Unidade 5: Modelagem de classes de análise (continuação)

3.2.7 Agregações e composições

Conceito: Toda associação tem uma semântica atrelada. **A semântica de uma associação** corresponde a seu significado, ou seja, à natureza conceitual da relação que existe entre os objetos que participam da associação. Ao darmos um nome a uma associação, tentamos deixar clara a semântica da mesma.

Conceito: Dentre todos os significados que uma associação pode ter, há um especial que representa relações do tipo todo-parte. **Uma relação todo-parte entre dois objetos** indica que um dos objetos **está contido** no outro. Também podemos dizer que um dos objetos **contém** o outro.

Conceitos: A UML define dois subtipos de relacionamentos todo-parte: a agregação e a composição. Tanto a **agregação** como a **composição** são casos especiais de associação. Portanto, todas as características válidas para a associação (multiplicidades, participações, papéis,...) valem para as primeiras. Além disso, sempre que for possível utilizar uma agregação ou uma composição, uma associação também poderá ser utilizada.

Algumas características das agregações são listadas a seguir.

- Agregações e 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;
- Agregações e composições **propagam comportamento**, pois um comportamento que se aplica ao todo certamente se aplica às partes;
- Nas agregações e composições **as partes são normalmente criadas e destruídas** pelo todo. Na classe do objeto todo, são definidas operações para adicionar e remover os objetos parte.

Regra para verificar a pertinência da aplicação de um relacionamento todo-parte (agregação ou composição):

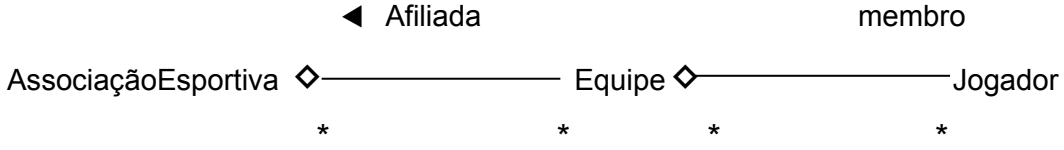
Sejam duas classes X e Y. Se uma das perguntas a seguir for respondida de maneira afirmativa, provavelmente haverá um relacionamento todo-parte envolvendo X e Y, no qual X é o todo e Y é a parte. (Se as respostas forem negativas, será melhor utilizar uma associação comum.)

"*X tem um ou mais Ys?*"

"*Y é parte de Y?*"

Notação gráfica para agregações:

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

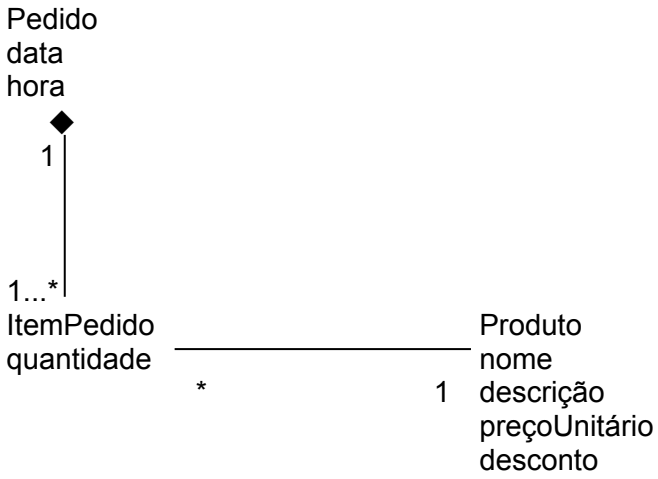


Este diagrama indica que uma associação esportiva é formada por diversas equipes e que cada equipe é formada por diversos jogadores. Por outro lado, um jogador pode fazer parte de várias equipes.

Notação gráfica para composições:

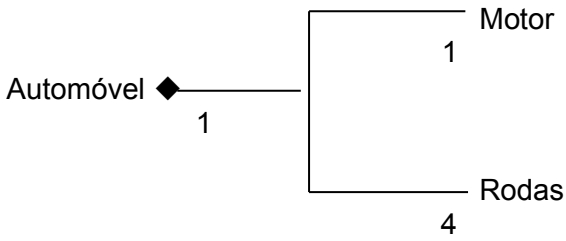
Uma composição é representada pela mesma linha mas com um losango negro do lado da classe todo.

