MODELAGEM DE DADOS CONCEITUAL

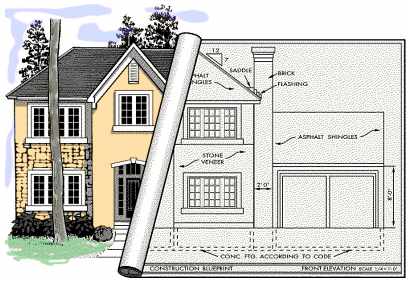
Prof. Ronaldo R. Goldschmidt

Introdução

- A construção de um banco de dados é um *processo* composto de algumas etapas:
  - Modelagem de Dados Conceitual
  - Projeto Lógico do BD
  - Projeto Físico do BD
  - Implementação
- A Modelagem de Dados Conceitual é a etapa responsável pela *identificação dos requisitos de informação* no negócio e pela *estruturação do banco de dados* em um nível lógico.

Diagrama de Classes da UML

- Utilizaremos o diagrama de classes da UML como ferramenta para desenhar o modelo de dados conceitual dos bancos de dados que desejamos projetar.

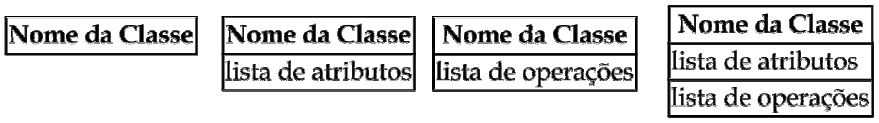


Classes

- Uma classe representa um grupo de objetos semelhantes.
- Uma classe descreve esses objetos através de ***atributos*** e ***operações***.
- Os **atributos** correspondem às informações que um objeto armazena.
- As **operações** correspondem às ações que um objeto sabe realizar.

Notação para uma classe

- Representada através de uma "caixa" com no máximo três compartimentos exibidos.
- Notação utilizada depende do nível de abstração desejado.



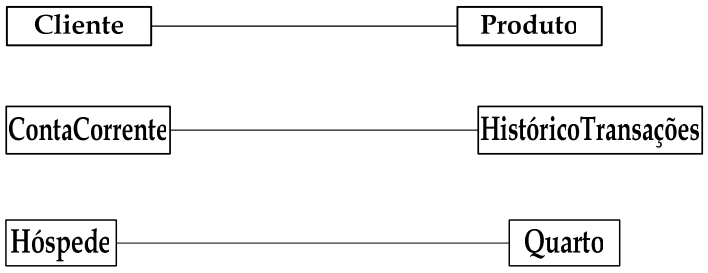
Exemplo (classe ContaBancária)

Associações

- Para representar o fato de que objetos podem se relacionar uns com os outros, utiliza-se a *associação*.
- Uma associação representa relacionamentos (ligações) que são formados entre objetos durante a execução do sistema.
  - embora as associações sejam representadas entre classes do diagrama, tais associações representam ligações possíveis entre *objetos* das classes envolvidas.

Associações - Notação

- Representada através de um segmento de reta ligando as classes cujos objetos se relacionam.
- Exemplos:



Multiplicidades

- Representam a informação dos limites inferior e superior da quantidade de objetos aos quais um outro objeto pode estar associado.
- Cada associação em um diagrama de classes possui duas multiplicidades, uma em cada extremo da linha de associação.

Multiplicidades

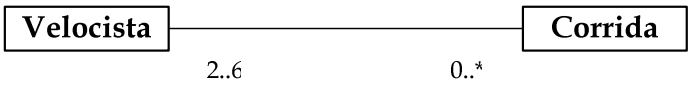
Nome	**Simbologia**
Apenas Um	1..1 (ou 1)
Zero ou Muitos	0..* (ou *)
Um ou Muitos	1..*
Zero ou Um	0..1
Intervalo Específico	$l_i..l_s$

Exemplo (multiplicidade)

- Pode haver um cliente que esteja associado a vários pedidos.
- Pode haver um cliente que não esteja associado a pedido algum.
- Um pedido está associado a um, e somente um, cliente.

Exemplo (multiplicidade)

- Uma corrida está associada a, no mínimo, dois velocistas
- Uma corrida está associada a, no máximo, seis velocistas.
- Um velocista *pode* estar associado a várias corridas.



