Especificação Formal em Modelagem Conceitual

© Prof. Raul Sidnei Wazlawick

UFSC-CTC-INE

2010

Fonte: Análise e Projeto de Sistemas de Informação Orientados a Objetos, 2ª Edição, Elsevier, 2010.

Análise de Domínio

- Descoberta das informações que são gerenciadas no sistema: *representação* e *transformação* da informação.
- Ocorre em pelo menos duas fases do Processo Unificado.
  - Na fase de *concepção* pode-se fazer um modelo conceitual preliminar.
  - Na fase de *elaboração* este modelo é refinado e complementado.

Aspectos da Análise de Domínio

- *Estático* ou *estrutural*, que pode ser representado no *modelo conceitual*.
- *Funcional, que* pode ser representado através dos *contratos de operações* e *consultas de sistema*.

Caracterização do Modelo Conceitual

- Deve ser independente da solução tecnológica que virá a ser adotada.
- Deve conter apenas elementos referentes ao domínio do problema em questão.
- Os elementos da solução ficam relegados à atividade de projeto :
  - Interfaces.
  - Formas de armazenamento (banco de dados).
  - Segurança de acesso.
  - Comunicação.
  - Etc.

O Modelo Conceitual é Estático

- Não podem existir no modelo conceitual referências a operações ou aspectos dinâmicos dos sistemas.
- Então, embora o modelo conceitual seja representando pelo diagrama de classes da UML, o analista não deve ainda adicionar métodos a essas classes.

Elementos do Modelo Conceitual

- **Atributos**: informações alfanuméricas simples, como números, textos, datas, etc.
- **Classes** ou **conceitos**: que são a representação da informação complexa que agrega atributos e que não pode ser descrita meramente por tipos alfanuméricos.
- **Associações**: que consistem em um tipo de informação que liga diferentes conceitos entre si.

Atributos

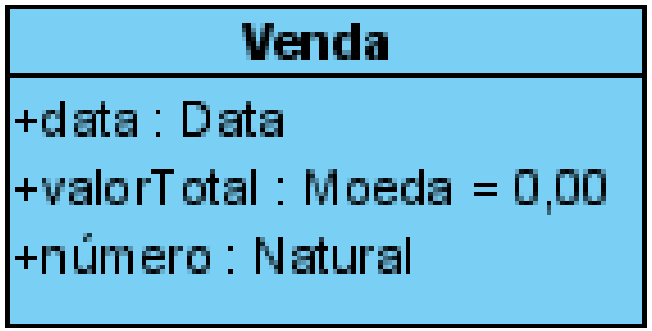
- São os tipos escalares
- NÃO são estruturas de dados como listas, tabelas e *arrays*
- São sempre representados no contexto de uma classe:

Tipagem

- Atributos podem ter tipos clássicos como string, inteiro, data, etc., ou tipos primitivos definidos pelo analista:

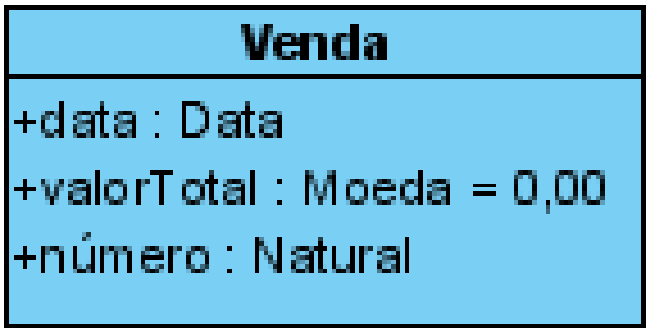
Valores Iniciais

- Atributos podem ser definidos com valores iniciais.
- Valores iniciais são produzidos no atributo no momento que as instâncias da classe correspondente forem criadas



OCL – Object Constraint Language

- Pode ser usada, entre outras coisas, para definir atributos iniciais:

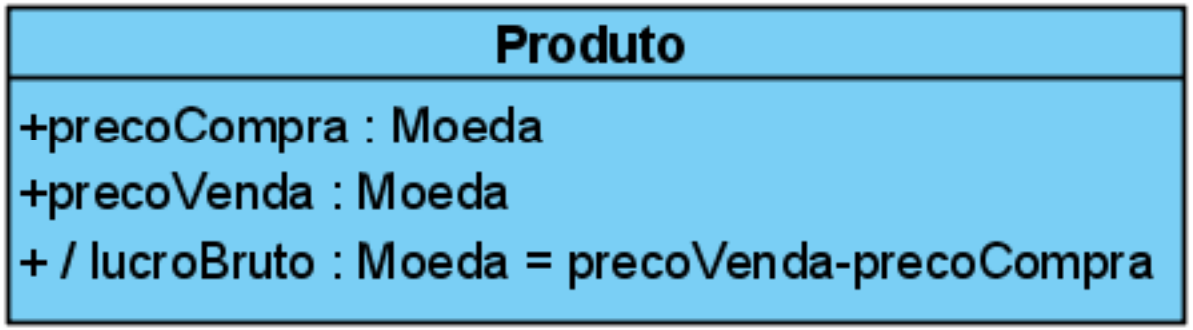


- `Context Venda::valorTotal:Moeda`
  `init: 0,00`

- `Context Venda::valorTotal`
  `init: 0,00`

Atributos Derivados

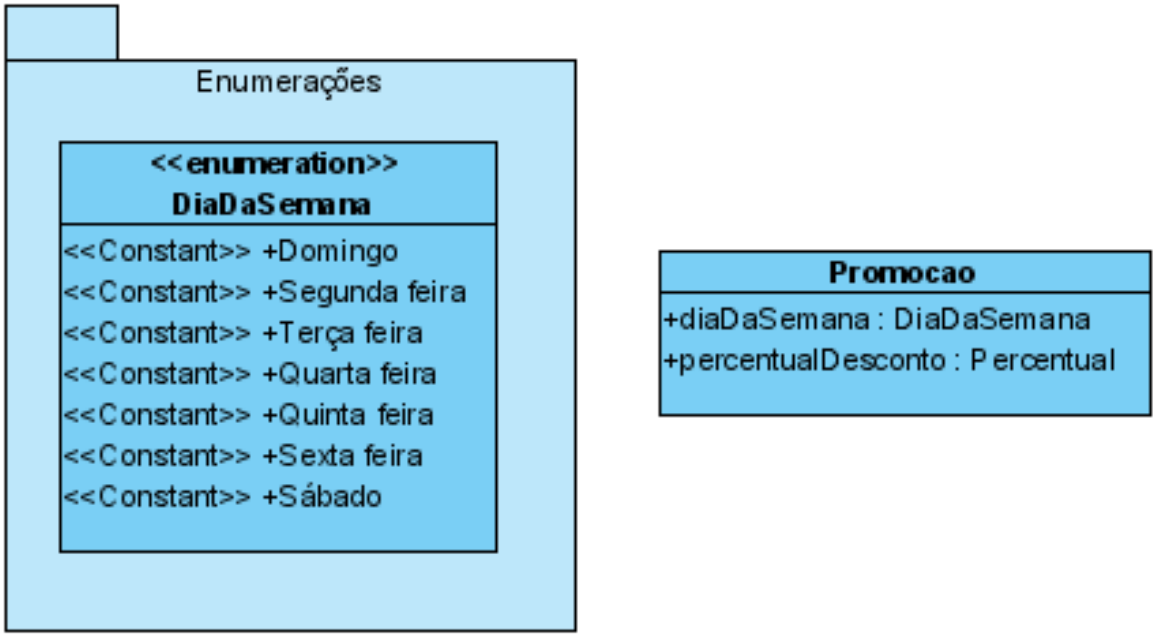
- Não são definidos diretamente, mas calculados



- `Context Produto::lucroBruto`
  `derive:`
  `self.precoVenda - self.precoCompra`

Enumerações

- São um meio termo entre o conceito e o atributo.
- São basicamente *strings* e se comportam como tal, mas há um conjunto predefinido de *strings* válidas que constitui a enumeração.



Características de Enumerações

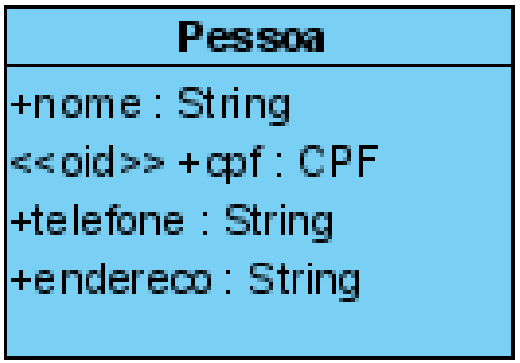
- NÃO podem ter associações com outros elementos.
- NÃO podem ter atributos.
- Se isso acontecer, então não se trata mais de uma enumeração, mas de um conceito complexo.
- Em OCL:
  - DiaDaSemana::Terca_Feira

Conceitos

- Conceitos são mais do que valores alfanuméricos.
- São também mais do que meramente um amontoado de atributos, pois:
  - Eles trazem consigo um significado e
  - Podem estar associados uns com os outros.

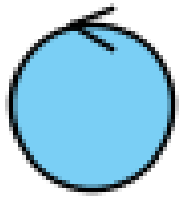
Identificador

- É um atributo que permite que uma instância de um conceito seja diferenciada de outras.
- Estereótipo: <<oid>> (*Object IDentifier*)
- Não existem duas instâncias do mesmo conceito com o mesmo valor para este atributo.



