretende obter um modelo puramente orientado por objectos

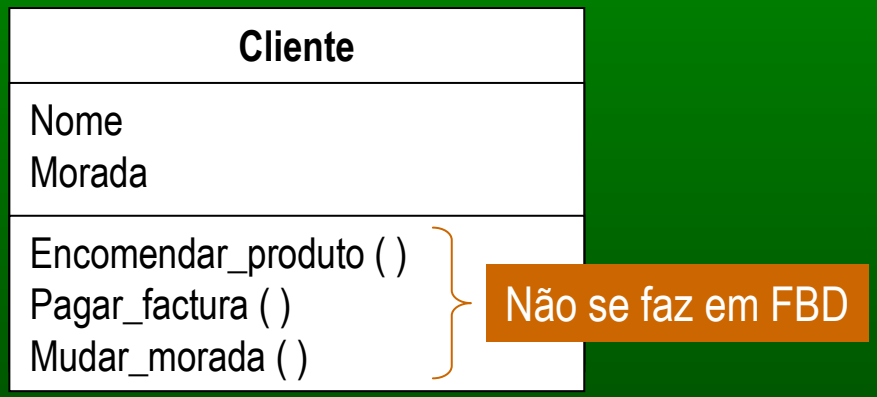




Desusos de UML em FBD

Não se especificam as operações das classes

Mas poderá fazer-se, para especificação de *stored procedures* ou *triggers* muito directamente relacionados com determinada classe



Relações 1

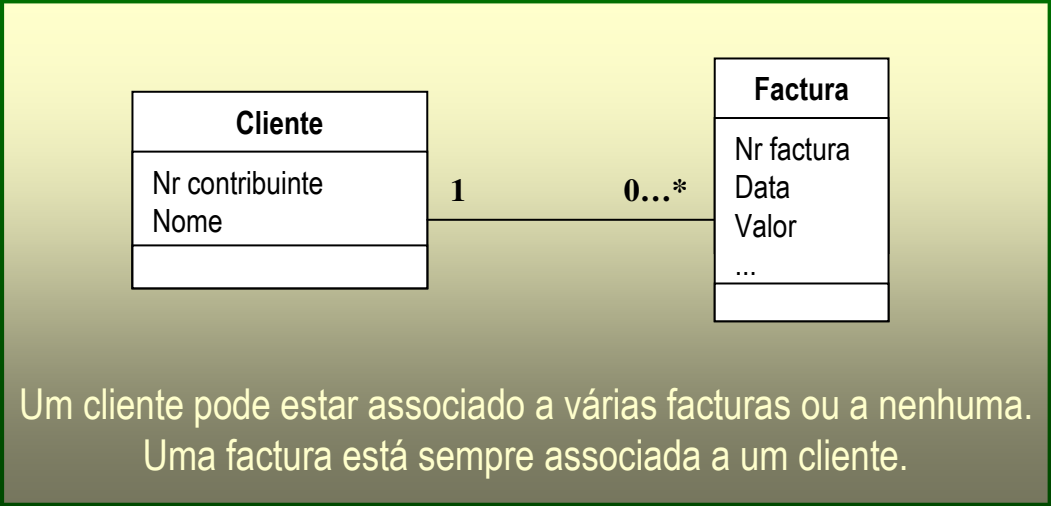
- Em qualquer sistema existem objectos que se relacionam entre si.
  - Sistema universitário
    - Objectos: alunos, cursos, exames;
    - os alunos relacionam-se com os cursos que frequentam e com os exames que fazem.
  - Sistema bancário
    - objectos – clientes, contas, balcões;
    - os clientes relacionam-se com as contas que possuem e as contas com os balcões em que estão sediadas.
- As ligações entre objectos relacionados também são informação
  - ☞ Quando há interesse em conhecer as ligações entre os objectos do sistema, especificam-se relações entre as classes desses objectos

Relações 2

- Em UML existem os seguintes tipos de relações, que expressam diferentes semânticas de interligação entre classes:
  - Associação
    - Tem dois casos especiais:
      - Agregação
      - Composição
  - Generalização
  - Relação de dependência

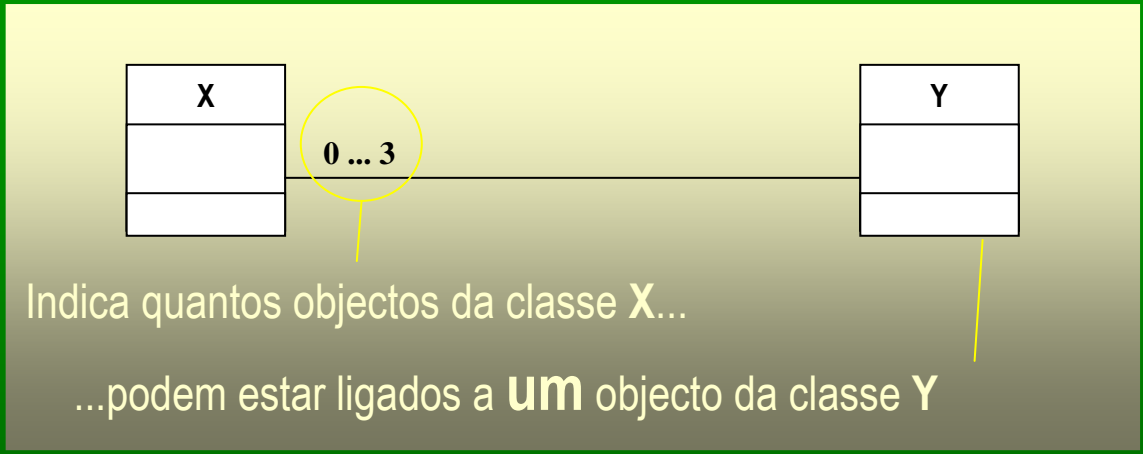
Associações

Uma associação é uma relação que permite especificar que objectos de uma dada classe se relacionam com objectos de outra classe, sendo importante saber para cada objecto quais os objectos que lhe estão associados.



Um cliente pode estar associado a várias facturas ou a nenhuma.
Uma factura está sempre associada a um cliente.

Multiplicidade das Associações

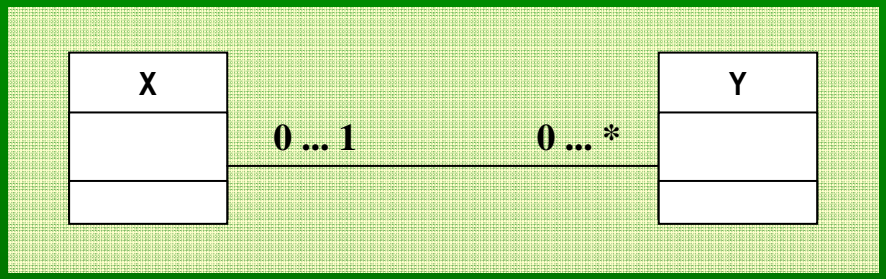


Algumas hipóteses:

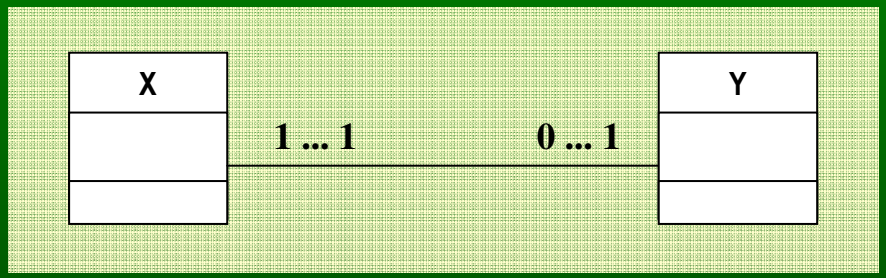
0...1 → zero ou 1	**1...1** → um e apenas um pode representar-se: **1**
0...* → zero ou vários, sem limitação de quantidade	**1...*** → no mínimo 1
0...3 → de zero a três	**1...3** → de 1 a 3

Multiplicidade das Associações

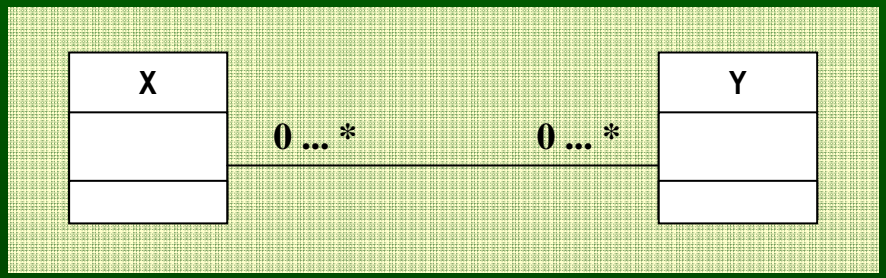
... infinitas combinações possíveis que são vulgar designarem-se como:



