Bases de Dados

Universidade do Porto

Faculdade de Engenharia

FEUP

1

Diagrama de Classes UML

Observação: adaptado de slides desenvolvidos pelos Profs. da FEUP :Ademar Aguiar, Gabriel David e João Pascoal Faria.

UML

Índice

UML

O que é a UML?

- UML = *Unified Modeling Language*
- UML é uma linguagem (notação com semântica associada) para
  - visualizar
  - especificar
  - construir
  - documentar

  os artefactos de um sistema com uma componente intensiva de software (*software intensive system*)
- UML não é uma metodologia
  - não diz quem deve fazer o quê, quando e como
  - UML pode ser usado segundo diferentes metodologias, tais como RUP (*Rational Unified Process*), FDD (*Feature Driven Development*), etc.
- UML não é uma linguagem de programação

UML

4

Modelos e Diagramas

- Um modelo é uma representação em **pequena escala**, numa perspectiva particular, de um sistema existente ou a criar
  - Atitude de **abstracção** (omissão de detalhes) fundamental na construção de um modelo
  - Modelos são a linguagem por excelência do projectista (*designer*)
  - Modelos são veículos para comunicação com vários interessados (*stakeholders*)
  - Modelos permitem raciocinar acerca do sistema real, sem o chegar a construir
- Ao longo do ciclo de vida de um sistema são construídos vários modelos, sucessivamente refinados e enriquecidos

UML

5

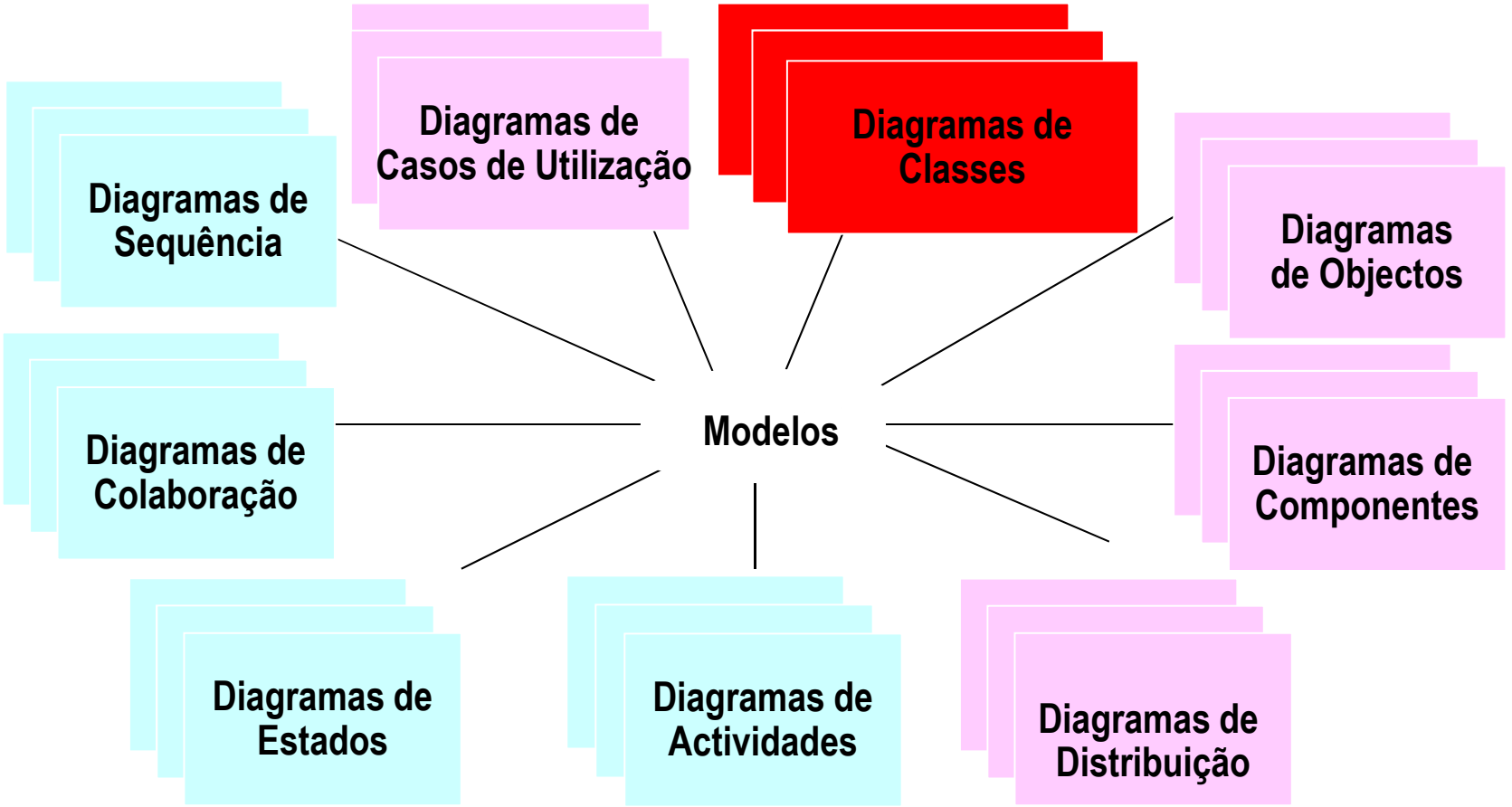
Modelos e Diagramas

- Um modelo é constituído por um conjunto de **diagramas** (desenhos) consistentes entre si, acompanhados de descrições textuais dos **elementos** que aparecem nos vários diagramas
  - Um **diagrama** é uma vista sobre um modelo
  - O mesmo **elemento** (exemplo: classe) pode aparecer em vários diagramas de um modelo
- No UML, há nove diagramas *standard*
  - Diagramas de visão estática: casos de utilização (*use case*), classes, objectos, componentes, distribuição (*deployment*)
  - Diagramas de visão dinâmica: sequência, colaboração, estados (*statechart*), actividades

UML

6

Modelos e Diagramas



UML: Diagrama de Classes

Objectivos

- Um diagrama de classes serve para modelar o vocabulário de um sistema, do ponto de vista do utilizador/problema ou do implementador/solução
  - Ponto de vista do utilizador/problema – na fase de captura e análise de requisitos, em paralelo com a identificação dos casos de utilização
  - Vocabulário do implementador/solução – na fase de projecto (*design*)
- Construído e refinado ao longo das várias fases do desenvolvimento do software, por analistas, projectistas (*designers*) e implementadores
- Também serve para:
  - Especificar colaborações (no âmbito de um caso de utilização ou mecanismo)
  - **Especificar esquemas lógicos de bases de dados**
  - Especificar vistas (estrutura de dados de formulários, relatórios, etc.)
- Modelos de objectos de domínio, negócio, análise e *design*

UML: Diagrama de Classes

Objectos

Um **objecto** é:

- algo com fronteiras bem definidas,
- relevante para o problema em causa,
- com estado,
  - modelado por valores de atributos (tamanho, forma, peso, etc.) e por ligações que num dado momento tem com outros objectos
- com comportamento
  - um objecto exibe comportamentos invocáveis (por resposta a chamadas de operações) ou reactivos (por resposta a eventos)

UML: Diagrama de Classes

Objectos

- e identidade
  - no espaço: é possível distinguir dois objectos mesmo que tenham o mesmo estado
    - exemplo: podemos distinguir duas folhas de papel A4, mesmo que tenham os mesmos valores dos atributos
  - no tempo: é possível saber que se trata do mesmo objecto mesmo que o seu estado mude
    - exemplo: se pintarmos um folha de papel A4 de amarelo, continua a ser a mesma folha de papel

UML: Diagrama de Classes

10

Objectos

Objectos do mundo real e objectos computacionais:

- No desenvolvimento de software orientado por objectos, procura-se imitar no computador o mundo real visto como um conjunto de objectos que interagem entre si
- Muitos objectos computacionais são imagens de objectos do mundo real
- Dependendo do contexto (análise ou projecto) podemos estar a falar em objectos do mundo real, em objectos computacionais ou nas duas coisas em simultâneo
- Exemplos de objectos do mundo real:
  - o Sr. João
  - a aula de ES no dia 11/10/2000 às 11 horas
- Exemplos de objectos computacionais:
  - o registo que descreve o Sr. João (imagem de objecto do mundo real)
  - uma árvore de pesquisa binária (objecto puramente computacional)

UML: Diagrama de Classes

Classes

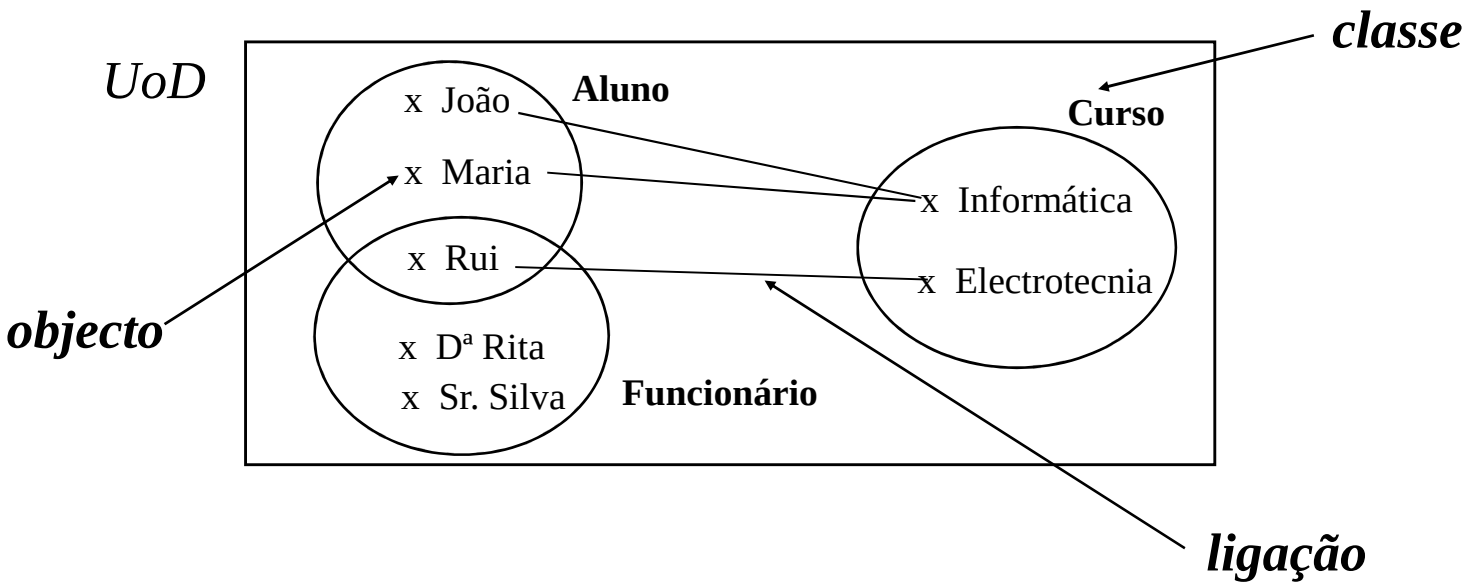
- No desenvolvimento de software OO, não nos interessam tanto os objectos individuais mas sim as classes de objectos
- Uma **classe** é um descritor de um conjunto de objectos que partilham as mesmas propriedades (semântica, atributos, operações e relações)
  - Trata-se de uma noção de classe em **compreensão**, no sentido de **tipo de objecto**, por oposição a uma noção de classe em **extensão**, como conjunto de objectos do mesmo tipo
- Um objecto de uma classe é uma **instância** da classe
- A **extensão** de uma classe é o conjunto de instâncias da classe
- Em Matemática, uma classe é um conjunto de "objectos" com uma propriedade em comum, podendo ser definida indiferentemente em compreensão ou em extensão

  $C = \{x \in \mathbb{N} : x \bmod 3 = 2\} = \{2, 5, 8, 11, 14, ...\}$

UML: Diagrama de Classes

Classes

- O conjunto de todos os objectos num determinado contexto forma um universo (*UoD - Universe of Discourse*)
- As extensões das classes são subconjuntos desse universo



UML: Diagrama de Classes

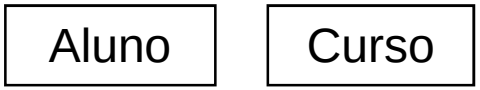
Classes

- Classes podem representar:
  - **Coisas concretas***:* Pessoa, Turma, Carro, Imóvel, Factura, Livro
  - **Papéis que coisas concretas assumem**: Aluno, Professor, Piloto
  - **Eventos***:* Curso, Aula, Acidente
  - **Tipos de dados***:* Data, Intervalo de Tempo, Número Complexo, Vector
- Decomposição orientada por objectos : começa por identificar os tipos de objectos (classes) presentes num sistema
  - contrapor com decomposição funcional ou algorítmica
  - tipos de objectos são mais estáveis do que as funções, logo a decomposição orientada por objectos leva a arquitecturas mais estáveis