Exemplo 1 de composição:



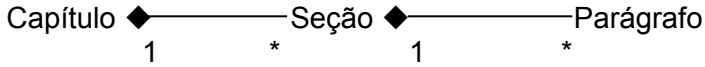
Este diagrama representa uma composição entre Pedido e ItemPedido

Exemplo 2 de composição:



Conceitos: Uma agregação pode se estender por diversos níveis, em **hierarquias de agregações**. De forma semelhante, pode-se falar de **hierarquia de composições**.

Exemplo de hierarquia de composição:



As diferenças entre a agregação e a composição não são muito bem definidas. Contudo, há duas **características distintivas** entre elas:

1. **Na agregação**, a destruição de um objeto todo não implica necessariamente a destruição do objeto parte. *Exemplo*: No contexto das associações esportivas do exemplo, se uma das equipes onde um jogador atua for extinta, o jogador pode ainda continuar atuando nas outras equipes. Por outro lado, *no exemplo* de pedidos de produtos, os itens pedidos não têm existência independente do pedido associado.
2. **Na composição**, os objetos parte pertencem a um único objeto todo. Por este motivo, a composição pode ser também denominada "**agregação não-compartilhada**". Por outro lado, numa agregação, pode ser que um mesmo objeto participe como componente de

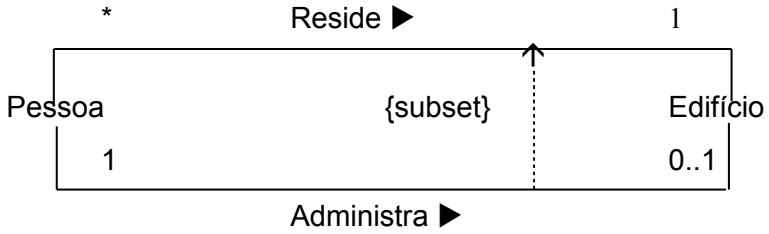
vários objetos todo. A agregação pode ser chamada, então, de "**agregação compartilhada**".

3.2.8 Restrições sobre associações

Restrições podem ser adicionadas às associações para aumentar a semântica. Duas das restrições predefinidas pela UML são subset (subconjunto) e xor (ou exclusivo). Ambas são representadas por uma linha pontilhada que liga duas ou mais associações.

Conceito: A restrição **subset** indica que os objetos conectados por uma associação constituem um subconjunto dos objetos conectados por outra associação.

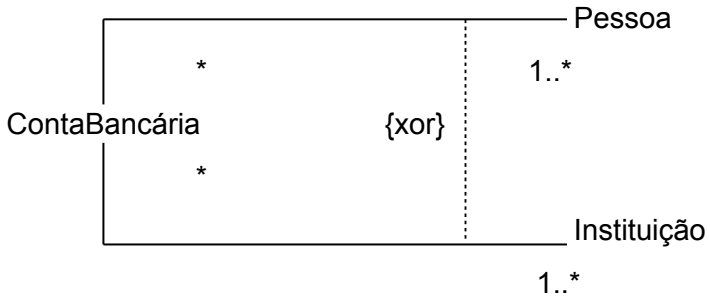
Exemplo:



Este diagrama indica que o conjunto de pessoas que administram um edifício é um subconjunto do conjunto de pessoas que residem nele. (a seta parte aponta para o conjunto maior.)

Conceito: A restrição **xor** indica que somente uma das associações entre envolvidas pode ocorrer entre os objetos das classes.

Exemplo:

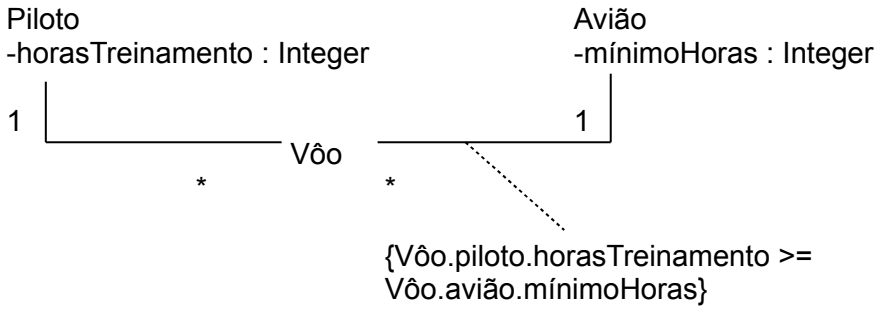


O diagrama representa a restrição de uma conta bancária: ou ela é de pessoa física ou de pessoa jurídica.