- Um-para-muitos
  ...ou Um-para-N

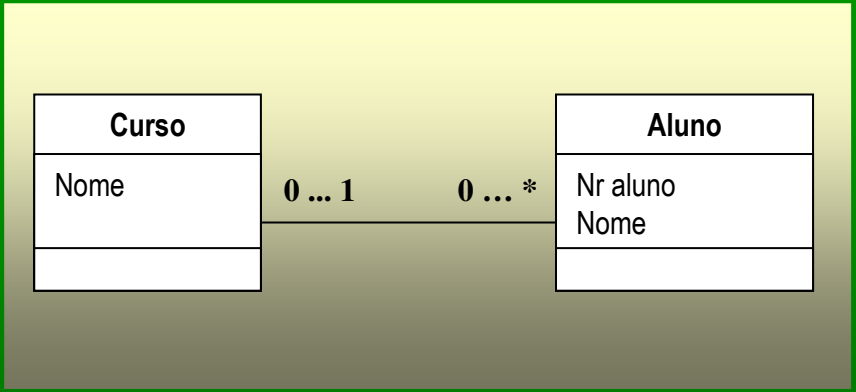


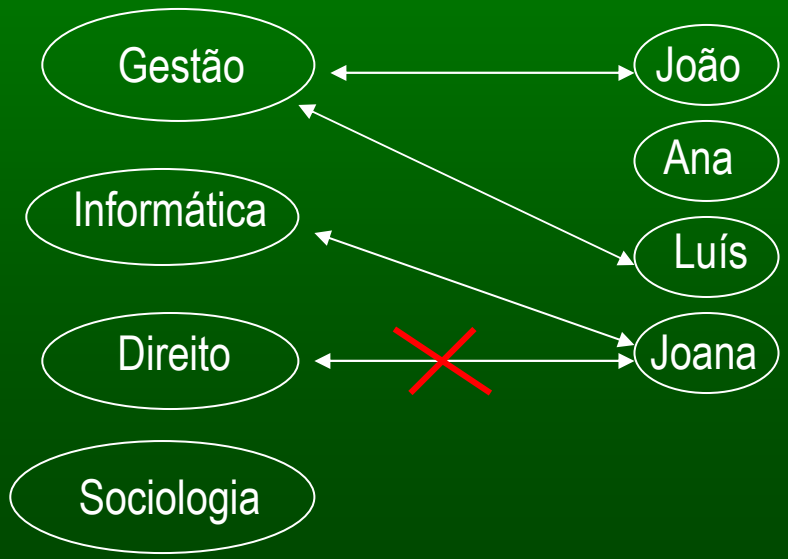
- Um-para-um



- Muitos-para-muitos
  ...ou M-para-N

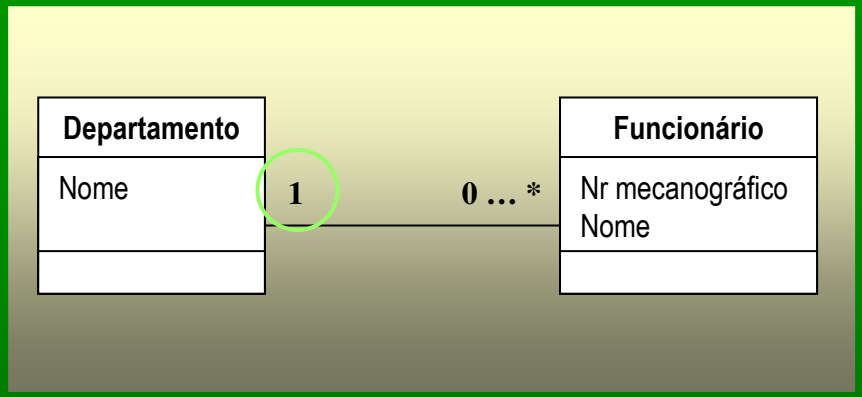
Associação *um-para-muitos* 1

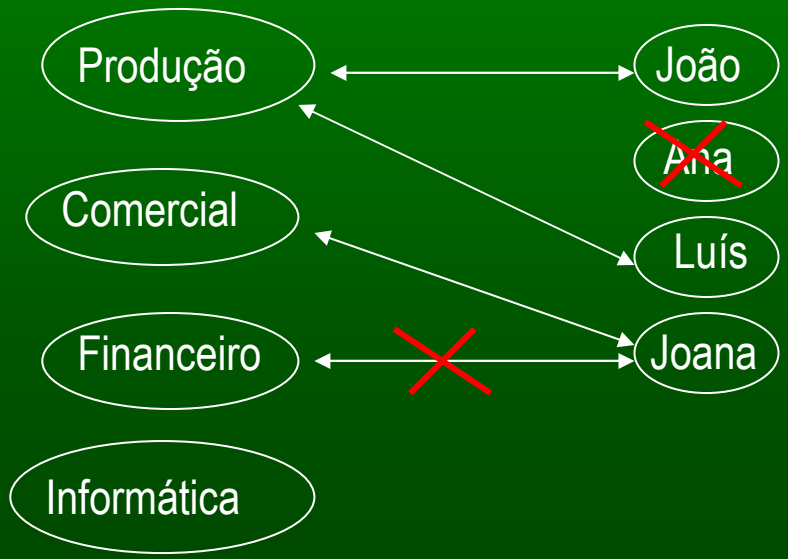




- *Semântica*
  - Um aluno pode estar associado a (inscrito em) um e apenas um curso
  - A um curso podem-se associar vários ou nenhum aluno.
- *Funcional*
  - Dado um aluno é possível determinar em que curso está inscrito, e
  - Dado um curso é possível identificar os seus alunos.

Associação *um-para-muitos* 2

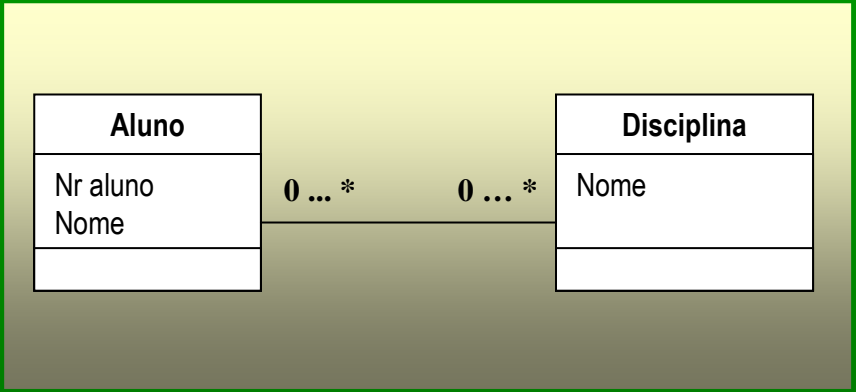


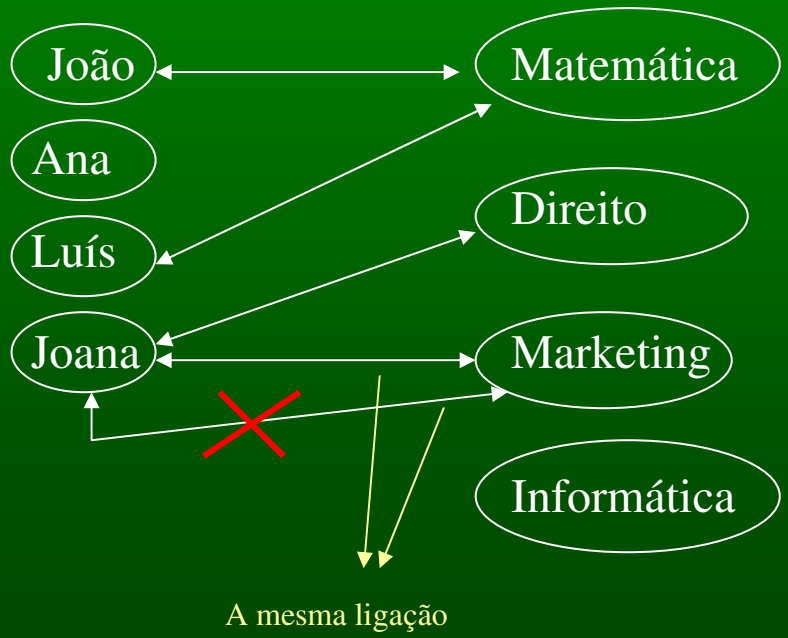


- *Semântica*
  - Um funcionário tem necessariamente que estar associado a um departamento

    Não é possível introduzir um funcionário no sistema sem que seja indicado o departamento a que pertence
- *Funcional*
  - Dado um funcionário é possível determinar em que departamento ele trabalha, e
  - Dado um departamento é possível identificar os seus funcionários.

Associação *muitos-para-muitos* 3

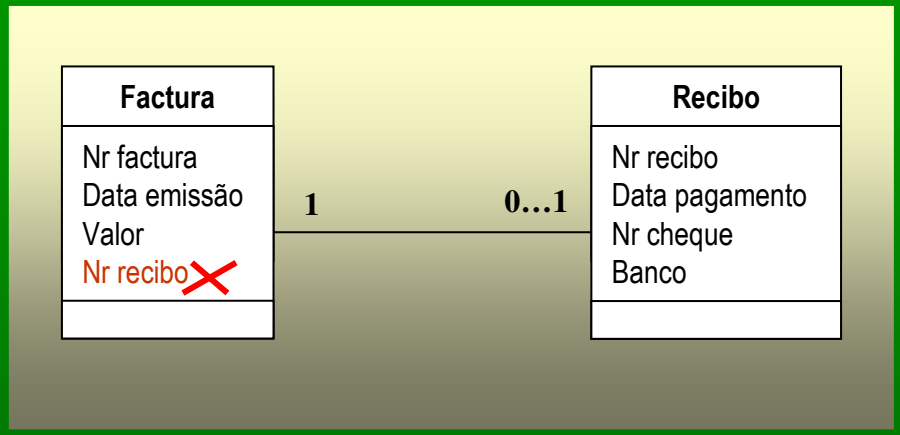




Um objecto não pode estar duplamente associado a outro objecto (Joana / Marketing).

- À semelhança das classes (em que os objectos são distintos), as associações também têm que ter ocorrências distintas.
- –Não há nada que distinga as duas ligações entre Joana e Marketing;
- –O sistema de base de dados deverá considerar a 2ª ligação como redundante: Não interessa registar duas vezes que Joana e Marketing estão ligados
- –Se interessa, então a associação muitos-para-muitos não é representação correcta

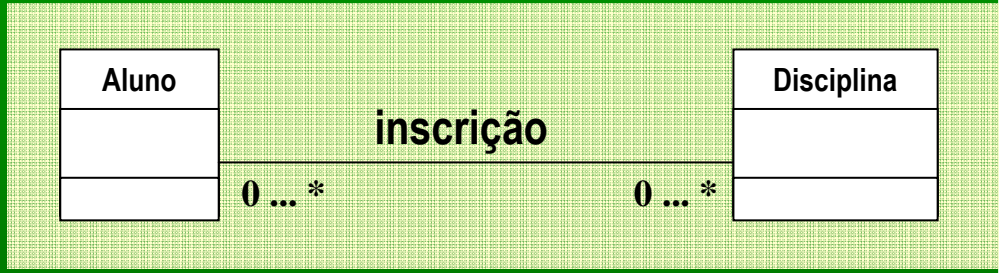
Associação *um-para-um*



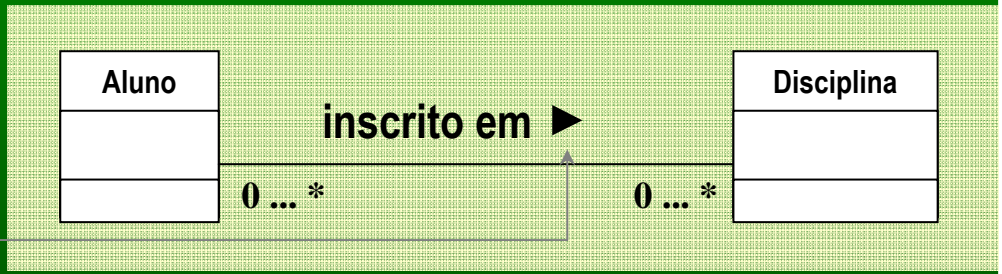
- É a associação que atribui um número de recibo à factura.
  - Caso contrário, uma factura poderia ficar associada a dois números de recibo (o indicado no atributo da factura e o indicado no objecto *Recibo* ao qual a factura estivesse ligada).
  - O atributo *Nr de recibo* na classe *Factura* é redundante – não deve ser especificado
- Apesar de haver uma correspondência um a um, não se deve especificar apenas uma classe, pois cada classe representa uma realidade
  - Inclusive, devia haver a classe Cheque.
  - Mais alguma?
- Pelas multiplicidades percebe-se que uma factura será necessariamente introduzida antes do respectivo recibo (quanto muito, simultaneamente)

Nomes de associações 1

- As associações podem (e devem) ter nomes
  - Substantivo →

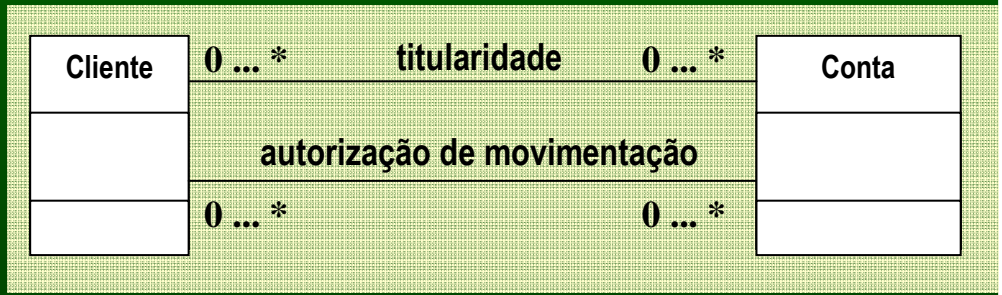


  - Verbo →



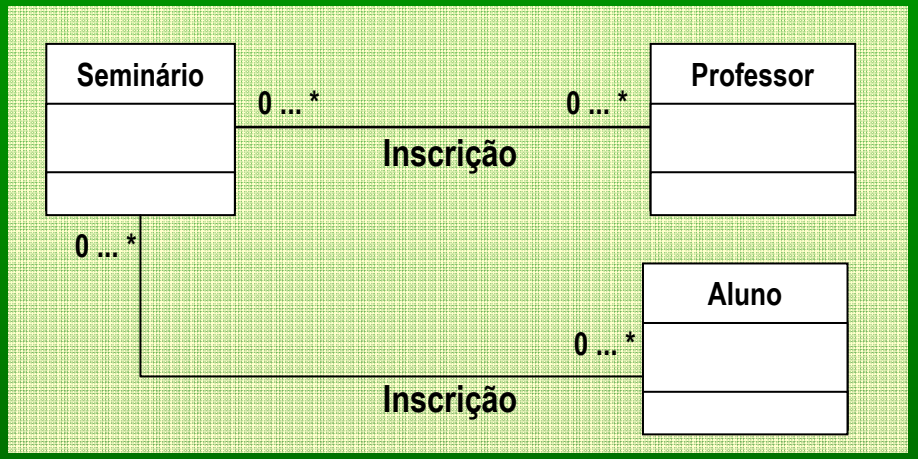
Indicar sempre o sentido de leitura

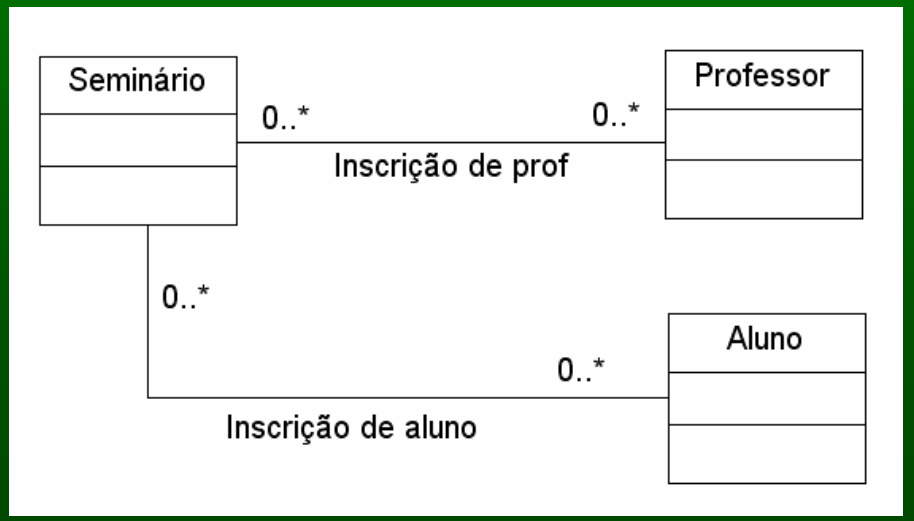
- Indispensável quando há duas associações entre as mesmas classes



Nomes de associações 2

- Em UML puro, duas associações podem ter o mesmo nome desde que sejam entre classes diferentes
- Algumas ferramentas impõem restrições mesmo que as associações sejam entre duas classes diferentes ⇒ duas associações não devem ter nome igual
  - É o caso da ferramenta *PowerDesigner*, usada nos laboratórios: associações com nomes iguais originam posteriormente duplicação de identificadores na base de dados.

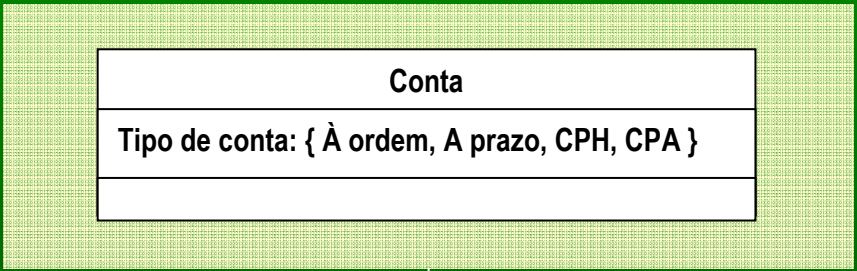


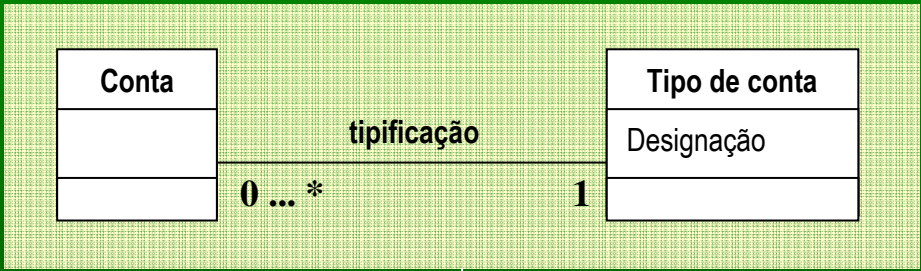


Atributo enumerado *vs.* Associação N-para-1

- Problema:

  Numa base de dados bancária, pretende-se guardar informação sobre cada conta, incluindo o seu tipo de conta.





Atributo enumerado: Deve usar-se apenas se, previsivelmente, as opções serão sempre as mesmas

Associação: Deve usar-se se as opções podem mudar e queremos possibilitar que seja o utilizador a gerir essas opções

- Se a quantidade de opções pode mudar.
- Se podem mudar de designação.

0... *vs.* 1...

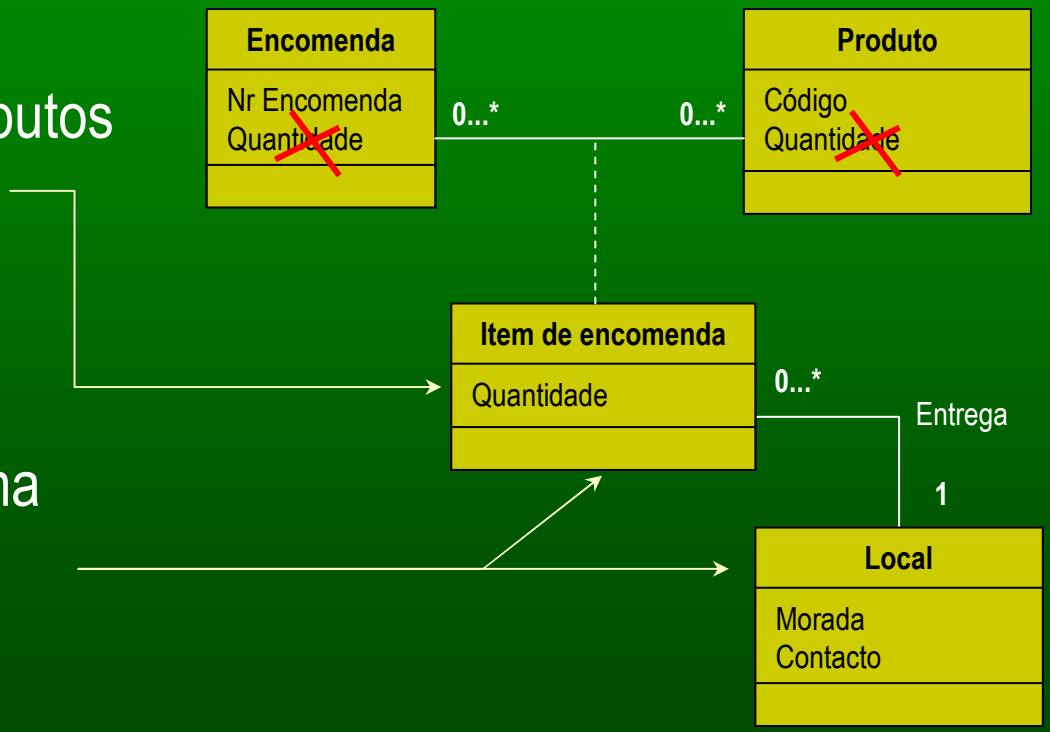
- Usar o 1...* quando, de todo, não se quer a informação do lado muitos sem a informação do lado 1. Quando não tem utilidade sem a informação do lado 1. Mesmo que na realidade a multiplicidade seja 1...*.
  - Exemplo:
    - Um curso tem pelo menos uma disciplina (portanto, na realidade: 1...*).
    - Mas podemos querer manter informação acerca do curso sem ainda termos indicado as suas disciplinas – logo: 0...*
  - Exemplo:
    - Uma receita médica prescreve um ou mais medicamentos.
    - O médico não deve poder guardar a receita sem ter indicado um dos medicamentos.
    - Sem pelo menos um medicamento, a informação sobre a receita não tem qualquer utilidade – logo: 1...*.

Classes associativas

São associações que se "transformam" em classes…

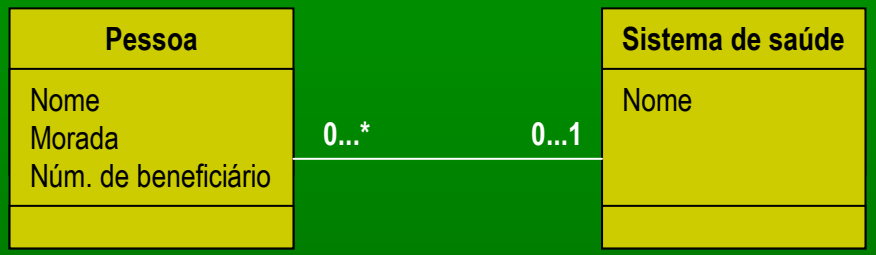
… quando :

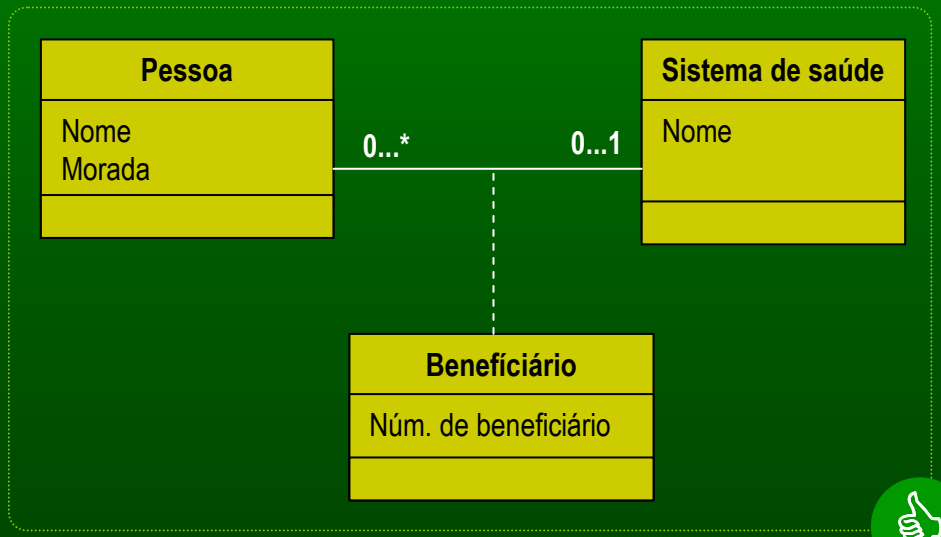
- É necessário colocar atributos na associação:
- É necessário associar uma classe a uma associação



Classes associativas

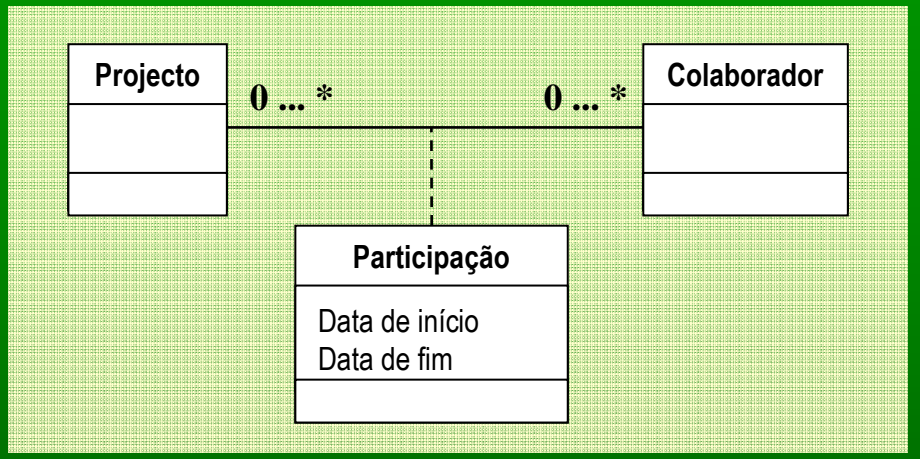
- As classes associativas são mais frequentemente necessárias nas associações *muitos para muitos*.
- Mas, em casos mais raros, fazem também sentido em associações de outras cardinalidades.



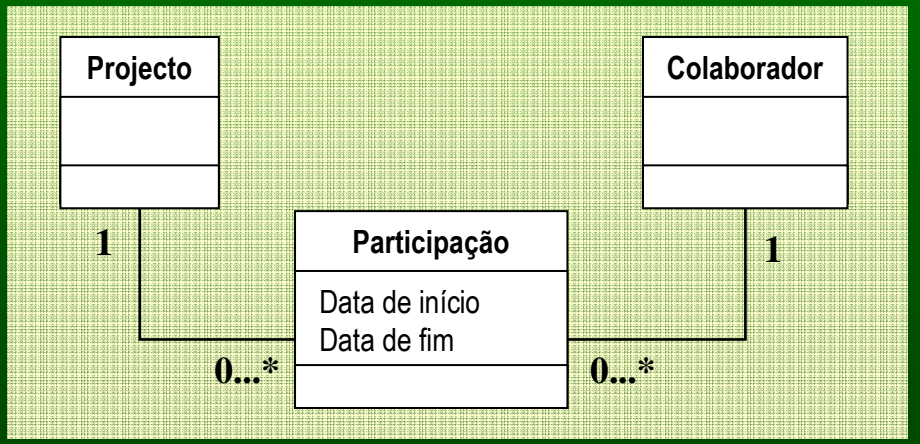


Classe associativa vs. Classe com duas associações *muitos-para-um*

- **Classe associativa**
  - Não permite ligar duas vezes os mesmos dois objectos;
  - No exemplo: Pode registar-se apenas umas vez que determinado colaborador participou em determinado projecto;

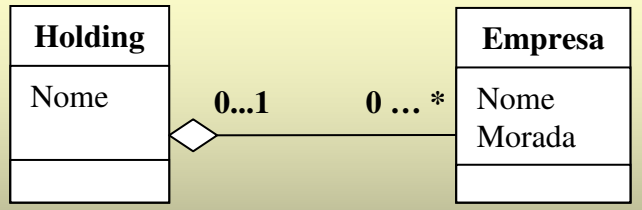


- **Classe com duas associações**
  - Permite ligar várias vezes os mesmos dois objectos;
  - No exemplo: Podem registar-se várias participações do mesmo colaborador no mesmo projecto

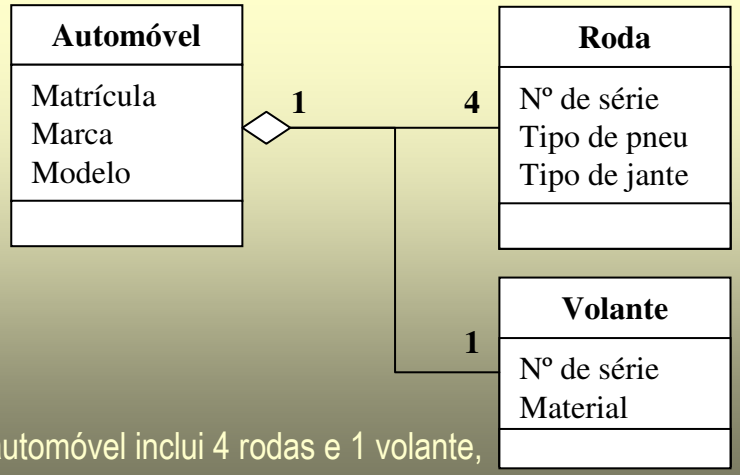


Agregações

- As Agregações são associações que se utilizam quando se pretende representar a noção de Todo/Parte (um todo constituído por partes).



Uma *holding* é constituída por empresas.
Uma empresa pode estar incluída numa holding.



Um automóvel inclui 4 rodas e 1 volante,
nem mais nem menos.

- As agregações são representadas graficamente por uma linha adornada com losângulo branco no extremo correspondente ao *todo*.

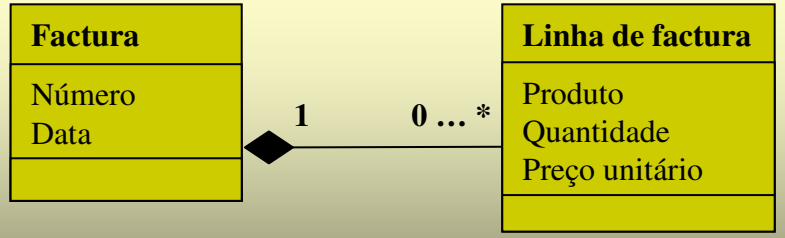
Composições

- As composições são um caso especial de agregações

  Isto é, tal como as agregações, representam situações em que um objecto de uma classe (composição) inclui um conjunto de objectos de outra classe (componentes)...

- ...mas têm semântica adicional:

  Os componentes apenas existem no contexto do *todo*.



Uma factura é uma composição de linhas:
- A factura é constituída por linhas;
- As linhas não existem senão dentro da factura.

Graficamente, o losângulo é preenchido a cheio quando a associação é do tipo composição.

Composições

O facto de numa composição os objectos *componentes* apenas existirem no contexto do objecto *composto* significa:

- Quando se remove o objecto composto, todos os seus componentes são automaticamente removidos
- Os objectos componente incluem no seu mecanismo de identificação o mecanismo de identificação do objecto composto
  - Uma linha de factura só se pode identificar univocamente se também mencionarmos a identificação da factura a que diz respeito
  - Os nome dos departamentos podem repetir-se entre empresas – se não juntarmos o nome da empresa não conseguiremos distinguir certos departamentos que possuam idêntica designação em empresas distintas

Composições

Factura nº 123 Data: 12/12/1999

Cliente: João Silva Nº Cont. 1234567

Linha	Produto	Quant.	P. Unit	Total
1	Produto A	2	5000 €	10000 €
2	Produto B	1	3000 €	3000 €
3	Produto X	3	2000 €	6000 €
Total				19000 €

Factura nº 100 Data: 12/10/1999

Cliente: Ana Silva Nº Cont. 1234568

Linha	Produto	Quant.	P. Unit	Total
1	Produto X	2	5000 €	10000 €
2	Produto B	1	3000 €	3000 €
3	Produto Y	3	2000 €	6000 €
Total				19000 €

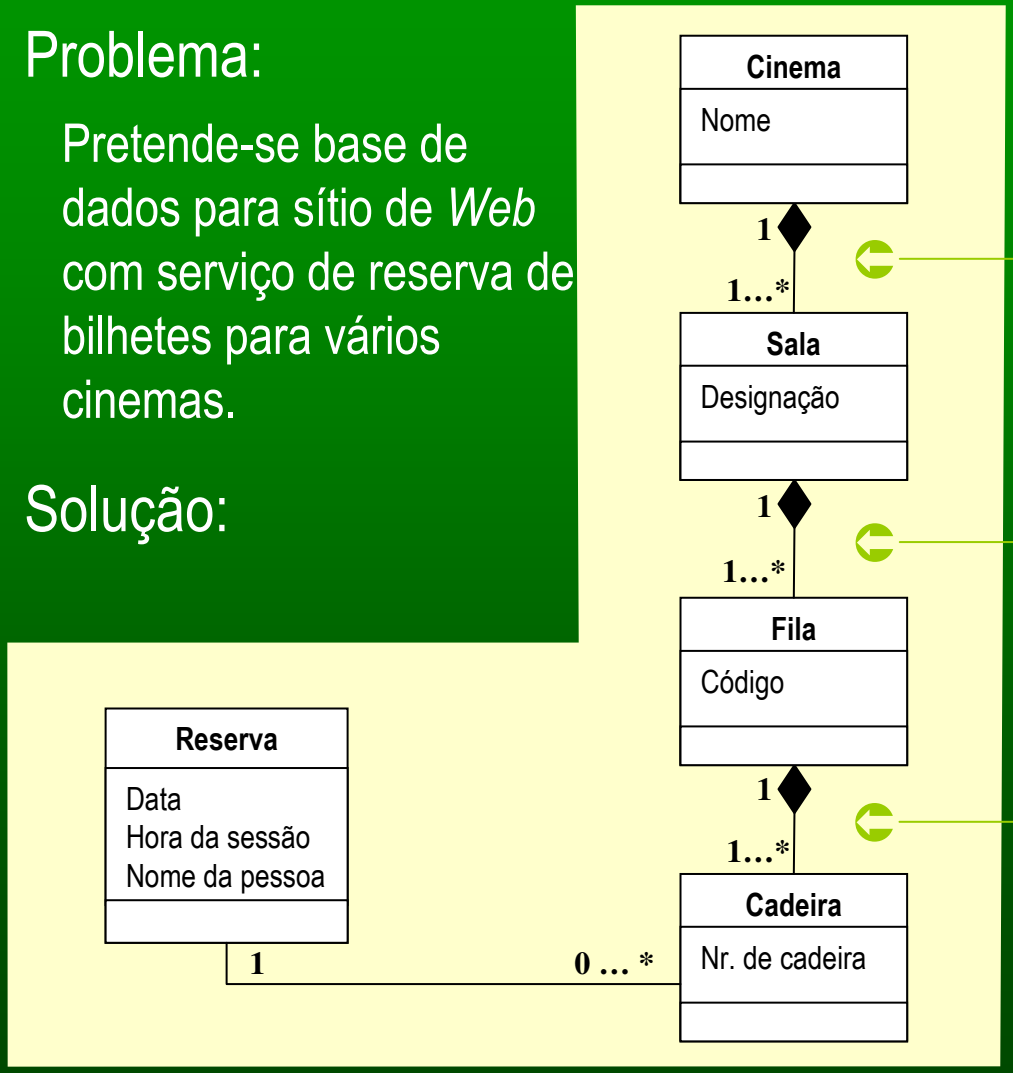
Linhas de factura diferentes, apesar de possuírem os mesmo valores.

Composições

- Problema:

  Pretende-se base de dados para sítio de *Web* com serviço de reserva de bilhetes para vários cinemas.

- Solução:



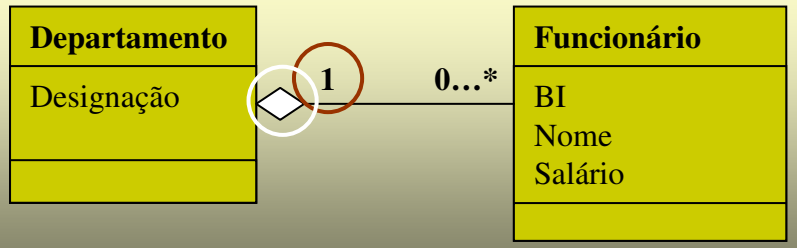
Não conseguimos identificar uma sala de cinema se não dissermos em que cinema está incluída

Não conseguimos identificar uma fila se não dissermos a que sala pertence

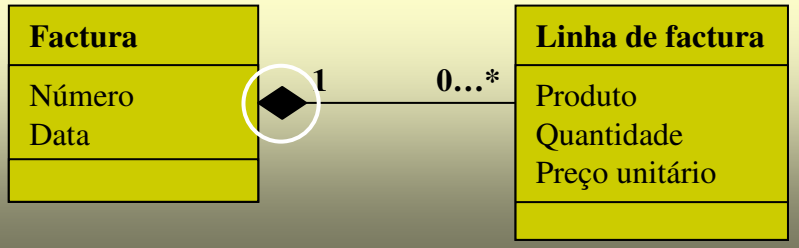
Não conseguimos identificar uma cadeira se não dissermos a que fila pertence

Composições *vs.* Agregações

- Apesar da obrigatoriedade existir em ambas as associações seguintes, são situações com semânticas distintas:



- Significa que um funcionário que não trabalhe num departamento não é relevante para o domínio em causa (se o seu departamento for removido ele terá que ser excluído ou reposicionado em outro departamento).
- No entanto, o funcionário existe *per si*, não necessita de estar associado a um departamento para ser referido/identificado



A Linha da factura só pode ser referida (distinguida das restantes) se for indicada a factura correspondente.

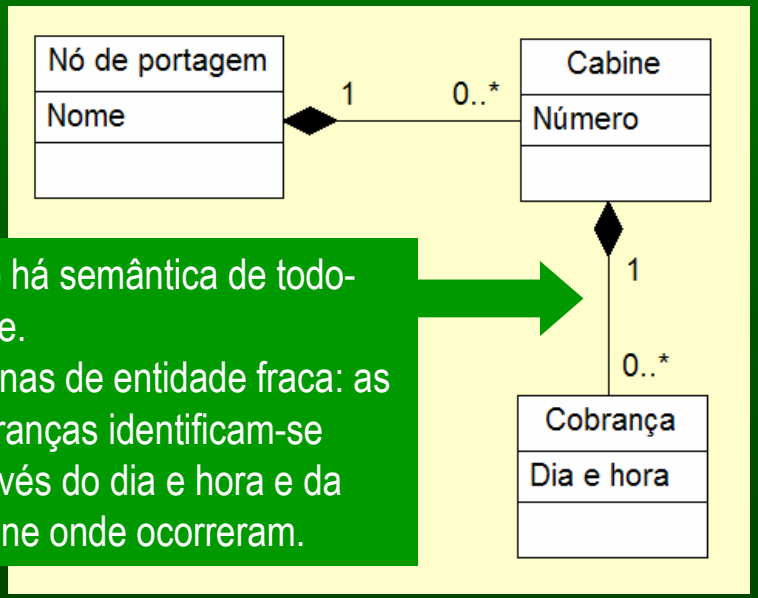
A Composição como conceito de identificação

- Em desenho de base de dados, designam-se por *entidades fracas* aquelas entidades que dependem de outras para se identificar
- A composição é a figura que em UML permite representar entidades fracas.
- Mesmo que não haja semântica de todo-parte, deve usar-se uma composição nas situações em que haja semântica de entidade fraca.

- Exemplo:

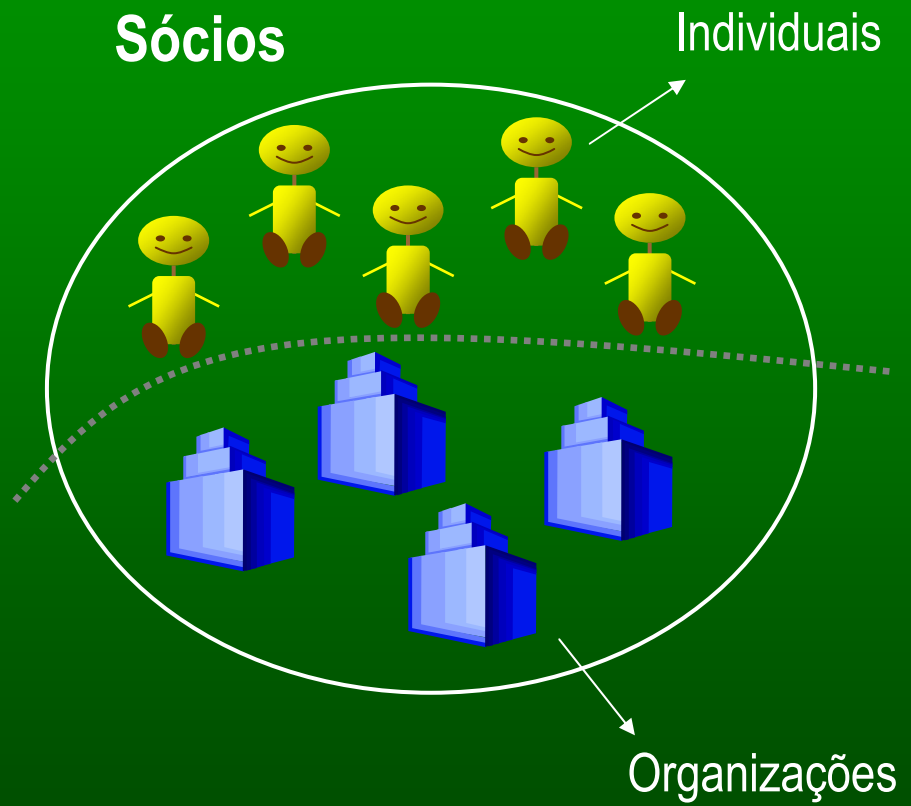
  Pretende-se manter numa base de dados informação acerca de todas as cobranças de portagem nas auto-estradas nacionais.

  Como identificar cada uma das cobranças?

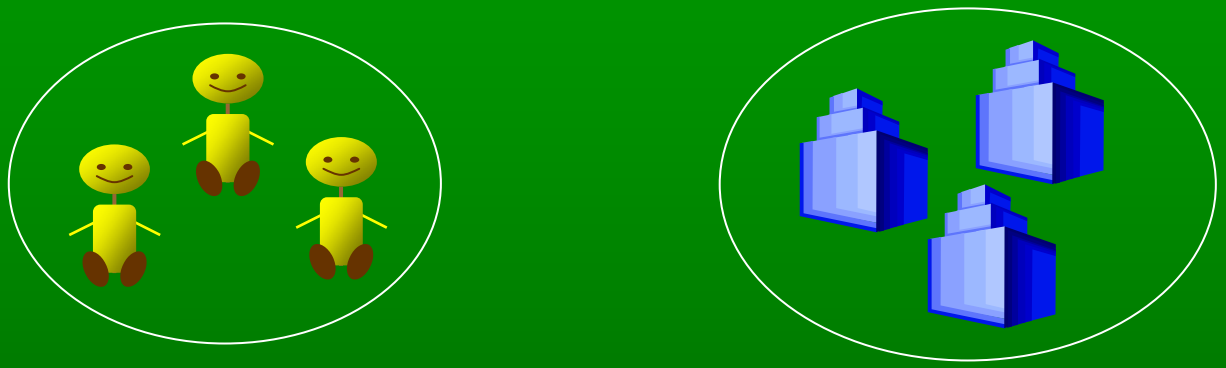


Generalização

- Generalização é uma relação que permite representar a noção de subdivisão e especificidade em conjuntos de objectos
- Todos os sócios partilham características comuns
  - Nome, morada, telefone, valor de quotização, etc.
- Mas há subgrupos com especificidades
  - Individuais: Idade
  - Organizações: Número de elementos



Generalização



Porquê distingui-los?

- Porque têm atributos específicos

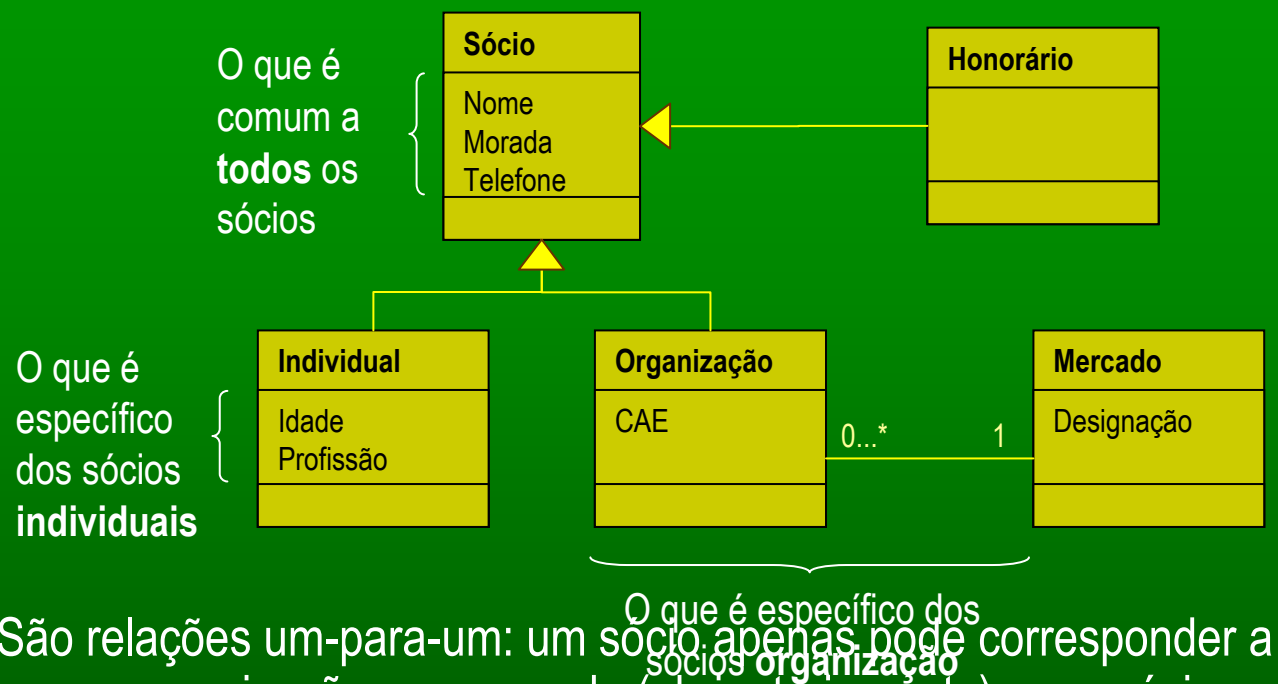
  exº: O CAE (Código de Actividade Económica) nas Organizações, a Idade nos Individuais

- Porque têm associações específicas

  exº: Mercados onde as Organizações actuam

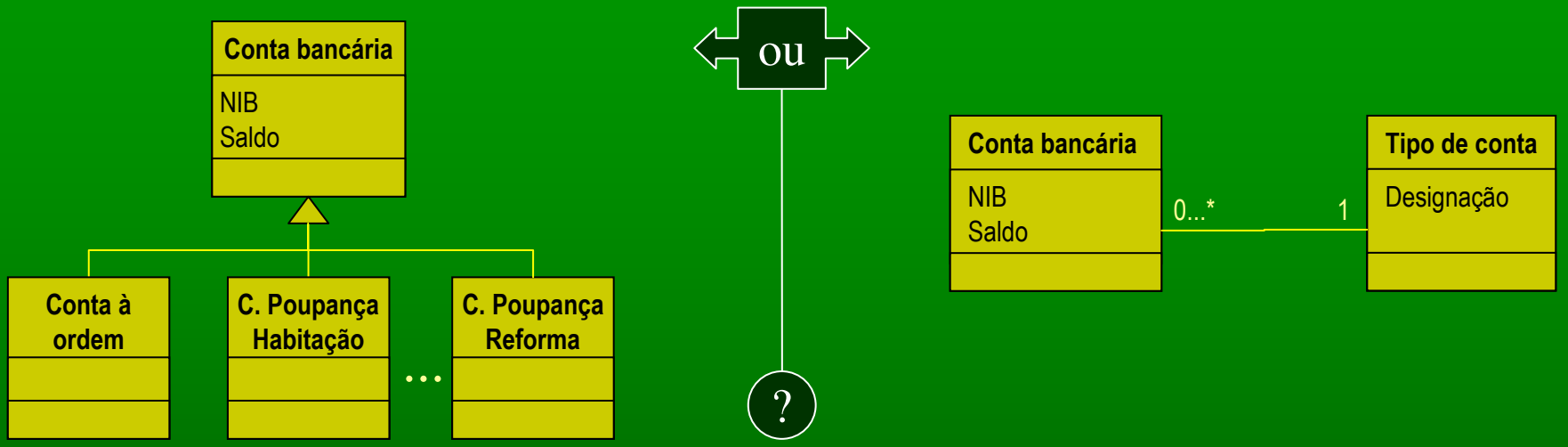
- Ou apenas porque se quer dar relevo a um subconjunto do domínio

Generalização



- São relações um-para-um: um sócio apenas pode corresponder a uma organização e uma organização corresponde (obrigatoriamente) a um sócio;
- Um Sócio **não** pode ser **simultaneamente** Individual e Organização;
- Um Sócio **pode** não ser **nem** Individual **nem** Organização;
- Um Sócio pode ser Honorário e Individual **ou** Honorário e Organização;
- As **subclasses** herdam todos os atributos da **supraclasse/superclasse**.

Generalização vs. associação *N-para-1*

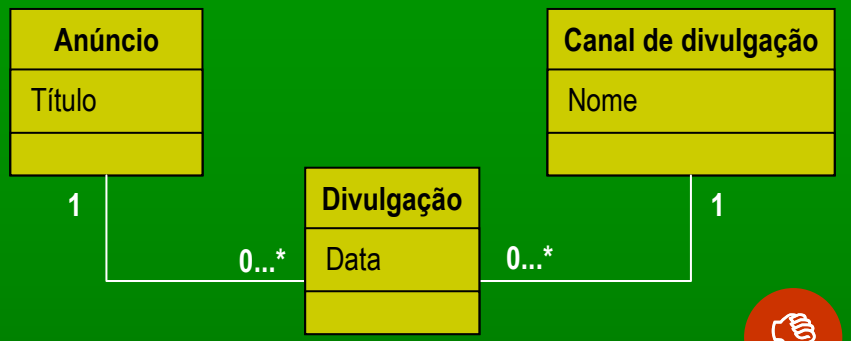


- Se o conjunto de opções é imutável
- Se as diferentes opções têm especificidades (atributos ou associações próprias)
- Se algumas opções irão ter um tratamento especial – por exemplo, permissões de acesso especiais
- Se as opções realçam conceitos importantes, por exemplo, se transmitem terminologia própria do domínio aplicacional
- Se com alguma probabilidade podem surgir novas opções ou as existentes podem necessitar de alterações
- Se não há especificidades
- Se todas as opções têm igual tratamento
- Se as opções não correspondem a conceitos de grande relevância no domínio aplicacional

Associações N-árias

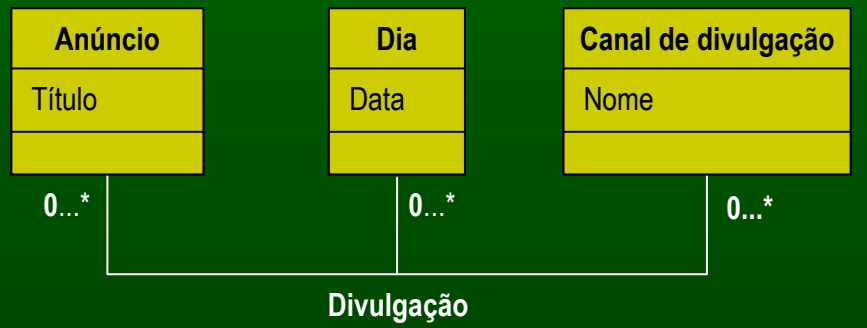
- Problema:

  Empresa deseja efectuar divulgação através de anúncios em vários meios de comunicação social. Para controlo de custos e pagamentos necessita de registar as divulgações.

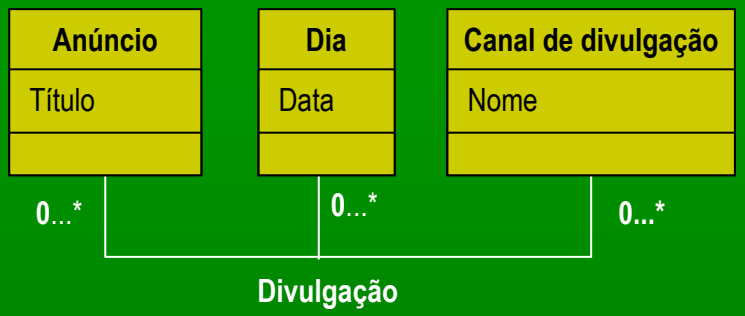


- Solução 1: Permite a introdução de informação duplicada (mesmo anúncio no mesmo canal no mesmo dia)