UML: Diagrama de Classes

Classes

- Em UML, uma classe é representada por um rectângulo com o nome da classe



- Habitualmente escreve-se o nome da classe no singular (nome de uma instância), com a 1ª letra em maiúscula
- Para se precisar o significado pretendido para uma classe, deve-se explicar o que é (e não é ...) uma instância da classe
  - Exemplo: "Um aluno é uma pessoa que está inscrita num curso ministrado numa escola. Uma pessoa que esteve no passado inscrita num curso, mas não está presentemente inscrita em nenhum curso, não é um aluno."
  - Em geral, o nome da classe não é suficiente para se compreender o significado da classe

UML: Diagrama de Classes

Atributos

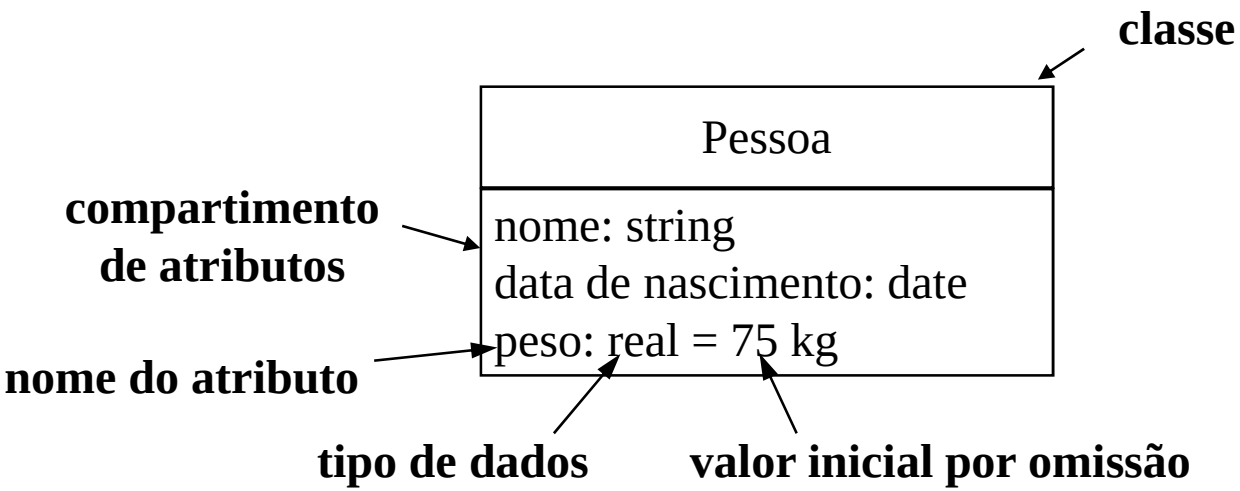
Atributos de instância:

- O estado de um objecto é dados por valores de atributos (e por ligações que tem com outros objectos)
- Todos os objectos de uma classe são caracterizados pelos mesmos atributos (ou variáveis de instância)
  - o mesmo atributo pode ter valores diferentes de objecto para objecto
- **Atributos** são definidos ao nível da classe, enquanto que os **valores dos atributos** são definidos ao nível do objecto
- Exemplos:
  - Uma pessoa (classe) tem os atributos nome, data de nascimento e peso
  - João (objecto) é uma pessoa com nome “João Silva”, data de nascimento “18/3/1973” e peso “68 Kg”

UML: Diagrama de Classes

Atributos

- Atributos são listados num compartimento de atributos (opcional) a seguir ao compartimento com o nome da classe
- Uma classe não deve ter dois atributos com o mesmo nome
- Os nomes dos tipos não estão pré-definidos em UML, podendo-se usar os da linguagem de implementação alvo

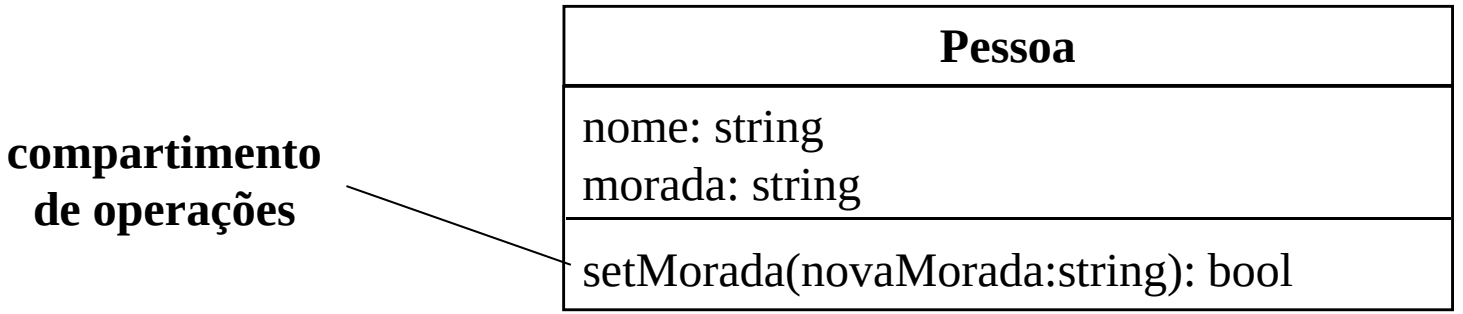


UML: Diagrama de Classes

Operações

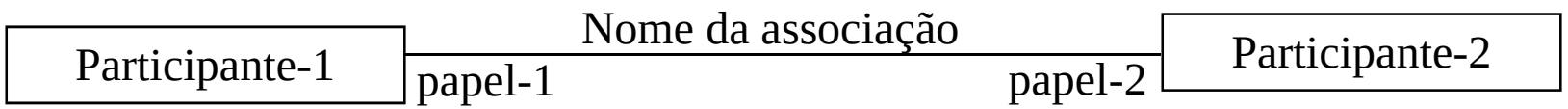
Operações de instância:

- Comportamento invocável de objectos é modelado por operações
  - uma operação é algo que se pode pedir para fazer a um objecto de uma classe
- Objectos da mesma classe têm as mesmas operações
- **Operações** são definidas ao nível da classe, enquanto que a **invocação de uma operação** é definida ao nível do objecto
- Princípio do **encapsulamento**: acesso e alteração do estado interno do objecto (valores de atributos e ligações) controlado por operações
- Nas classes que representam objectos do mundo real é mais comum definir **responsabilidades** em vez de operações



UML: Diagrama de Classes

Associações

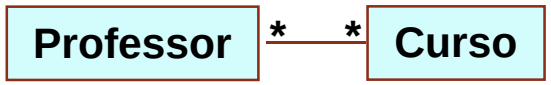


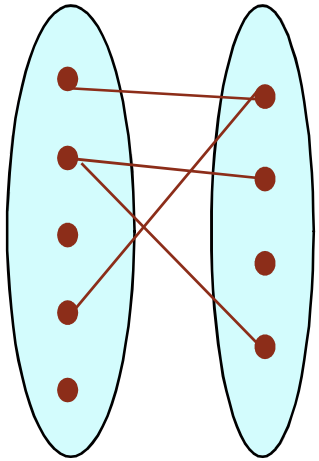
- Uma **associação** é uma relação entre objectos das classes participantes (um objecto de cada classe em cada ligação)
- Não gera novos objectos
- Matematicamente, uma associação binária é uma relação binária, i.e., um subconjunto do produto cartesiano das extensões das classes participantes
- Assim como um objecto é uma instância duma classe, uma **ligação** é uma instância de uma associação
- Pode haver mais do que uma associação (com nomes diferentes) entre o mesmo par de classes
- Papéis nos extremos da associação podem ter indicação de visibilidade (pública, privada, etc.)

UML: Diagrama de Classes

Multiplicidade das associações

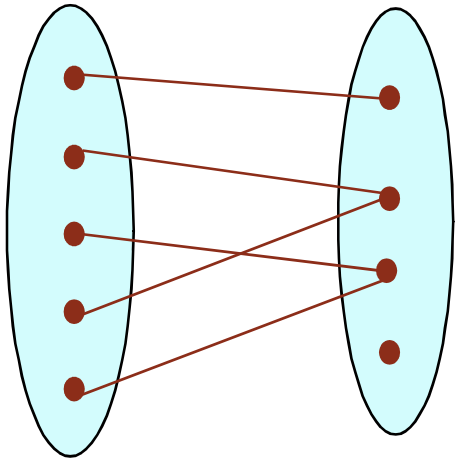
Muitos-para-Muitos



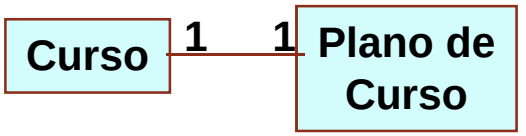


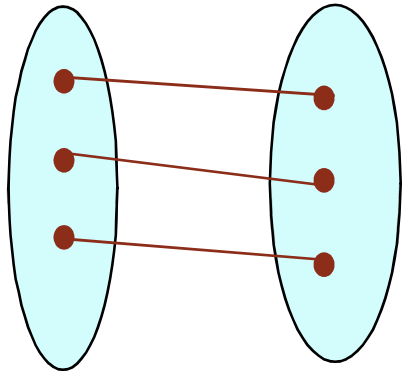
(sem restrições)

Muitos-para-1



1-para-1





UML: Diagrama de Classes

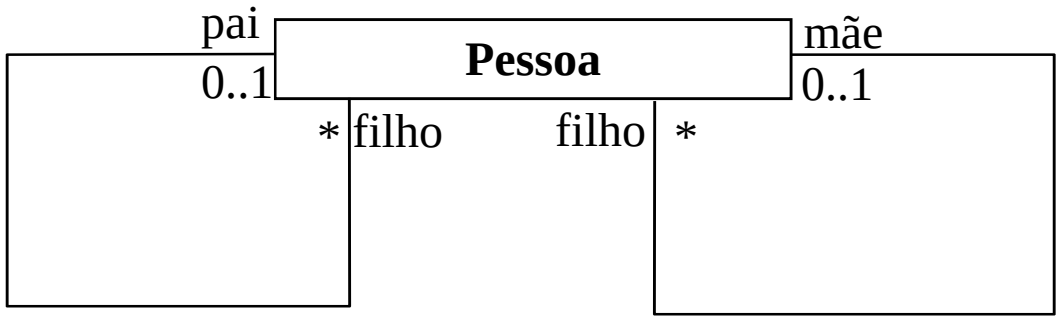
Multiplicidade das associações

1	- exactamente um
0..1	- zero ou um (zero a 1)
*	- zero ou mais
0..*	- zero ou mais
1..*	- um ou mais
1, 3..5	- um ou três a 5

UML: Diagrama de Classes

Associações reflexivas

- Pode-se associar uma classe com ela própria (em papéis diferentes)



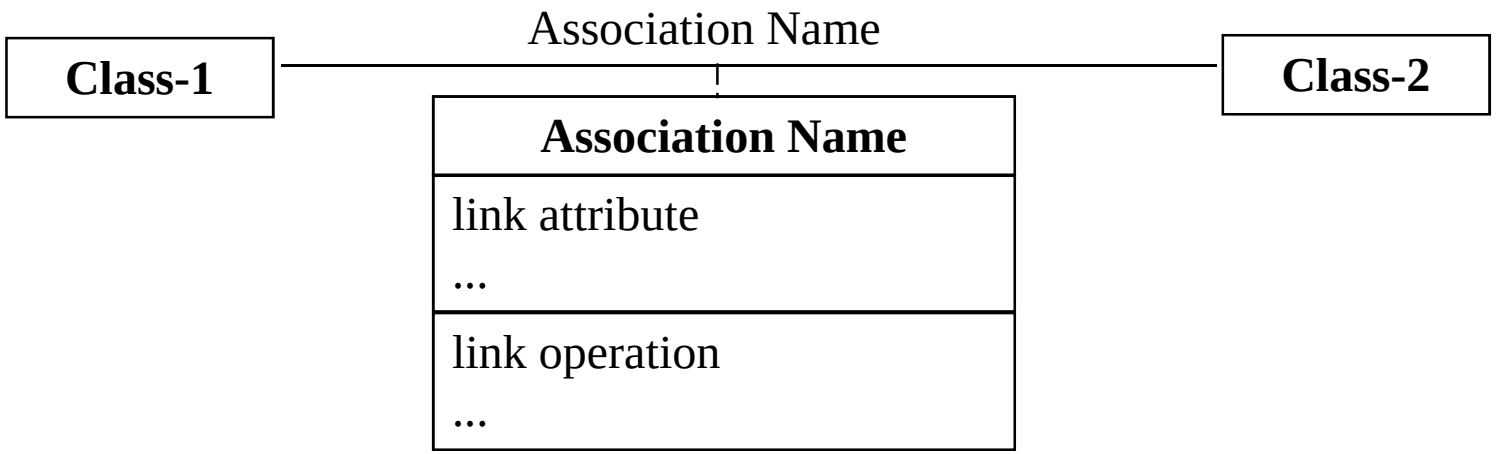
UML: Diagrama de Classes

Nomes de associações

- A indicação do nome é opcional
- O nome é indicado no meio da linha que une as classes participantes
- Pode-se indicar o sentido em que se lê o nome da associação