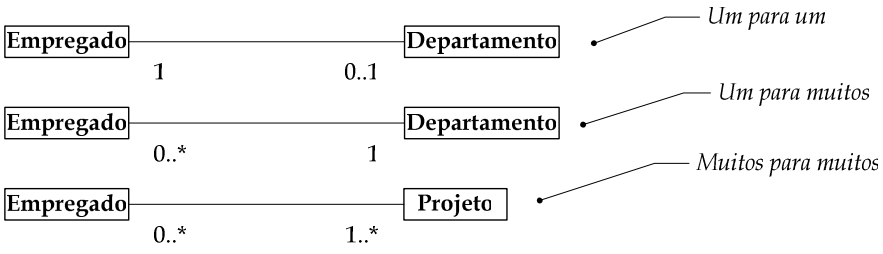
Conectividade

- A **conectividade** corresponde ao tipo de associação entre duas classes: "*muitos para muitos*", "*um para muitos*" e "*um para um*".
- A conectividade da associação entre duas classes depende dos símbolos de multiplicidade que são utilizados na associação.

Conectividade X Multiplicidade

Conectividade	Em um extremo	No outro extremo
Um para um	0..1 1	0..1 1
Um para muitos	0..1 1	* 1..* 0..*
Muitos para muitos	* 1..* 0..*	* 1..* 0..*

Exemplo (conectividade)



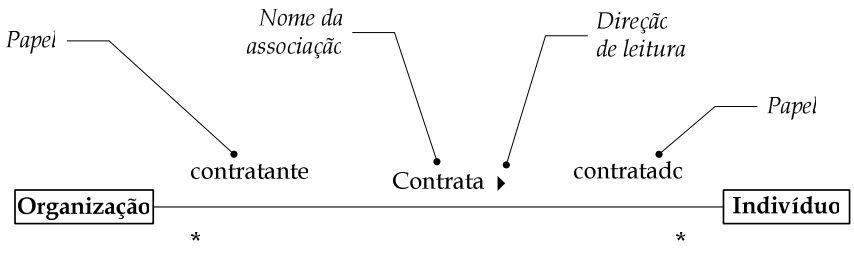
Participação

- Uma característica de uma associação que indica a necessidade (ou não) da existência desta associação entre objetos.
- A participação pode ser *obrigatória* ou *opcional*.
  - 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

- Para melhor esclarecer o significado de uma associação no diagrama de classes, a UML define três recursos de notação:
  - ***Nome da associação***: fornece algum significado semântico a mesma.
  - ***Direção de leitura***: indica como a associação deve ser lida
  - ***Papel***: para representar um papel específico em uma associação.

Exemplo (Nome de associação, direção de leitura e papéis)

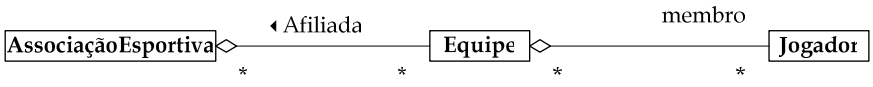


Agregação

- É um caso especial da associação
  - conseqüentemente multiplicidades, participações, papéis, etc. podem ser usados igualmente
- Utilizada para representar conexões que guardam uma relação *todo-parte* entre si.
- Em uma agregação, um objeto está contido no outro, ao contrário de uma associação.
- Onde se puder utilizar uma agregação, uma associação também poderá ser utilizada.

Notação para Agregação

- *Representada como uma linha conectando as classes relacionadas, com um losango branco perto da classe que representa o todo.*
- Exemplo:

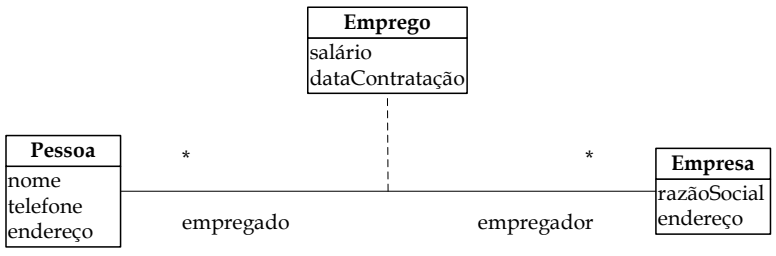


Classe Associativa

- É uma classe que está ligada a uma associação, ao invés de estar ligada a outras classes.
- É normalmente necessária quando duas ou mais classes estão associadas, e é necessário manter informações sobre esta associação.
- Uma classe associativa pode estar ligada a associações de qualquer tipo de conectividade.