Classe Controladora de Sistema

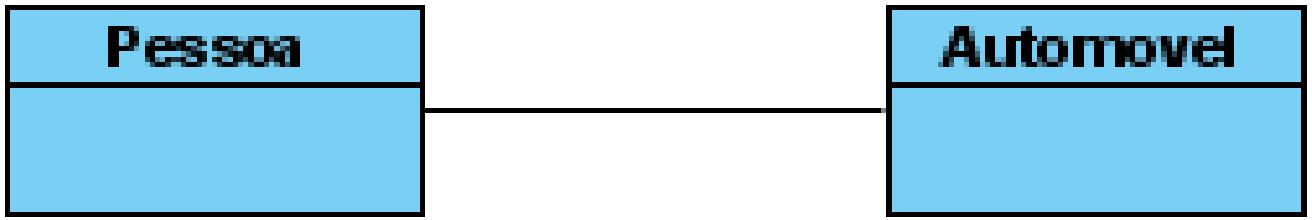
- Representa o Sistema como um todo.
- É o ponto de partida para as conexões das associações.
- Tem uma única instância estereotipada com <<control>> ou na notação de Jacobson:



Livir

Associações

- Relacionam dois ou mais conceitos entre si.



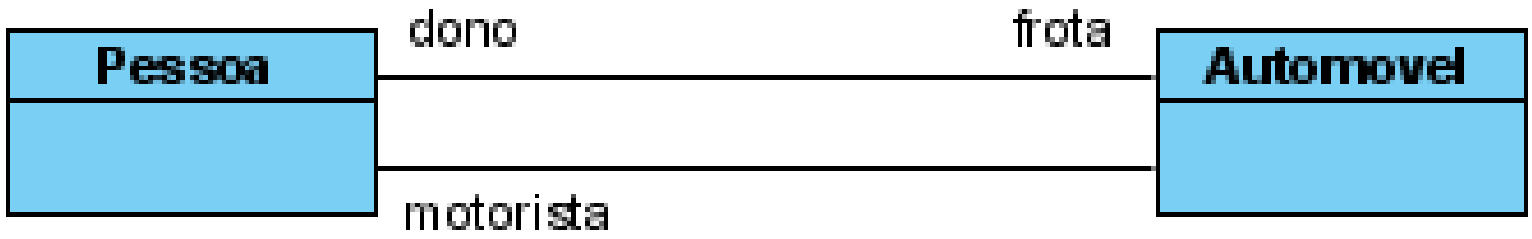
Associação x Operação

- *Associação* é uma relação estática que pode existir entre conceitos complexos, complementando a informação que se tem sobre eles em um determinado instante, ou referenciando informação associativa nova.
- *Operação* é o ato de transformar a informação, fazendo-a passar de um estado para outro, mudando, por exemplo, a configuração das associações, destruindo e/ou criando novas associações ou objetos, ou modificando o valor dos atributos.

Papeis

- Correspondem à função que um lado da associação representa em relação aos objetos do lado oposto.

Múltiplas Associações Demandam Papeis



Multiplicidade de Papel

- Indica quantos objetos podem se associar.
- Sempre há um limite inferior.
- Pode haver um limite superior.

Consideração de Multiplicidade

- O papel é obrigatório ou não?
  - Uma pessoa é obrigada a ter pelo menos um automóvel?
  - Um automóvel deve obrigatoriamente ter um dono?
- A quantidade de instâncias que podem ser associadas através do papel tem um limite conceitual definido?
  - Existe um número máximo ou mínimo de automóveis que uma pessoa pode possuir?

Armadilha da Obrigatoriedade:

- A toda venda corresponde um pagamento.
- Mas isso não torna a associação obrigatória, pois a venda pode existir sem um pagamento.
- Um dia ela possivelmente será paga, mas ela pode existir sem o pagamento por algum tempo.
- Então esse papel *não* é obrigatório para a venda.

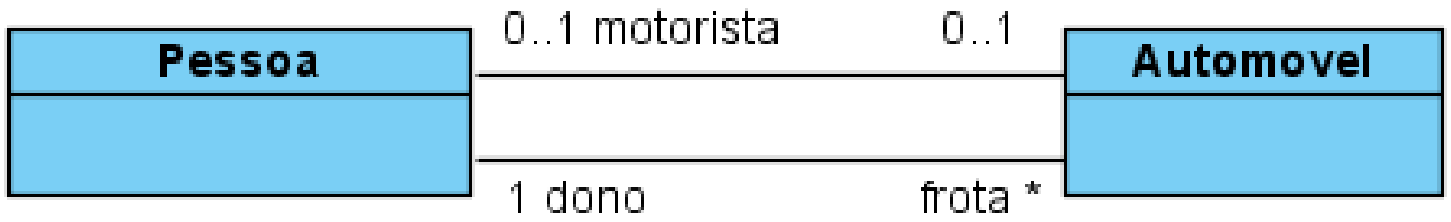
Armadilha do Limite Máximo

- O número máximo de automóveis que uma pessoa pode possuir é o número de automóveis que existe no planeta.
- Mas à medida que outros automóveis venham a ser construídos, esse magnata poderá possuí-los também.
- Embora exista um limite físico, não há um limite lógico para a posse.
- Então o papel deve ser considerado virtualmente sem limite superior.

