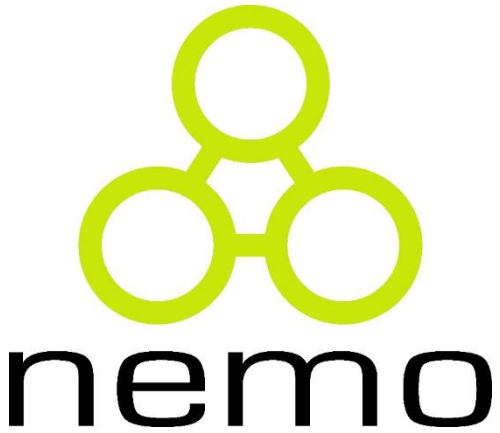
# Engenharia de Requisitos de Software




Jordana S. Salamon

jssalamon@inf.ufes.br

jordanasalamon@gmail.com

DEPARTAMENTO DE INFORMÁTICA

CENTRO TECNOLÓGICO

UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO

# Modelagem Conceitual Estrutural

- Um modelo conceitual é uma abstração da realidade segundo uma conceituação. Ele pode ser usado para comunicação, aprendizado e análise de aspectos relevantes do domínio subjacente.

- O modelo conceitual estrutural de um sistema tem por objetivo **descrever as informações que esse sistema deve representar e gerenciar.**

nemo

# Modelagem Conceitual Estrutural

- A modelagem conceitual é a atividade de descrever alguns dos aspectos do mundo físico e social a nossa volta, com o propósito de entender e comunicar.

- Os modelos resultantes das atividades de modelagem conceitual são essencialmente destinados a serem usados por pessoas e não por máquinas, então modelos conceituais devem ser concebidos com foco no domínio do problema e não no domínio da solução .

# Modelagem Conceitual Estrutural

- **As informações a serem capturadas em um modelo conceitual estrutural devem existir independentemente da existência de um sistema computacional para tratá-las.**

- Diferentes tipos de modelos podem ser usados na modelagem conceitual estrutural, cada um obedecendo a um paradigma específico, ou seja, obedecendo uma forma específica de ver o mundo.

# Modelagem Conceitual Estrutural

Por exemplo:

*Paradigma Estruturado:* adota uma visão de desenvolvimento baseada em um modelo entrada-processamento-saída. No paradigma estruturado, os dados são considerados separadamente das funções que os transformam e a decomposição funcional é usada intensamente.

*Paradigma Orientado a Objetos:* parte do pressuposto que o mundo é povoado por objetos, ou seja, a abstração básica para se representar as coisas do mundo são os objetos.

# Modelagem Conceitual Estrutural

- Na modelagem conceitual segundo o paradigma orientado a objetos, tipos de entidades são modelados como **classes** e tipos de relacionamentos são modelados como **atributos e associações**.

- Assim, o propósito da modelagem conceitual estrutural orientada a objetos é definir as classes, atributos e associações que são relevantes para tratar o problema a ser resolvido.

# Modelagem Conceitual Estrutural

- Para tal, as seguintes tarefas devem ser realizadas:

- • Identificação de Classes
- • Identificação de Atributos e Associações
- • Especificação de Hierarquias de Generalização/Especialização

- É importante notar que essas atividades são dependentes umas das outras e que, durante o desenvolvimento, elas são realizadas de forma paralela e iterativa, sempre visando ao entendimento do domínio do problema, desconsiderando aspectos de implementação.

nemo

# Modelagem Conceitual Estrutural

## O Paradigma Orientado a Objetos

- O mundo é visto como sendo composto por objetos, onde um objeto é uma entidade que combina estrutura de dados e comportamento funcional.

Carro

Pessoa

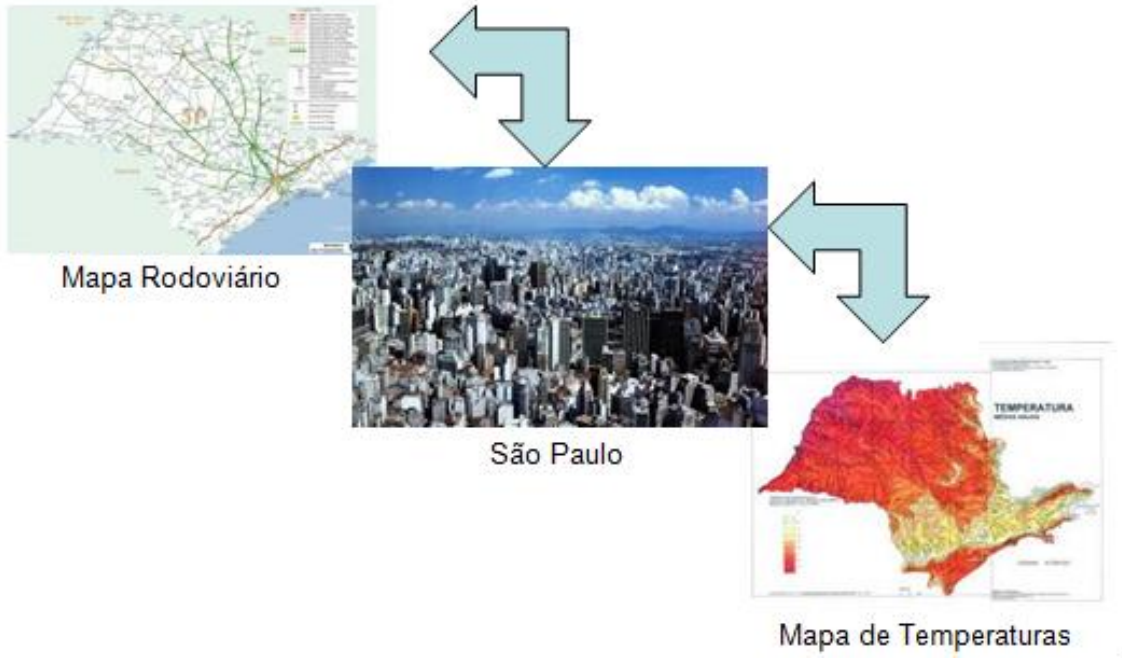

Turma

# Modelagem Conceitual Estrutural

## Princípios da Orientação a Objetos

### a) Abstração

Visão simplificada de algo, onde apenas os elementos relevantes são considerados.

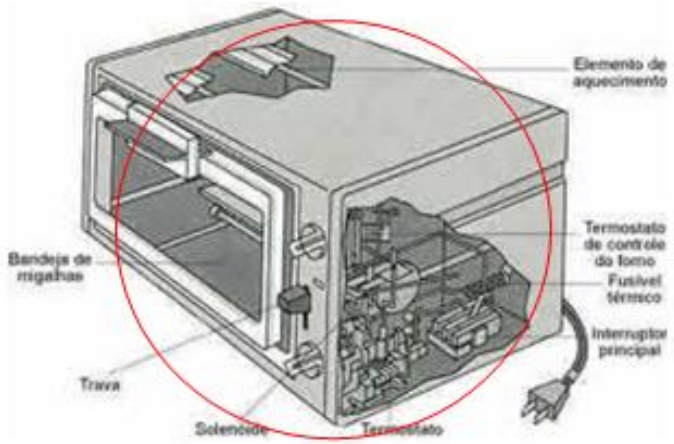


# Modelagem Conceitual Estrutural

## b) Encapsulamento

- Separação dos aspectos externos de um objeto, acessíveis por outros objetos, de seus detalhes internos de implementação, que ficam ocultos dos demais objetos.

Estrutura Interna




Interface

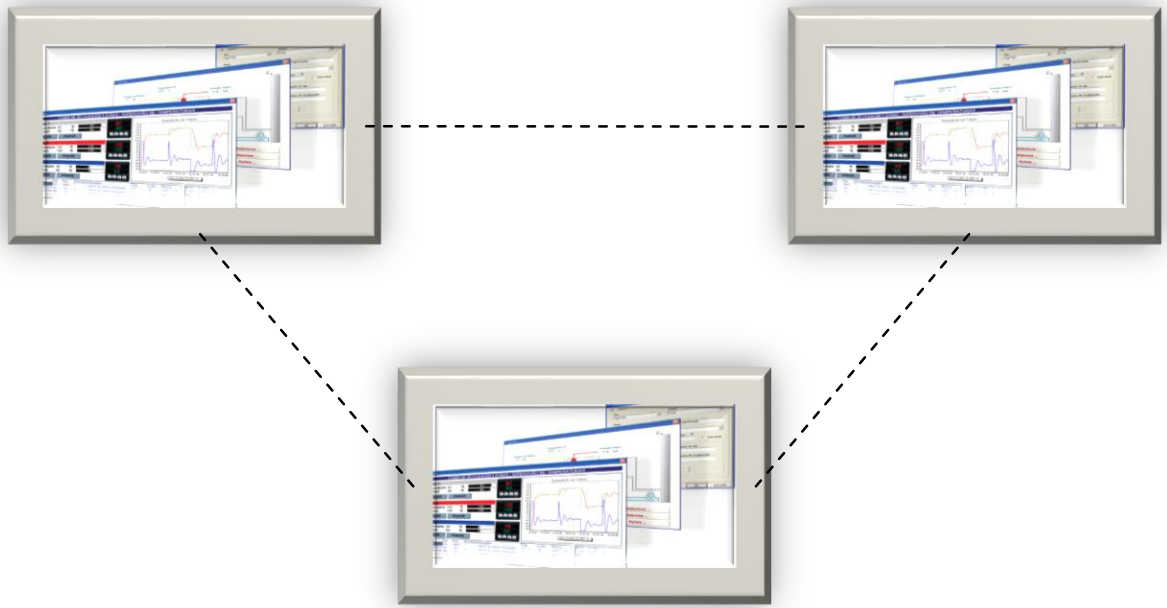
- Abstração e encapsulamento são conceitos complementares: enquanto a abstração enfoca o comportamento observável de um objeto, o encapsulamento oculta a implementação que origina esse comportamento.