Restrições também podem ser definidas formalmente por meio de expressões da OCL da forma **Item.selector**, que permite ter acesso às propriedades (atributos, operações e associações) de uma classe.

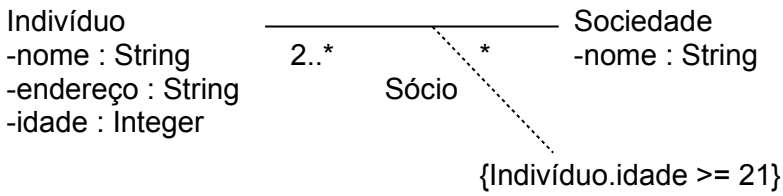
Estas expressões podem ser combinadas para formar expressões mais complexas. Para isso, recorre-se aos operadores aritméticos (+, -, *, /) e lógicos (>, =, <, < >).

Exemplo:



Esta restrição indica que o número de horas de vôo de um piloto deve ser maior ou igual ao número de horas exigidas para o avião ser pilotado. (**Semântica X representação**!)

Exemplo:



Neste exemplo, a restrição representa que o indivíduo deve ter pelo menos 21 anos para participar de uma sociedade.

3.3 Generalizações e especializações

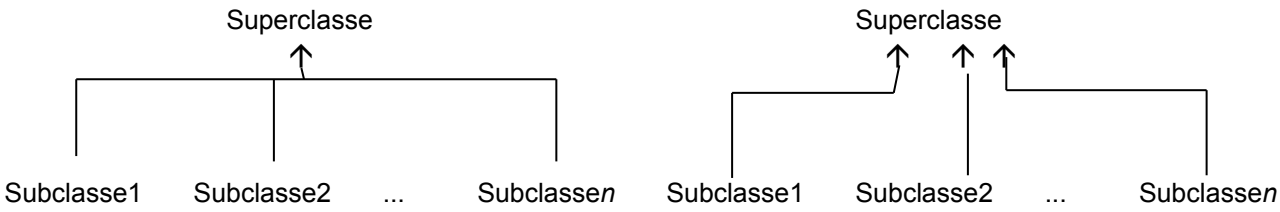
Além de relacionamentos entre objetos, o modelo de classes também aceita relacionamentos entre classes.

Conceito: O relacionamento de **herança entre classes** é também chamado de **generalização/especialização**, ou **gen/espec.** Isto porque ambos relacionamentos são parte da mesma propriedade, vista de duas óticas complementares. (Se A é uma generalização de B, B é uma especialização de A.)

Também são usados, associados a esta propriedade entre classes, os termos **subclasse** (para a classe "filha") e **superclasse** (para a classe "pai"), e, respectivamente, **subtipo** e **supertipo**.

Outros termos usados em linguagens OO são **classe base** (para a classe pai) e **classe herdeira**, assim como **ancestral** e **descendente**, para fazer alusão ao relacionamento geral em vários níveis.

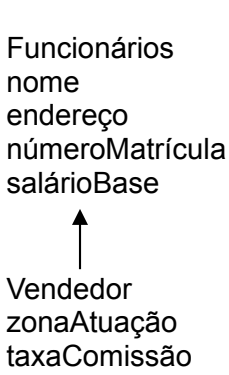
Notação:



Para compreender o conceito de herança é necessário lembrar que uma classe determina um conjunto de objetos que partilham um conjunto de propriedades (atributos, operações,

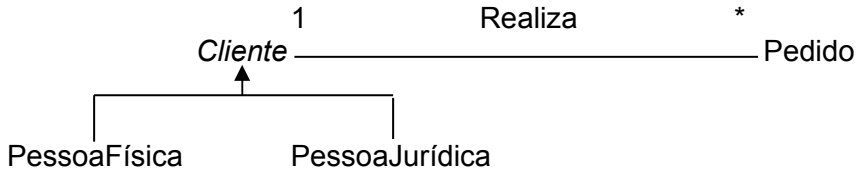
associações). No entanto, alguns objetos, muito semelhantes a outros, podem possuir propriedades que os outros não possuem. Eles formam, assim, um tipo especial dos outros (uma especialização ou subclasse).