Exemplos de Multiplicidade

- 1 exatamente um.
- 0..1 zero ou um.
- * de zero a infinito.
- 1..* de um a infinito.
- 2..5 de dois a cinco.
- 2,5 dois ou cinco.
- 2,5..8 dois ou de cinco a oito

Uso no Diagrama



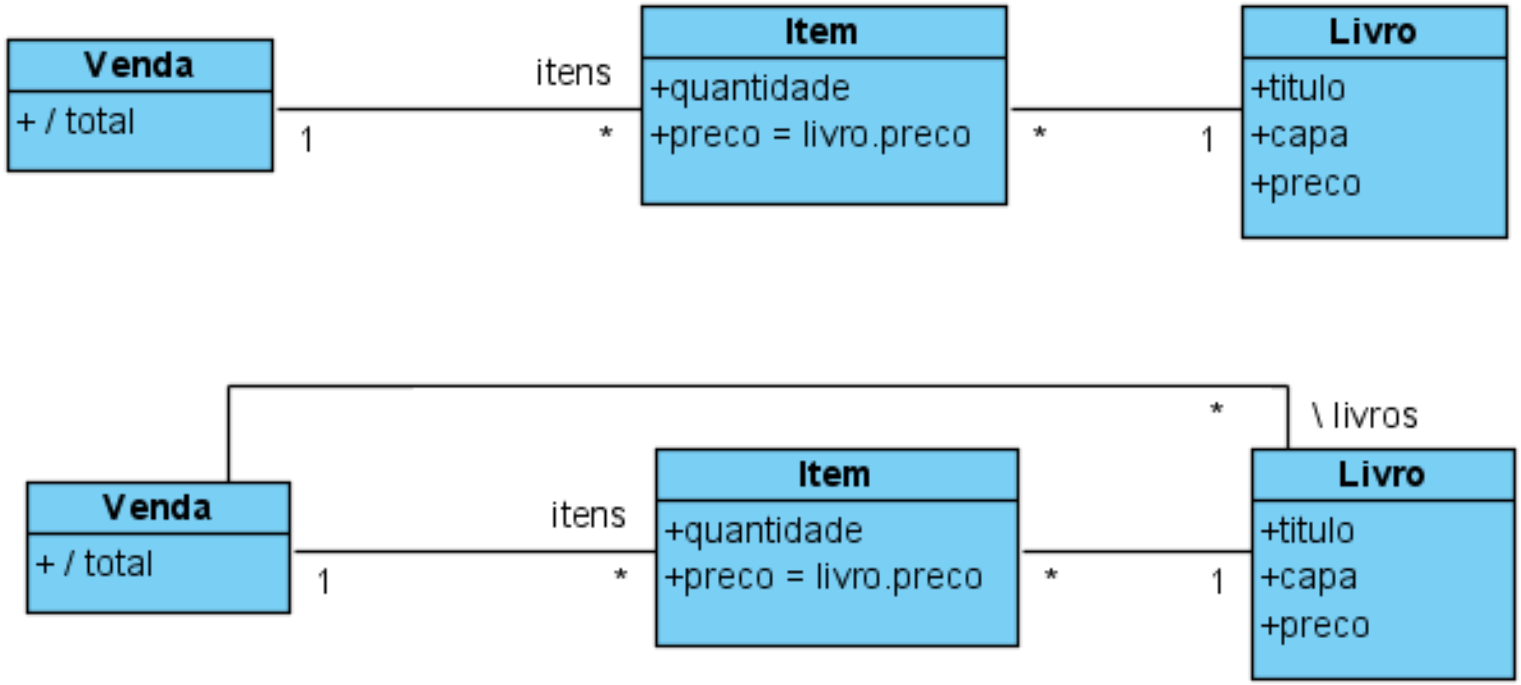
Direção das Associações

- Uma associação, no modelo conceitual, deve ser *não-direcional*.

Associação Derivada

- É calculada a partir de outras.

Exemplo




```
Context Venda::livros
  derive: self.itens.livro
```

Coleções

- Coleções de objetos são representadas nas associações com papel múltiplo, e não como conceitos.