Notação de Classes Associativas

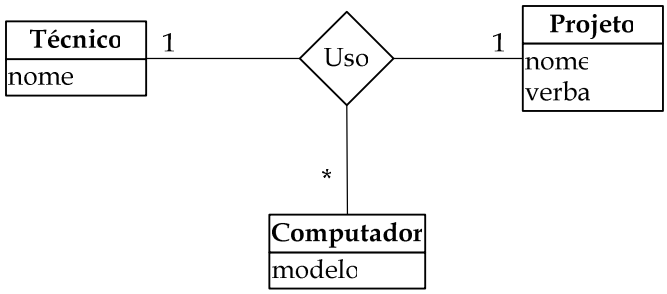
- *Representada pela notação utilizada para uma classe. A diferença é que esta classe é ligada a uma associação.*
- Exemplo:



Associações n-árias

- São utilizadas para representar a associação existente entre objetos de n classes.
- Uma ***associação ternária*** são uma caso mais comum (menos raro) de associação n-ária (n = 3).
- Na notação da UML, as linhas de associação se interceptam em um losango.

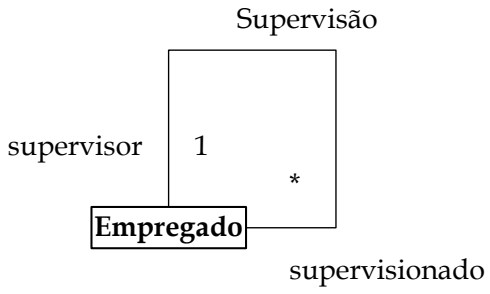
Exemplo (associação ternária)



Associações Reflexivas

- Associa objetos da mesma classe.
  - Cada objeto tem um papel distinto na associação.
- A utilização de papéis é bastante importante para evitar ambigüidades na leitura da associação.
- Uma associação reflexiva *não* indica que um objeto se associa com ele próprio.
  - Ao contrário, indica que objetos de uma mesma classe se associam

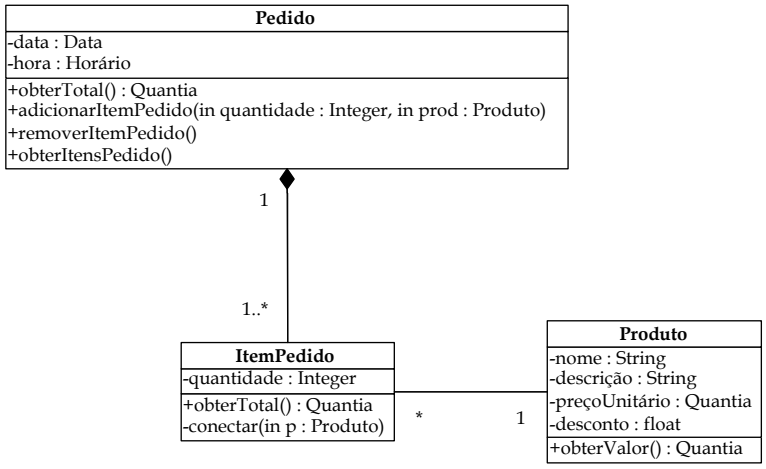
Exemplo (Associação Reflexiva)



Transformação de associações em agregações ou composições

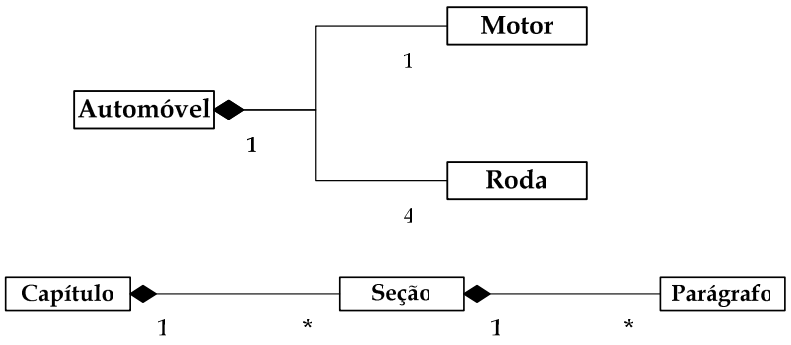
- Na agregação:
  - Os objetos que fazem parte do todo são criados e destruídos independentemente deste último.
  - Um objeto parte pode ser utilizado para compor diversos objetos todos.
  - A destruição de um desses objetos todo não implica na destruição do objeto parte.
- Na composição:
  - os objetos *parte* pertencem a um único *todo*.
  - objetos parte são sempre criados e destruídos pelo objeto todo.
  - Usualmente os clientes do objeto todo não têm conhecimento da estrutura das partes.
    - No objeto composto, são definidas operações para adicionar e remover objetos componentes.

Transformação de associações em agregações ou composições



Transformação de associações em agregações ou composições

- Note que composições (agregações) podem se estender por diversos níveis, formando hierarquias de composição (agregação).



Exercícios e Leitura Complementar

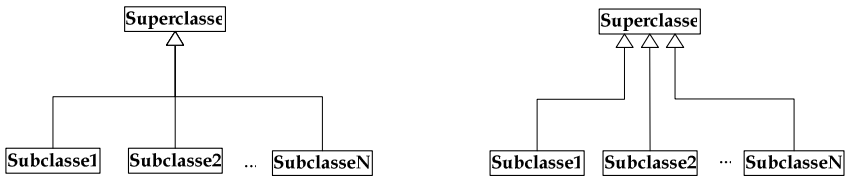
- Elaborar o modelo de dados conceitual dos exercícios de modelagem 1, 2, 3 e 4 da lista de exercícios.

- Ler o capítulo 5 do livro *Princípios de Análise e Projeto de Sistemas com UML* – Eduardo Bezerra – Ed. Campus.

Generalização – Terminologia

- *subclasse* X *superclasse*.
- *supertipo* X *subtipo*.
- *classe base* X *classe herdeira*.
- classe de *especialização* X classe de *generalização*.
- *ancestral* e *descendente* (generalização em vários níveis)

Notação para Generalização

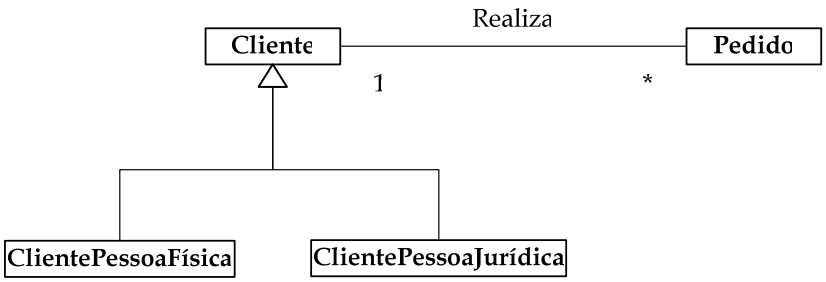


Generalização X Associação

- Importante: a generalização difere da associação (agregação, composição) porque a primeira se trata de um relacionamento *entre classes*.
  - "Gerentes são tipos especiais de funcionários".
  - "Gerentes chefiam departamentos".
- Na associação, objetos específicos de uma classe se associam entre si ou com objetos específicos de outras classes.

Herança de Associações

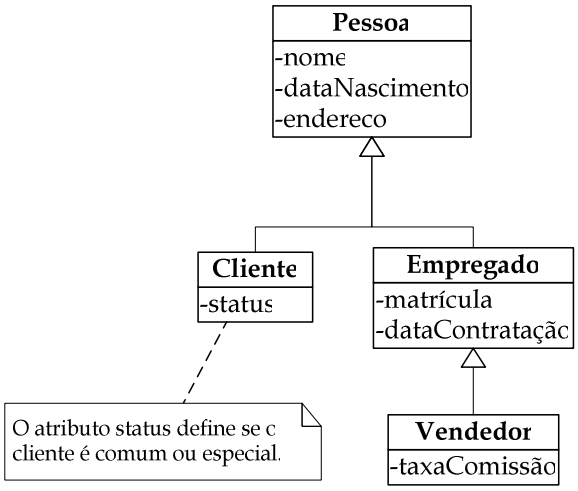
- Atributos e operações e associações são herdados pelas subclasses.



Hierarquias de Generalização

- A generalização pode ser aplicada em vários níveis (*hierarquia de generalização*).
  - uma classe que herda propriedades de uma outra classe pode ela própria servir como superclasse.
- Características importantes:
  - ***Transitividade***: uma classe em uma hierarquia herda propriedades e relacionamentos de todos os seus ancestrais.
  - ***Assimetria***: 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 generalização.

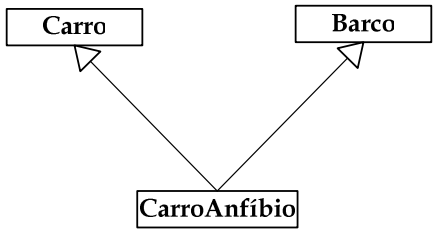
Hierarquias de Generalização



Herança Múltipla

- ***Herança múltipla:*** Uma classe pode ter mais de uma superclasse.
  - Tal classe herda de todas a suas superclasses.
- O uso de herança múltipla deve ser evitado.
  - Esse tipo de herança é difícil de entender.
  - Algumas LPs não dão suporte à implementação desse tipo de herança (Java e Smalltalk).

Exemplo (Herança Múltipla)

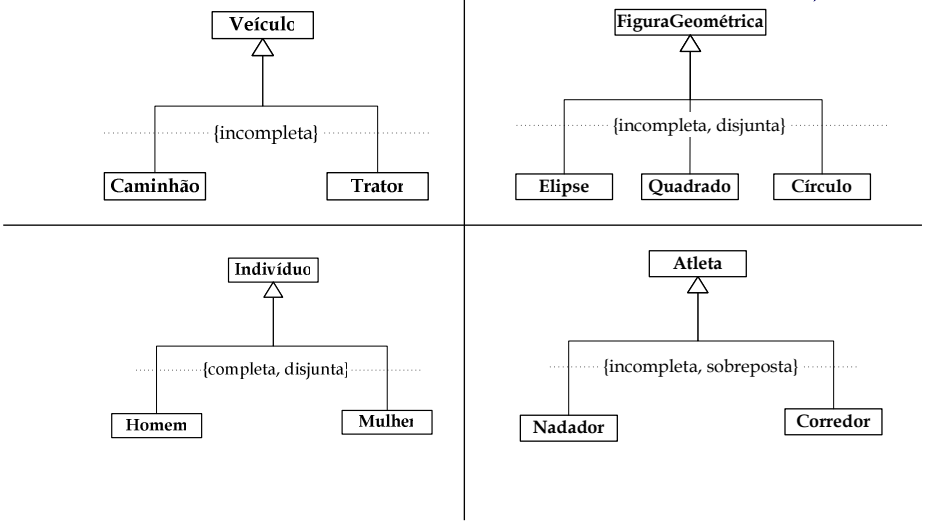


Restrições sobre Generalizações

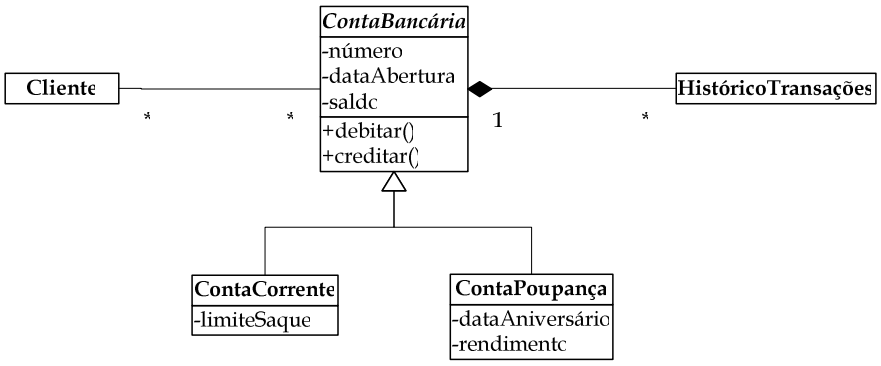
- Restrições sobre generalizações são representadas (entre chaves) no diagrama de classes próximas à linha do relacionamento.
- Restrições predefinidas pela UML:

Sobreposta **Disjunta**
Completa **Incompleta**

Exemplos (Restrições sobre Generalizações)



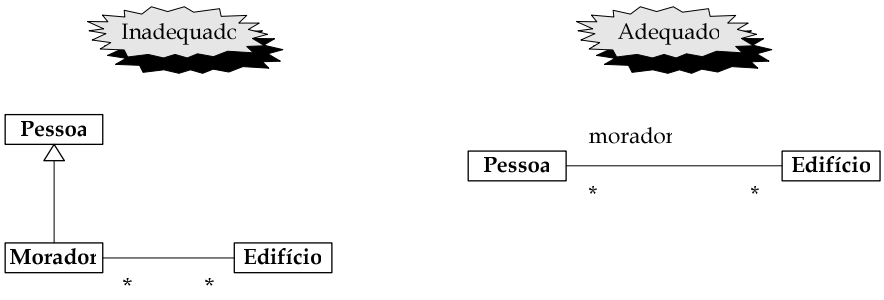
Exemplo de Heranças e Associações



Dicas

- Deve-se evitar a construção de hierarquias de generalização muito profundas (com mais de três níveis)
  - dificultam a leitura do diagrama.
- Papéis e subclasses não devem ser confundidos.
  - Um papel corresponde ao uso de uma certa classe em uma associação.Uma classe pode assumir vários papéis.
  - O modelador deve evitar a criação de subclasses em situações que podem ser resolvidas através da utilização de papéis.

Papel X Subclasse



Exercícios e Leitura Complementar

- Elaborar o modelo de dados conceitual dos exercícios de modelagem 5, 6, 7 e 8 da lista de exercícios.
- Ler o capítulo 9 do livro *Princípios de Análise e Projeto de Sistemas com UML* – Eduardo Bezerra – Ed. Campus.