UML: Diagrama de Classes

Classes de associação

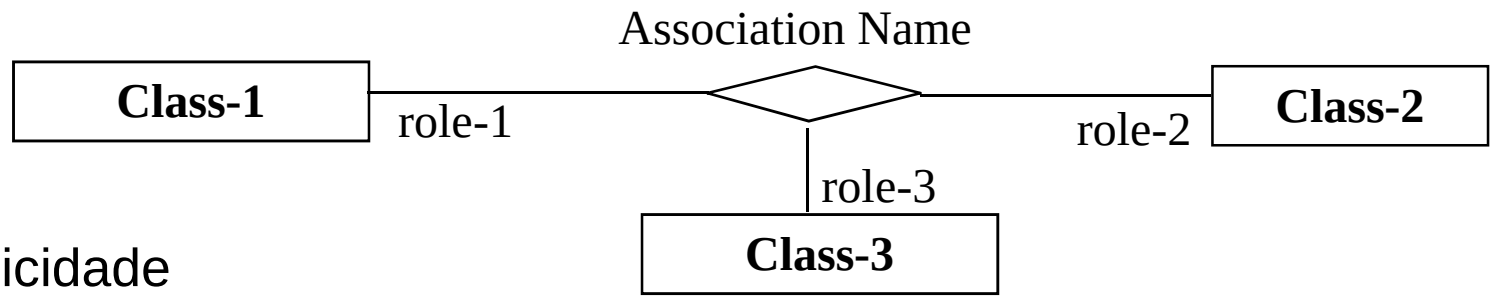


- reúne as propriedades de associação e classe
- o nome pode ser colocado num sítio ou noutro, conforme interessa realçar a natureza de associação ou de classe, mas a semântica é a mesma
- o nome também pode ser colocado nos dois sítios
- não é possível repetir combinações de objectos das classes participantes na associação

UML: Diagrama de Classes

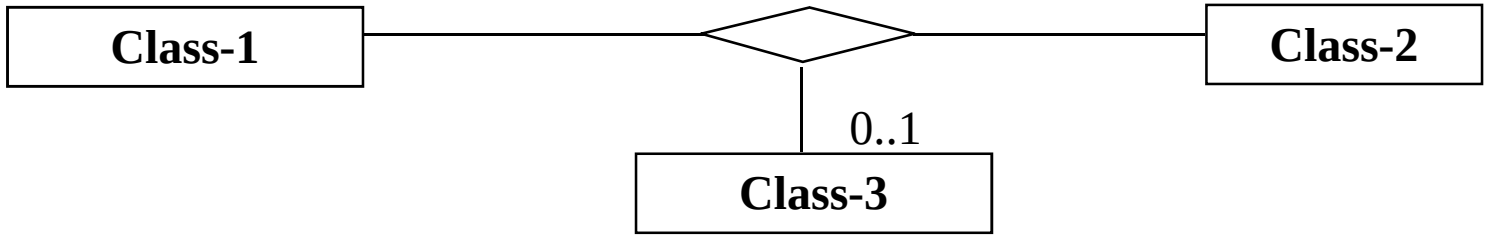
Associações n-árias

- Notação



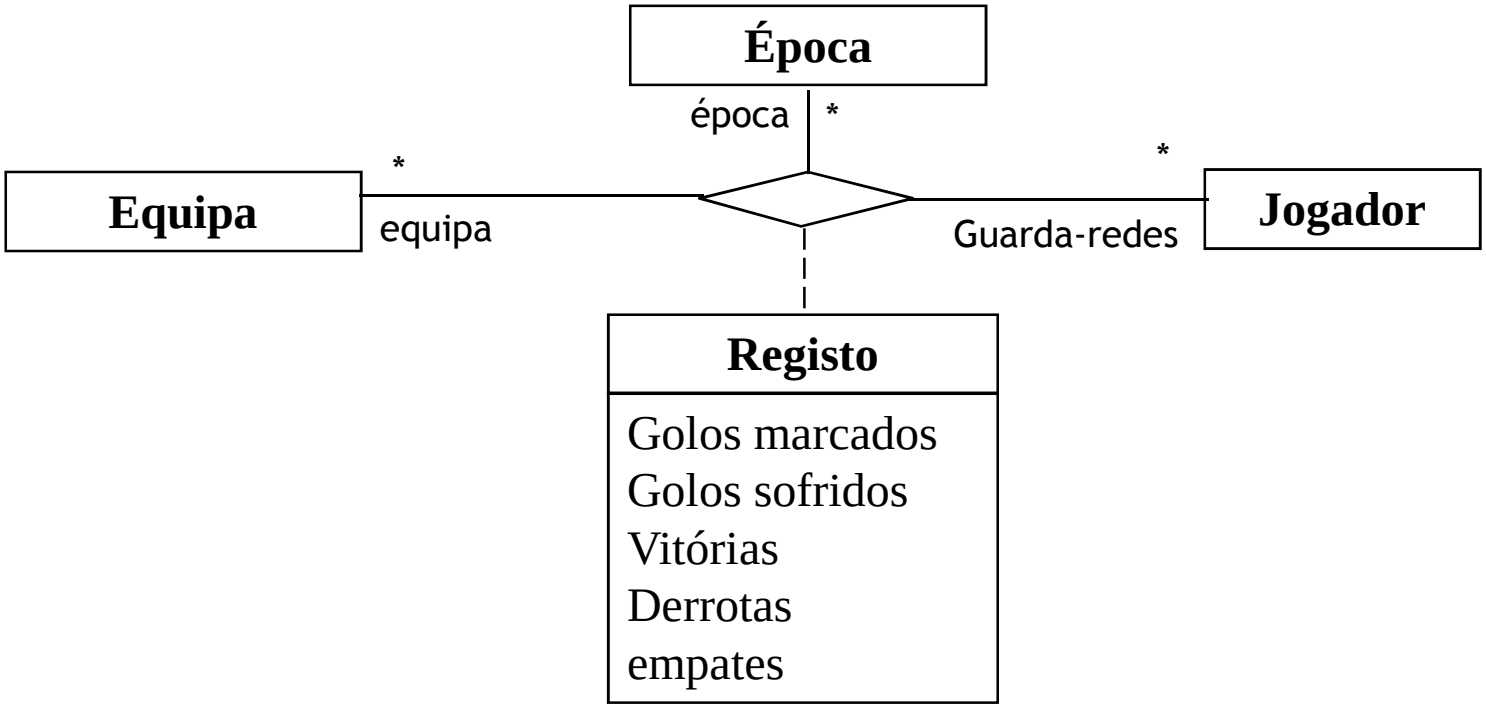
- Multiplicidade

  a cada par de objectos das restantes classes (1 e 2), correspondem 0 ou 1 objectos da classe 3



UML: Diagrama de Classes

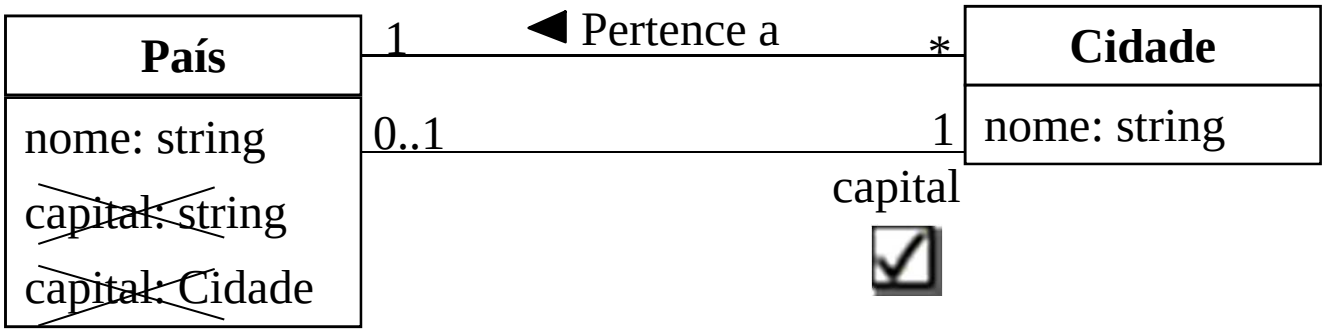
Associações n-árias



UML: Diagrama de Classes

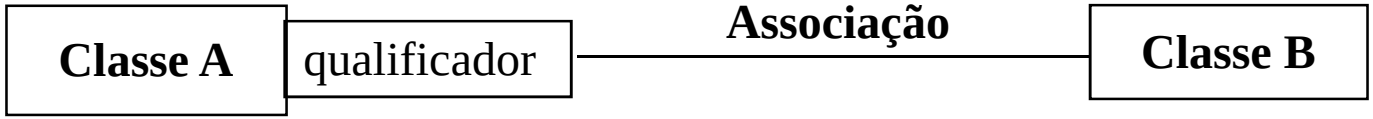
Associação vs. atributo

- Uma propriedade que designa um objecto de uma classe presente no modelo, deve ser modelada como uma associação e não como um atributo
- Exemplo:

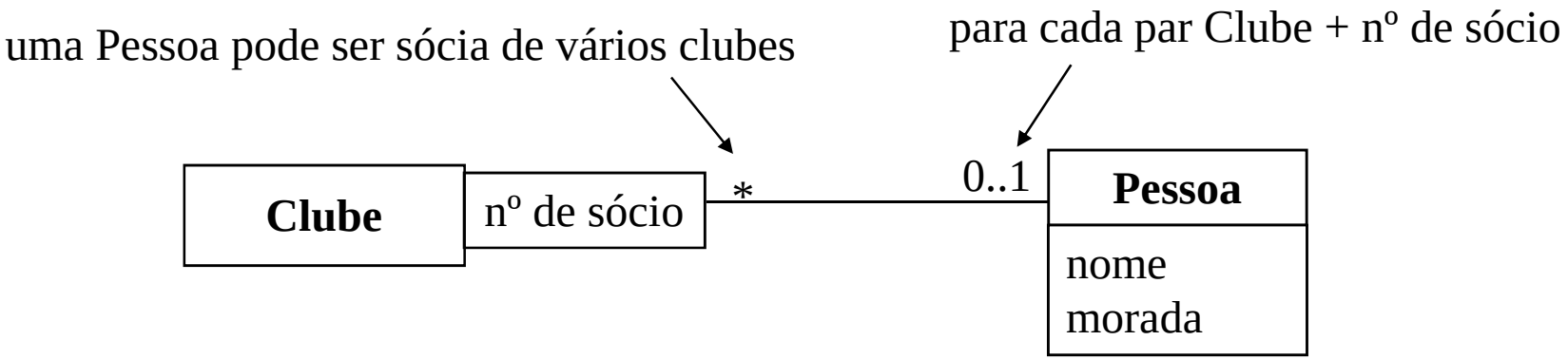


UML: Diagrama de Classes

Associações qualificadas



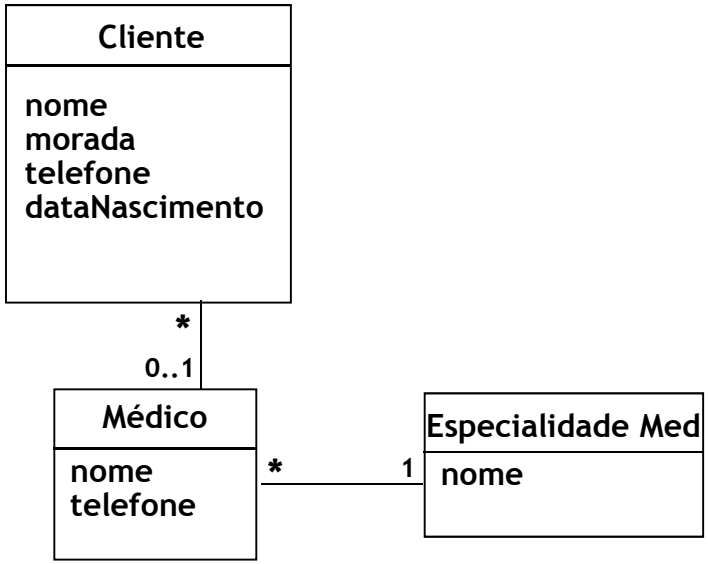
- Qualificador: lista de um ou mais atributos de uma associação utilizados para navegar de A para B
- "Chave de acesso" a B (acesso a um objecto ou conjunto de objectos) a partir de um objecto de A



Exercício

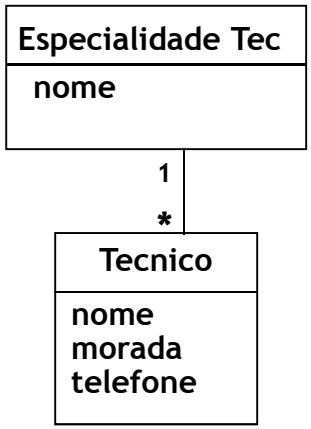
O gabinete de psicologia HiperEgo pretende contratar o desenvolvimento de um sistema de informação que permita organizar a informação sobre os seus clientes e técnicos.