nemo

# Modelagem Conceitual Estrutural

**c) Modularidade**

Obtenção de sistemas decompostos em um conjunto de módulos coesos e fracamente acoplados.



# Modelagem Conceitual Estrutural

## Conceitos da Orientação a Objetos

a) **Objetos:** entidades que interagem entre si, onde cada uma delas desempenha um papel específico.

O carro do João

João

b) **Classes:** descrevem um conjunto de objetos com as mesmas propriedades (atributos e associações) e o mesmo comportamento (operações).

*Objetos são instâncias das classes*

**Carro**

Carro A

Carro B

Carro C

**Pessoa**

João

Maria

Cecy

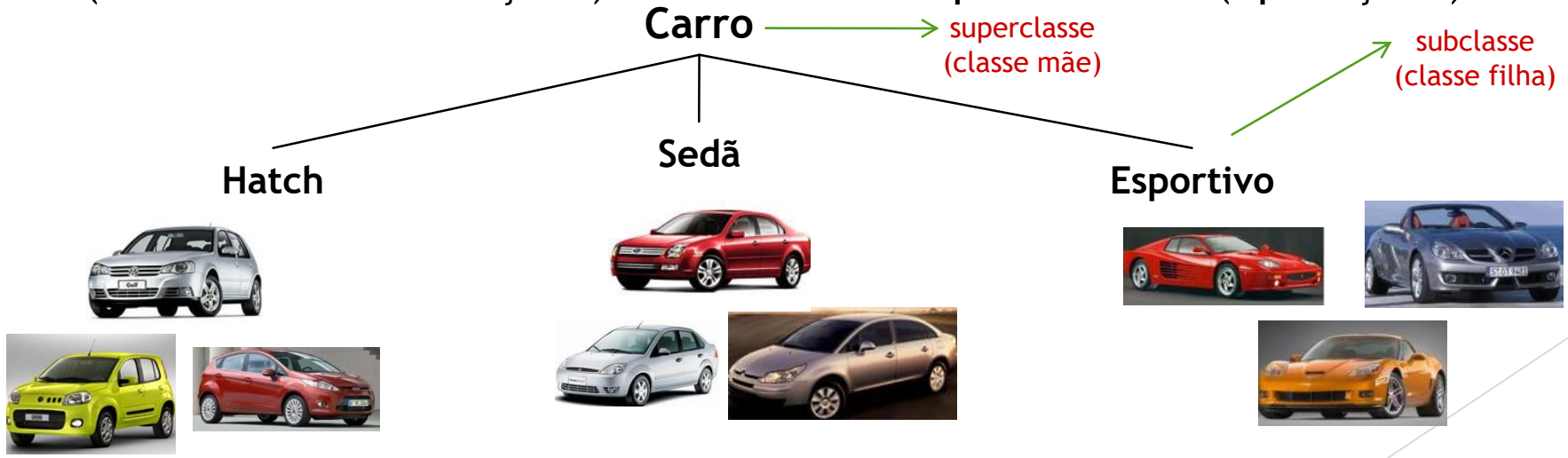
# Modelagem Conceitual Estrutural

c) **Ligações e Associações:** relacionamentos entre objetos e classes (respectivamente).



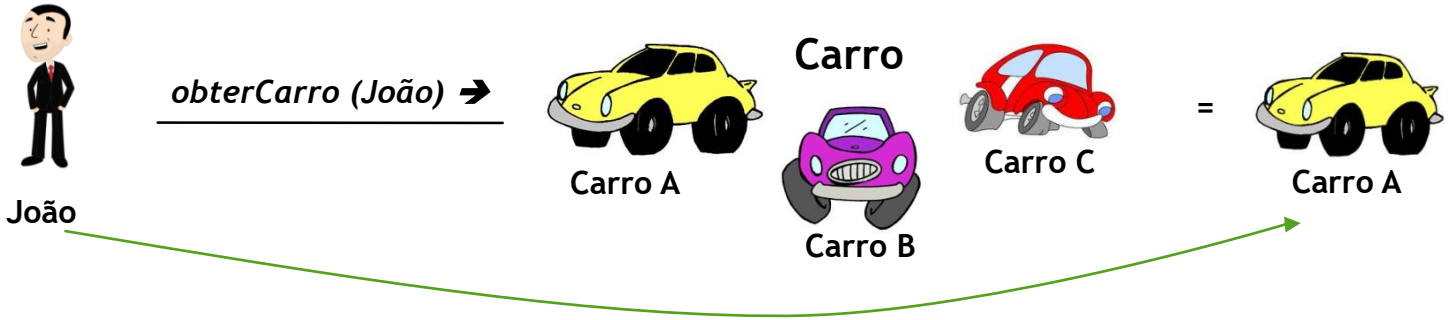


d) **Herança:** descrevem um conjunto de classes com as mesmas propriedades (atributos e associações) e o mesmo comportamento (operações).

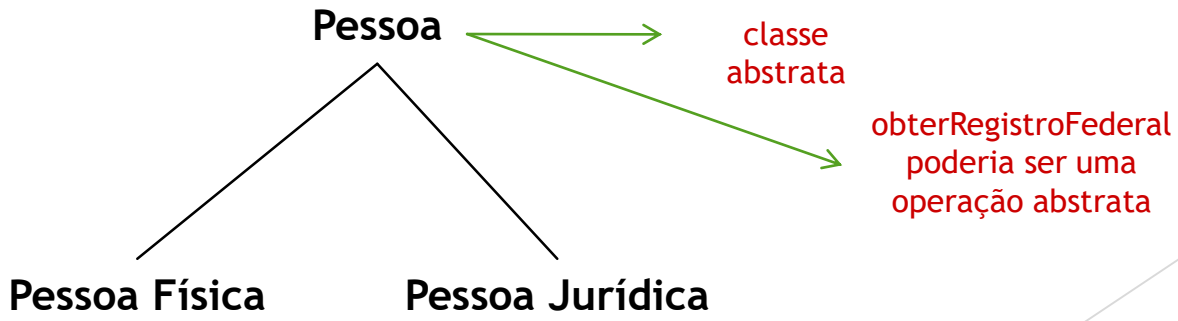


# Modelagem Conceitual Estrutural

e) **Mensagens e Métodos:** forma de comunicação entre os objetos.




d) **Classes e Operações Abstratas:** classes que não possuem instância e operações que não são implementadas nas classes (são apenas assinatura). A classe abstrata existe meramente para que um comportamento comum a um conjunto de classes possa ser colocado em uma localização comum e definido uma única vez.



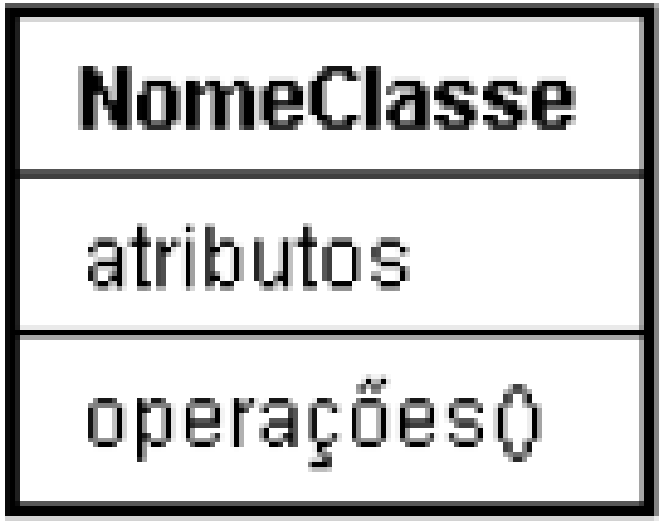

nemo

# Identificação de Classes

- Tipos de entidade são um dos mais importantes elementos em modelos conceituais.
- Um tipo de entidade pode ser definido como um tipo cujas instâncias em um dado momento são objetos individuais identificáveis que se consideram existir no domínio naquele momento.
- Na orientação a objetos, tipos de entidade são representados por **classes**, enquanto as instâncias de um tipo de entidade são **objetos.**

# Identificação de Classes

- Na UML, classes são representadas por um retângulo com três compartimentos: o compartimento superior é relativo ao nome da classe; o compartimento do meio é dedicado à especificação dos atributos da classe; e o compartimento inferior é dedicado à especificação das operações da classe:



# Identificação de Classes

- Tomando por base os requisitos iniciais de cliente, relatórios de atividades de levantamento de requisitos e, sobretudo, descrições de casos de uso, é possível iniciar a modelagem conceitual estrutural.