- Errado:

Tipos Abstratos de Dados

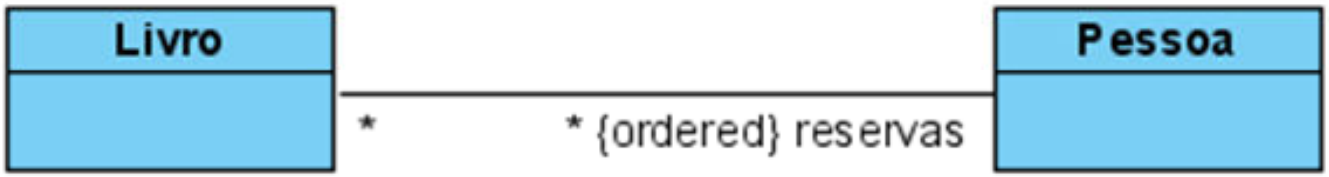
- *
  - Conjunto ou Set
  - Não repete elementos e não tem ordem
- * {ordered}
  - Conjunto Ordenado ou OrderedSet
  - Não repete elementos mas tem ordem
- * {bag}
  - Multiconjunto ou Bag
  - Repete elementos mas não tem ordem
- * {sequence}
  - Lista ou Sequence
  - Repete elementos e tem ordem
- n
  - array

Conjunto

- Um papel de associação *, na falta de maiores detalhes, representa um *conjunto*, ou seja, elementos não se repetem e não há nenhuma ordem definida entre eles.
- A *frota* é um *conjunto* de automóveis de uma pessoa.
  - Se um mesmo automóvel for adicionado a essa associação para a mesma pessoa, o efeito é nulo, pois ele já pertence ao conjunto.

Conjunto Ordenado {ordered}

- Existe ordem, mas os elementos não se repetem



Multiconjunto {bag}

- Elementos podem se repetir, mas a ordem não importa

Lista {sequence}

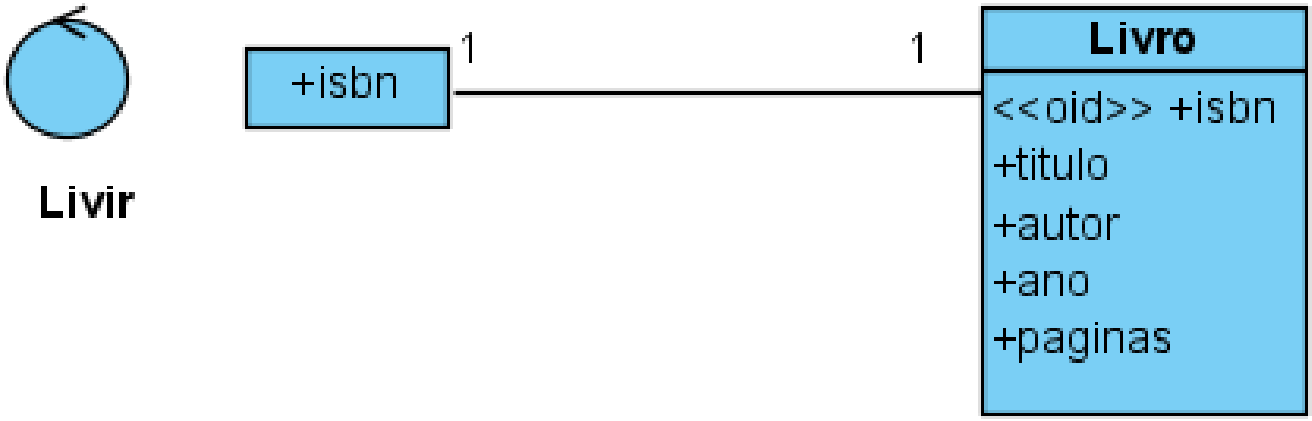
- Há ordem e pode haver repetição

Casos Especiais de Lista

- Pilha {stack}
- Fila {queue}

Mapeamento

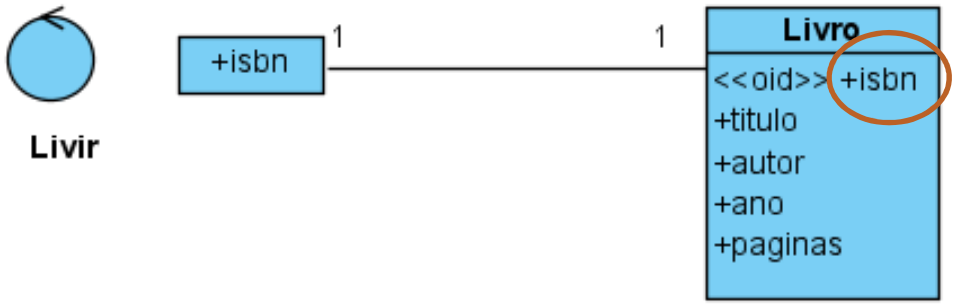
- Associa um valor alfanumérico a um objeto
- Usa-se um *qualificador* na associação