Exemplo: Todos os funcionários de uma empresa possuem atributos comuns: nome, endereço, número de matrícula, e salário base. Mas somente um tipo especial de funcionário, o dos vendedores, possui atributos específicos, como zona de atuação e taxa de comissão. Os vendedores têm todas as propriedades dos funcionários, mas eles possuem características próprias que os diferenciam em relação aos funcionários em geral, portanto, formam uma subclasse.



Observação: É importante notar que as subclasses herdam não somente os atributos e as operações, mas, também, as associações em que a superclasse participa. Assim, se há uma associação definida para a superclasse de uma classe, ela não necessita ser redefinida para a subclasse, pois ela é herdada juntamente com o restante das características.

Exemplo:



Observação: Por outro lado, se há uma associação que somente se refere a uma subclasse ela deve ser representada de forma limitada à mesma.

Conceito: **Classes abstratas** são definidas para um contêiner de definições e propriedades. Elas não agrupam objetos de forma direta. Elas são indicadas *em itálico*. O conceito de classe abstrata é particularmente relevante na modelagem das classes de projeto.

Observações: É importante notar a diferença entre o conceito de **herança**, que se refere a **classes**, e os conceitos de **associações** (associações propriamente ditas, agregações e composições), definidas **entre objetos** ou **instâncias de classes**. Isto equivale a dizer que objetos específicos de uma classe se associam a objetos específicos de outras classes.

3.3.1 Propriedades do relacionamento de herança

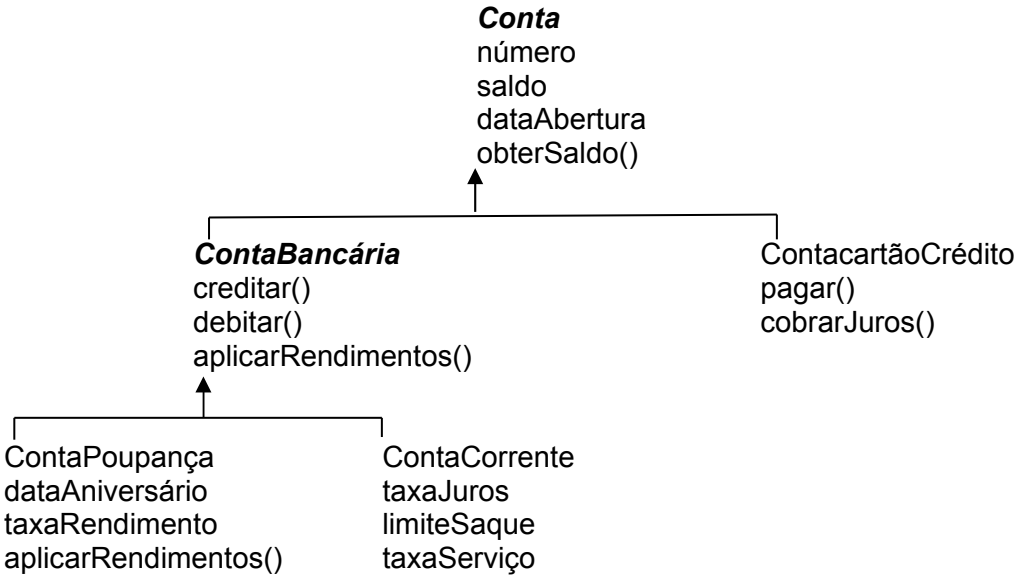
O relacionamento de herança tem duas propriedades: transitividade e assimetria.

Conceitos: A herança pode ser aplicada em vários níveis, criando uma **hierarquia de herança**.

A **transitividade** indica que uma classe em uma hierarquia herda tanto propriedades de sua

superclasse imediata como das superclasses indiretas.

Exemplo:



Conceito: A **assimetria** significa que não pode haver ciclos numa hierarquia de herança. Assim, dadas duas classes A e B, se A for uma generalização de B, então B não pode ser uma generalização de A.

3.3.2 Refinamento do modelo de classes via gen/espec

Conceitos: Os relacionamentos generalização e especificação permitem o **refinamento** do modelo de classes no sentido da sua consistência. Isto é feito a partir da identificação de **abstrações** mais genéricas ou mais específicas que outras no diagrama de classes, que são incluídas no diagrama por meio das estratégias de generalização e especialização.