- Esse trabalho começa com a descoberta de quais classes devem ser incluídas no modelo.

- As classes de um modelo representam a expressão inicial do sistema. As atividades subsequentes da modelagem estrutural buscam obter uma descrição cada vez mais detalhada, em termos de associações e atributos.

nemo

# Identificação de Classes

- Sugere-se que os seguintes elementos sejam considerados como candidatos a classes:

- • Agentes: entidades do domínio do problema que têm a capacidade de agir com intenção de atingir uma meta.

- Em sistemas de informação, há dois tipos principais de agentes: os agentes físicos (tipicamente pessoas) e os agentes sociais (organizações, unidades organizacionais, sociedades etc.). Em relação às pessoas, deve-se olhar para os papéis desempenhados pelas diferentes pessoas no domínio do problema.

nemo

# Identificação de Classes

- • Objetos: entidades sem a capacidade de agir, mas que fazem parte do domínio de informação do problema.

- Podem ser também classificados em físicos (p.ex., carros, livros, imóveis) e sociais (p.ex., cursos, disciplinas, leis). Entretanto, há também outros tipos de objetos, tais como objetos de caráter descritivo usados para organizar e descrever outros objetos de um domínio (p.ex., modelos de carro).

- Objetos sociais e de descrição tendem a ser coisas menos tangíveis, mas são tão importantes para a modelagem conceitual quanto os objetos físicos.

# Identificação de Classes

- • Eventos: representam a ocorrência de ações no domínio do problema que precisam ser registradas e lembradas pelo sistema.
- Eventos acontecem no tempo e, portanto, a representação de eventos normalmente envolve a necessidade de registrar, dentre outros, quando o evento ocorreu (ponto no tempo ou intervalo de tempo).
- Deve-se observar que muitos eventos ocorrem no domínio do problema, mas grande parte deles não precisa ser lembrada.
- Para capturar os eventos que precisam ser lembrados e, portanto, registrados, devem-se focalizar os principais eventos de negócio do domínio do problema.

nemo

# Identificação de Classes

- Exemplo de eventos:

- Em um sistema de locação de automóveis, são potenciais classes de eventos: Locação, Devolução e Reserva.

- Por outro lado, a ocorrência de eventos cadastrais, tais como os cadastros de clientes e carros, tende a ser de pouca importância, não sendo necessário lembrar a ocorrência desses eventos.

# Identificação de Classes

- É sempre importante que o analista tenha em mente os **objetivos** do sistema durante a modelagem conceitual.

- Não se devem representar informações irrelevantes para o sistema e, portanto, **a relevância para o sistema é o principal critério a ser adotado para decidir se um determinado elemento deve ou não ser incluído no modelo conceitual estrutural do sistema.**

- O resultado principal da atividade de identificação de classes é a obtenção de uma lista de potenciais classes para o sistema em estudo.

nemo

# Identificação de Classes

- Um modelo conceitual estrutural para uma aplicação complexa pode conter dezenas de classes e, portanto, pode ser necessário definir uma representação concisa capaz de orientar um leitor em um modelo dessa natureza.

- O agrupamento de classes em subsistemas serve basicamente a este propósito, podendo ser útil também para a organização de grupos de trabalho em projetos extensos.

- Quando uma coleção de classes colabora entre si para realizar um conjunto coeso de responsabilidades (casos de uso), elas podem ser vistas como um subsistema. Assim, um subsistema é uma abstração que provê uma referência para mais detalhes em um modelo de análise, incluindo tanto casos de uso quanto classes.

# Identificação de Classes

- Uma vez identificadas as potenciais classes, deve-se proceder uma avaliação para decidir o que efetivamente considerar ou rejeitar.

- Além do critério de relevância, os seguintes critérios devem ser considerados nessa avaliação:

- • Estrutura complexa: o sistema precisa tratar informações sobre os objetos da classe? Tipicamente, uma classe deve ter, pelo menos, dois atributos. Se uma classe apresentar apenas um atributo, avalie se não é melhor tratá-la como um atributo de uma classe existente.

nemo

# Identificação de Classes

- • Atributos e associações comuns: os atributos e as associações da classe devem ser aplicáveis a todas as suas instâncias, isto é, a todos os objetos da classe.

- • Existência de instâncias: toda classe deve possuir instâncias. Uma potencial classe que possua uma única instância não deve ser considerada em um modelo conceitual estrutural.

- Tipicamente uma classe possui várias instâncias e a população da classe varia ao longo do tempo.

nemo

# Identificação de Atributos e Associações

- De uma perspectiva prática, atributos ligam classes do domínio do problema a tipos de dados.
- Associações, por sua vez, consistem em um tipo de informação que liga diferentes classes do domínio entre si.
- Tipos de dados podem ser primitivos ou específicos de domínio.
  - Os tipos de dados primitivos são aplicáveis aos vários domínios e sistemas e são considerados como sendo predefinidos.
  - Os tipos de dados específicos de um domínio de aplicação, por outro lado, precisam ser definidos.

# Identificação de Atributos e Associações

- São exemplos de tipos de dados específicos: CPF, ISBN de livros, endereço etc.
- Exemplos de tipos de dados primitivos:
  - • String: cadeia de caracteres;
  - • boolean: admite apenas os valores verdadeiro e falso;
  - • Integer (ou int): números inteiros;
  - • Float (ou float): números reais;
  - • Currency: valor em moeda (reais, dólares etc.);
  - • Date: datas, com informação de dia, mês e ano;
  - • Time: horas em um dia, com informação de hora, minuto e segundo;
  - • DateTime: combinação dos dois anteriores;
  - • YearMonth: informação de tempo contendo apenas mês e ano;
  - • Year: informação de tempo contendo apenas ano.

# Atributos

- Um atributo é uma informação de estado para a qual cada objeto em uma classe tem o seu próprio valor.
- Os atributos adicionam detalhes às abstrações e são apresentados na parte central do símbolo de classe.
- Atributos possuem um tipo de dado, que pode ser primitivo ou específico de domínio.
- Ao identificar um atributo como sendo relevante, deve-se definir qual o seu tipo de dado.

# Tipos de Dados

- Caso nenhum dos tipos de dados primitivos se aplique, deve-se definir, então, um tipo de dados específico.
- Exemplo:

- Em domínios que lidem com livros, é necessário definir o tipo ISBN15, cujas instâncias são ISBNs válidos.

- Em domínios que lidem com pessoas físicas e jurídicas, CPF e CNPJ também devem ser definidos como tipos de dados específicos.

- Usar um tipo de dados primitivo nestes casos, tal como String ou int, é insuficiente, pois não são quaisquer cadeias de caracteres ou números que se caracterizam como ISBNs, CPFs ou CNPJs válidos.

# Tipos de Dados

- Tipos de dados específicos podem apresentar propriedades. Por exemplo, CPF é um número de 11 dígitos, que pode ser dividido em duas partes: os 9 primeiros dígitos e os dois últimos, que são dígitos verificadores.

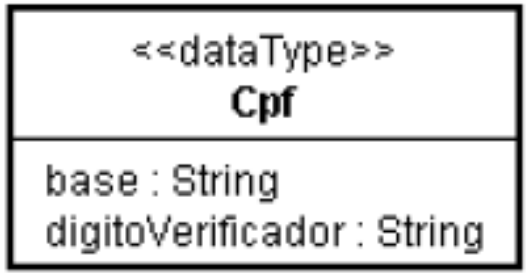
- Um tipo de dados especial é a enumeração. Na enumeração, os valores do tipo são enumerados explicitamente na forma de literais, como é o caso do tipo DiaSemana, que é tipicamente definido como um tipo de dados compreendendo sete valores: {Segunda, Terça, Quarta, Quinta, Sexta, Sábado e Domingo}.