Tipos de Qualificador

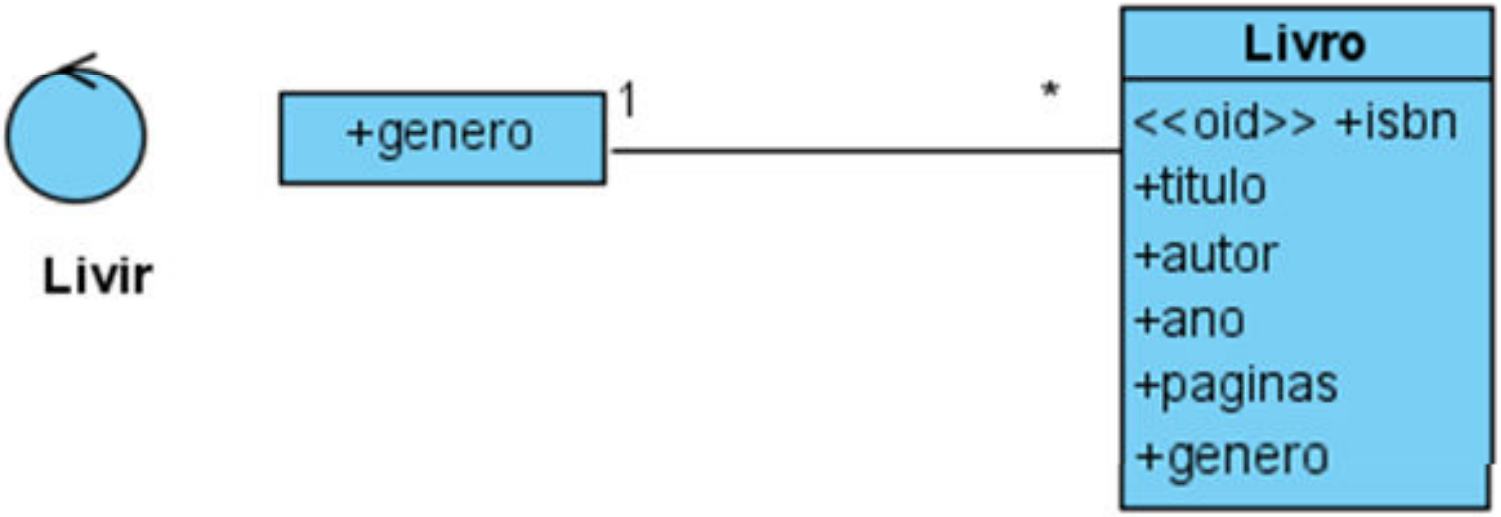
- Interno: é atributo da classe qualificada
- Externo: não é atributo da classe qualificada

- Exemplo de qualificador interno:



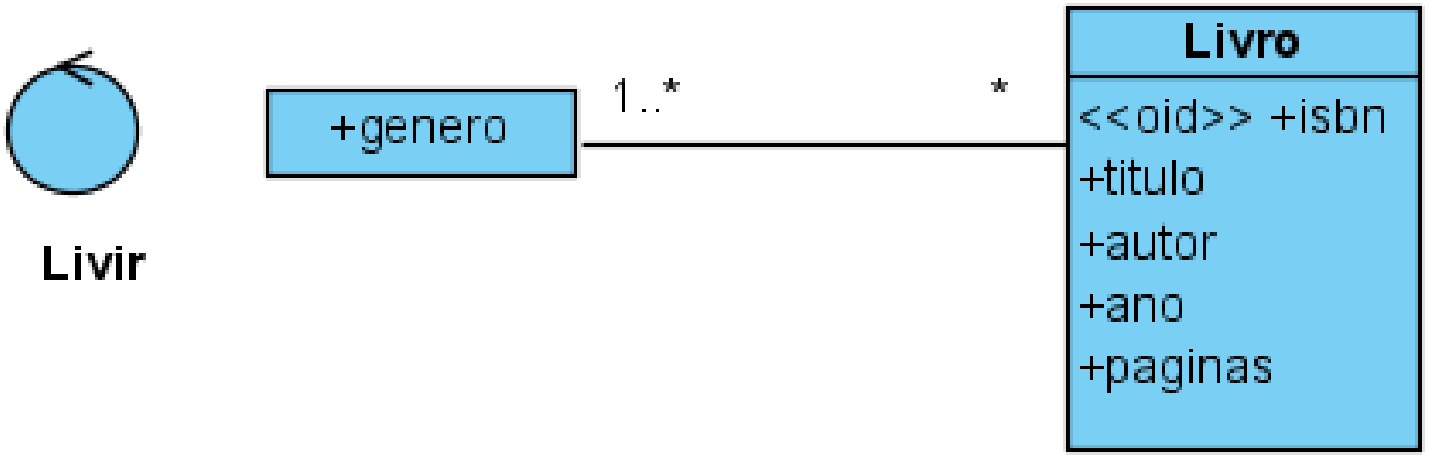
Partição

- Associa um conjunto a cada qualificador



Relação

- Associa um conjunto a um qualificador e cada instância pode ser qualificada várias vezes