- Solução 2: **Associação ternária**

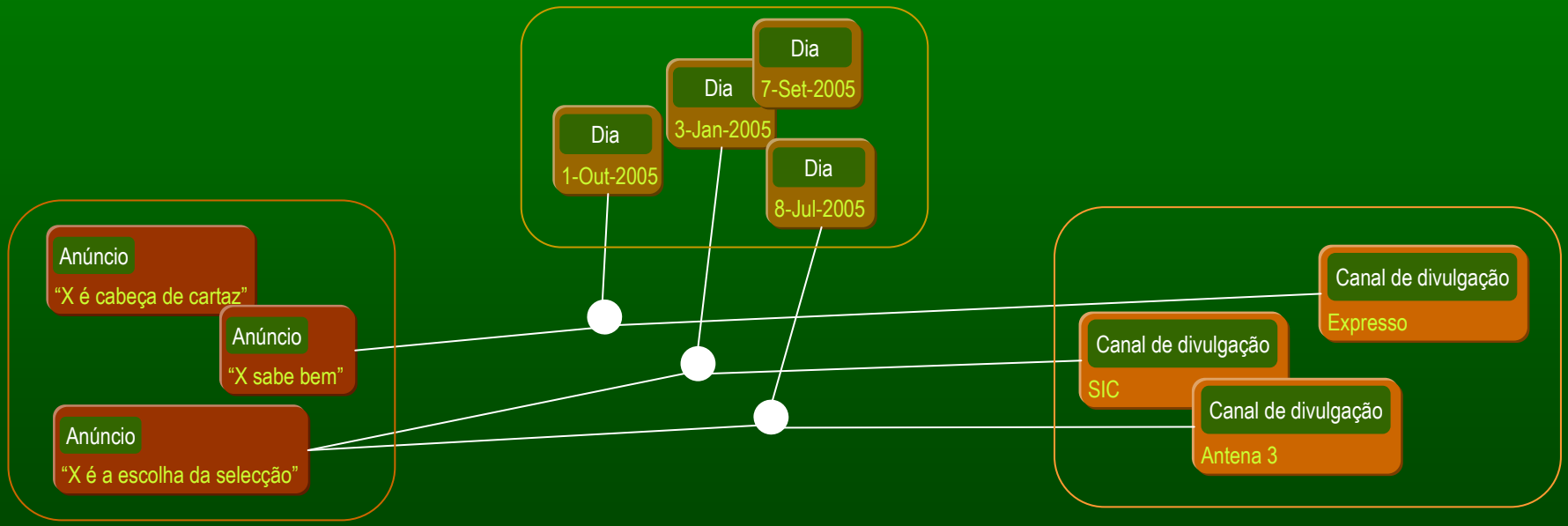


Associações N-árias



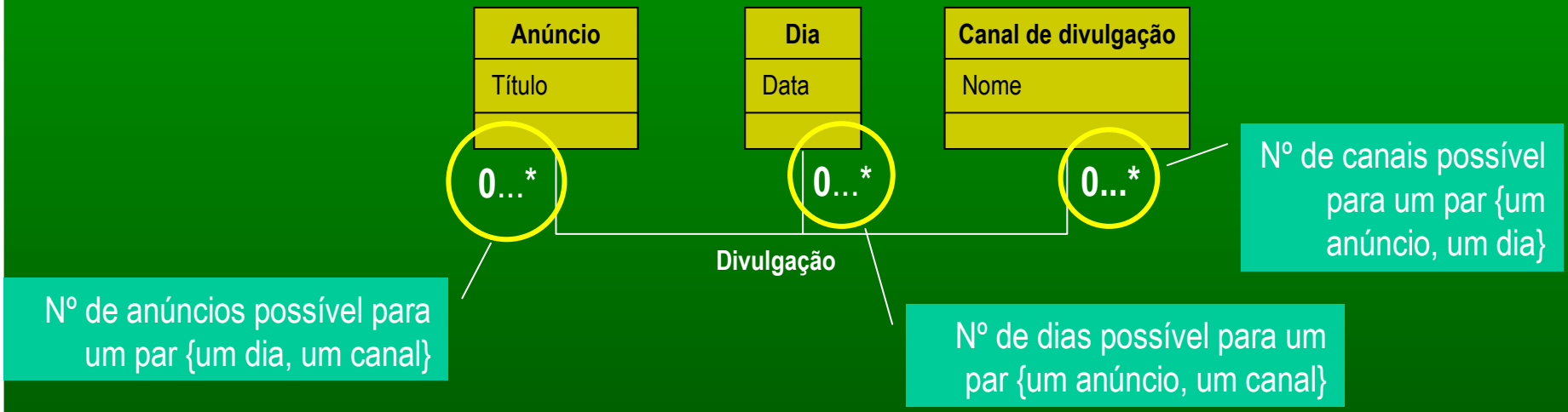
- No plano dos objectos, uma ligação ***N***-ária envolve sempre ***N*** objectos

  ...um de cada classe argumento:



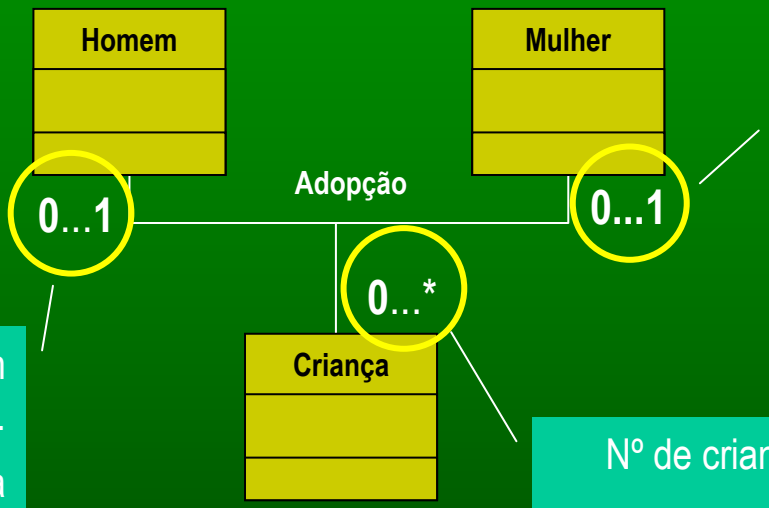
Multiplicidades em associações N-árias

- Definição de multiplicidades numa relação ternária



Multiplicidade em associações N-árias

- Definição de multiplicidades numa relação ternária



Nº de mulheres possíveis para um par {um homem, uma criança}.

Nº de homens possível para um par {uma mulher, uma criança}.

Adoptar **o Joãozinho** com a **Ana**, apenas pode ter sido com **um homem**. Se o Joãozinho não foi adoptado pela Ana, então há **zero homens** associados ao par {Ana, Joãozinho}. Logo: **0 ou 1** homens.

Nº de crianças possíveis para um par {um homem, uma mulher}.

O **Pedro** e a **Ângela**, **juntos**, podem ter **0 (zero) ou várias crianças** adoptadas.

Atenção que **não é** o nº de crianças envolvidas **numa adopção**!

Associações N-árias vs. (N-1) associações binárias

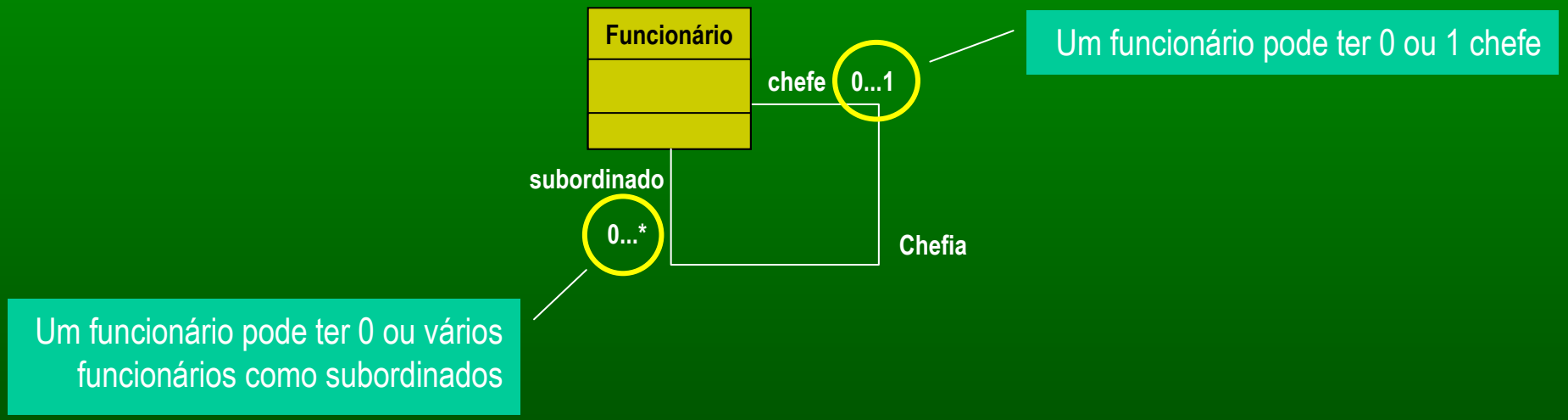
- *Exemplos:*
  - *Inscrição: Aluno-disciplina-ano lectivo*
  - *Docência: Docente-disciplina-ano lectivo*
  - *Adopção: Casal-criança*
  - *Atribuição de óscar: Óscar-Filme-Ano*
  - *Reserva: Pessoa-Cadeira-Sessão-Dia*
  - *Vendas*

Associações n-árias: notação alternativa

- *Ilustrar a notação losangular utilizada por várias ferramentas (por exemplo: Visio).*

Associações reflexivas

- Também chamadas associações singulares

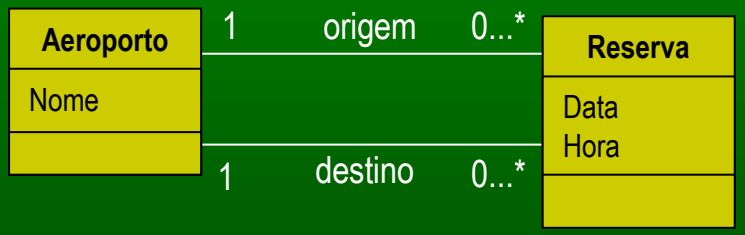


Associações reflexivas

Problema:

Agência de viagens pretende registar reservas de voo.

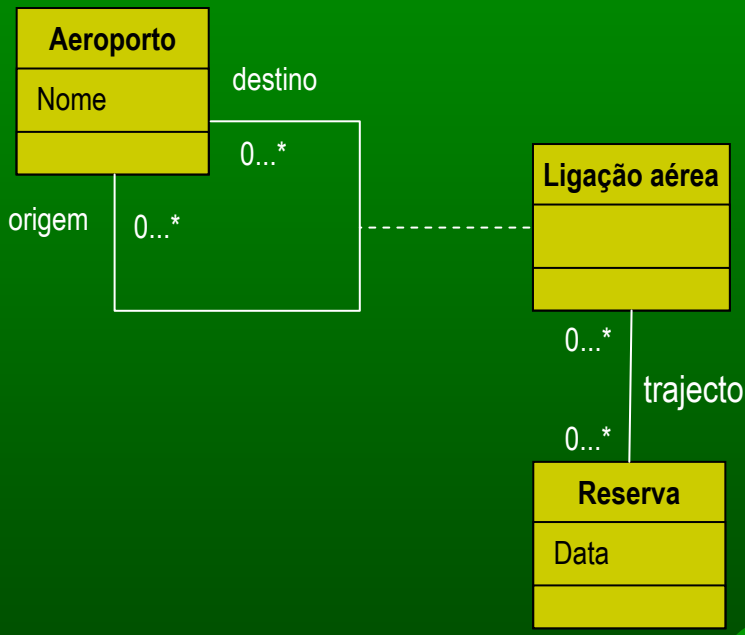
- Solução 1:



Não permite registar ligações (aéreas) nem escalas.