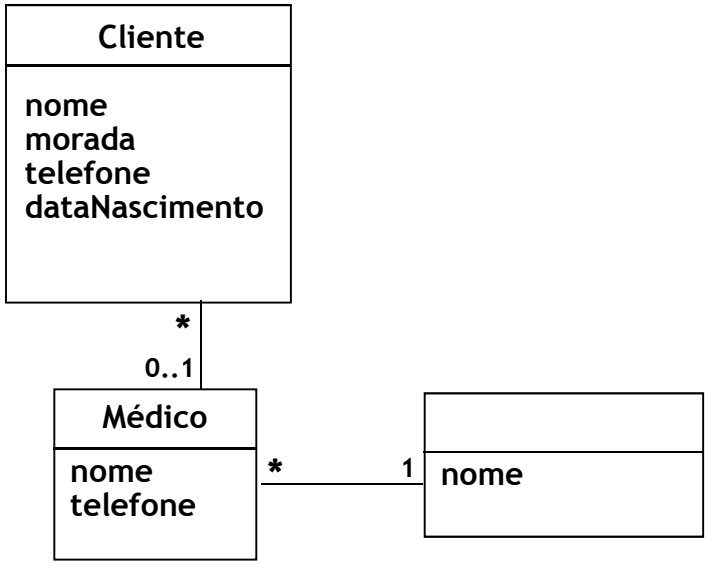
Sobre os clientes é necessário saber o nome, a morada, o telefone e a data de nascimento. É importante ainda saber se o apoio prestado pelo gabinete foi ou não receitado por um médico e, no caso afirmativo, é necessário saber o nome do médico, a sua especialidade e o número de telefone.



Exercício

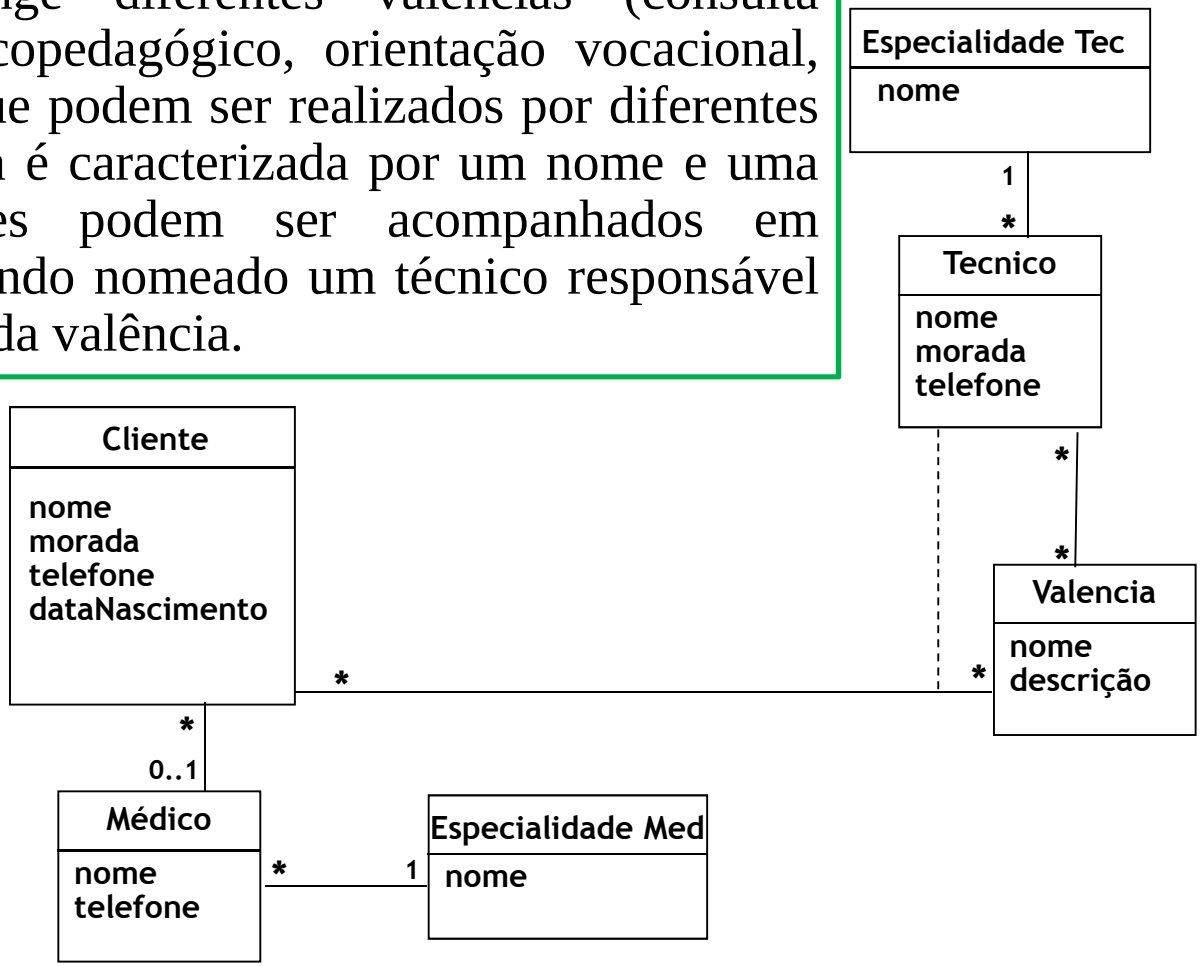
O gabinete conta com serviços de diferentes técnicos. Sobre cada um é necessário saber o nome, a morada, o telefone e a especialidade (psicólogo, terapeuta, professor especializado, etc.).





Exercício

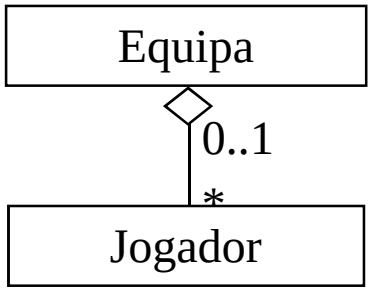
O gabinete abrange diferentes valências (consulta psicológica, apoio psicopedagógico, orientação vocacional, terapia da fala, etc.), que podem ser realizados por diferentes técnicos. Cada valência é caracterizada por um nome e uma descrição. Os clientes podem ser acompanhados em diferentes valências, sendo nomeado um técnico responsável para cada cliente em cada valência.



UML: Diagrama de Classes

Agregações

- Associação com o significado contém (é constituído por) / faz parte de (*part of*)
- Relação de inclusão nas instâncias das classes
- Hierarquias de objectos
- Exemplo:



• Uma equipa **contém** *0 ou mais* jogadores

• Um jogador **faz parte de** *uma* equipa (num dado momento), mas também pode estar desempregado

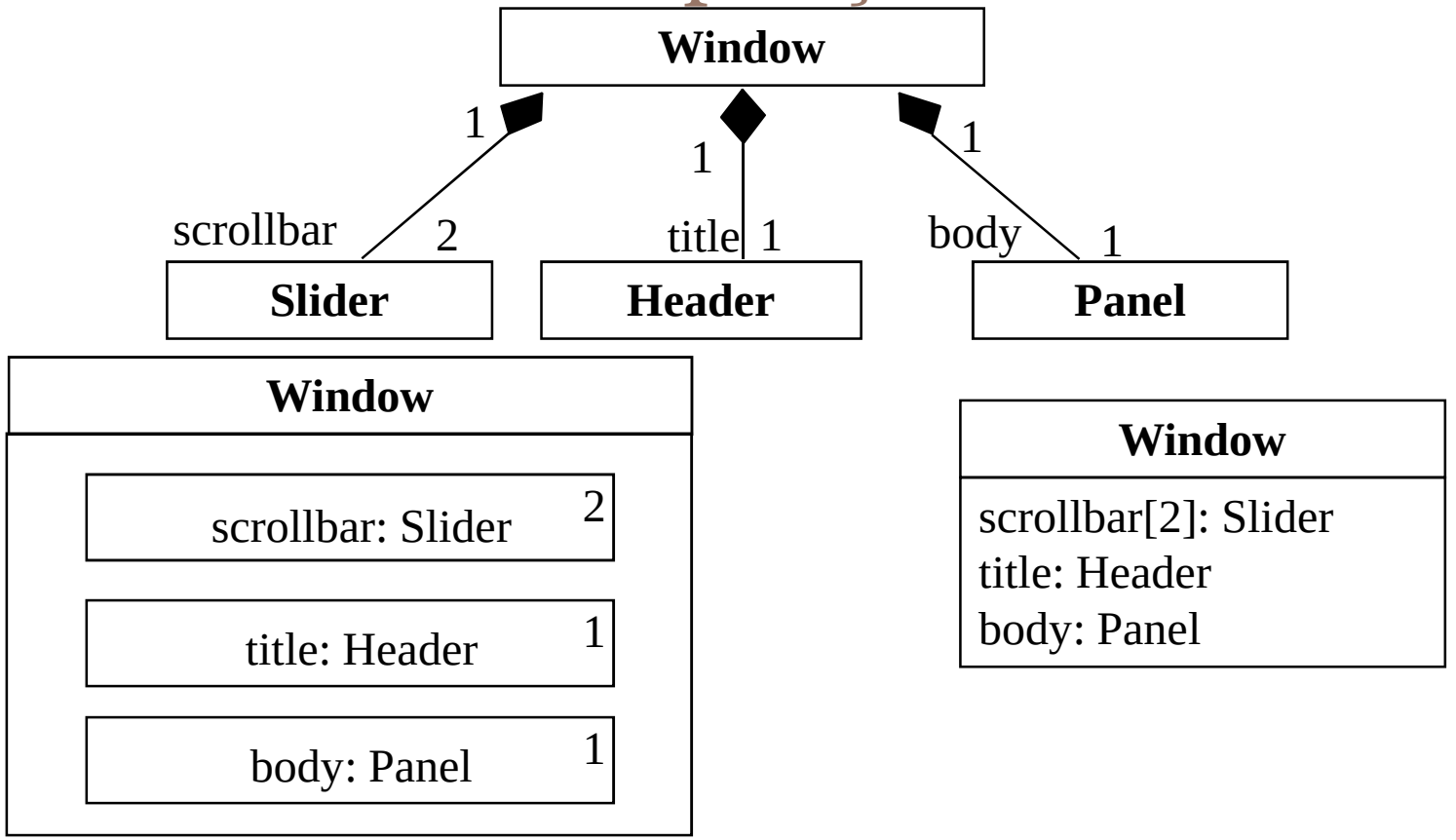
UML: Diagrama de Classes

Composições

- Forma mais forte de agregação aplicável quando:
  - existe um forte grau de pertença das partes ao todo
  - cada parte só pode fazer parte de um todo (i.e., a multiplicidade do lado do todo não excede 1)
  - o topo e as partes têm tempo de vida coincidente, ou, pelo menos, as partes nascem e morrem dentro de um todo
  - a eliminação do todo propaga-se para as partes, em cascata
- Notação: losango cheio (◆) ou notação encaixada
- Membros-objecto em C++