Primeiramente, pode ser que duas ou mais classes semelhantes tenham sido identificadas. Neste caso, pode ser adequado criar uma **generalização**, que pode ser **abstrata** (não gerar objetos), servindo apenas para **organizar o modelo**.

Em segundo lugar, pode ser criada uma **especialização**, que consiste em criar classes mais específicas a partir de uma classe preexistente. Isto é feito normalmente ao se identificar propriedades de um subconjunto (um tipo ou uma categoria) dos objetos da classe, que não é passível de extensão à classe inteira.

Regra de substituição: Seja qual for a derivação aplicada ao modelo, existe uma regra que pode verificar a pertinência da substituição: Sejam duas classes A e B, onde A é uma generalização de B. Do ponto de vista dos clientes de A, não pode haver diferenças entre utilizar instâncias de B ou de A.

Corolário: O uso do relacionamento gen ou espec é inadequado sempre que nem todas as propriedades e comportamentos da superclasse façam sentido para as subclasses.

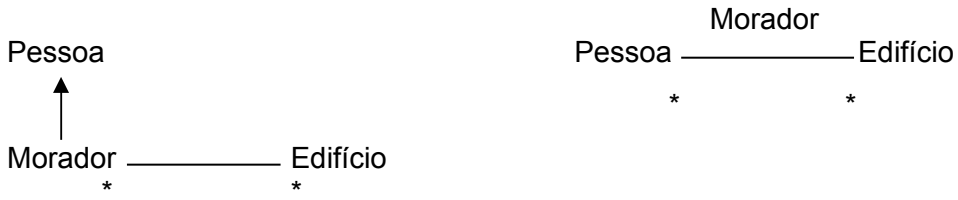
Outra regra para a aplicação ou não das derivações: Se a pergunta "*X é um tipo de Y?*" tiver resposta afirmativa, então Y é superclasse de X.

Observação: A partir da generalização e da especialização, classes mais gerais podem ser definidas com base em classes mais específicas e classes mais específicas pode ser definidas a partir de classes mais gerais. Entretanto, na aplicação da generalização ou da especialização

devem ser evitadas estruturas hierárquicas excessivamente profundas, pois elas dificultam a leitura do diagrama (**da sua semântica no mundo real**).

Outra dica: Subclasses não devem ser confundidas com papéis. O modelador deve evitar a criação de subclasses em situações que podem ser representadas por papéis.

Exemplo:



(solução inadequada) (solução adequada)

3.3.3 Restrições aos relacionamentos gen/espec

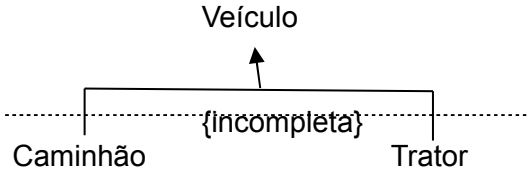
A UML define algumas restrições que podem ser aplicadas aos relacionamentos de generalização e especialização.

Restrição	**Significado**
Sobreposta	Possibilidade de criação posterior de mais subclasses que herdem de uma superclasse (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 explicitadas na hierarquia.

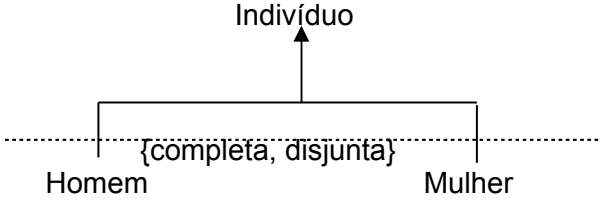
Observação: Cada um dos pares (sobreposta,disjunta) e (completa,incompleta) são mutuamente exclusivos.

Exemplos:

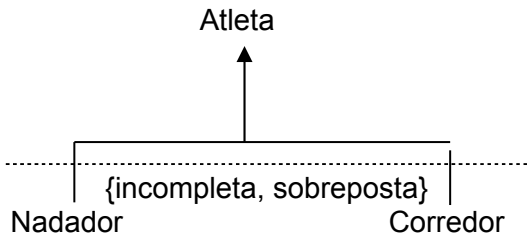
herança incompleta:



herança completa e disjunta:



herança incompleta sobreposta:



herança incompleta e disjunta: