Diagrama de Classes

Herysson R. Figueiredo
herysson.figueiredo@ufn.edu.br

Diagrama de classes

O diagrama de classes é utilizado na construção do modelo de classes desde o nível de análise até o nível de especificação. De todos os diagramas da UML, esse é o mais rico em termos de notação.

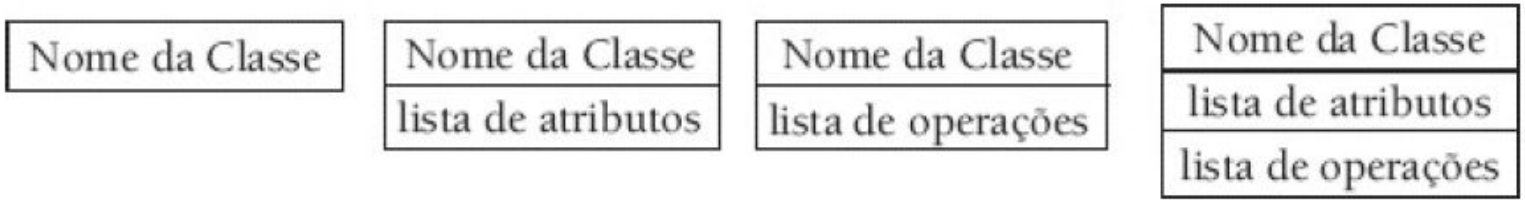
Classes

Uma classe é representada por uma “caixa” com, no máximo, três compartimentos exibidos.

- No primeiro compartimento (de cima para baixo) é exibido o seu nome.
- No segundo compartimento, são declarados os atributos que correspondem às informações que um objeto armazena.
- No terceiro compartimento, são declaradas as operações, que correspondem às ações que um objeto sabe realizar.

Classes

As possíveis notações da UML para representar classes são:



O grau de abstração desejado em certo momento do desenvolvimento do modelo de classes direciona a utilização de uma ou outra notação.

Classes

Estruturalmente, uma classe é composta de **atributos** e de **operações**.

- Os **atributos** correspondem à descrição dos dados armazenados pelos objetos de uma classe.
  - A cada atributo de uma classe está associado um conjunto de valores que esse atributo pode assumir

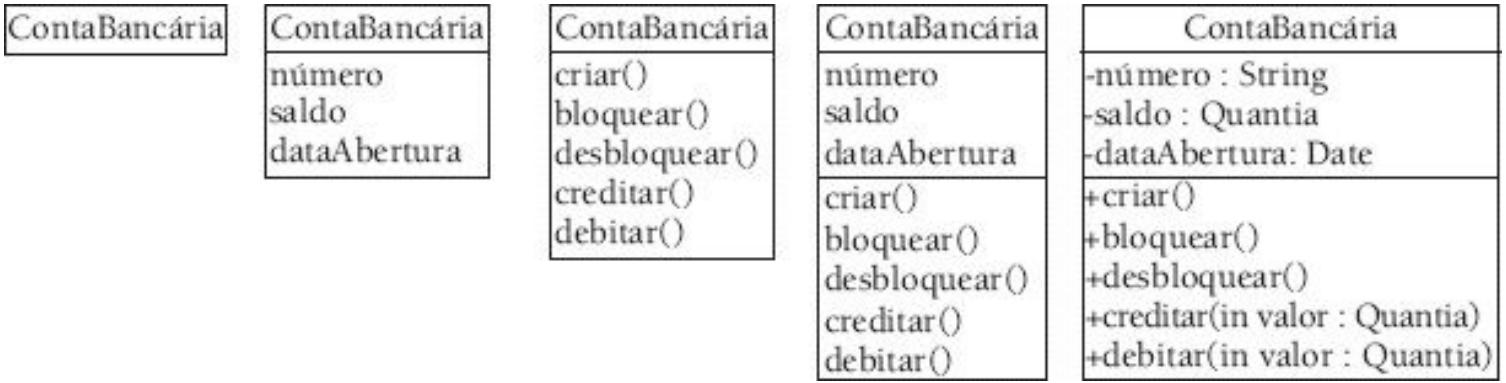
Classes

Estruturalmente, uma classe é composta de **atributos** e de **operações**.

- As **operações** correspondem à descrição das ações que os objetos de uma classe sabem realizar.
  - Ao contrário dos atributos, os objetos de uma classe compartilham as mesmas operações.
  - O nome de uma operação normalmente contém um verbo e um complemento, terminando com um par de parênteses

Classes

Exemplos de representação de uma mesma classe, **ContaBancária**, em diferentes graus de abstração.



Associações

A ocorrência de uma classe é chamada de **objeto** ou **instância.** Estes objetos de um sistema podem se relacionar uns com os outros e a existência de um **relacionamento** entre dois objetos possibilita a **troca de mensagens**.

Portanto, em última análise, **relacionamentos entre objetos permitem que eles colaborem** entre si para produzir as funcionalidades do sistema.

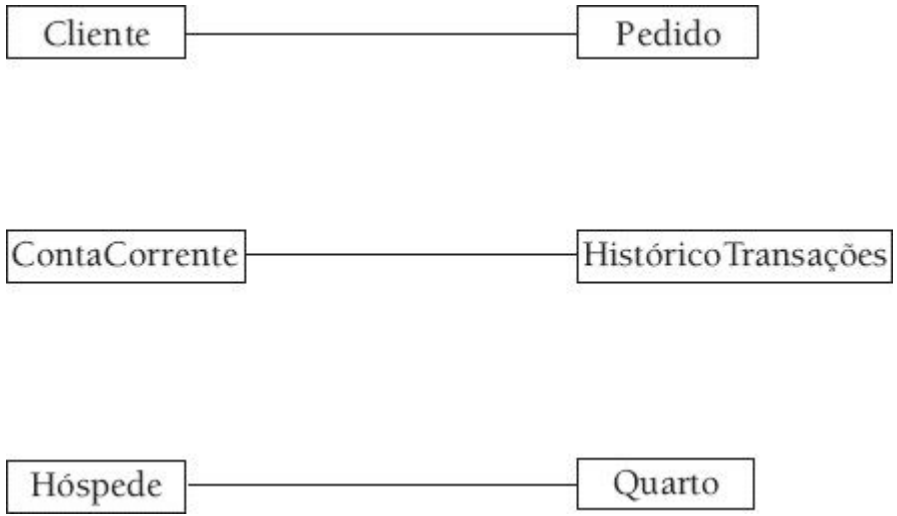
Associações

A associação representa relacionamentos que são formados entre objetos durante a execução do sistema.

Uma associação é representada no diagrama de classes por uma **linha** (normalmente um segmento de reta) ligando as classes às quais pertencem os objetos relacionados.

Associações

1. no domínio de vendas, um *cliente* **compra** *produtos*;
2. no domínio bancário, uma *contacorrente* **possui** um *histórico de transações*;
3. Em um hotel, há vários hóspedes, assim como há vários quartos. Os *hóspedes* do hotel **ocupam** *quartos*.



Associações

As Associações possuem diversas características importantes: **multiplicidades, nome, direção de leitura, papéis, tipo de participação e conectividade.**

Multiplicidade

As associações permitem a representação da informação dos limites inferior e superior da **quantidade de objetos** aos quais outro objeto pode estar associado. Esses limites são chamados de **multiplicidades** na terminologia da UML.

Multiplicidade

As associações permitem a representação da informação dos limites inferior e superior da **quantidade de objetos** aos quais outro objeto pode estar associado. Esses limites são chamados de **multiplicidades** na terminologia da UML.

Cada associação em um diagrama de classes possui duas multiplicidades, uma em cada extremo da linha que a representa.

Multiplicidade

Os símbolos possíveis para representar uma multiplicidade são:

Nome	Simbologia
Apenas Um	1
Zero ou Muitos	0..*
Um ou Muitos	1..*
Zero ou Um	0..1
Intervalo Específico	$I_i..I_s$

Multiplicidade

Como exemplo, **Pedido** e **Cliente**, e uma associação entre as mesmas.

Multiplicidade

Como exemplo, **Pedido** e **Cliente**, e uma associação entre as mesmas.

A leitura dessa associação nos informa que pode haver **um objeto da classe Cliente** que esteja associado a **vários objetos da classe Pedido.**

Multiplicidade

Como exemplo, **Pedido** e **Cliente**, e uma associação entre as mesmas.

Além disso, essa mesma leitura nos informa que **pode haver um objeto da classe Cliente que não esteja associado a pedido algum.**

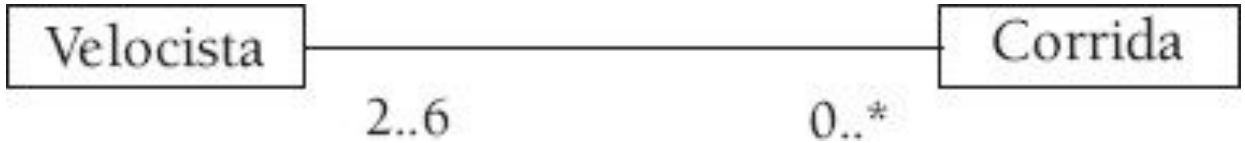
Multiplicidade

Como exemplo, **Pedido** e **Cliente**, e uma associação entre as mesmas.

O * denota, em todos os símbolos nos quais aparece, que **não há um limite superior** predefinido para a quantidade máxima de objetos com os quais outro objeto pode se associar.

Multiplicidade

Como exemplo, Velocista e Corrida, e uma associação entre as mesmas.



Multiplicidade

Como exemplo, Velocista e Corrida, e uma associação entre as mesmas.

O valor para I_i **dois** (uma corrida está associada a, **no mínimo, dois velocistas**), e o valor para I_s é **seis** (uma corrida está associada a, no máximo, seis velocistas).

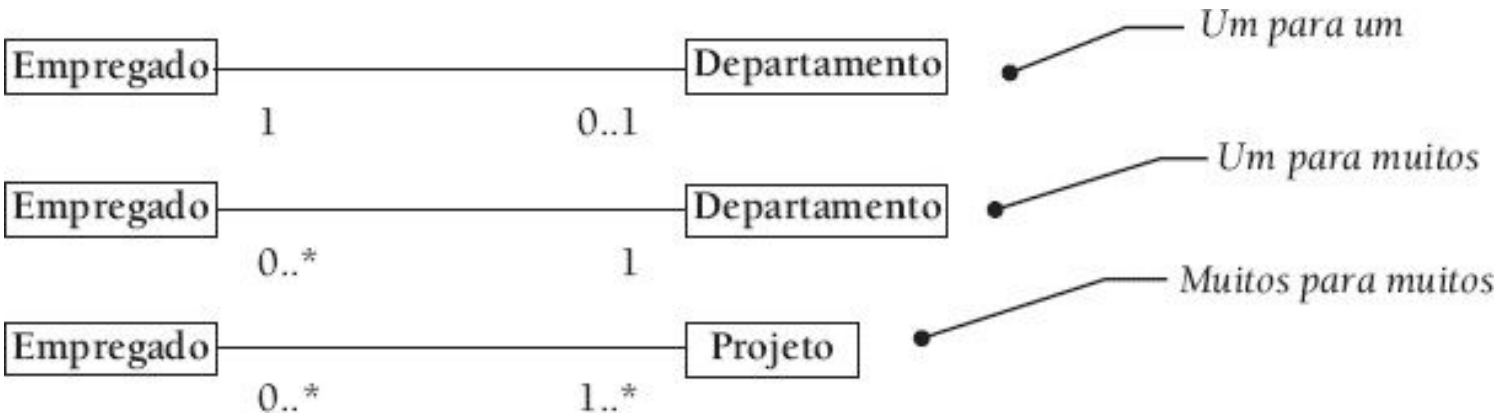
Multiplicidade

As **associações** podem ser agrupadas em apenas três tipos: **“muitos para muitos”, “um para muitos”** e **“um para um”.**

Conectividade	Multiplicidade de um extremo	Multiplicidade do outro extremo
Um para um	0..1 ou 1	0..1 ou 1
Um para muitos	0..1 ou 1	* ou 1..* ou 0..*
Muitos para muitos	* ou 1..* ou 0..*	* ou 1..* ou 0..*

Multiplicidade

As **associações** podem ser agrupadas em apenas três tipos: **“muitos para muitos”, “um para muitos”** e **“um para um”.**

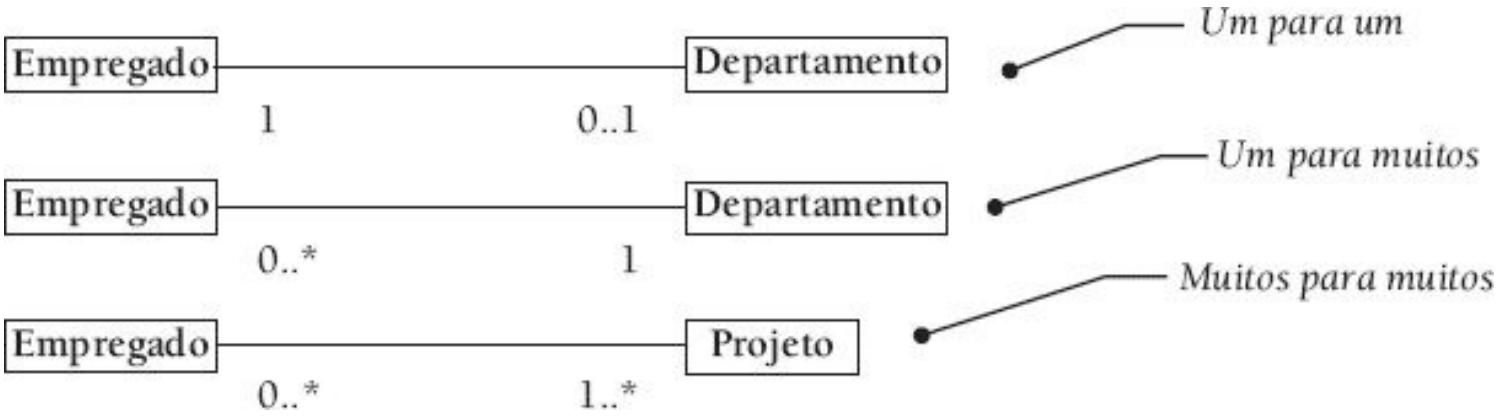


Participação

Uma característica importante de uma associação está relacionada à necessidade ou não da existência dessa associação entre objetos. Essa característica é denominada **participação**. A participação pode ser **obrigatória** ou **opcional**.

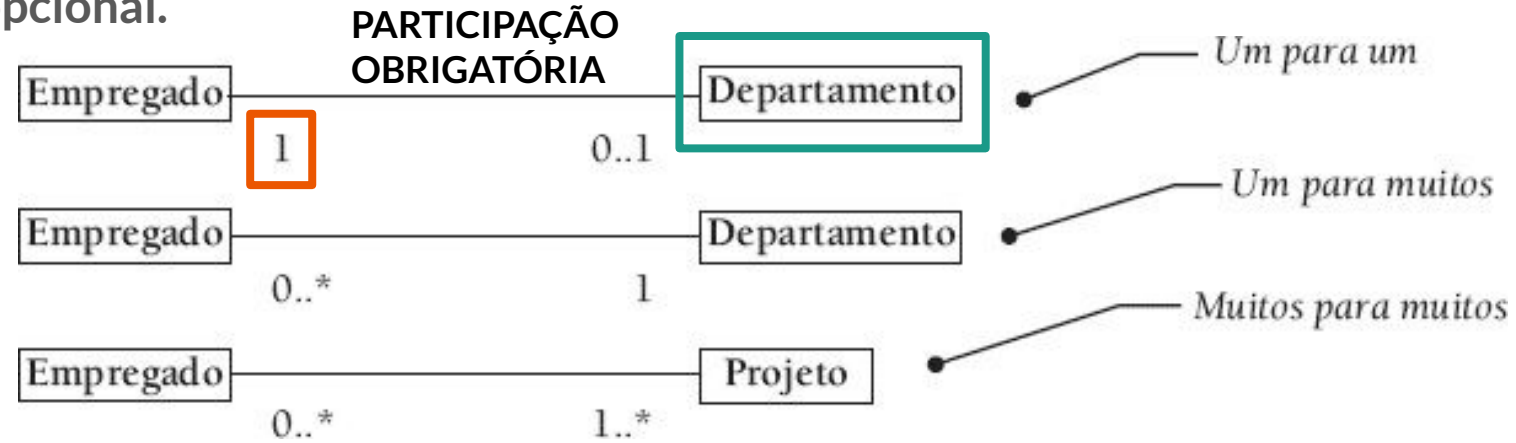
Participação

Se o valor *mínimo da multiplicidade* de uma associação *é igual a 1 (um)*, significa que a participação é **obrigatória**. Caso contrário, a participação é **opcional.**



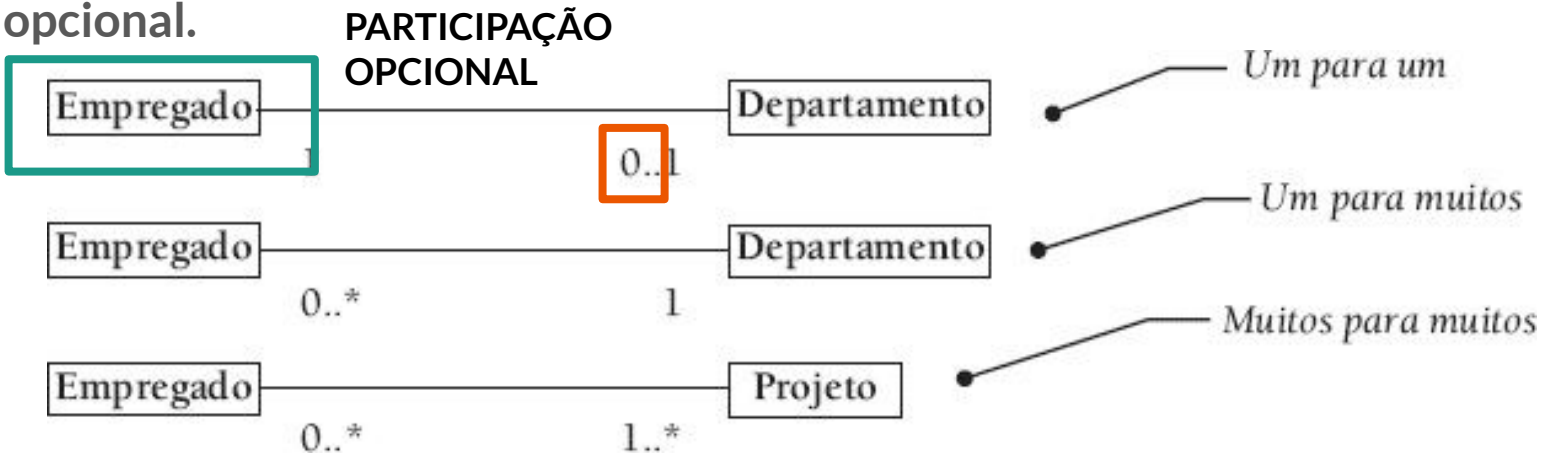
Participação

Se o valor *mínimo da multiplicidade* de uma associação *é igual a 1 (um)*, significa que a participação é **obrigatória**. Caso contrário, a participação é **opcional.**



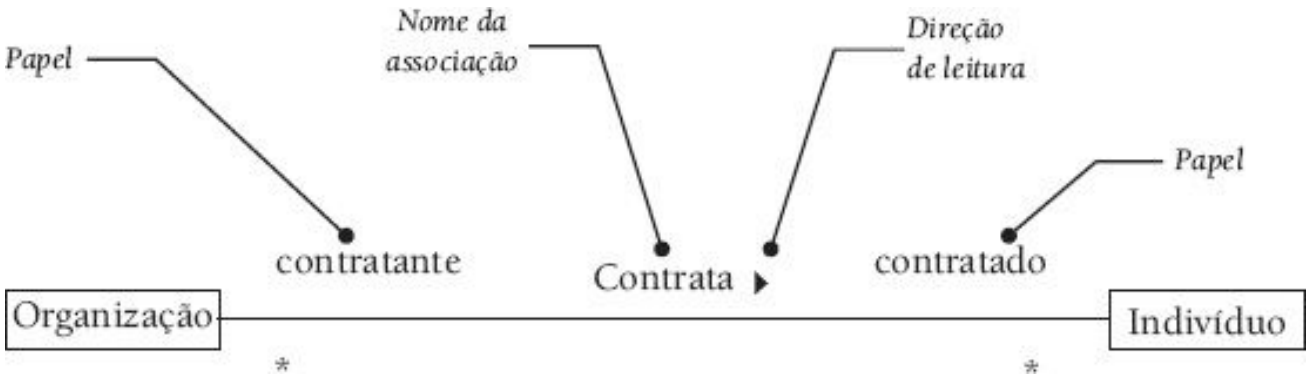
Participação

Se o valor *mínimo da multiplicidade* de uma associação *é igual a 1 (um)*, significa que a participação é **obrigatória**. Caso contrário, a participação é **opcional.**