- **IMPORTANTE**: Tipos de dados enumerados só devem ser usados quando se sabe a priori quais são os seus valores e eles são fixos. Assim, são bons candidatos a tipos enumerados informações como sexo (M/F), estado civil, etc.

nemo

# Tipos de Dados

- Notação de Tipos de Dados da UML:





nemo

# Atributos

- Uma dúvida típica e recorrente na modelagem estrutural é se um determinado item de informação deve ser modelado como uma classe ou como um atributo.

- Para que o item seja considerado uma classe, ele tem de passar nos critérios de inclusão já discutidos.

- Entretanto, há alguns itens de informação que passam nesses critérios, mas que ainda assim podem ser melhor modelados como atributos, tendo como tipo um tipo de dado complexo, específico de domínio.

# Atributos

- Um atributo deve capturar um conceito atômico, ou seja, um único valor ou um agrupamento de valores fortemente relacionados que sirva para descrever outro objeto.

- Além disso, para que um item de estrutura complexa seja modelado como um atributo, ele deve ser compreensível pelos interessados simplesmente pelo seu nome.

- **IMPORTANTE**: Uma vez que atributos e associações são tipos de relacionamentos, não devemos incluir na lista de atributos de uma classe, atributos representando associações (ou atributos representando “chaves estrangeiras” como se a classe fosse uma tabela de um banco de dados relacional).

nemo

# Atributos

- A sintaxe de atributos na UML é a seguinte:
- **<visibilidade> nome: tipo [multiplicidade] = valorInicial {propriedades}**
- A visibilidade de um atributo indica em que situações esse atributo é visível por outras classes.
- Na UML há quatro níveis de visibilidade, indicados pelos seguintes símbolos:
- + público : o atributo pode ser acessado por qualquer classe;
- # protegido: o atributo só é passível de acesso pela própria classe ou por uma de suas especializações;
- - privado: o atributo só pode ser acessado pela própria classe;
- ~ pacote: o atributo só pode ser acessado por classes declaradas dentro do mesmo pacote da classe a que pertence o atributo.

nemo

# Atributos

- O *tipo* indica o tipo de dado do atributo;

- A *multiplicidade* é a especificação do intervalo permitido de itens que o atributo pode abrigar. A multiplicidade deve ser informada, indicando o valor mínimo e o valor máximo, da seguinte forma:
  - **valor_mínimo .. valor_máximo**
- Exemplo:
- • nome: String -> instâncias da classe têm obrigatoriamente um e somente um nome.
- • cpf: Cpf [0..1] -> instâncias da classe têm um ou nenhum cpf.
- • telefones: Telefone [0..*] -> instâncias da classe têm um ou vários telefones.
- • pessoasContato: String [2] -> instâncias da classe têm exatamente duas pessoas de contato.

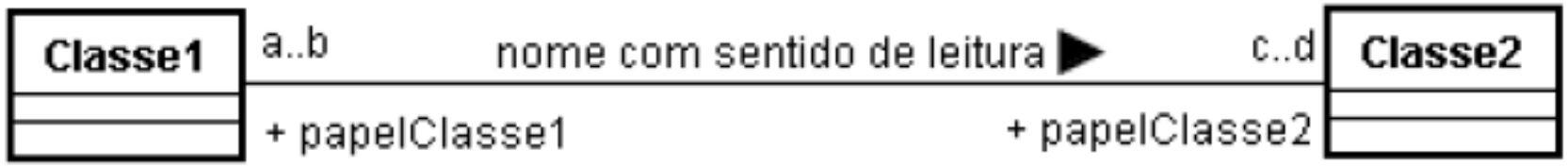
nemo

# Associações

- Uma associação é um tipo de relacionamento que ocorre entre instâncias de duas ou mais classes.

- Associações podem ser nomeadas. Além disso, cada classe envolvida na associação desempenha um papel, ao qual pode ser dado um nome.

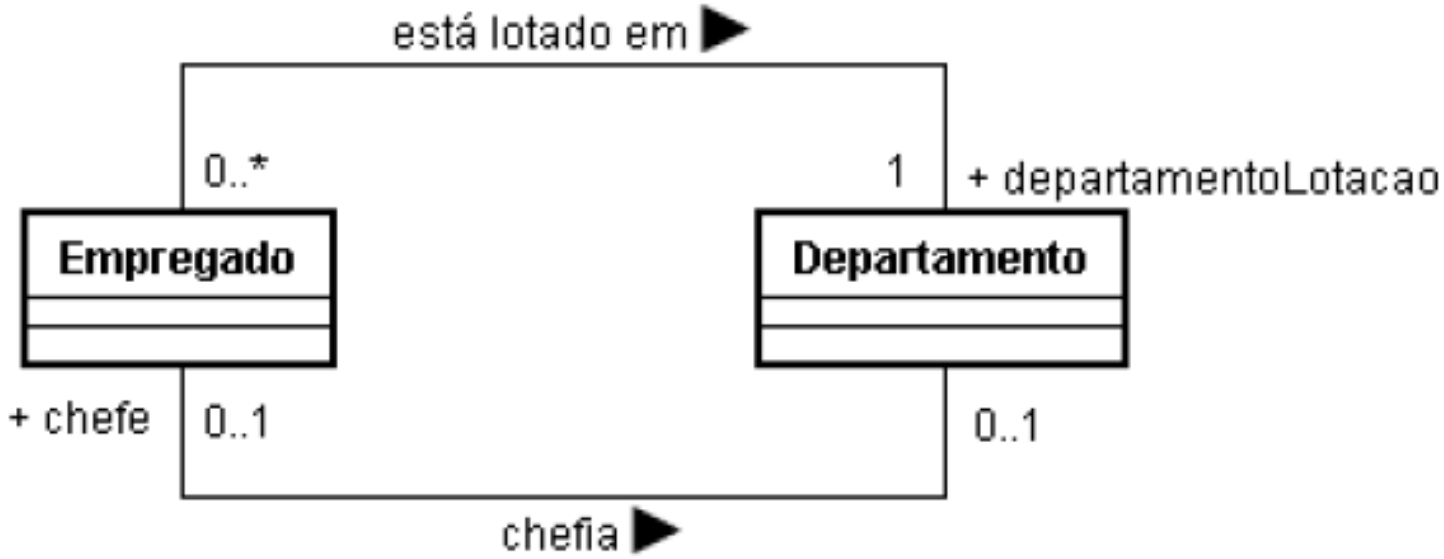
- Cada classe envolvida na associação possui também uma multiplicidade nessa associação, que indica quantos objetos podem participar de uma instância dessa associação .




nemo

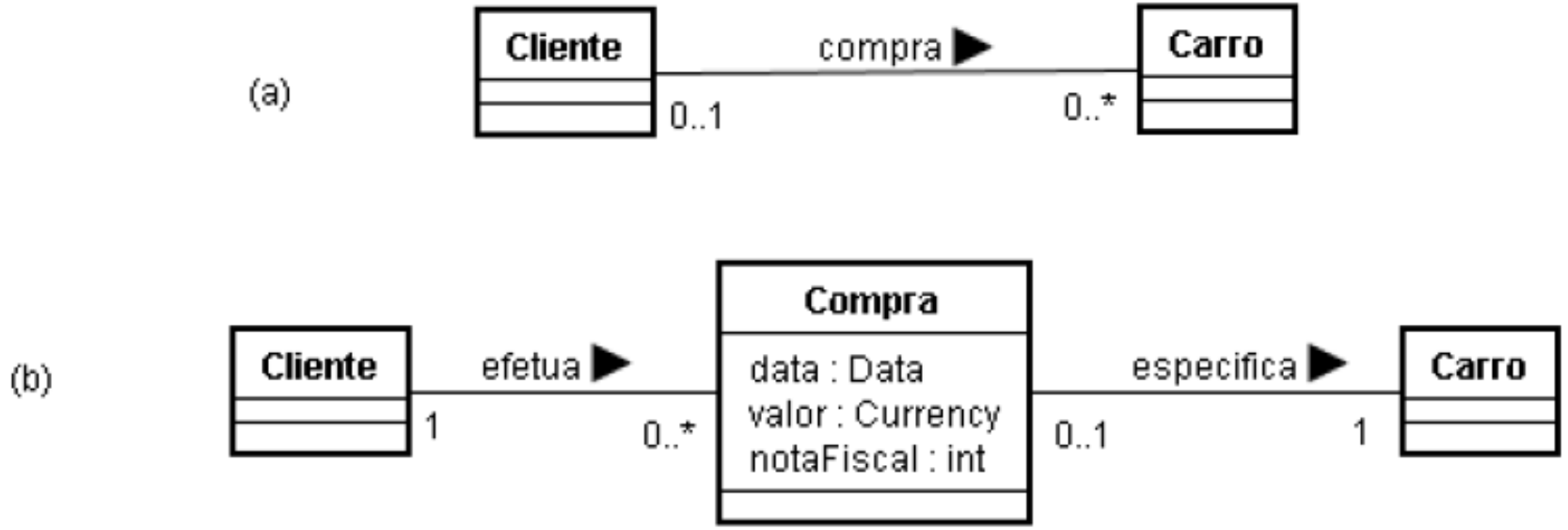
# Associações

- Exemplo de associação:



# Associações

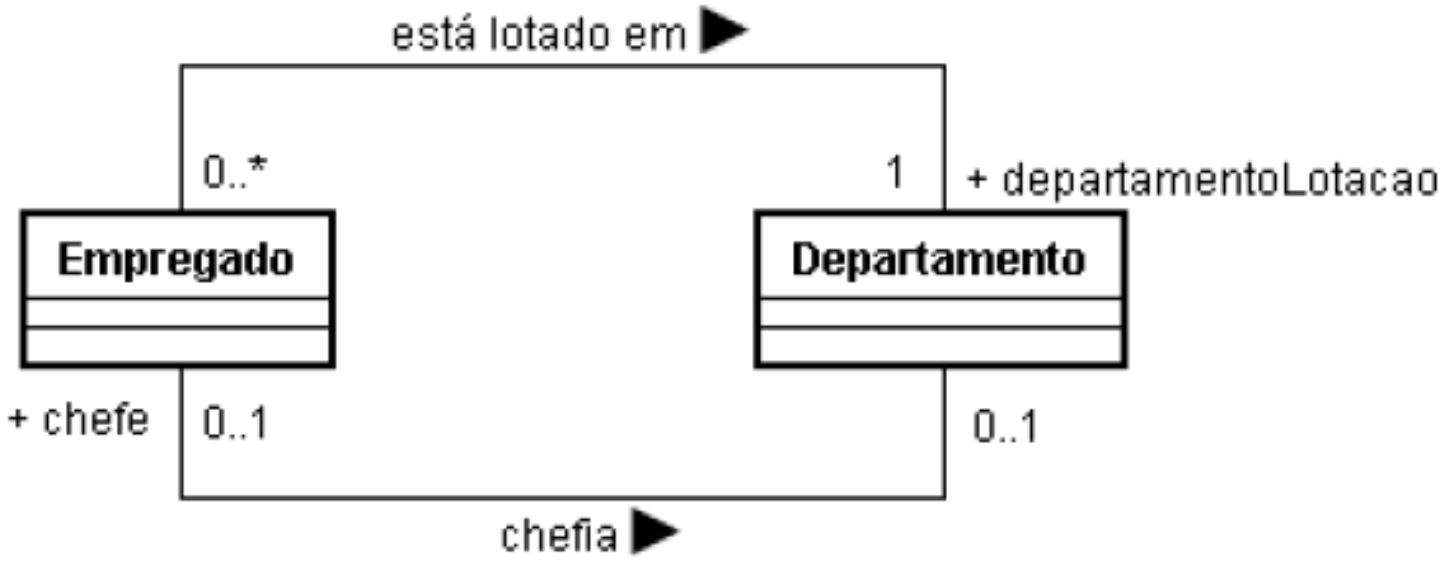
- Como há alguns eventos que precisam ter sua ocorrência registrada, eles são tipicamente mapeados como classes.
- Esses eventos estão descritos nos casos de uso e podem ter sido capturados como associações.
- Exemplo:



# Associações

- Na modelagem conceitual é fundamental saber a quantidade de objetos que uma associação admite em cada um de seus papéis, o que é capturado pelas multiplicidades da associação.

- Voltando ao exemplo:



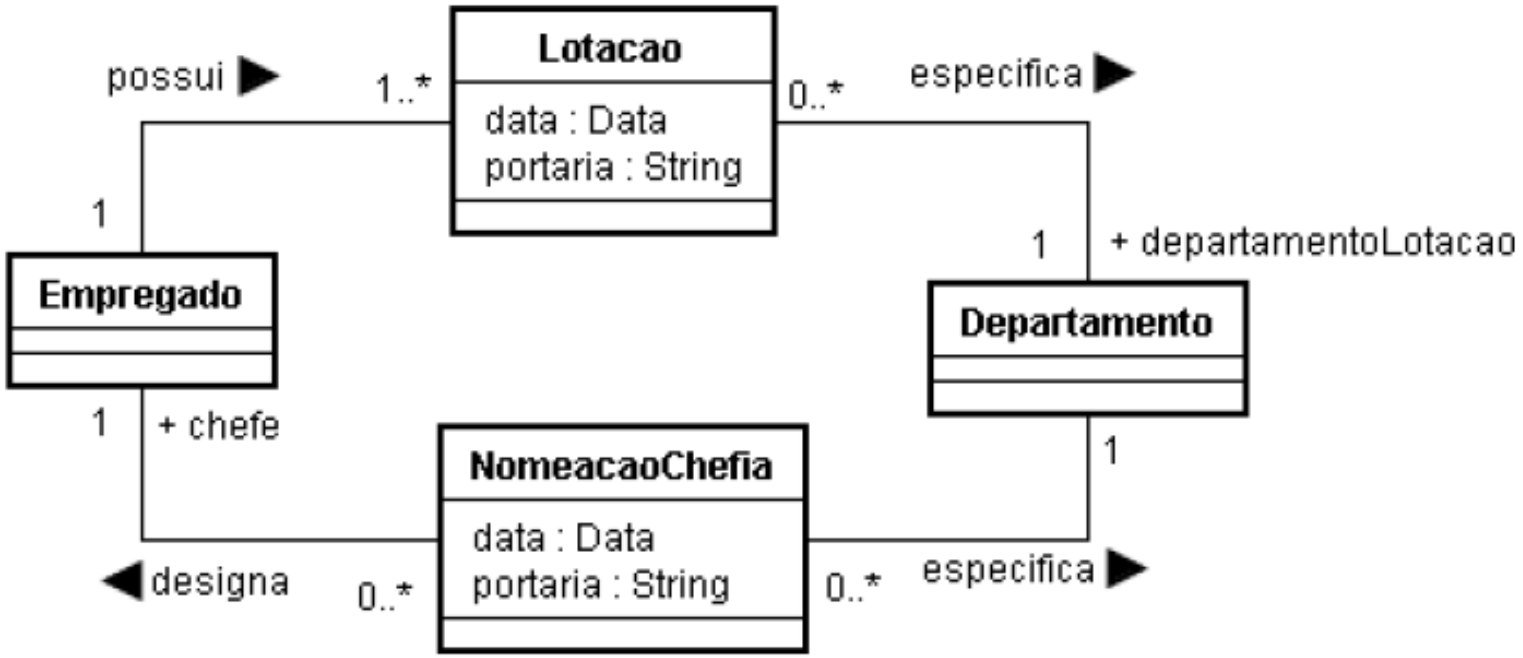
# Associações

- Para definir precisamente as multiplicidades, é necessário investigar os seguintes aspectos:
- Um empregado pode mudar de lotação?
- Se sim, é necessário registrar apenas a lotação atual ou é necessário registrar o histórico de lotações dos empregados (ou seja, registrar o evento de lotação de um empregado em um departamento)?
- Um departamento pode, ao longo do tempo, mudar de chefe?
- Se sim, é necessário registrar o histórico de chefias do departamento (ou seja, registrar o evento de nomeação do chefe do departamento)?

nemo

# Associações

- Retificando o modelo:




nemo

# Associações

- Infelizmente, o modelo é incapaz de responder a algumas perguntas.

- Para eliminar essas ambiguidades, é necessário capturar regras de negócio do tipo *restrições de integridade*. No exemplo, as seguintes regras se aplicam:
- • Um empregado só pode estar lotado em um único departamento em um dado momento.
- • Um empregado só pode estar designado como chefe de um único departamento em um dado momento.
- • Um departamento só pode ter um empregado designado como chefe em um dado momento.

nemo

# Associações

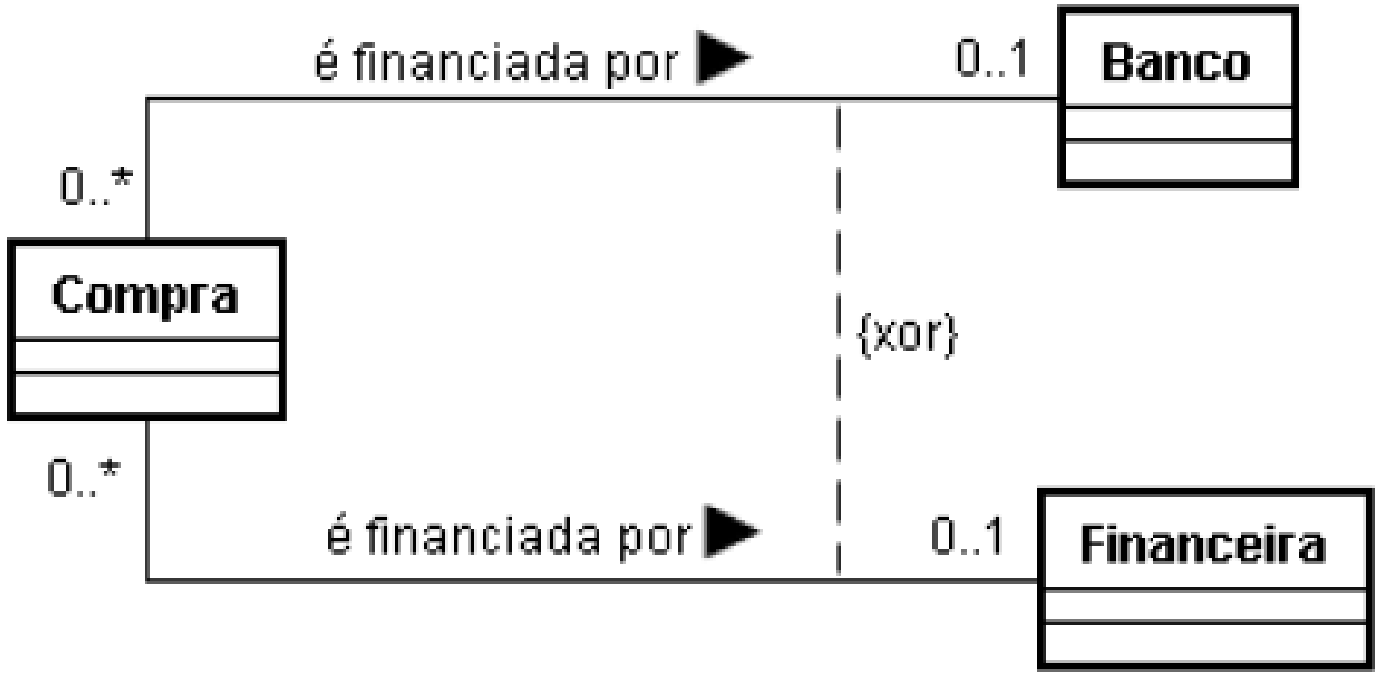
- Vale ressaltar que a UML provê alguns mecanismos para representar restrições de integridade em um modelo gráfico.

- As próprias multiplicidades são uma forma de capturar restrições de integridade (ditas restrições de integridade de cardinalidade).

- Além das multiplicidades, a UML provê o recurso de restrições, as quais são representadas entre chaves ({restrição}). Restrições podem ser usadas, dentre outros, para restringir a ocorrência de associações.

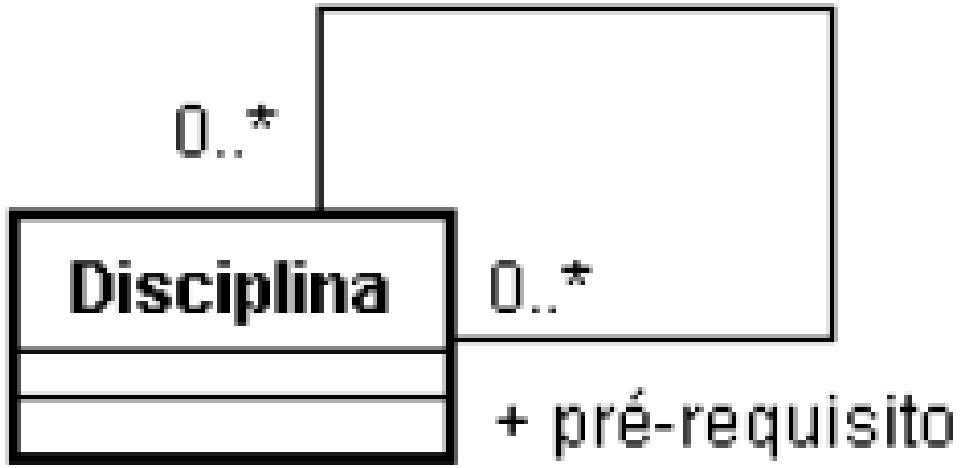
# Associações

- Exemplo:



# Associações

- Ainda em relação às multiplicidades, vale frisar que associações muitos-para-muitos são perfeitamente legais em um modelo orientado a objetos.



# Associações

- Deve-se observar, no entanto, que muitas vezes, uma associação muitos-para-muitos oculta a necessidade de uma classe do tipo evento a ser lembrado.

- Exemplo: em uma organização, empregados são alocados a projetos. Um empregado pode ser alocado a vários projetos, enquanto um projeto pode ter vários empregados a ele alocados.

(a)

# Associações

- Contudo, se quisermos registrar as datas de início e fim do período em que o empregado esteve alocado ao projeto, esse modelo é insuficiente e deve ser alterado para comportar uma classe do tipo evento lembrado Alocação.

(b)

# Associações

- De maneira geral pode-se pensar que, se uma associação apresenta atributos, é melhor tratá-la como uma nova classe.

- Seja o seguinte exemplo: em uma loja, um cliente efetua um pedido, discriminando vários produtos, cada um deles em uma certa quantidade.

(a)

# Associações

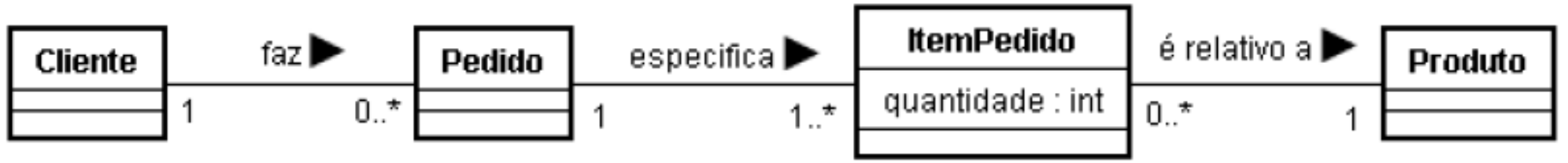
- Essa informação não pode ficar em Produto, pois diferentes pedidos pedem quantidades diferentes de um mesmo produto.

- Também não pode ficar em Pedido, pois um mesmo pedido tipicamente especifica diferentes quantidades de diferentes produtos.

- 

- De fato, quantidade não é nem um atributo da classe Pedido nem um atributo da classe Produto, mas sim um atributo da associação especifica.

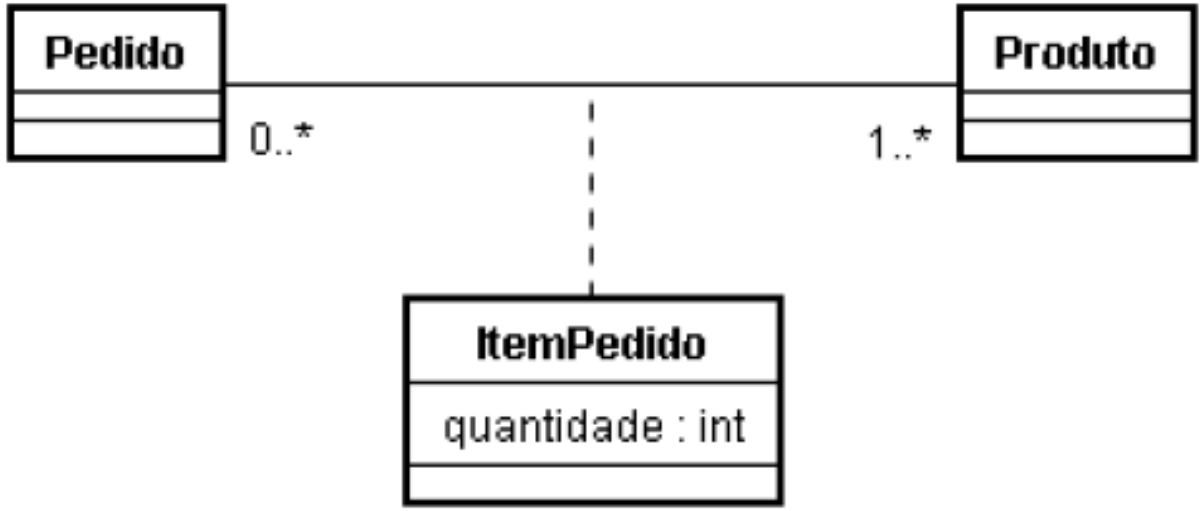
(b)



# Associações

- A UML oferece uma primitiva de modelagem, chamada classe de associação, que pode ser usada para reificar associações.

- Uma classe de associação pode ser vista como uma associação que tem propriedades de classe.




nemo

# Associações

- Classes associativas são ainda representações de associações.
- Assim como uma instância de uma associação, uma instância de uma classe associativa é um par ordenado conectando duas instâncias das classes envolvidas na associação.
- Assim, se Pedido100 é uma instância de Pedido, Lápis é uma instância de Produto e o Pedido100 especifica 5 Lápis, então uma instância de ItemPedido é a tupla ((Pedido100, Lápis), 5).

# Associações

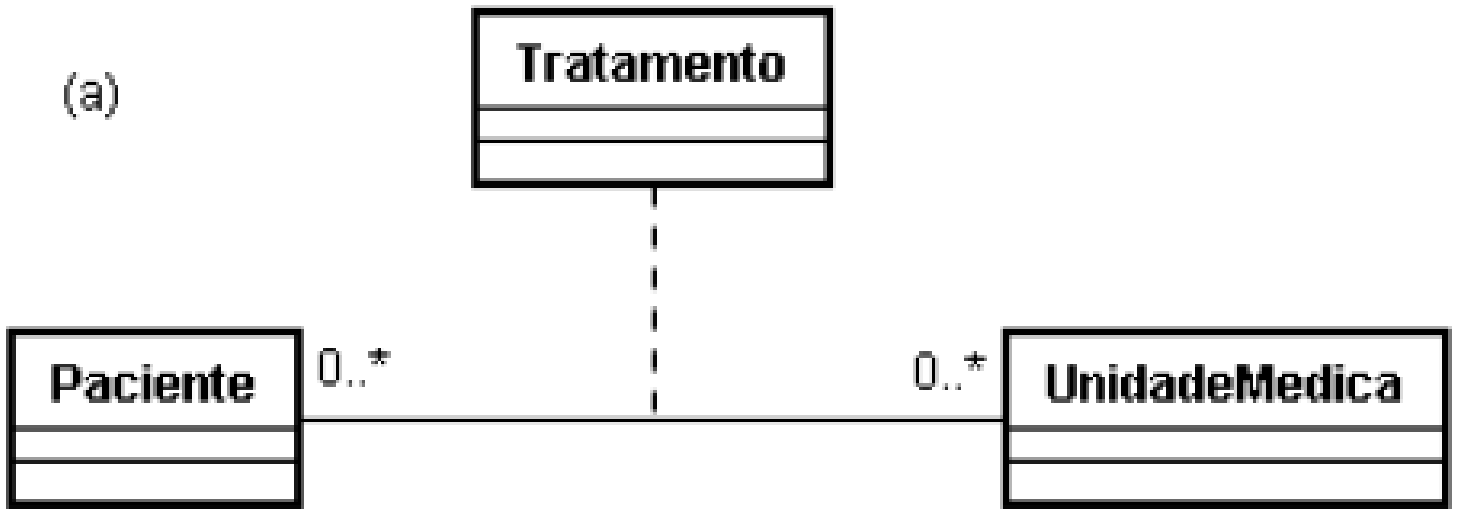
- Classes associativas podem ser usadas também para representar eventos cuja ocorrência precisa ser lembrada.

- Entretanto, é importante observar que o uso de classes associativas nesses casos pode levar a problemas de modelagem.

- Seja o seguinte contexto: em um hospital, pacientes são tratados em unidades médicas. Um paciente pode ser tratado em diversas unidades médicas diferentes, as quais podem abrigar diversos pacientes sendo tratados.

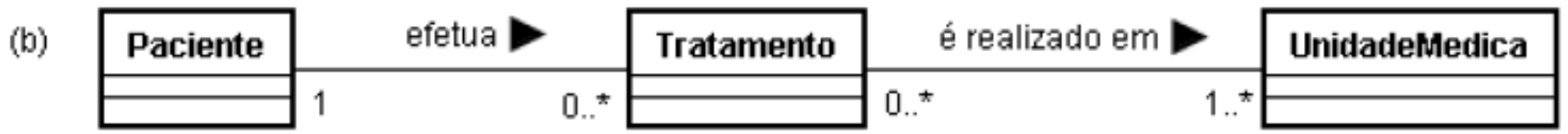
# Associações

- Modelo relacionado ao exemplo:



# Associações

- Pode-se imaginar que um tratamento é um tratamento de um paciente em várias unidades médicas.
- A classe de associação não permite representar isso.
- Assim, um modelo mais fiel ao domínio é aquele que representa Tratamento como uma classe do tipo evento a ser lembrado e que está relacionada com Paciente e Unidade Médica.

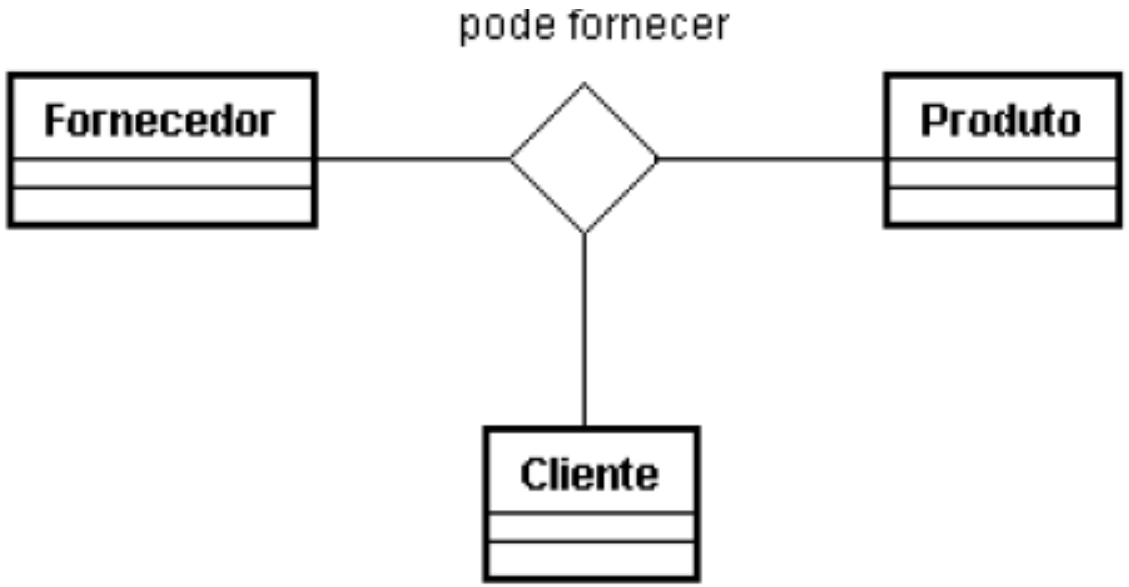


# Associações

- Até o momento, todas as associações mostradas foram associações binárias

- Entretanto, associações n-árias são também possíveis, ainda que bem menos corriqueiramente encontradas.

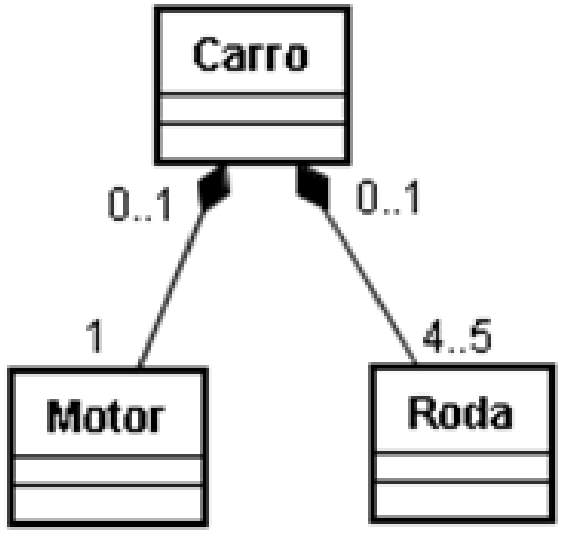
- Exemplo:




nemo

# Associações Todo-Parte

- Existem dois tipos de associação todo-parte: **agregação** e **composição**.




**Composição**

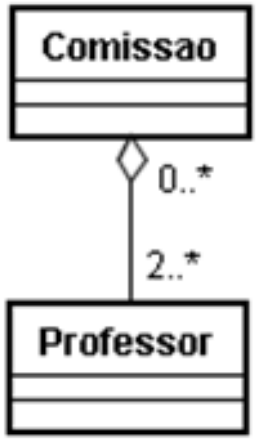
Um objeto-parte só pode ser parte de um único todo.

- Na composição, a parte tem responsabilidade na existência do todo.
- Se o objeto todo é excluído, todas as suas partes são também excluídas.
- Contudo, um objeto parte pode ser removido do objeto todo antes que este último seja excluído e, portanto, o objeto parte, nesse caso, não seria excluído junto com o objeto todo

nemo

# Associações Todo-Parte

- Existem dois tipos de associação todo-parte: **agregação** e **composição**.




**Agregação**

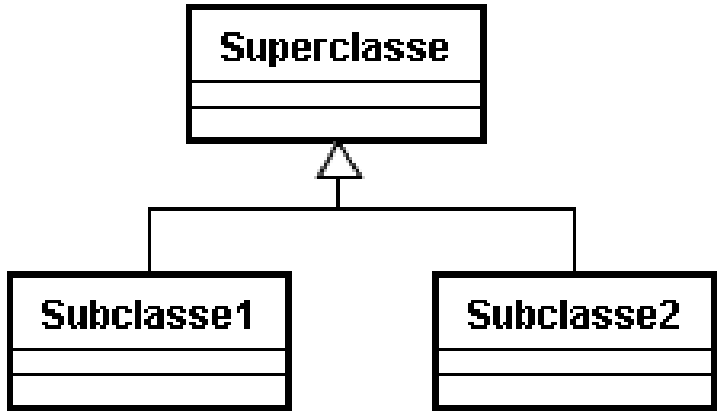
Um objeto-parte pode ser parte de mais de um todo.

# Associações - Generalização/Especialização

- Com relações de generalização/especialização é possível capturar similaridades entre classes, dispondo-as em hierarquias.

- **Atributos** descrevem propriedades dos objetos de uma classe. Quando um conjunto de classes possuir semelhanças e diferenças, então elas podem ser organizadas em uma hierarquia de classes, de forma a agrupar em uma superclasse os elementos de informação comuns, deixando as especificidades nas subclasses.

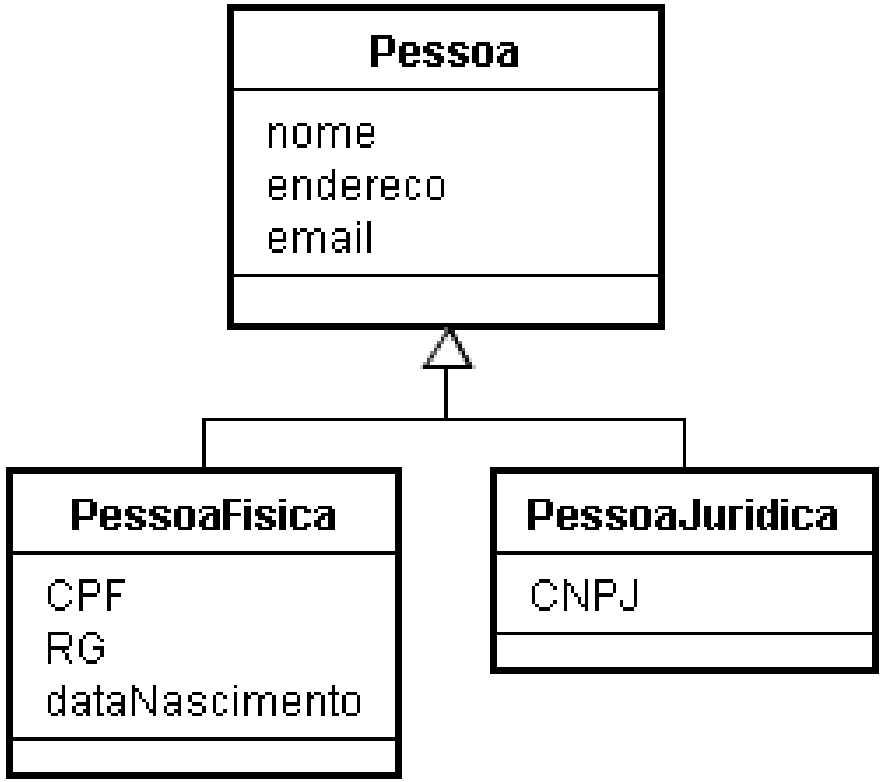
*Notação:*




nemo

# Associações - Generalização/Especialização

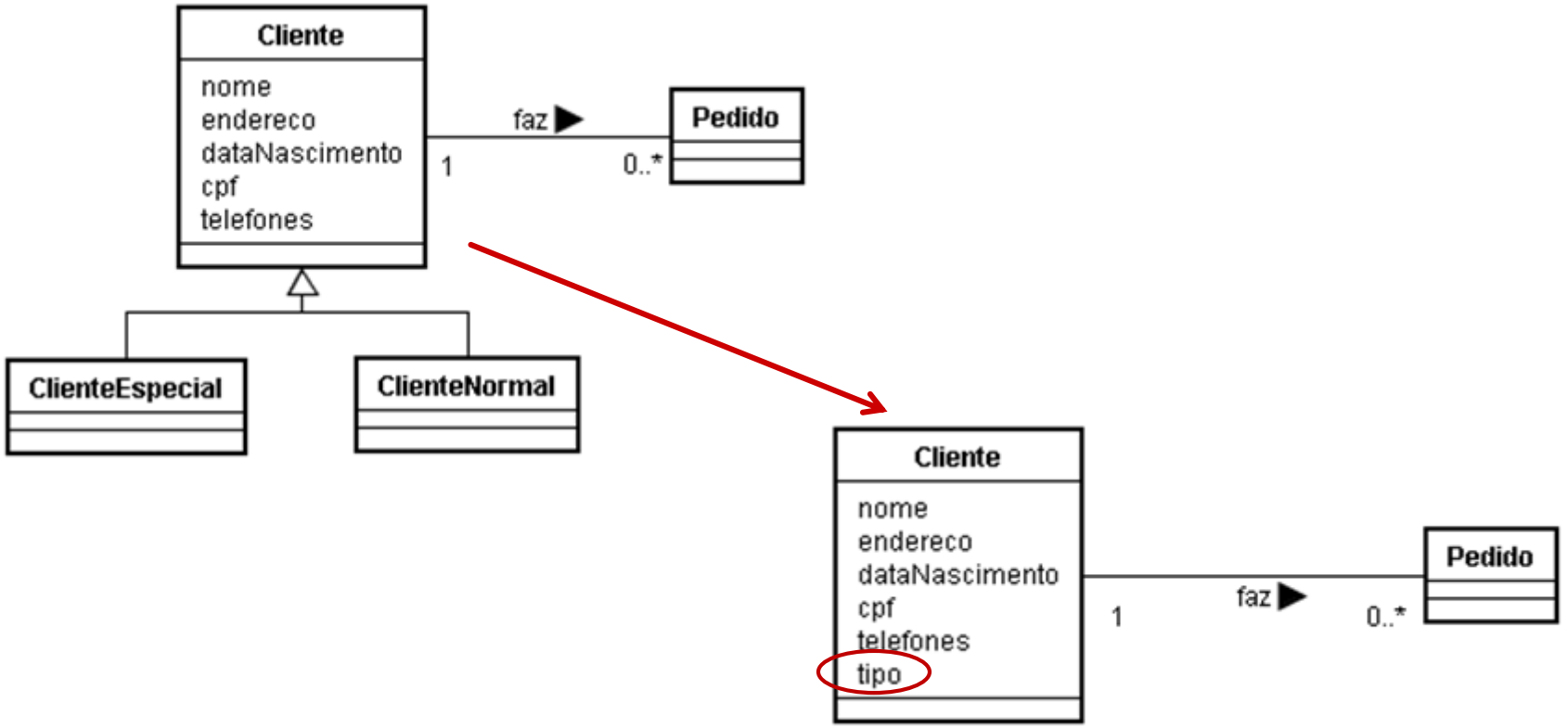
- Exemplo:




nemo

# Associações - Generalização/Especialização

**Uso ou Não da Herança**




nemo

# Associações - Generalização/Especialização

- Também não faz sentido criar uma hierarquia de classes em que a superclasse não tem nenhum atributo ou associação.
- Informações de estados pelos quais um objeto passa também não devem ser confundidas com subclasses.
- É importante considerar alguns critérios para incluir uma subclasse (ou superclasse) em um modelo conceitual.
- O principal deles é o fato da especialização (ou generalização) estar dentro do domínio de responsabilidade do sistema. Apenas subclasses (superclasses) relevantes para o sistema em questão devem ser consideradas.

nemo

# Associações - Generalização/Especialização

- Além desse critério básico, os seguintes critérios devem ser usados para analisar hierarquias de herança:

- • Uma hierarquia de classes deve modelar relações “é-um-tipo-de”, ou seja, toda subclasse deve ser um subtipo específico de sua superclasse.

- • Uma subclasse deve possuir todas as propriedades (atributos e associações) definidas por suas superclasses e adicionar mais alguma coisa (algum outro atributo, associação ou operação).

- • Todas as instâncias de uma subclasse têm de ser também instâncias da superclasse.

nemo

# Associações - Generalização/Especialização

- No que se refere à modelagem de superclasses, deve-se observar se uma superclasse é concreta ou abstrata.
- Se a superclasse puder ter instâncias próprias, que não são instâncias de nenhuma de suas subclasses, então ela é uma classe concreta.
- Por outro lado, se não for possível instanciar diretamente a superclasse, ou seja, se todas as instâncias da superclasse são antes instâncias das suas subclasses, então a superclasse é abstrata.
- Classes abstratas são representadas na UML com seu nome escrito em itálico.

# That’s all Folks!