UML: Diagrama de Classes

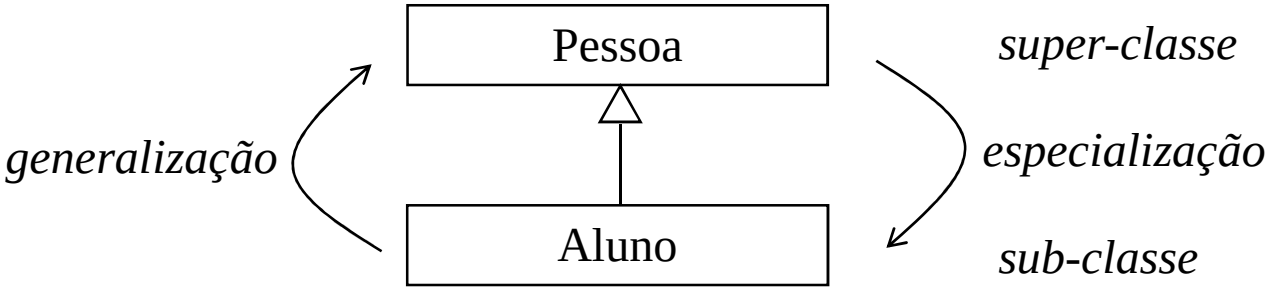
Composições



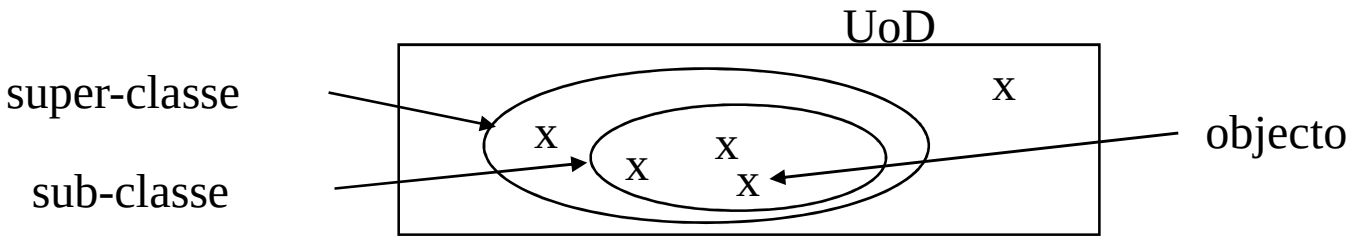
(sub-objectos no compartimento dos atributos)

UML: Diagrama de Classes

Generalizações



- **Relação semântica "is a" ("é um" / "é uma") :** um aluno é uma pessoa
- **Relação de inclusão nas extensões das classes:**



extensão (sub-classe) ⊂ extensão (super-classe)

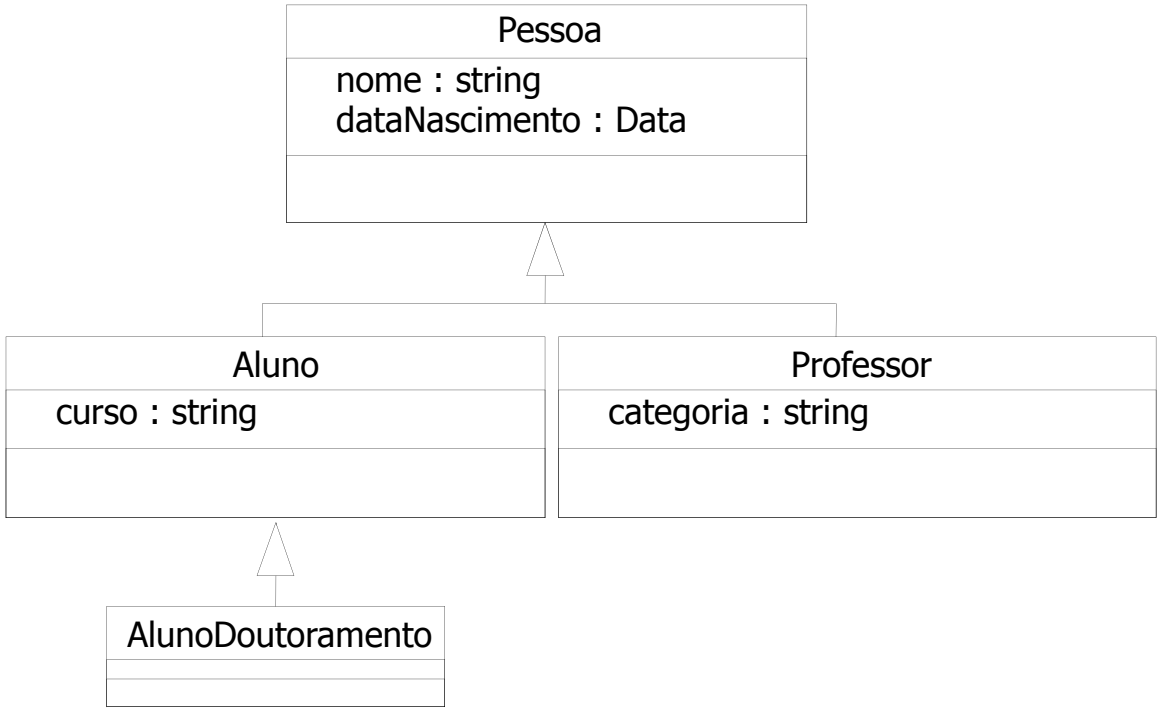
- **Relação de herança nas propriedades:** A sub-classe herda as propriedades (atributos, operações e relações) da super-classe, podendo acrescentar outras

UML: Diagrama de Classes

Generalizações: hierarquias de classes

- Em geral, pode-se ter uma hierarquia de classes relacionadas por herança / generalização
  - em cada classe da hierarquia colocam-se as propriedades que são comuns a todas as suas subclasses

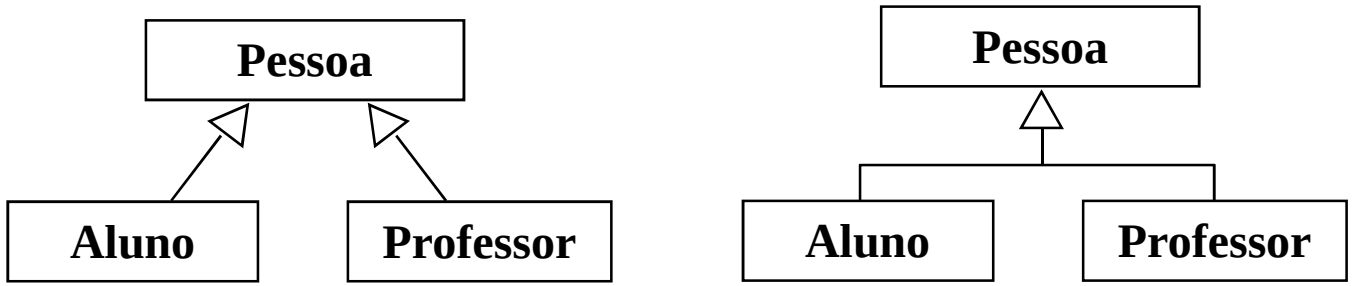
  ⇒ **evita-se redundância, promove-se reutilização!**



UML: Diagrama de Classes

Generalizações: hierarquias de classes

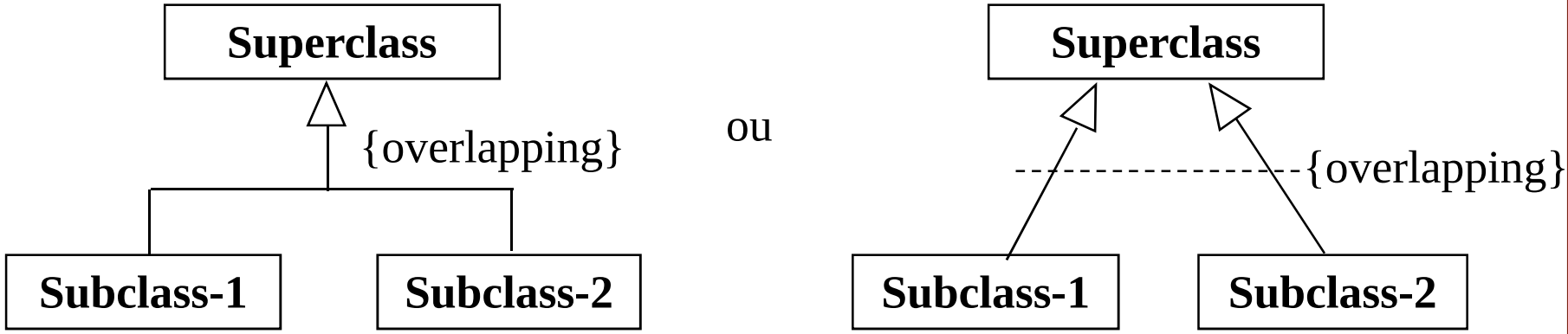
Notações alternativas:



UML: Diagrama de Classes

Generalizações: subclasses sobrepostas

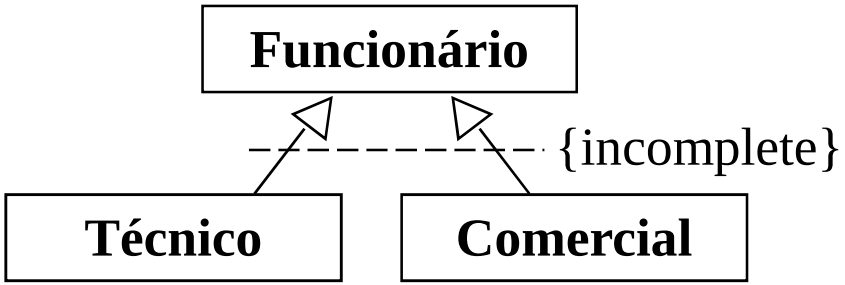
- caso em que um objecto da superclasse pode pertencer simultaneamente a mais do que uma subclasse
- indicado por restrição {overlapping}
- o contrário é {disjoint} (situação por omissão?)



UML: Diagrama de Classes

Generalizações: subclasses incompletas

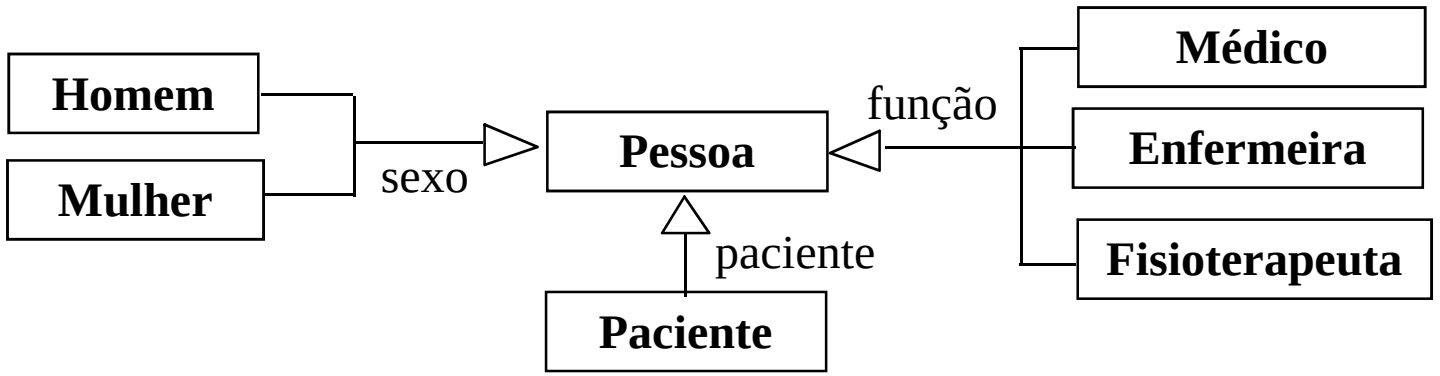
- caso em que um objecto da superclasse pode não pertencer a nenhuma das subclasses
- indicado por restrição {incomplete}
- o contrário é {complete} (situação por omissão?)