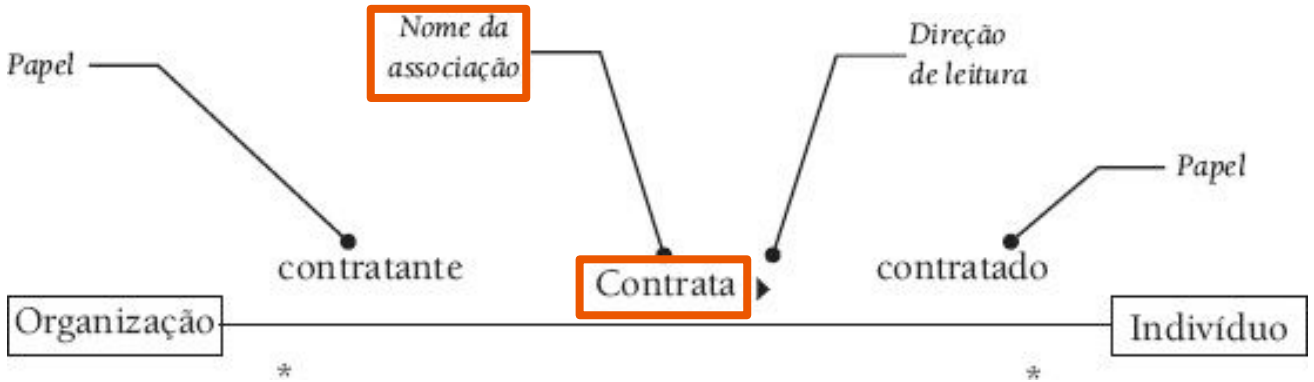
Nome de associação, direção de leitura e papéis

A UML define três recursos de notação: **nome de associação, direção de leitura e papel.**



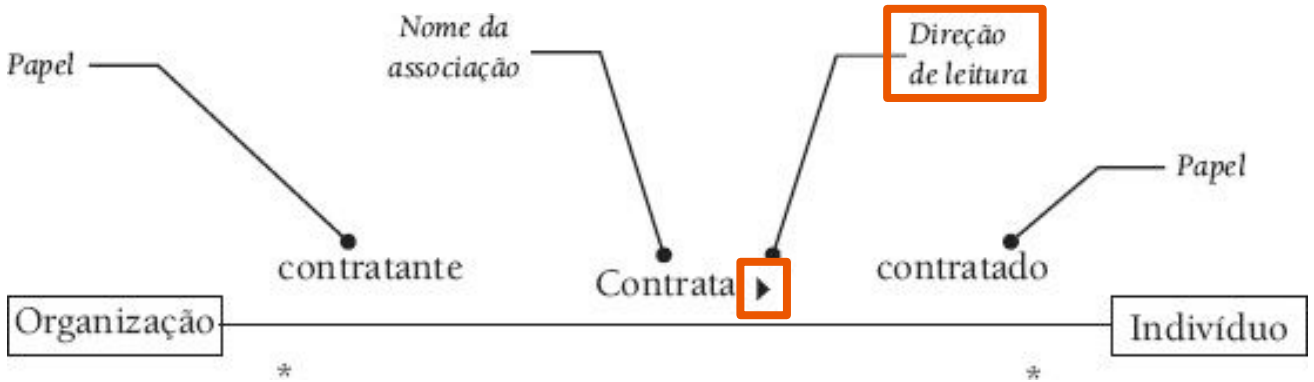
Nome de associação, direção de leitura e papéis

O **nome da associação** é posicionado na linha da associação, a meio caminho das classes envolvidas.



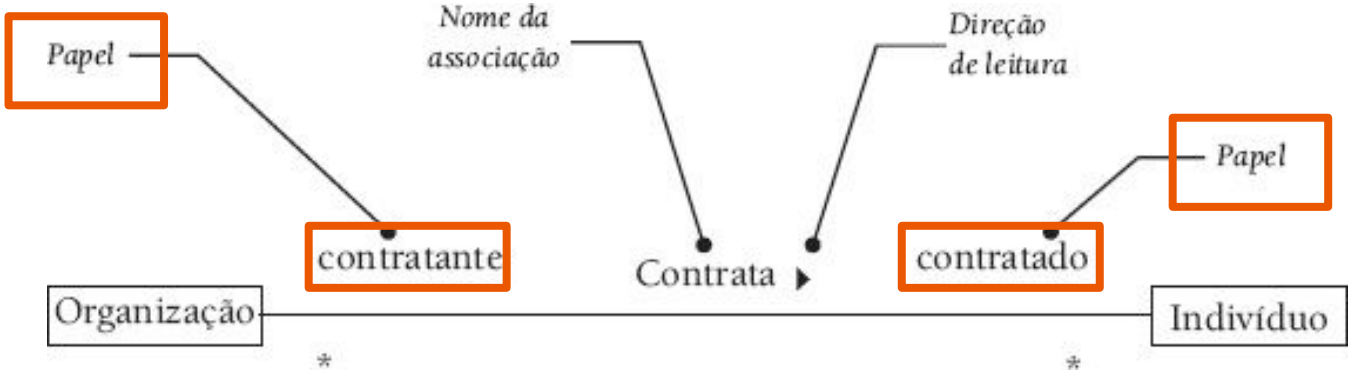
Nome de associação, direção de leitura e papéis

A **direção** de leitura indica como a associação deve ser lida. Essa direção é representada por um **pequeno triângulo** posicionado próximo a um dos lados do nome da associação.



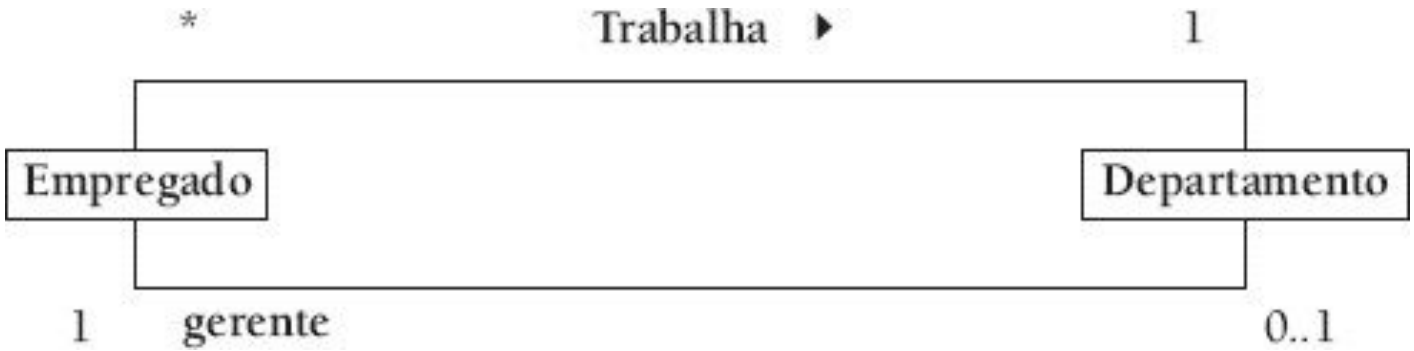
Nome de associação, direção de leitura e papéis

Uma característica complementar à utilização de nomes e de direções de leitura é a indicação de **papéis** (*roles*) para cada uma das classes participantes em uma associação.



Nome de associação, direção de leitura e papéis

Exemplo, considere duas classes: **Empregado** e **Departamento**. Considerando, ainda, que um departamento precisa saber quais são os seus empregados e quem é o seu gerente.



Nome de associação, direção de leitura e papéis

O nome da associação deve ser simples e exprimir o significado da mesma. É preferível não nomear associações com usar nomes vagos ou óbvios demais.

O mesmo vale para os papéis: em situações em que o significado da associação for intuitivo, a utilização de papéis só serve para “carregar” o diagrama.

O ponto é tentar equilibrar clareza e concisão.

Classes associativas

Classes associativas são classes que estão ligadas a associações, em vez de estarem ligadas a outras classes. São também chamadas de classes de associação.

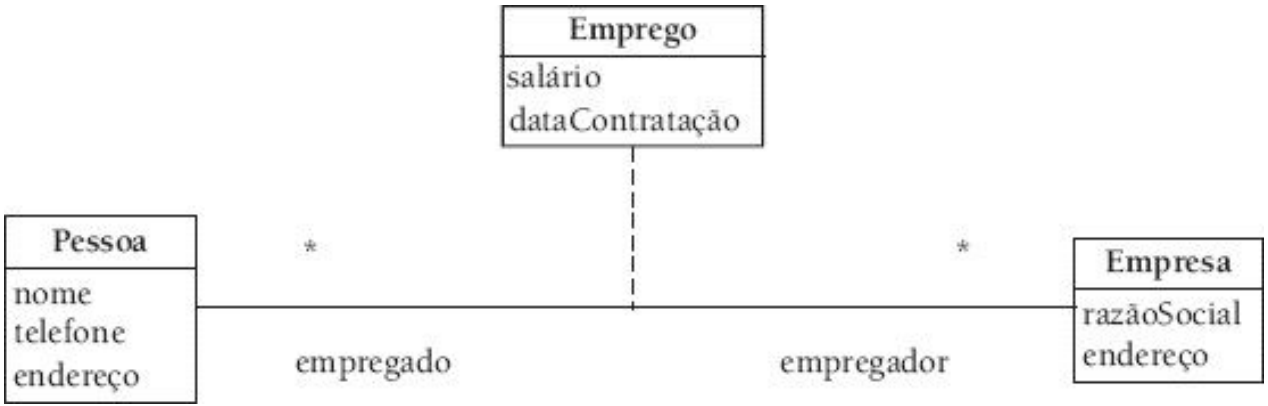
Esse tipo de classe normalmente aparece quando duas ou mais classes estão associadas e é necessário manter informações dessa associação.

Classes associativas

Embora seja mais comum encontrar classes associativas ligadas a **associações de conectividade muitos para muitos**, uma classe associativa pode estar ligada a associações de qualquer conectividade.

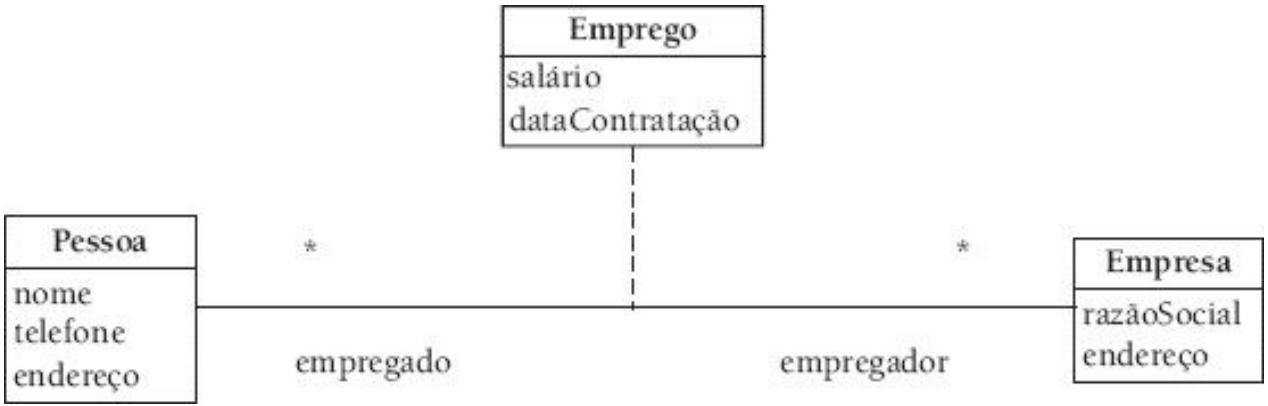
Classes associativas

Na UML, uma classe associativa é representada pela mesma notação utilizada para uma classe comum. A diferença é que esta classe é ligada por uma linha tracejada a uma associação.



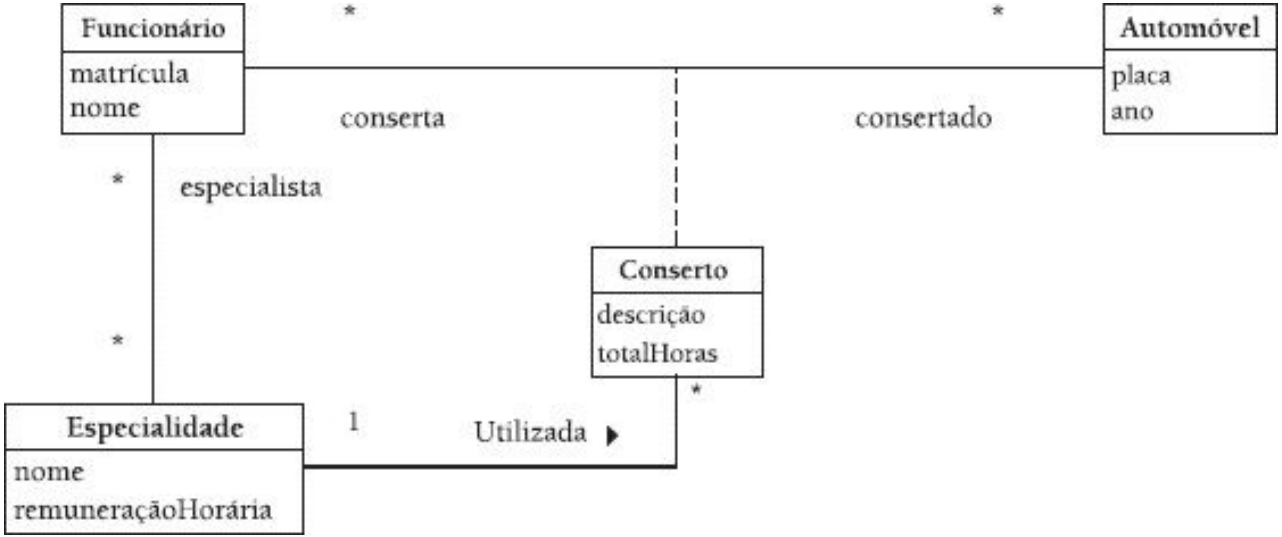
Classes associativas

Como dica de modelagem, **não se deve nomear a linha da associação** de uma classe associativa.



Classes associativas

Uma classe associativa pode participar de outros relacionamentos



Classes associativas

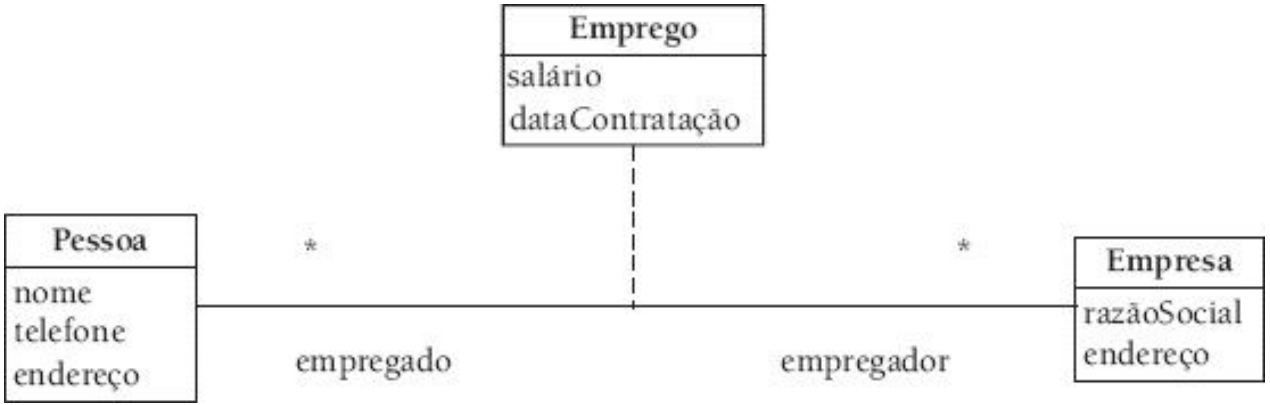
Pelos exemplos anteriores, pode-se notar que uma classe associativa é um elemento **híbrido**: tem **características de uma classe**, mas t**ambém de uma associação**.

Classes associativas

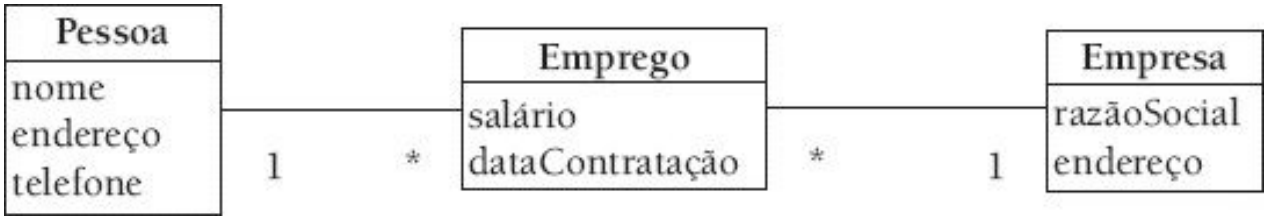
Um diagrama de classes que contém uma classe associativa pode ser modificado para retirá-la, sem perda de informação no modelo em questão. Isso pode ser feito em dois passos:

1. eliminação da associação correspondente à classe associativa;
2. criação de associações diretas desta última com as classes que antes eram conectadas pela associação eliminada no passo

Classes associativas



Classes associativas

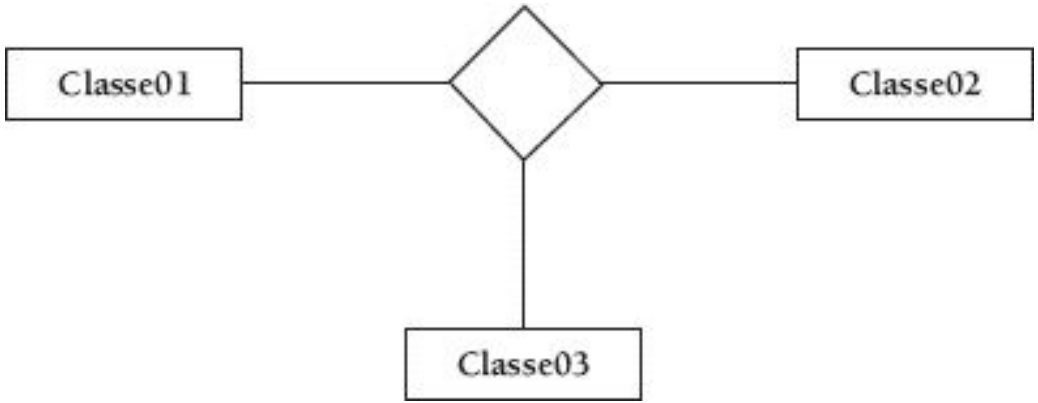


Associações ternárias

Define-se o **grau** de uma associação como a q**uantidade de classes envolvidas na mesma**. Na maioria dos casos práticos de modelagem, as associações normalmente são binárias, ou seja, representam a ligação entre objetos de duas classes (tem grau igual a dois).

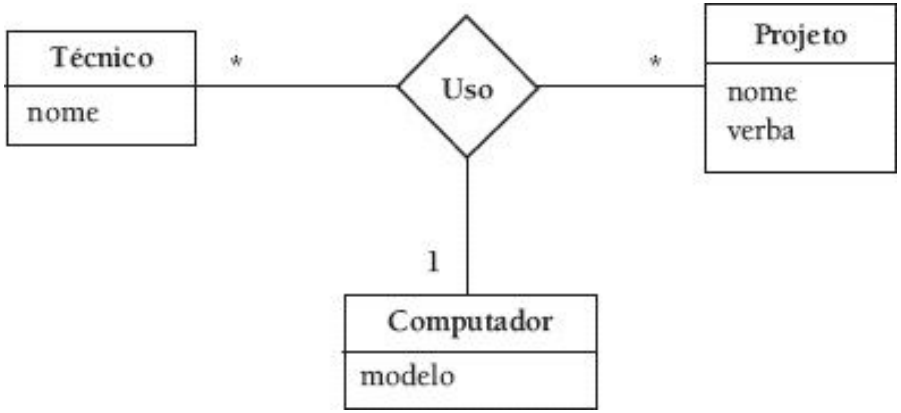
Associações ternárias

Quando o **grau** de uma associação é **igual a três**, dizemos que ela é **ternária**. Associações ternárias são necessárias quando é preciso associar **três objetos distintos.**



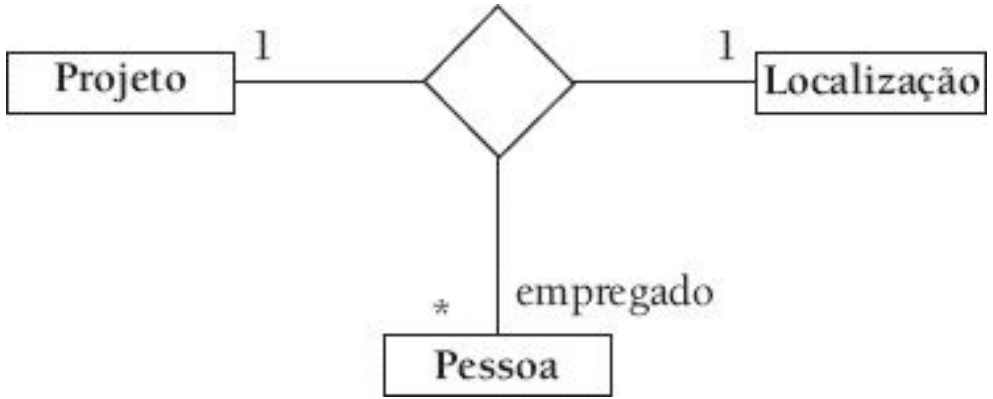
Associações ternárias

Um exemplo de associação ternária: Um **técnico** utiliza exatamente um **computador** para cada **projeto** em que trabalha. Cada computador pertence a um técnico para cada projeto.



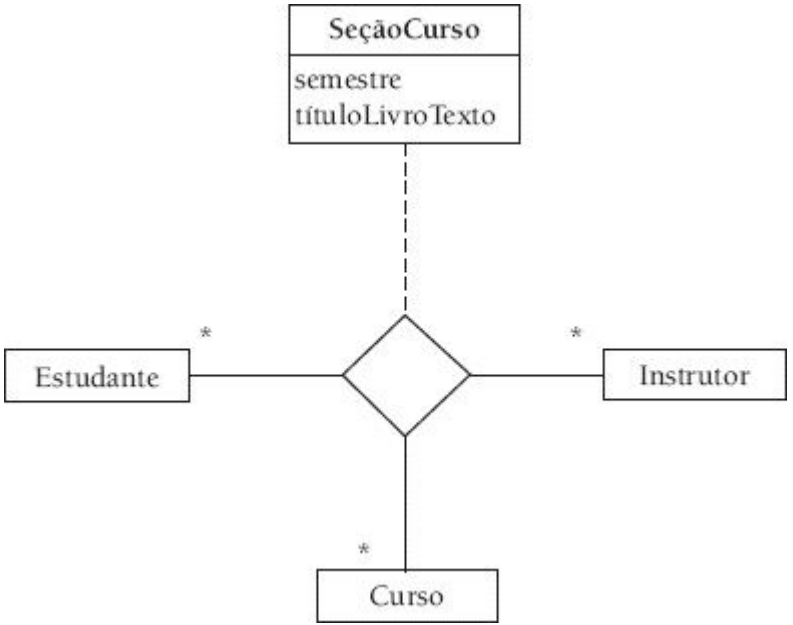
Associações ternárias

Outro exemplo de associação ternária. Desta vez, vê-se que cada **empregado** associado a um **projeto** trabalha exclusivamente em uma **localização**, mas pode estar em diferentes localizações em projetos diferentes.



Associações ternárias

Classe associativa e de **associação ternária** podem ser misturados no conceito de classe associativa ternária, no qual existe uma classe associativa ligada a uma associação ternária.

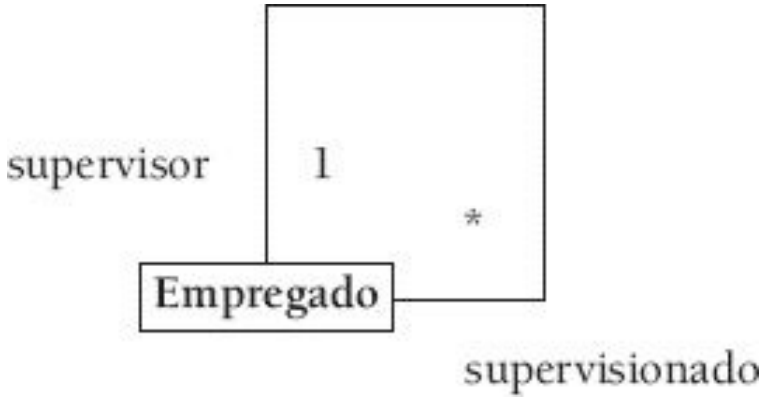


Associações reflexivas

Uma associação reflexiva (também denominada **autoassociação**) liga objetos da mesma classe. Cada objeto tem um papel distinto nessa associação.

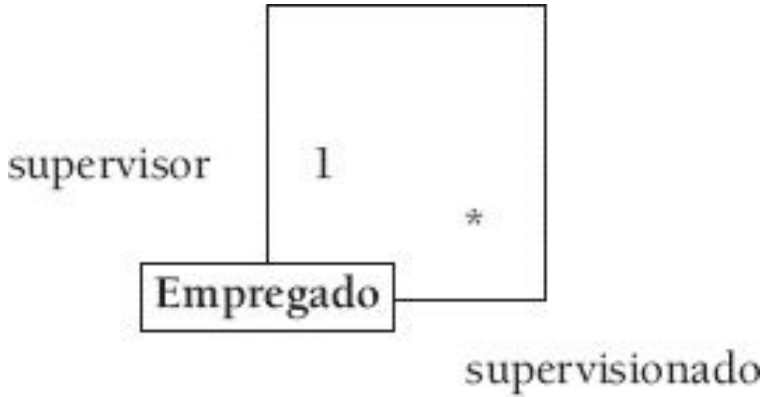
Associações reflexivas

Exemplo uma associação reflexiva entre objetos de Empregado. Nesse exemplo de uso de autoassociação, em que há objetos que assumem o papel de supervisor e outros que tomam para si o papel de supervisionado.



Associações reflexivas

Uma **associação reflexiva** não indica que um objeto se associa a ele próprio. Em vez disso, uma autoassociação indica que um objeto de uma classe se associa com outros objetos da mesma classe.



Agregação e composição

À toda associação, podemos atrelar uma **semântica**. A semântica de uma associação corresponde ao seu significado, ou seja, à natureza conceitual da relação que existe entre os objetos que participam daquela associação.

Agregação e composição

De todos os significados diferentes que uma associação pode ter, há uma categoria especial de significados que representa relações **todo-parte**.

Esse tipo de relação entre dois objetos indica que um deles **está contido** no outro. Podemos também dizer que um objeto **contém o outro**.

A UML define dois tipos de relacionamentos todo-parte, a **agregação** e a **composição**.

Agregação e composição

Características particulares das **agregações** e **composições** que as diferem das associações simples:

- Agregações/composições são assimétricas, no sentido de que, se um objeto A é parte de um objeto B, o objeto B não pode ser parte do objeto A.

Agregação e composição

Características particulares das **agregações** e **composições** que as diferem das associações simples:

- Agregações/composições propagam comportamento, de forma que um comportamento que se aplica a um todo automaticamente se aplica também às suas partes.

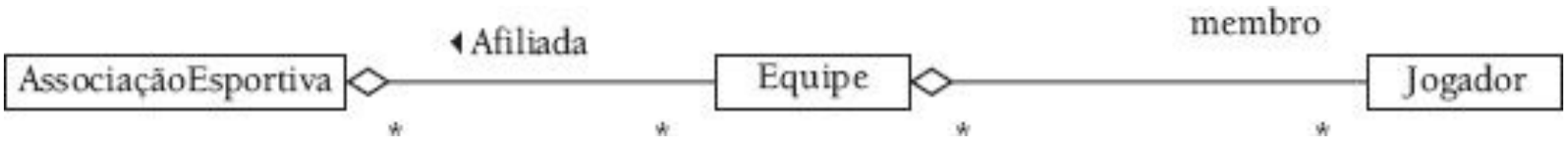
Agregação e composição

Características particulares das **agregações** e **composições** que as diferem das associações simples:

- Nas agregações/composições, as partes são normalmente criadas e destruídas pelo todo. Naclasse do objeto todo, são definidas operações para adicionar e remover as partes.

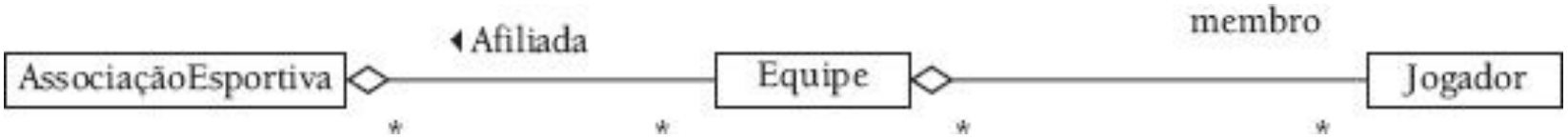
Agregação e composição

Uma **agregação** é representada como uma linha que conecta as classes relacionadas, com um **diamante (losango) branco** perto da classe que representa o todo



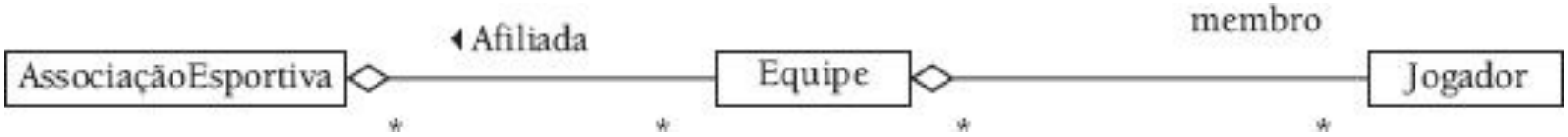
Agregação e composição

Esse diagrama indica que uma **associação esportiva** é formada por diversas **equipes**. Cada uma delas é formada por **diversos jogadores**. Por outro lado, um jogador pode fazer parte de diversas equipes.



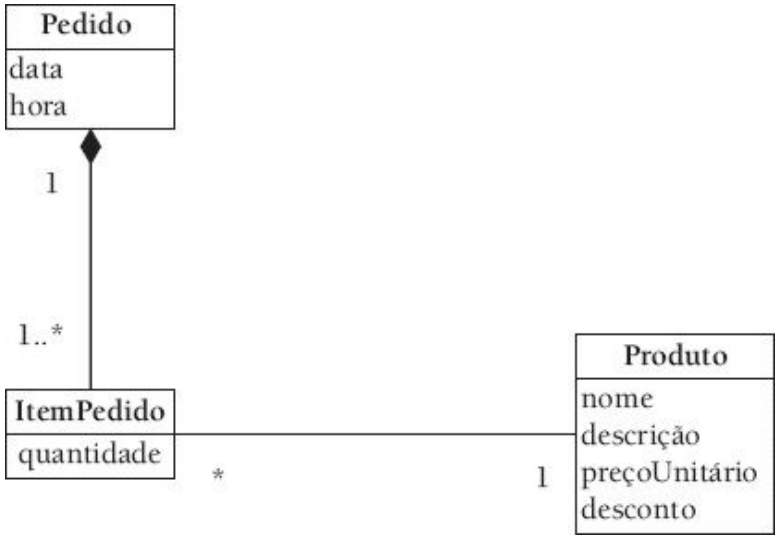
Agregação e composição

Na **agregação**, a **destruição** de um objeto todo **não implica necessariamente** a destruição do objeto parte. Por exemplo. Se uma das equipes das quais um jogador é membro for extinta por algum motivo, ele ainda poderá continuar membro de outras equipes.



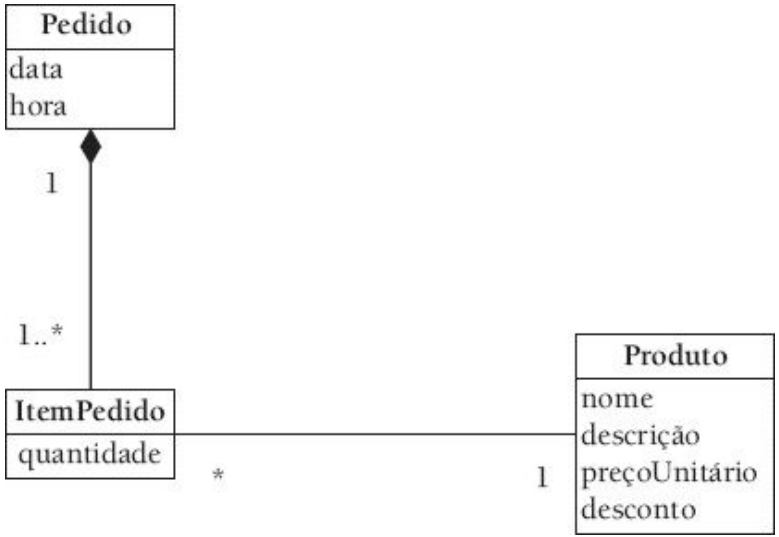
Agregação e composição

Uma **composição** é representada na UML por meio de um **diamante negro**, para contrapor com o diamante branco da agregação.



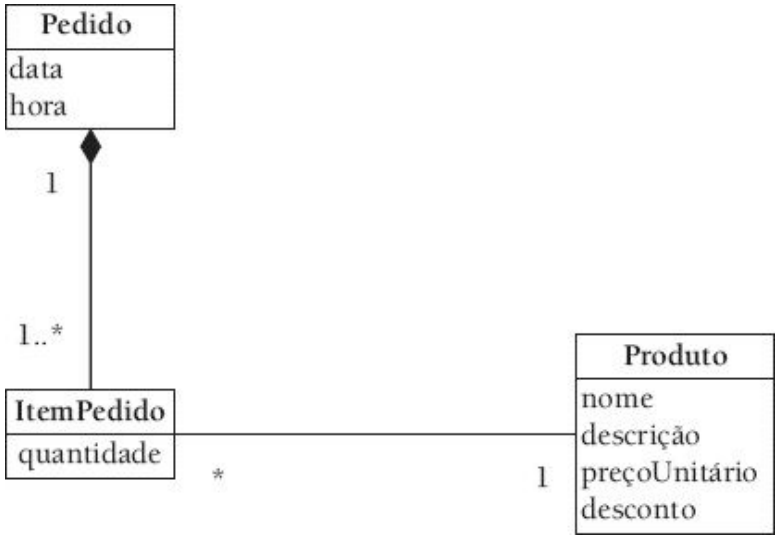
Agregação e composição

Exemplo de **composição,** considere os itens de um pedido de compra. É comum esse tipo de pedido incluir vários itens. Cada item diz respeito a um produto faturado. Os itens têm identidade própria (é possível distinguir um item de outro no mesmo pedido).



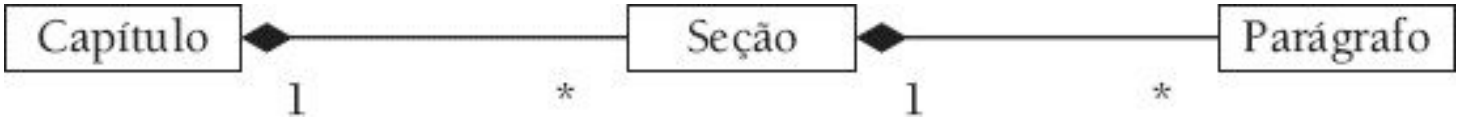
Agregação e composição

Nessa situação, os itens **não têm existência independente** do pedido ao qual estão conectados. Quando o pedido deixa de existir, o mesmo acontece com os seus itens.



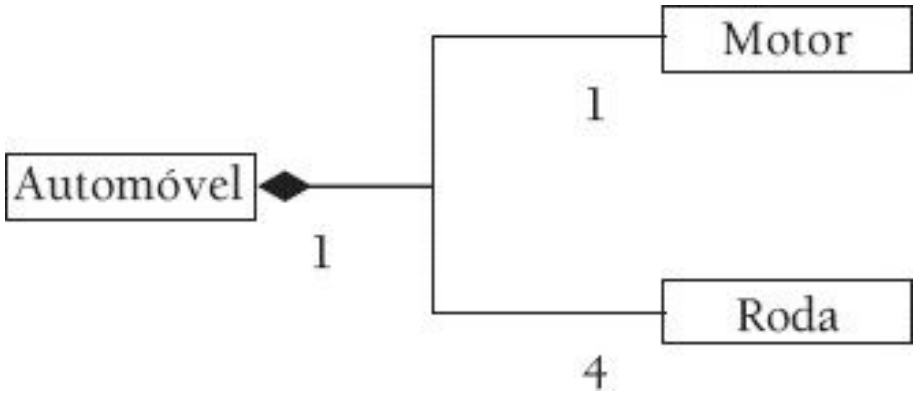
Agregação e composição

Na composição, os objetos parte pertencem a um único todo. Por essa razão, a composição é também denominada agregação não-compartilhada.



Agregação e composição

Na composição, os objetos parte pertencem a um único todo. Por essa razão, a composição é também denominada agregação não-compartilhada.



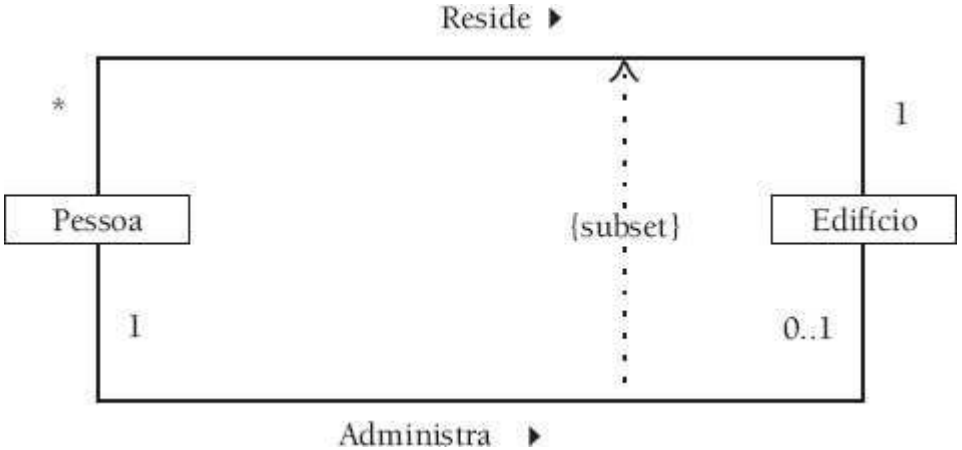
Restrições sobre associações

Restrições podem ser adicionadas sobre uma associação para adicionar mais semântica a ela.

Duas das restrições sobre associações predefinidas pela UML são **subset** (subconjunto) e **xor** (ou exclusivo).

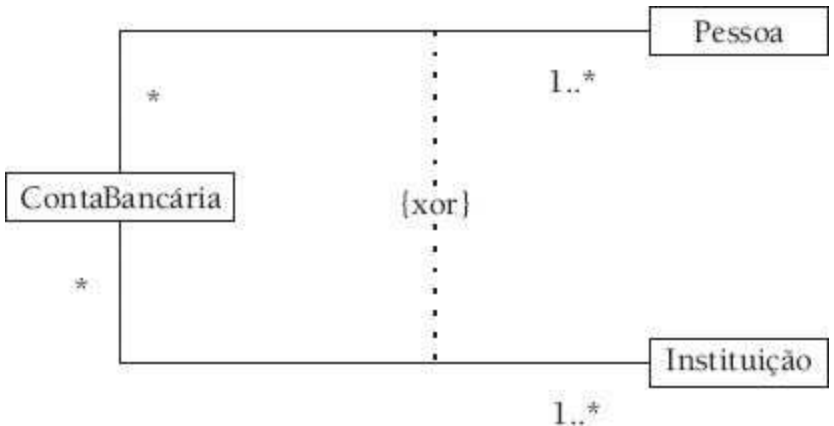
Restrições sobre associações

A restrição **subset** indica que os objetos conectados por uma associação constituem um subconjunto dos objetos conectados através de uma outra associação



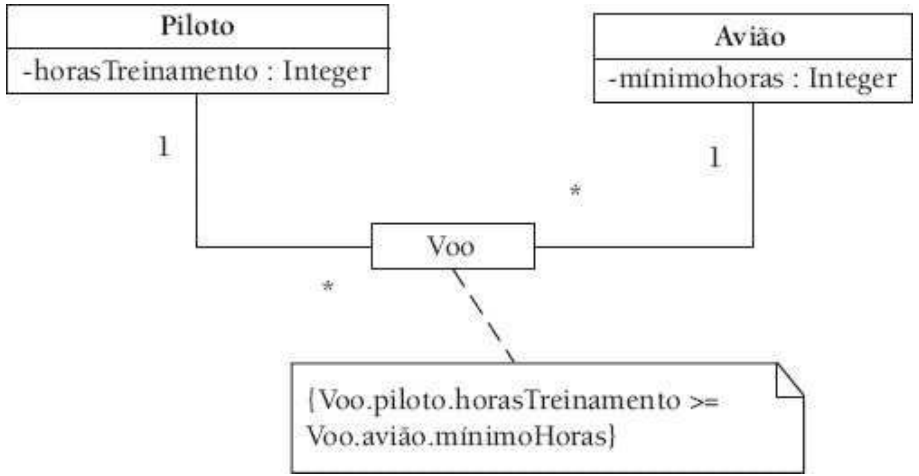
Restrições sobre associações

Na restrição **xor**, há duas ou mais classes ligadas pela linha pontilhada. Essas classes devem ter associações com uma classe em comum. Essa restrição significa que somente uma das associações envolvidas pode ocorrer entre os objetos.



Restrições sobre associações

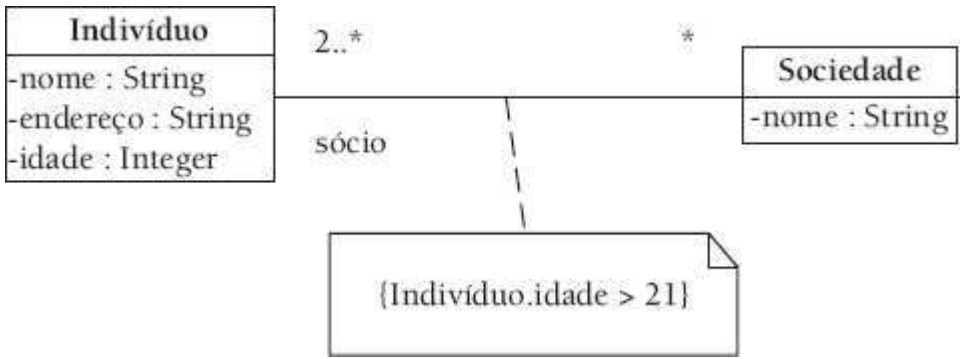
As restrições também podem ser definidas formalmente em OCL. Expressões nessa linguagem podem ser definidas utilizando-se uma expressão da forma Item.seletor, que permite o acesso às propriedades de uma classe (atributos, operações e associações)



Object Constraint Language - Linguagem de Restrição de Objetos

Restrições sobre associações

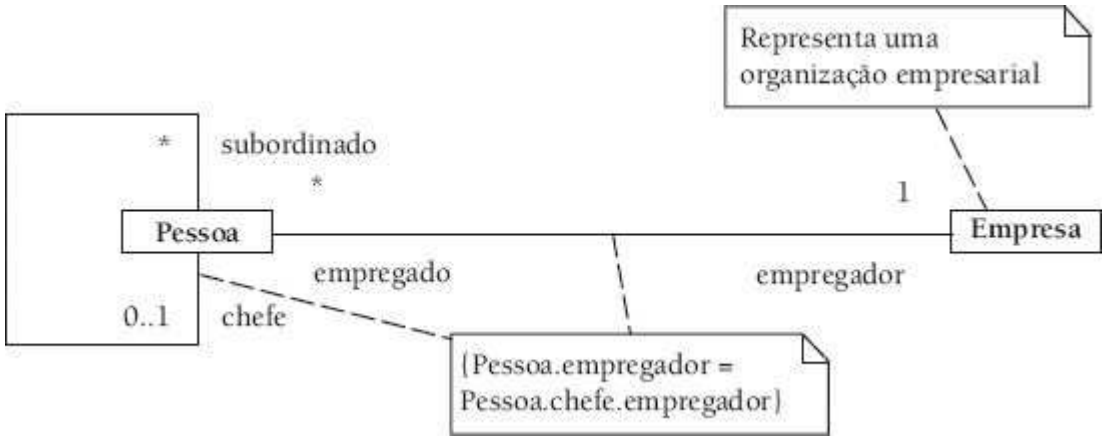
As restrições também podem ser definidas formalmente em OCL.



Object Constraint Language - Linguagem de Restrição de Objetos

Restrições sobre associações

As restrições também podem ser definidas formalmente em OCL.



Object Constraint Language - Linguagem de Restrição de Objetos

Generalização e especializações

O modelo de classe pode também representar relacionamentos de **generalidade** ou **especificidade** entre as classes envolvidas.

O relacionamento de herança é também chamado de relacionamento de **generalização/especialização**, ou simplesmente **gen/espec**.

Generalização e especializações

Generalização e a **especialização** são dois pontos de vista do mesmo relacionamento: dadas duas classes A e B, se A é uma generalização de B, então B é uma especialização de A.

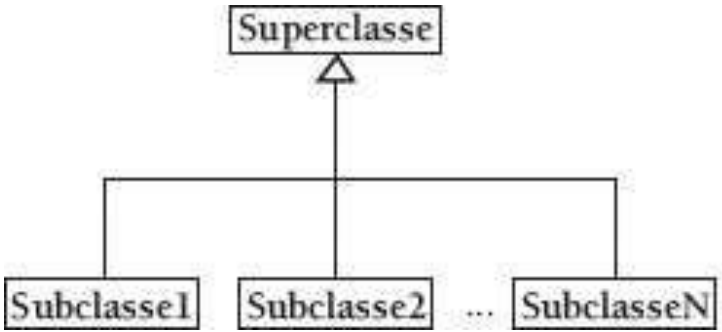
Generalização e especializações

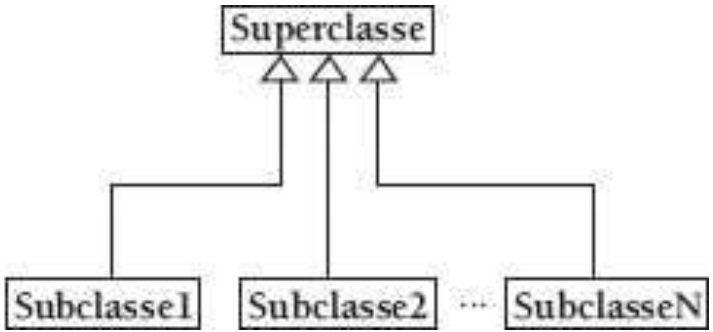
Termos utilizados para denotar relacionamento de herenças são bastante variados:

- **subclasse e superclasse;**
- **supertipo e subtipo;**
- **classe base e classe herdeira;**
- **ancestral e descendente.**

Generalização e especializações

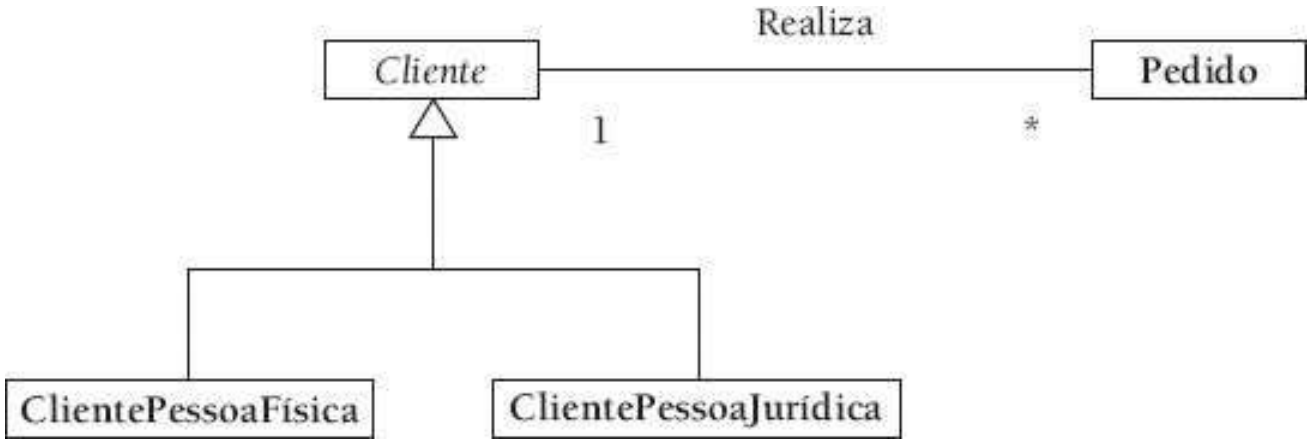
No diagrama de classes, a herança é representada na UML por uma flecha partindo da subclasse em direção à superclasse.





Generalização e especializações

No diagrama de classes, a herança é representada na UML por uma flecha partindo da subclasse em direção à superclasse.



Propriedades do relacionamento de herança

O relacionamento de herança possui duas propriedades importantes, **transitividade** e **assimetria**:

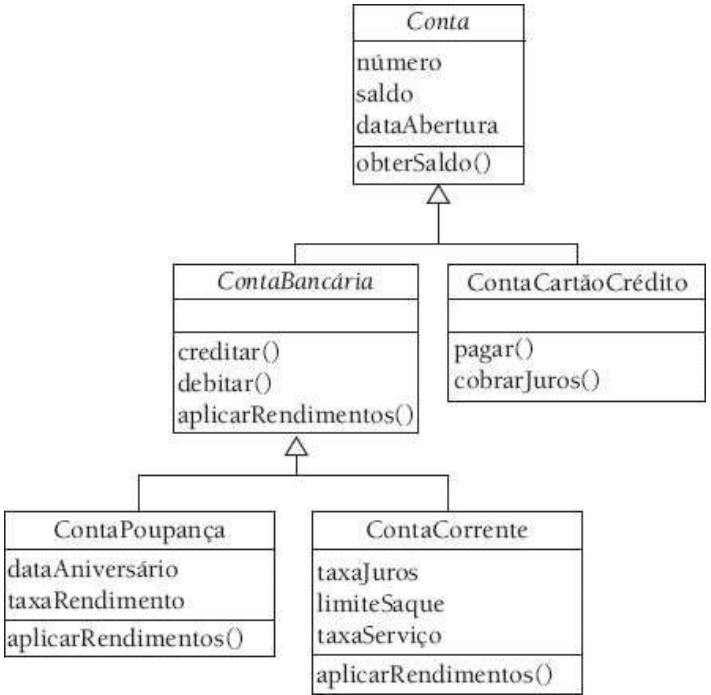
- **Transitividade** indica que uma classe em uma hierarquia herda tanto propriedades e relacionamentos de sua superclasse imediata quanto de suas não imediatas (classes em um nível mais alto da hierarquia).

Propriedades do relacionamento de herança

O relacionamento de herança possui duas propriedades importantes, **transitividade** e **assimetria**:

- **Assimetria** essa propriedade significa que dadas duas classes A e B, se A for uma generalização de B, então B não pode ser uma generalização de A. Ou seja, não pode haver ciclos em uma hierarquia de herança.

Propriedades do relacionamento de herança



Refinando o modelo de classes com gen/espec

A ideia básica é identificar abstrações mais genéricas ou mais específicas que outras no diagrama de classes do sistema. Essa identificação de abstrações corresponde a refinamentos no modelo de classes que podem ser obtidos a partir de duas estratégias alternativas e complementares: a **generalização** e a **especialização**.