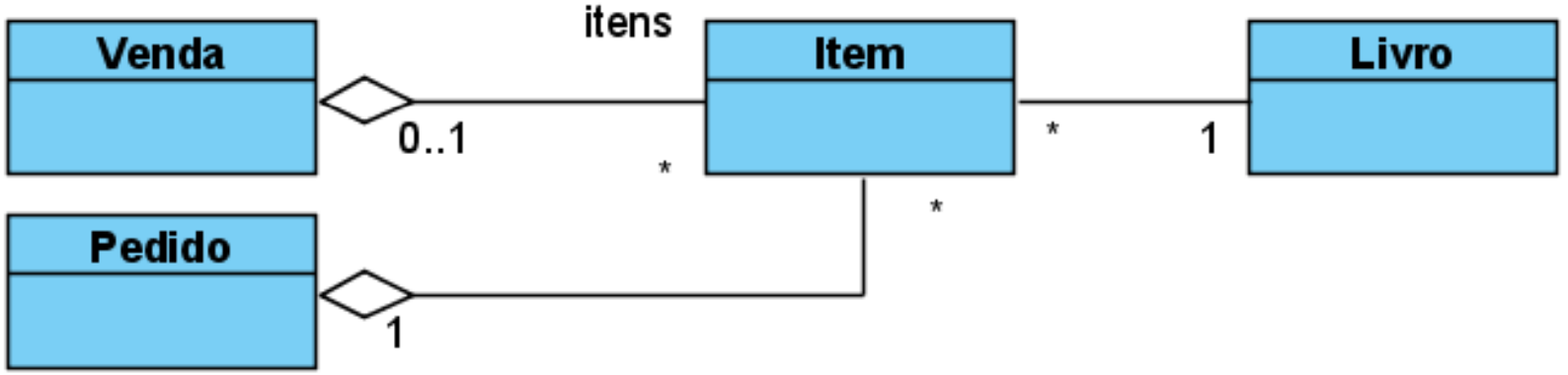


Composição

- Objetos efetivamente FAZEM PARTE de outros

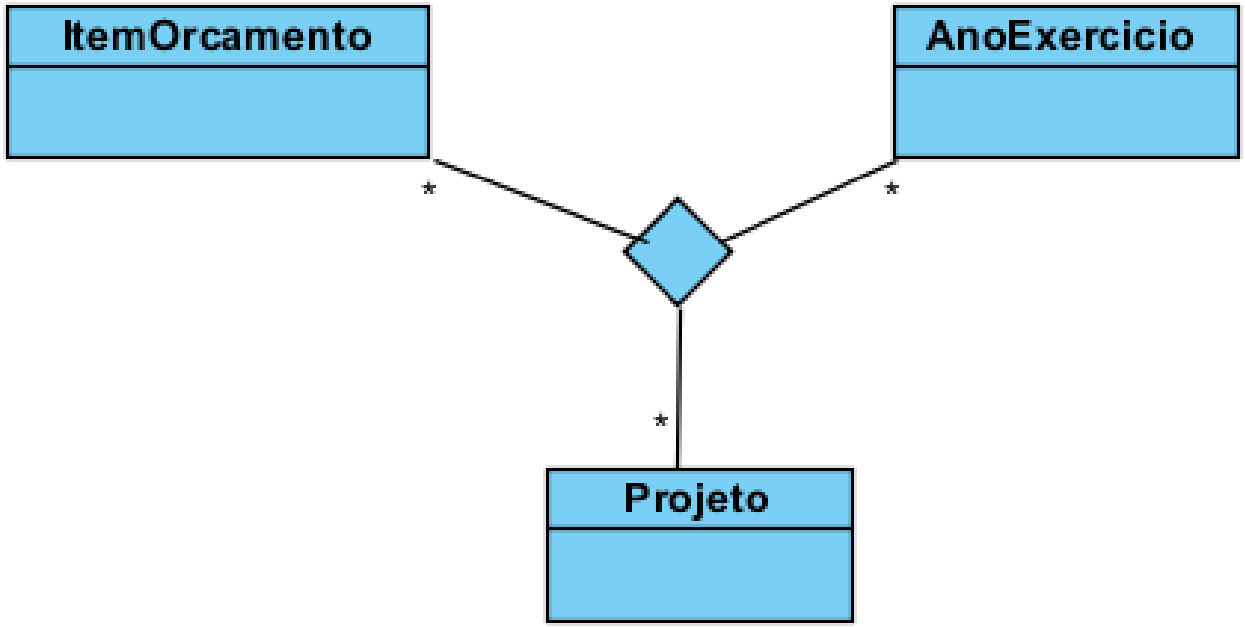
Agregação

- Objetos formam outros, mas podem ser compartilhados



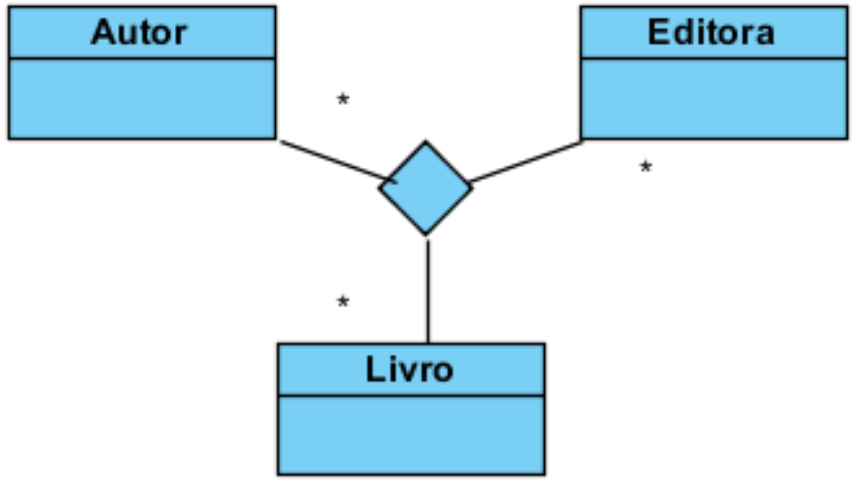
Associações n-árias

- São raras, mas podem acontecer

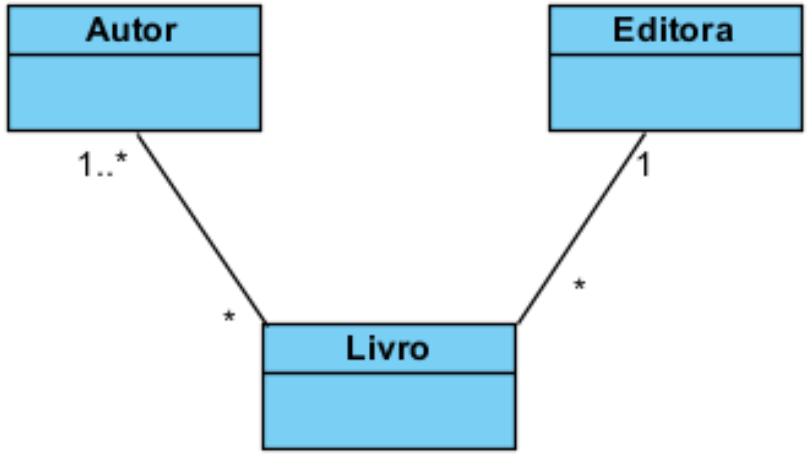


Armadilha das n-árias

- Errado



- Correto

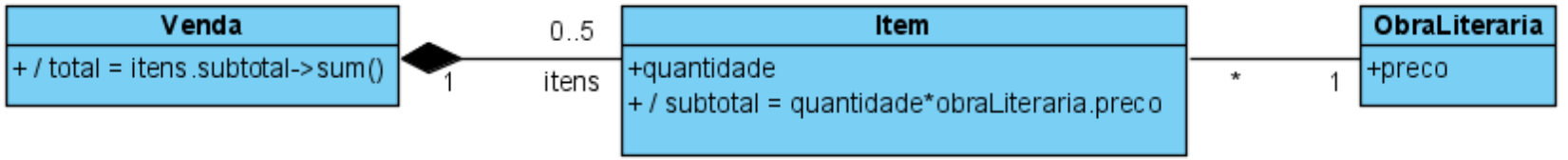


Invariantes