UML: Diagrama de Classes

Generalizações: classificação múltipla

- caso em que um objecto pode pertencer num dado momento a várias classes, sem que exista uma subclasse que represente a intersecção dessas classes (com herança múltipla)
- geralmente não suportado pelas linguagens de programação OO (pode então ser simulada por agregação de papéis)



exemplo de combinação legal: {Mulher, Paciente, Enfermeira}

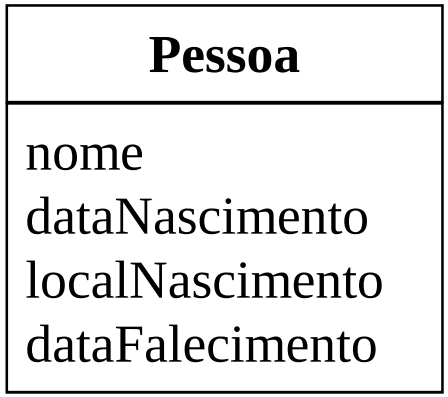
UML: Diagrama de Classes

Restrições

- Uma restrição especifica uma condição que tem de se verificar no estado do sistema (objectos e ligações)
- Uma restrição é indicada por uma expressão ou texto entre chavetas ou por uma nota posicionada junto aos elementos a que diz respeito, ou a eles ligada por linhas a traço interrompido (sem setas, para não confundir com relação de dependência)
- Podem ser formalizadas em UML com a OCL - "Object Constraint Language"
- Também podem ser formalizadas (como invariantes) numa linguagem de especificação formal como VDM++

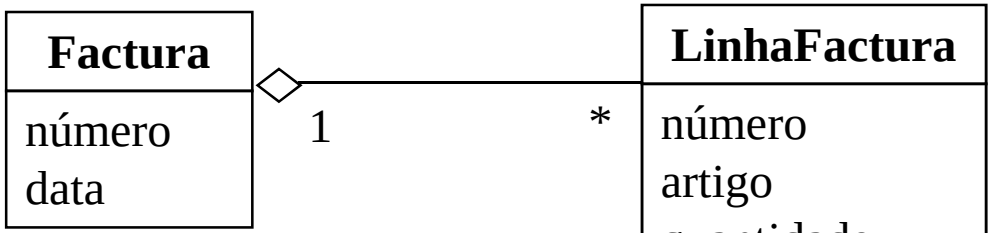
UML: Diagrama de Classes

Restrições: em classes



{chave candidata: (nome, dataNascimento, localNascimento)}

{dataFalecimento > dataNascimento}

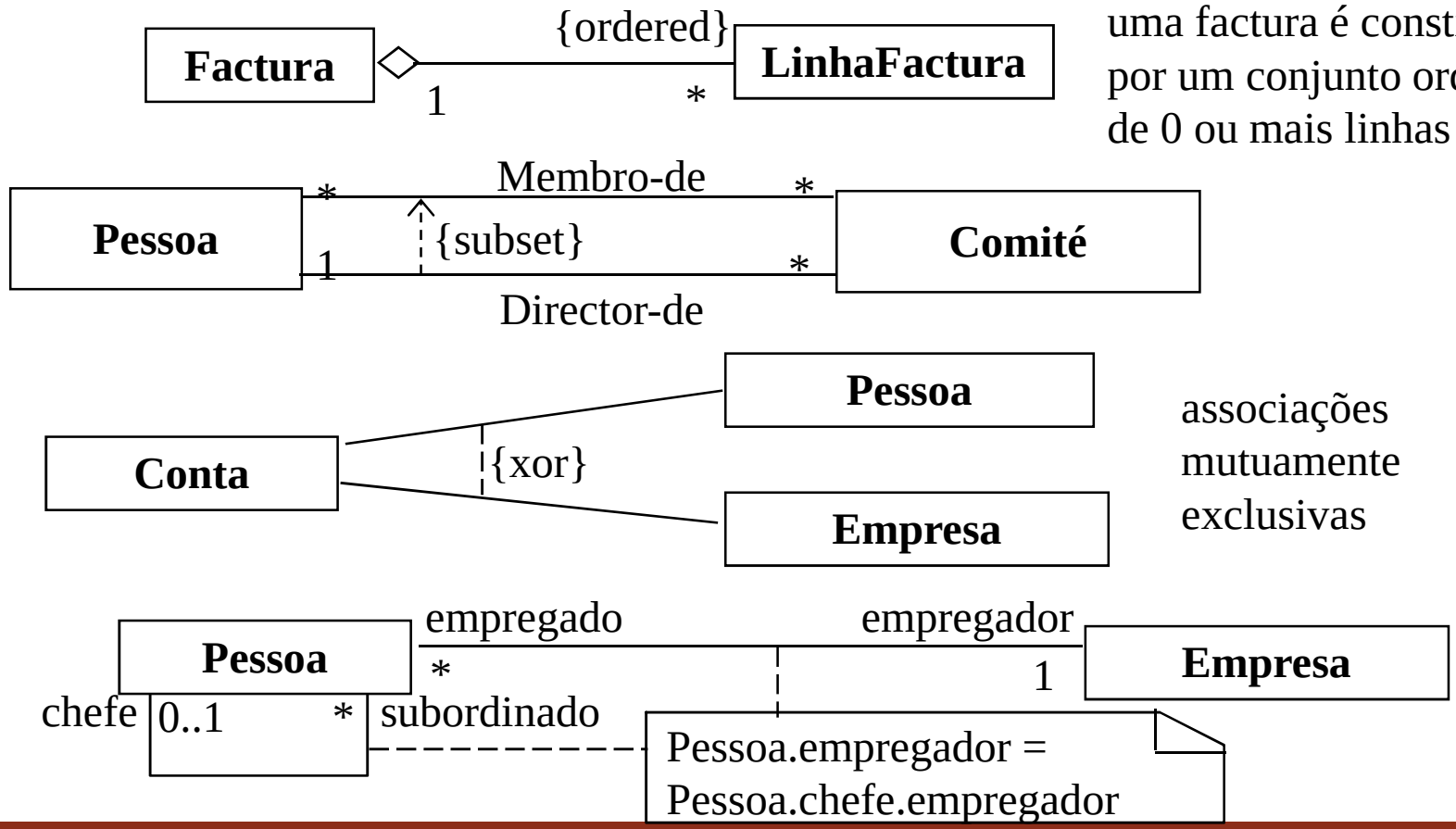


{chave candidata: (número)}

{chave candidata: (factura, número)}

UML: Diagrama de Classes

Restrições: em associações



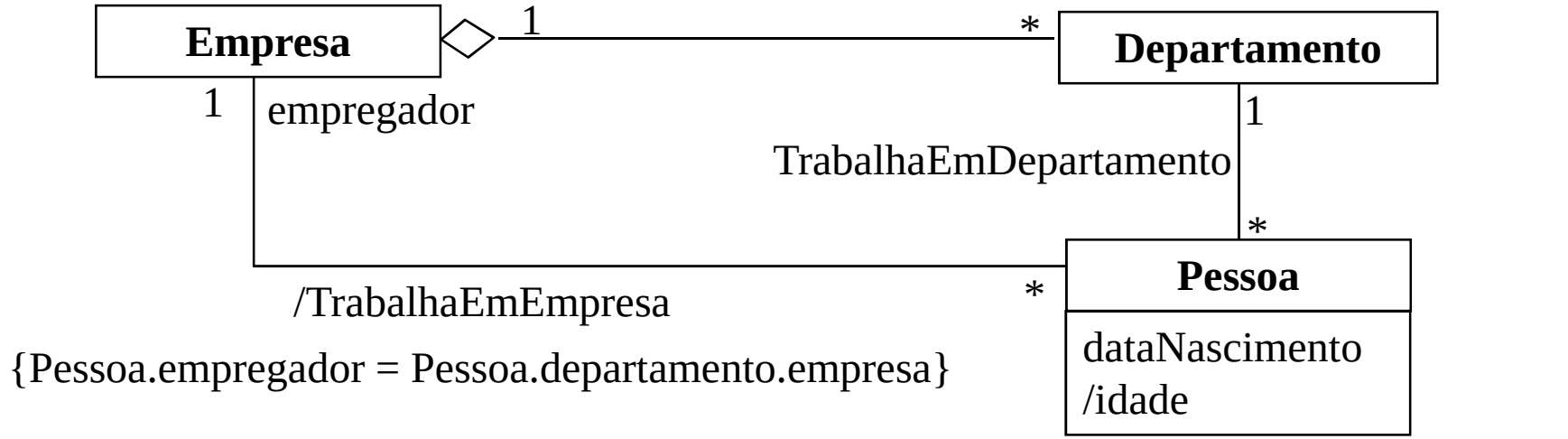
UML: Diagrama de Classes

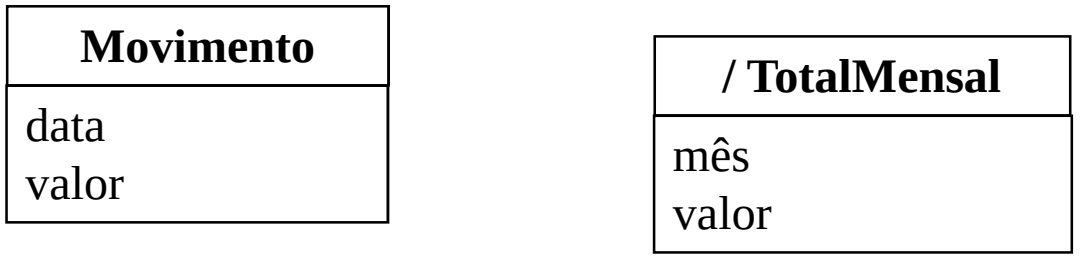
Elementos derivados

- Elemento derivado (atributo, associação ou classe): elemento calculado em função de outros elementos do modelo
- Notação: barra "/" antes do nome do elemento derivado
- Um elemento derivado tem normalmente associada uma restrição que o relaciona com os outros elementos