- A uma reserva precisamos de poder associar vários pares origem-destino

- Solução 2:

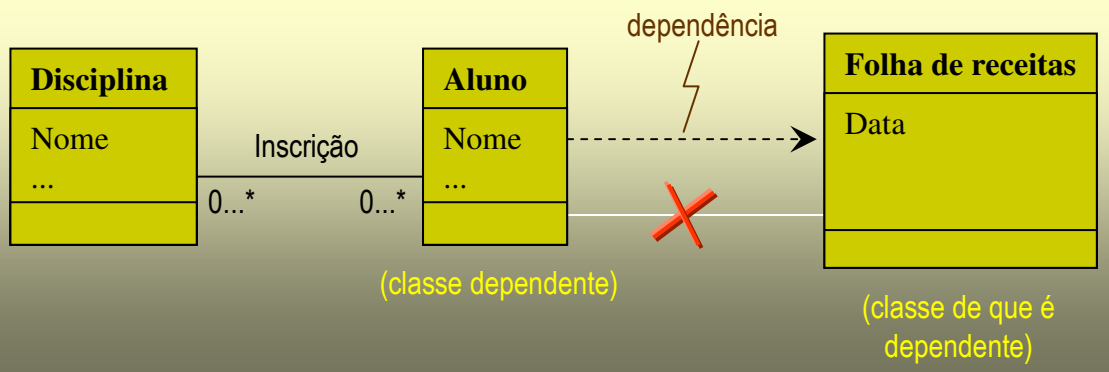


Perguntas extra:

- Permite voos Lisboa-Lisboa?
- Onde deverá ficar o atributo *Hora*?

Relações de dependência

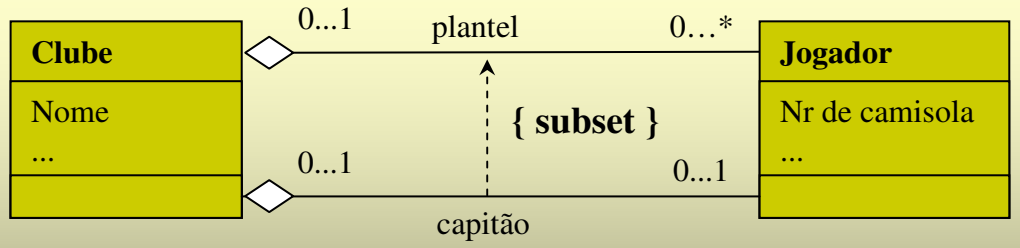
- As relações de dependência permitem evidenciar dependências que não têm expressão como relações estruturais.
- Existe uma relação de dependência quando uma alteração na especificação de uma classe origina uma alteração na especificação de outra classe.
  - Surgem quando alguma acção sobre os objectos de uma classe (a classe dependente) envolve a utilização de objectos de outra
  - Só se especificam se não existir uma relação já definida entre essas duas classes – esta, existindo, já dá indicação de dependência entre as classes



- A inscrição de um aluno numa disciplina origina uma entrada de receitas – a classe *Aluno* depende da classe *Folha de receitas*.
- Não interessa manter registo das ligações entre alunos e folhas de receitas – não se especifica uma associação

Constraint: subset

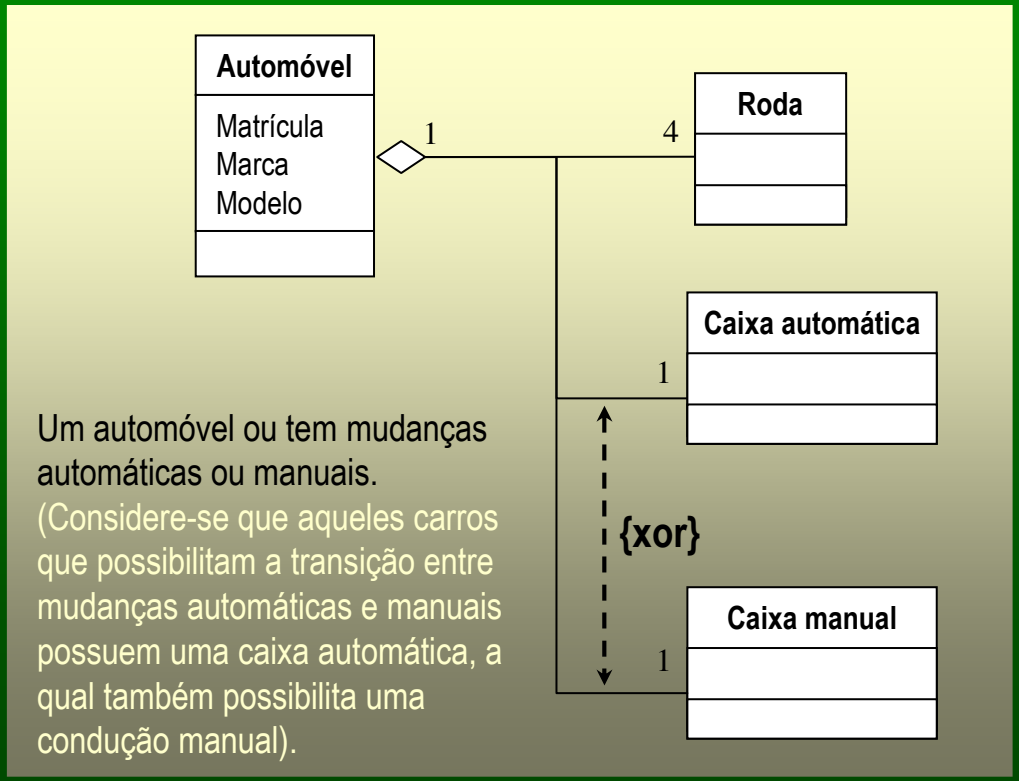
- Usado para expressar que as ligações estabelecidas no contexto de uma dada associação são um subconjunto das estabelecidas no contexto de outra associação



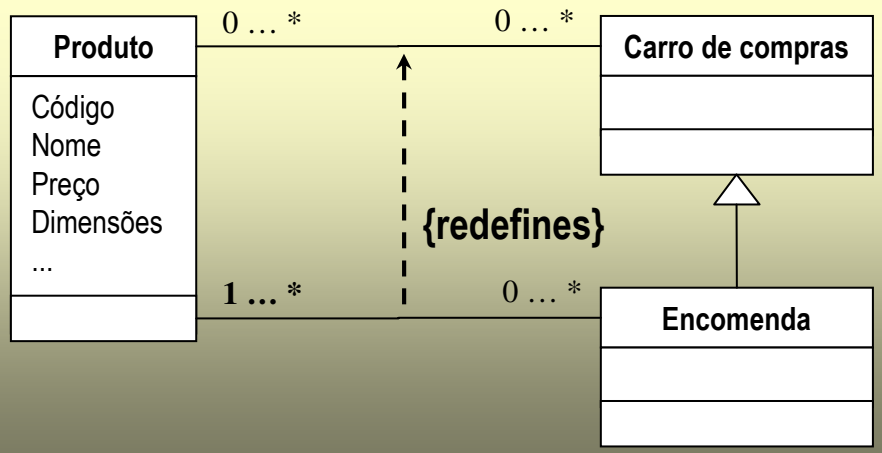
O capitão de uma equipa (de um clube) é um dos elementos que figura no seu plantel.
Este esquema indica que a base de dados não deverá deixar apontar como capitão de uma equipa um jogador que não faça também parte do seu plantel.

Constraint: xor

- Usa-se quando duas associações são exclusivas relativamente à classe que têm em comum – i. é, cada objecto desta classe só pode estar associada a outro objecto por via de uma destas associações, nunca por via das duas associações simultaneamente;
- Lê-se: “égzór”.

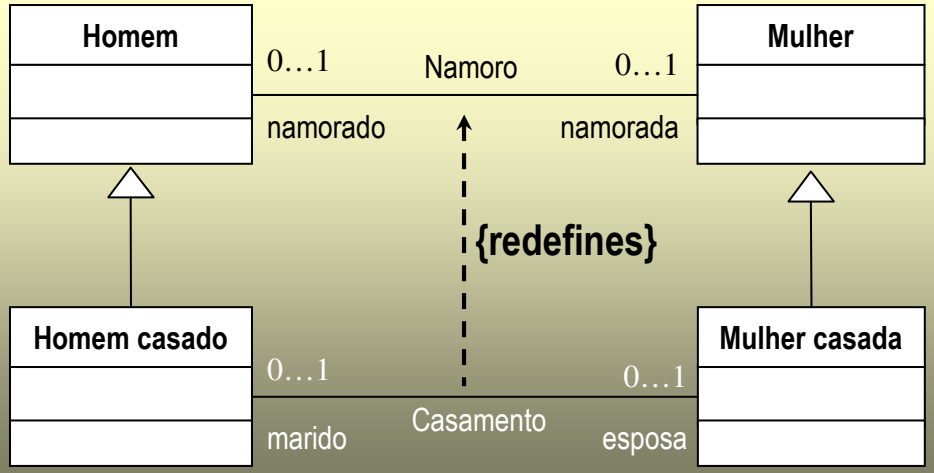


Constraint: redefines



O domínio de aplicação é um sítio de vendas *on-line*.

- Os visitantes do sítio colocam produtos no seu carro de compras.
- As encomendas são carros de compras que foram aprovados para compra (o cliente deu ordem de compra).
- Necessariamente, ao passar a encomenda, um carro de compras tem que ter pelo menos um produto – notar o "1...* " na associação de baixo

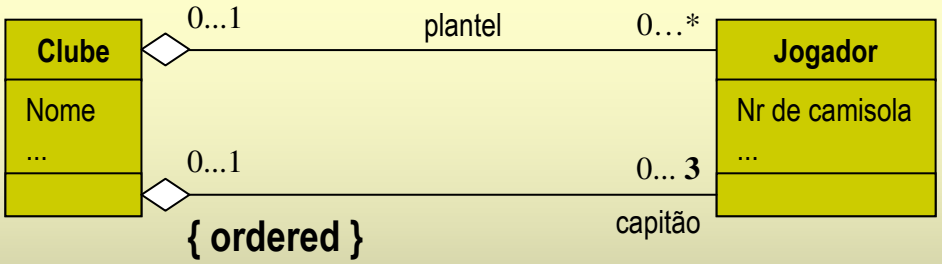


O conceito representado por uma relação homem-mulher muda quando estes passam a ser pessoas casadas.
Uma ligação de casamento substitui sempre uma ligação de namoro.

• Pergunta: Porque é que a cardinalidade da associação *Casamento* é 0...1 e não 1...1?

Constraint: ordered

- Usa-se quando se pretende expressar que a base de dados deve manter uma ordenação quanto às ligações estabelecidas para cada objecto



- Um clube pode ter até 3 jogadores designados para capitães de equipa.
- Um é o 1º capitão e é este quem habitualmente desempenha a função de chefia de equipa em campo. O 2º capitão só desempenha esta função se o 1º estiver ausente. O mesmo se passa entre o 3º e 2º capitão.

☞ É importante registar a ordem pela qual os 3 capitães de uma equipa (clube) estão designados, pois é o 1º capitão quem desempenha.

Outras notações gráficas comuns em modelação de dados

- Mostrar aqui que muitos dos conceitos apresentados existem em outros formalismos, sendo frequente encontrar estes mesmos conceitos representados de outras formas, nomeadamente:
  - Notação *Crow's foot*
  - Notação Chen-ERD (E-R original)