Refinando o modelo de classes com gen/espec

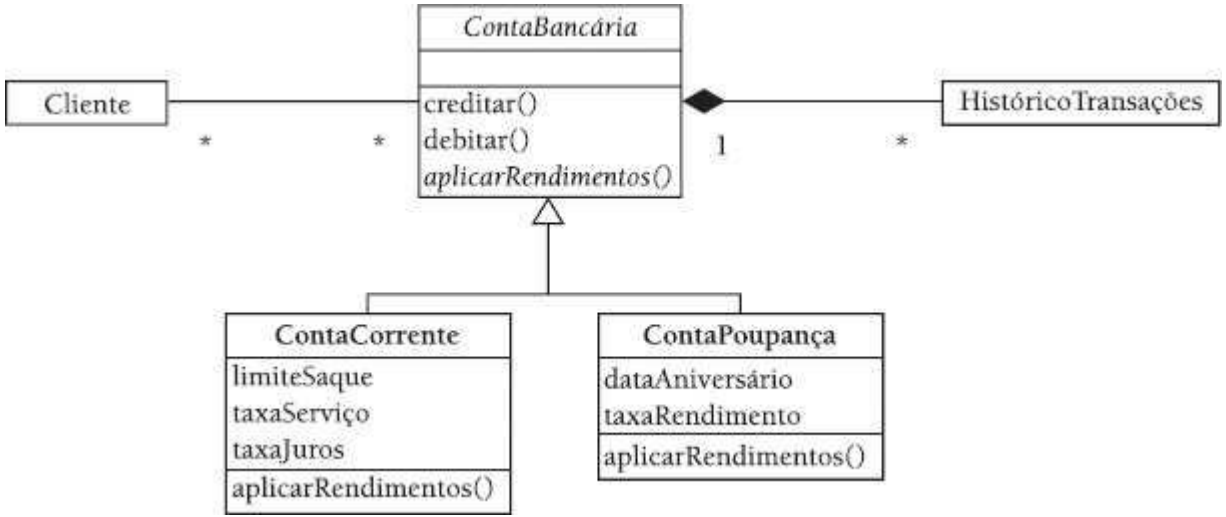
Pode ser que duas ou mais classes semelhantes tenham sido identificadas. Neste caso, talvez seja adequado criar uma **generalização**, ou seja, uma classe mais genérica, e definir as classes anteriores como subclasses desta última.

Pode ser que a superclasse não gere instâncias próprias e esteja sendo utilizada somente para organizar classes semelhantes em uma hierarquia (classe **abstrata**)

Refinando o modelo de classes com gen/espec

Em segundo lugar, também é possível aplicar a especialização, que corresponde ao processo de criar classes mais específicas a partir de uma classe preexistente.

Refinando o modelo de classes com gen/espec



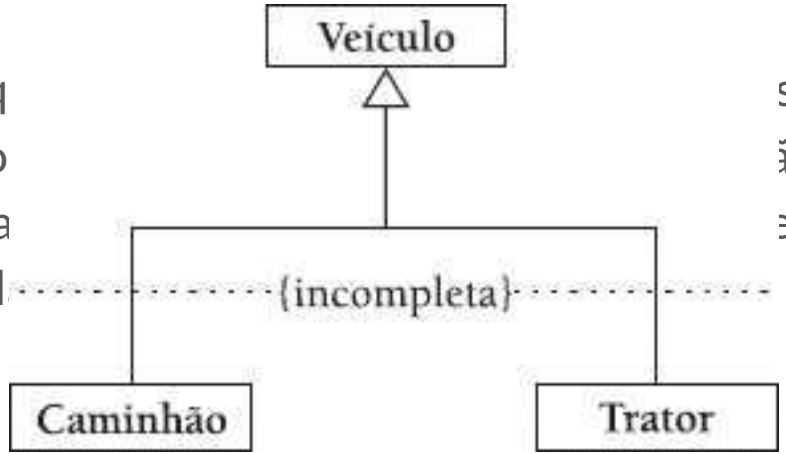
Refinando o modelo de classes com gen/espec

Definição de restrições sobre gen/espec

UML permite que determinadas restrições sejam associadas a elementos de um modelo. As restrições sobre gen/espec são representadas no diagrama de classes, próximas à linha do relacionamento. Essas restrições são apresentadas entre chaves.

Definição de restrições sobre gen/espec

UML permite q sociadas a elementos de um modelo ão representadas no diagrama de cla ento. Essas restrições são apresentad



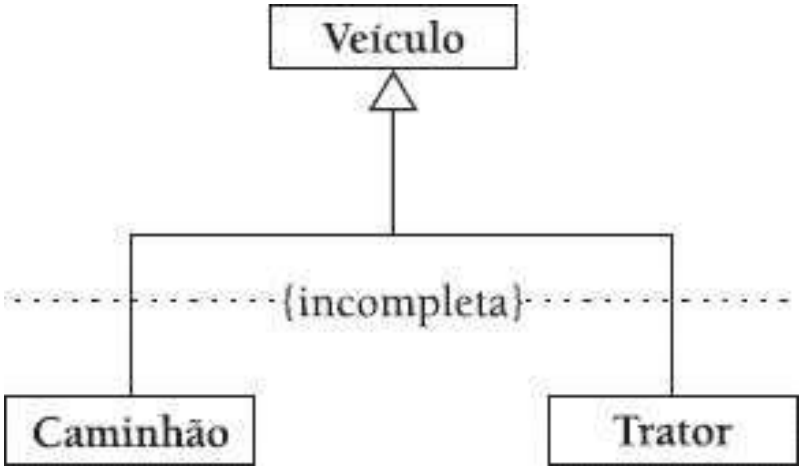
Definição de restrições sobre gen/espec

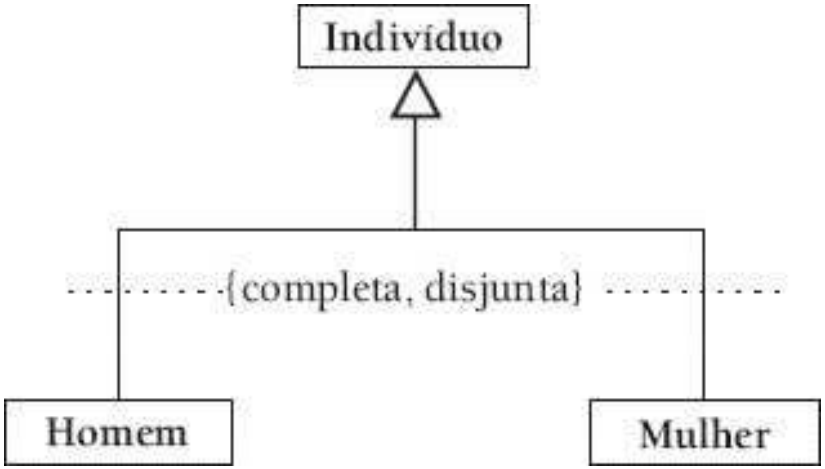
Estas restrições podem ser de quatro tipos: sobreposta, disjunta, completa, incompleta.

Definição de restrições sobre gen/espec

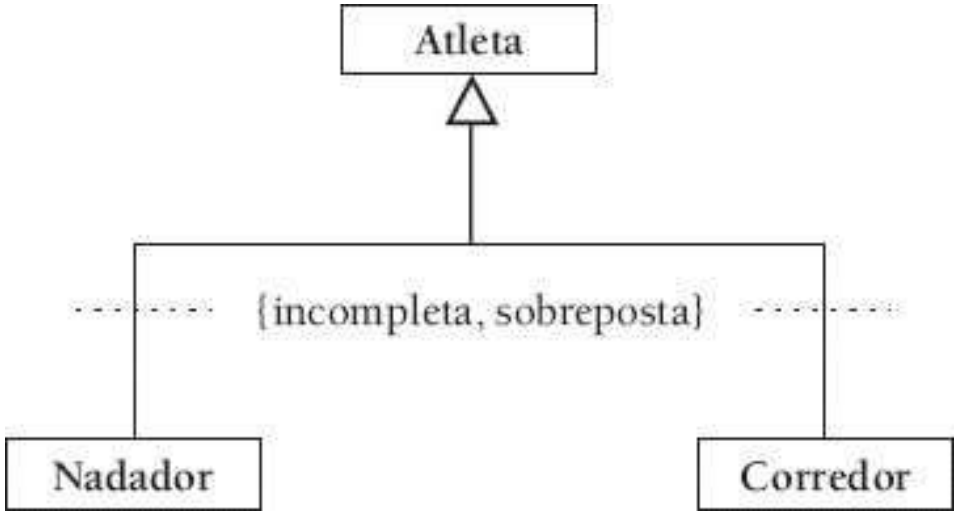
Restrição	Significado
sobreposta	Posteriormente podem ser criadas subclasses que herdem de mais de uma subclasse (herança múltipla).
disjunta	Quaisquer subclasses criadas posteriormente poderão herdar de somente uma subclasse.
completa	Todas as subclasses possíveis foram enumeradas na hierarquia.
incompleta	Nem todas as subclasses foram enumeradas na hierarquia.

Definição de restrições sobre gen/espec





Definição de restrições sobre gen/espec



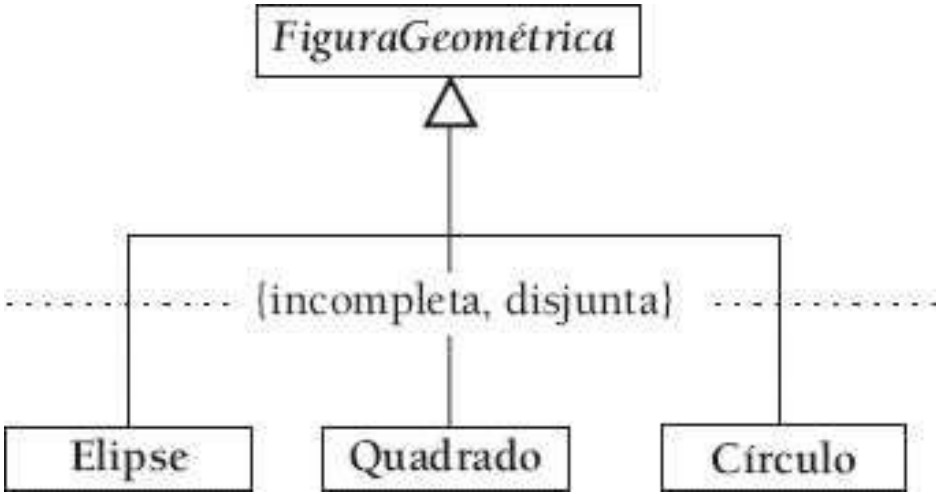


Diagrama de objetos

Fazer exercício - Juca está fazendo compras online para volta às aulas.

Bibliografia

BEZERRA, E. Princípios de Análise e Projeto de Sistemas com UML. 2. ed. Rio de Janeiro: Elsevier, 2006.

BOOCH, G.; RUMBAUGH, J.; JACOBSON, I. UML: guia do usuário. 2. ed. Rio de Janeiro: Campus, 2006.

PRESSMAN, R. S. Engenharia de software. Rio de Janeiro: McGraw-Hill, 2006.

SOMMERVILLE, I. Engenharia de software. São Paulo: Addison Wesley, 2007.