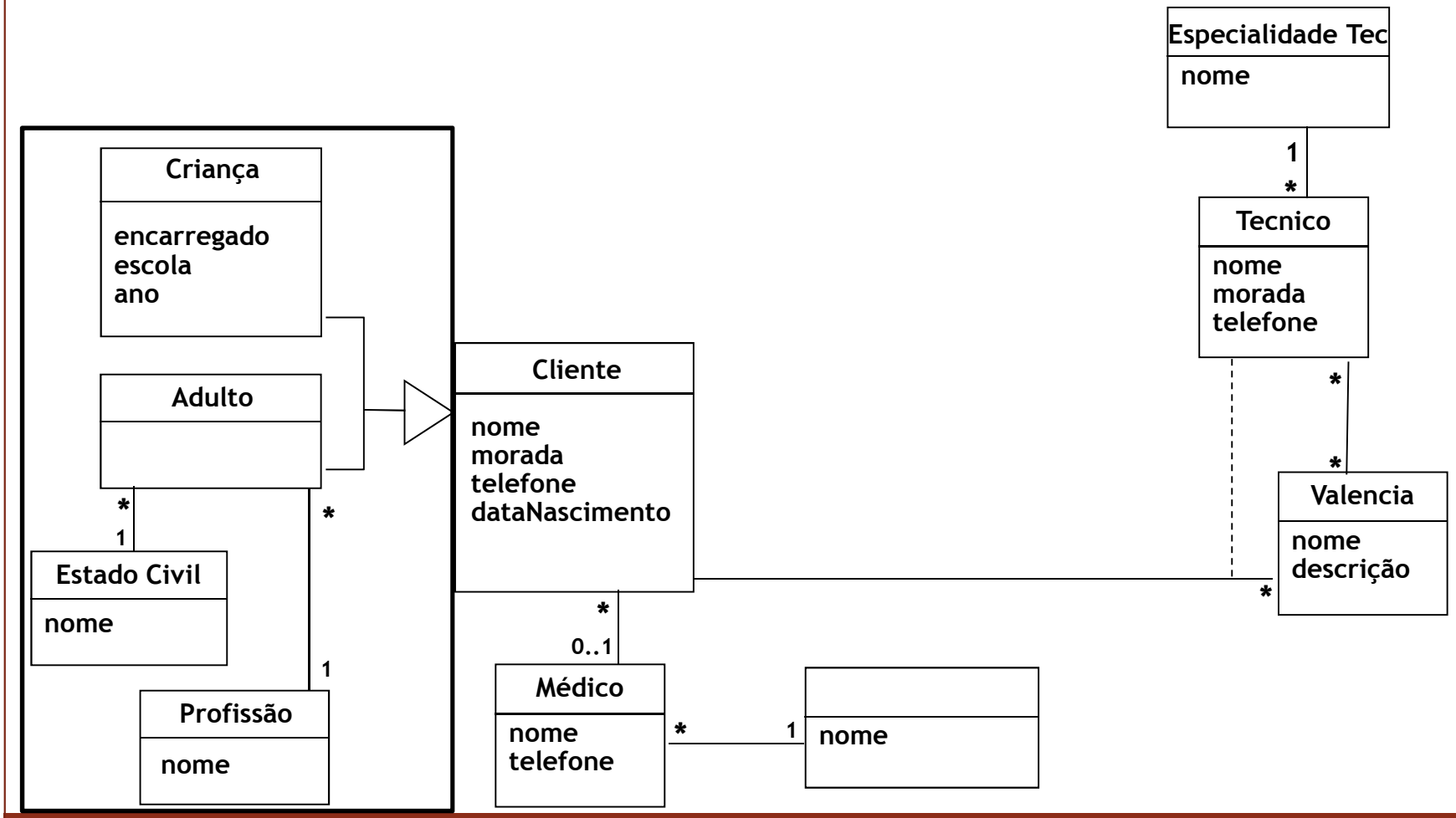
UML: Diagrama de Classes

Elementos derivados

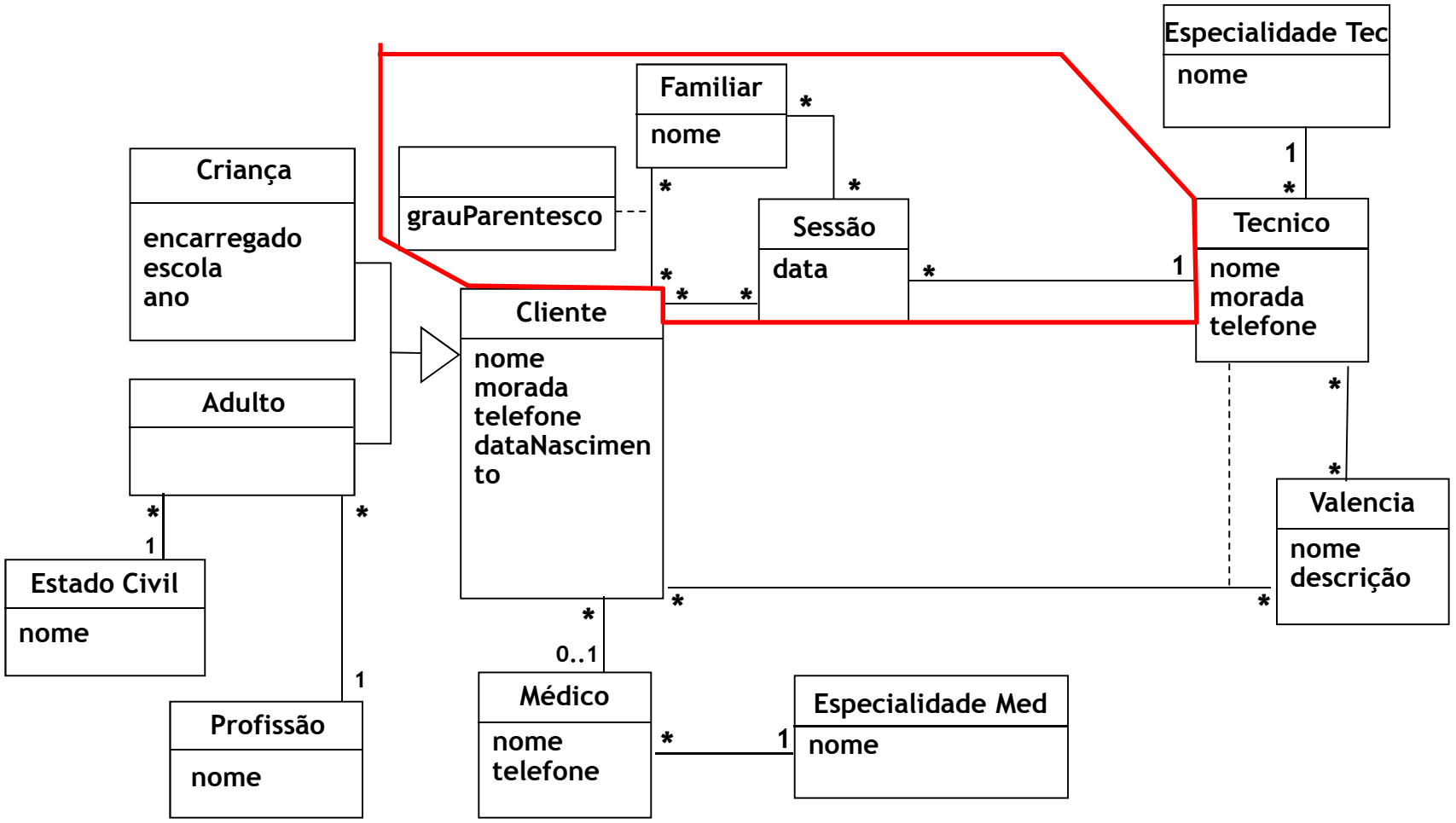




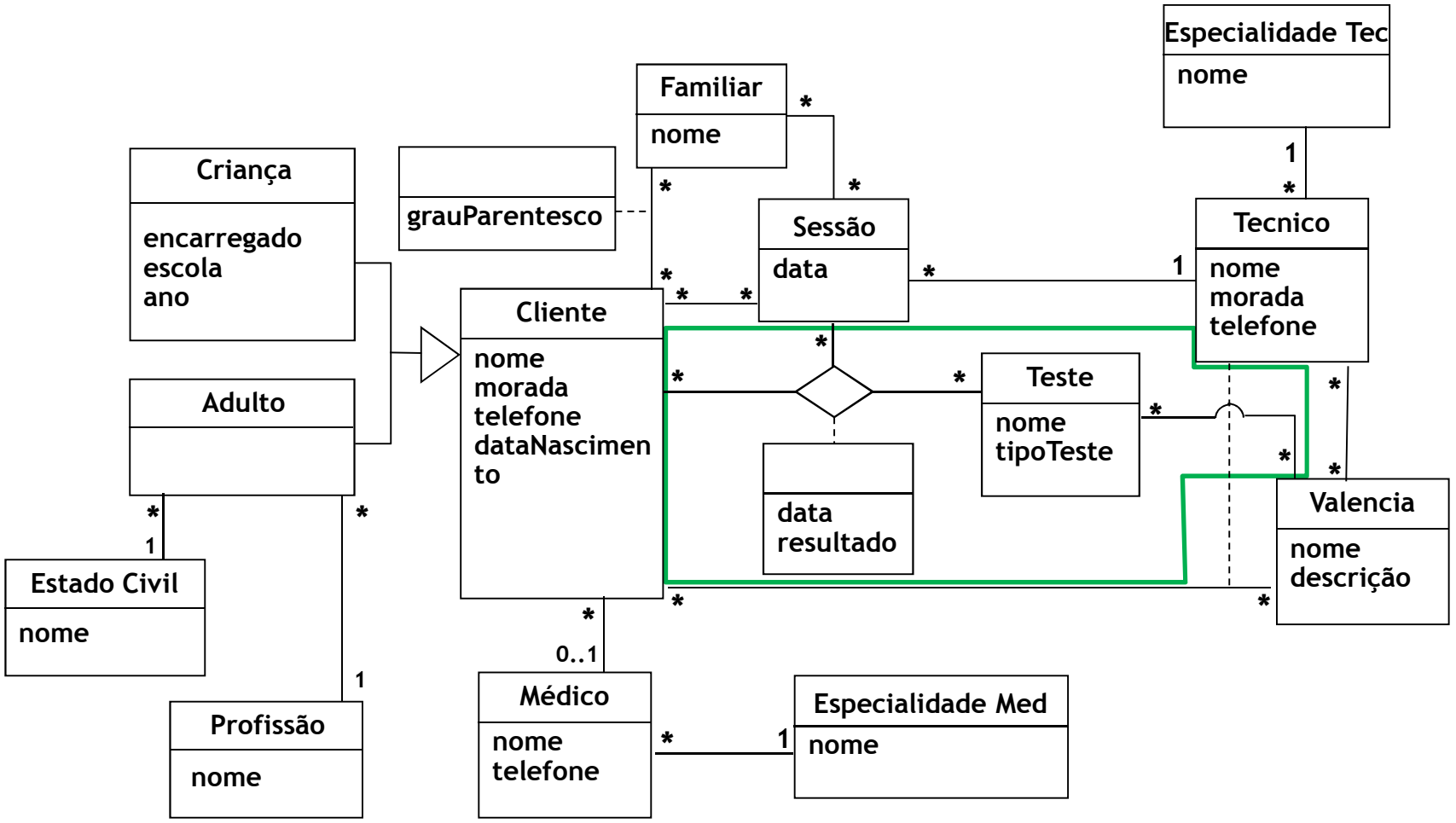
Os clientes do gabinete são maioritariamente crianças, mas um pequeno número são adultos. Sobre as crianças é necessário saber quem é o encarregado de educação, o nome da escola que frequentam e o ano em que estão. Sobre os adultos é necessário saber a profissão e o estado civil.



As actividades de gabinete são realizadas em sessões que podem ser em grupo ou individuais. Sobre cada sessão é importante saber a data em que se realizou e os participantes. Em cada sessão, além de um técnico do gabinete, podem participar vários clientes e também alguns familiares. Sobre estes é necessário saber o seu nome e o grau de parentesco com o cliente.



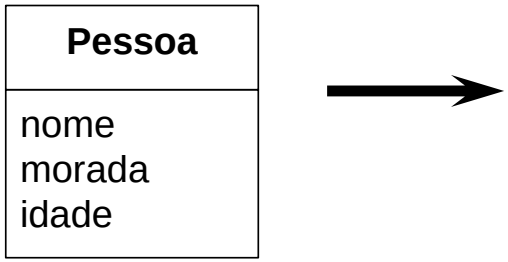
Durante as sessões os clientes podem efectuar diversos testes (psicotécnicos, WISC, COPS, etc.), sendo importante saber para cada um o nome, o tipo de teste e as valências com que está relacionado. De cada vez que um cliente realiza um teste é importante guardar a data e o resultado do teste.



UML: Diagrama de Classes

Mapeamento para MR: classes

- Cada classe é mapeada para uma relação (tabela)
  - a tabela terá os mesmos atributos que a classe respectiva, mais o atributo referente ao identificador de objecto
  - decide-se para cada atributo da tabela se poderá ou não ter valores nulos
  - atribui-se a cada atributo um domínio, pré-definido ou definido pelo utilizador



	Atributo	Nulos?	Domínio
Tabela	pessoa-ID	N	ID
Pessoa	nome	N	Nome
	morada	Y	Endereço
	idade	Y	Idade
	Chave candidata:	pessoa-ID	
	Chave primária:	pessoa-ID	
	Atributos mais acedidos:	(pessoa-ID) (nome)	

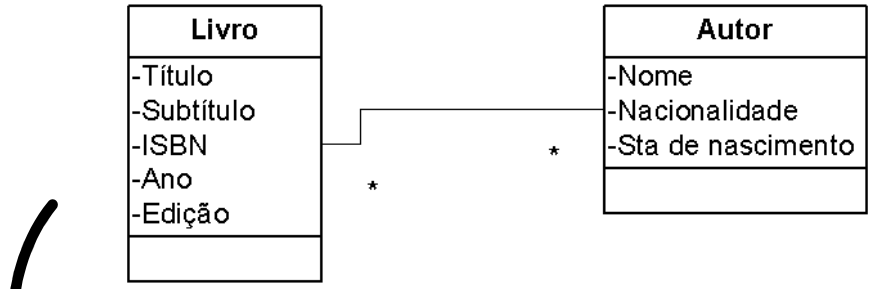
UML: Diagrama de Classes

Mapeamento para MR: associações muitos-para-muitos

- Em geral, uma associação pode ou não ser mapeada para uma tabela, dependendo:
  - do tipo e multiplicidade da associação
  - das preferências do projectista em termos de extensibilidade, número de tabelas e compromissos de desempenho
- Uma associação muitos-para-muitos é sempre mapeada para uma tabela distinta
  - a chave primária será constituída pela concatenação das chaves primárias de cada uma das tabelas envolvidas na associação

UML: Diagrama de Classes

Mapeamento para MR: associações muitos-para-muitos



	Atributo	Nulos?	Domínio
Tabela	idLivro	N	ID
Autores	idAutor	N	ID
	Chave candidata:	(idLivro, idAutor)	
	Chave primária:	(idLivro, idAutor)	
	Atributos mais acedidos:	(idLivro) (idAutor)	

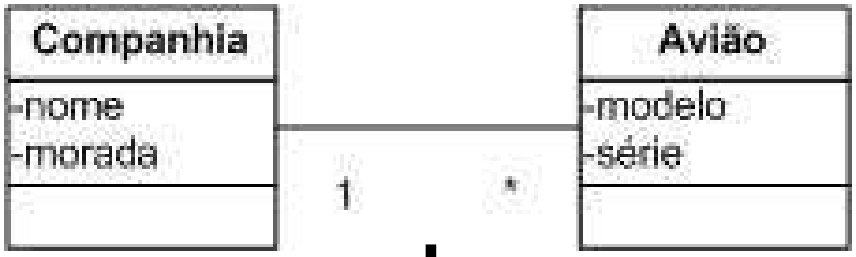
UML: Diagrama de Classes

Mapeamento para MR: associações um-para-muitos

- Uma associação um-para-muitos pode ser mapeada de 2 formas
  - criação de uma tabela distinta para a associação
  - adicionar uma chave-externa na tabela relativa a muitos
- Vantagens de não criar uma tabela distinta
  - menos tabelas no esquema final
  - maior desempenho devido a um menor número de tabelas a navegar
- Desvantagens de não criar uma tabela distinta
  - menos rigor em termos de projecto do esquema
  - extensibilidade reduzida

UML: Diagrama de Classes

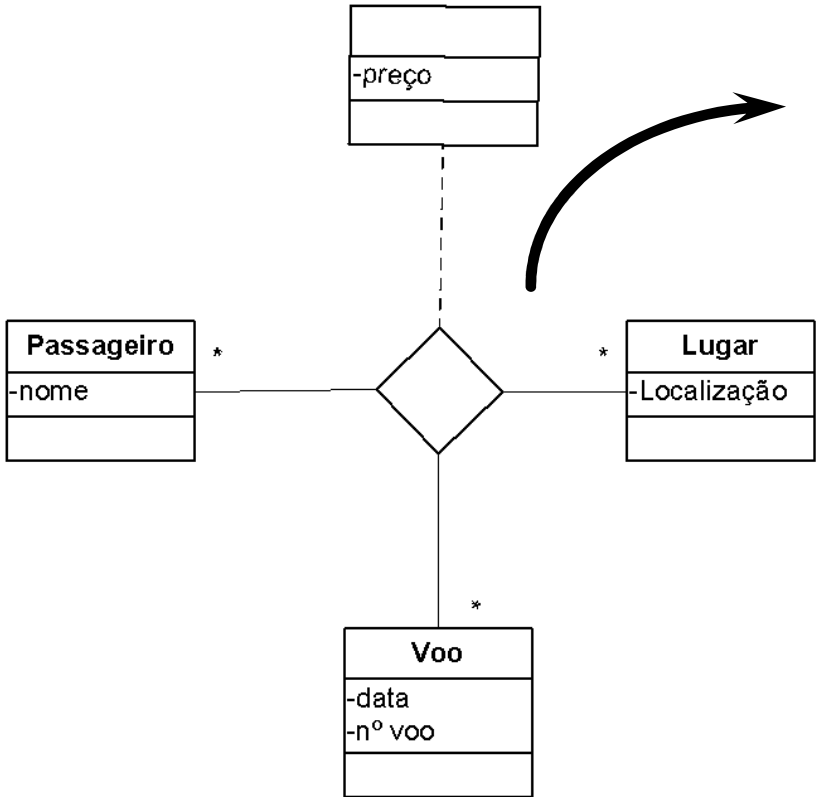
Mapeamento para MR: associações um-para-muitos



Atributo	Nulos?	Domínio
avião-ID	N	ID
modelo	Y	Text
no_serie	Y	Text
companhia-ID	Y	ID
Chave candidata:	(avião-ID)	
Chave primária:	(avião-ID)	
Atributos mais acedid	(avião-ID) (no_serie) (companhia-ID)	

UML: Diagrama de Classes

Mapeamento para MR: associações ternárias



	Atributo	Nulos?	Domínio
Tabela	passageiro-ID	N	ID
Preço	voo-ID	N	ID
	lugar-ID	N	ID
	preço	Y	Montante
	Chave candidata:	(passageiro-ID, voo-ID, lugar-ID)	
	Chave primária:	(passageiro-ID, voo-ID, lugar-ID)	
	Atributos mais acedidos	(passageiro-ID) (voo-ID) (lugar-ID)	

E como será o mapeamento de associações ternárias com outras multiplicidades?

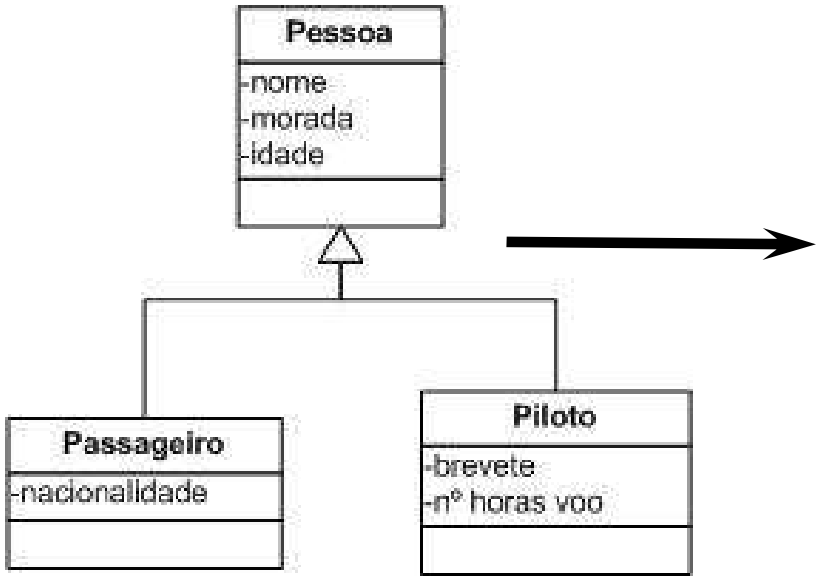
UML: Diagrama de Classes

Mapeamento para MR: generalizações (1)

1. Tanto a superclasse como cada uma das subclasses são mapeadas para tabelas distintas
   - abordagem logicamente correcta e extensível
   - envolve várias tabelas, pelo que a navegação da superclasse para as subclasses pode tornar-se lenta

UML: Diagrama de Classes

Mapeamento para MR: generalizações (1)



	Atributo	Nulos?	Domínio
Tabela	pessoa-ID	N	ID
Pessoa	nome	N	Nome
	morada	Y	Endereço
	idade	Y	Idade
	tipo-de-pessoa	N	Tipo-Pessoa
	Chave candidata:	(pessoa-ID)	
	Chave primária:	(pessoa-ID)	
	Atributos mais acedidos	(pessoa-ID) (nome)	

	Atributo	Nulos?	Domínio
Tabela	pessoa-ID	N	ID
Passageiro	nacionalidade	Y	País
	Chave candidata:	(pessoa-ID)	
	Chave primária:	(pessoa-ID)	
	Atributos mais acedidos	(pessoa-ID)	

	Atributo	Nulos?	Domínio
Tabela	pessoa-ID	N	ID
Piloto	brevete	Y	Text
	horas_de_voo	Y	Inteiro
	Chave candidata:	(pessoa-ID)	
	Chave primária:	(pessoa-ID)	
	Atributos mais acedidos	(pessoa-ID)	

UML: Diagrama de Classes

Mapeamento para MR: generalizações (2)

- 2. Eliminação da navegação da superclasse-para-subclasse
  - elimina a tabela da superclasse e replica todos os atributos dela em cada subclasse
  - ideal, quando a superclasse possui poucos atributos e as subclasses muitos atributos
  - não se pode garantir a unicidade dos valores dos atributos da superclasse

UML: Diagrama de Classes

Mapeamento para MR: generalizações (2)

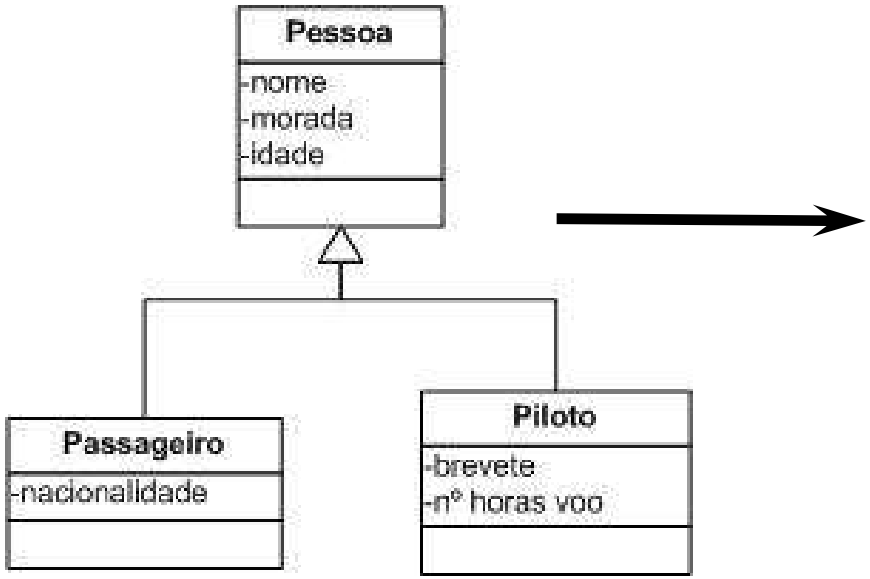


Tabela **Passageiro**

Atributo	Nulos?	Domínio
pessoa-ID	N	ID
nome	N	Nome
morada	Y	Endereço
idade	Y	Idade
nacionalidade	Y	País

Chave candidata: (pessoa-ID)
Chave primária: (pessoa-ID)
Atributos mais acedidos: (pessoa-ID)

Tabela **Piloto**

Atributo	Nulos?	Domínio
pessoa-ID	N	ID
nome	N	Nome
morada	Y	Endereço
idade	Y	Idade
brevete	Y	Text
horas_de_voo	Y	Inteiro

Chave candidata: (pessoa-ID)
Chave primária: (pessoa-ID)
Atributos mais acedidos: (pessoa-ID)

UML: Diagrama de Classes

Mapeamento para MR: generalizações (3)

- 3. Uma tabela para a superclasse
  - criação de apenas uma tabela para a superclasse
  - todos os atributos das subclasses são acrescentados à tabela da superclasse
  - cada subclasse utiliza apenas os atributos que lhe pertencem, deixando os restantes com valores nulos
  - viola a terceira forma normal
  - abordagem interessante quando em presença de poucas subclasses (2 ou 3)

UML: Diagrama de Classes

Mapeamento para MR: generalizações (3)

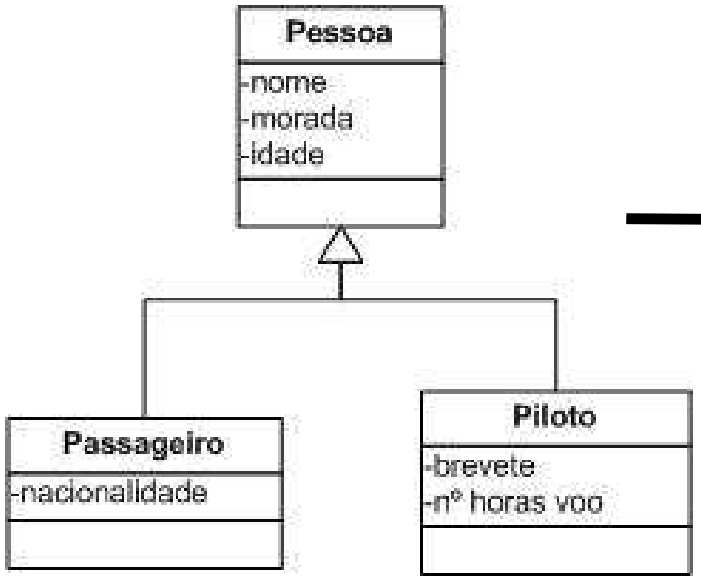


Tabela **Pessoa**

Atributo	Nulos?	Domínio
pessoa-ID	N	ID
nome	N	Nome
morada	Y	Endereço
idade	Y	Idade
tipo-de-pessoa	N	Tipo-Pessoa
nacionalidade	Y	País
brevete	Y	Text
horas_de_voo	Y	Inteiro

Chave candidata: (pessoa-ID)
Chave primária: (pessoa-ID)
Atributos mais acedidos: (pessoa-ID)

UML: Diagrama de Classes

Mapeamento para MR

- O mapeamento para BD Relacionais não é único
  - existem três formas de mapear uma generalização
  - é necessário adicionar detalhes que não existem no modelo de objectos, tais como, chaves primárias e chaves candidatas, se um atributo pode ter valores nulos ou não
  - obriga à atribuição de um domínio a cada atributo
  - e obriga a listar os atributos mais frequentemente acedidos

UML: Diagrama de Classes

Mapeamento para MR

- Resumindo
  - criar um atributo *id* correspondente a cada classe
  - *classe* implementada por *relação*
  - *associação* muitos-para-muitos ou ternária implementada por *relação*
  - *associação* muitos-para-um implementada por *atributo extra* (*id* do lado 1) na relação do lado muitos (*)
  - traduzir hierarquias de uma das três formas apresentadas

Exercício (continuação)

Mapeamento para MR

- Converter o diagrama de classes obtido anteriormente para o modelo relacional.

Exercício (continuação)

Mapeamento para MR

- Cliente(idCliente, nome, morada, telefone, dataNasc, idMedico->Medico, tipoCliente)
- Crianca(idCrianca->Cliente.idCliente, encarregado, escola, ano)
- Adulto(idAdulto->Cliente.idCliente, idEstadoCivil->EstadoCivil, idProfissao->Profissao)
- EstadoCivil(idEstadoCivil, nome)
- Profissao(idProfissao, nome)
- Medico(idMedico, nome, telefone, idEspecialidadeMed->EspecialidadeMed)
- EspecialidadeMed(idEspecialidadeMed, nome)
- Familiar(idFamiliar, nome)
- Sessao(idSessao, data, idTecnico->Tecnico)
- Teste(idTeste, nome, tipoTeste)
- Parentesco(idFamiliar->Familiar, idCliente->Cliente, grauParentesco)
- ParticipacoesFam(idFamiliar->Familiar, idSessao->Sessao)
- ParticipacoesCli(idCliente->Cliente, idSessao->Sessao)
- ResultadosTeste(idCliente->Cliente, idSessao->Sessao, idTeste->Teste, data, resultado)
- Tecnico(idTecnico, nome, morada, telefone, idEspecialidadeTec->EspecialidadeTec)
- EspecialidadeTec(idEspecialidadeTec, nome)
- Valencia(idValencia, nome, descricao)
- ValenciaTeste(idValencia->Valencia, idTeste->Teste)
- ValenciaCliente(idValencia->Valencia, idCliente->Cliente, idTecnico->Tecnico)
- ValenciaTecnico(idValencia->Valencia, idTecnico->Tecnico)