- Existem situações onde a expressividade gráfica do diagrama de classes é insuficiente para representar determinadas regras do modelo conceitual.
- Nestes casos necessita-se fazer uso de invariantes.
- *Invariantes* são restrições sobre as instâncias e classes do modelo

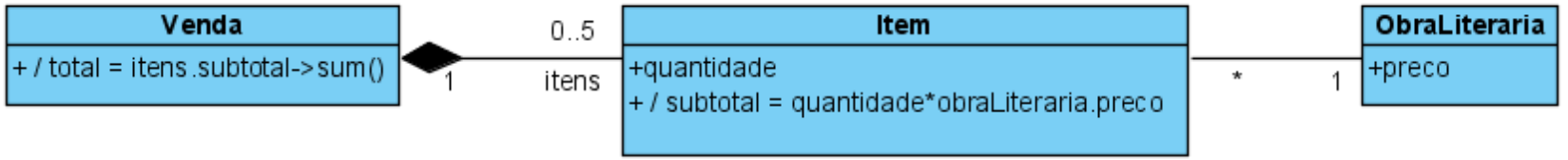
Uma invariante que pode ser representada graficamente

- Uma venda não pode ter mais de 5 itens:



Uma invariante que *não* pode ser representada graficamente

- "nenhuma venda pode ter valor superior a mil reais"

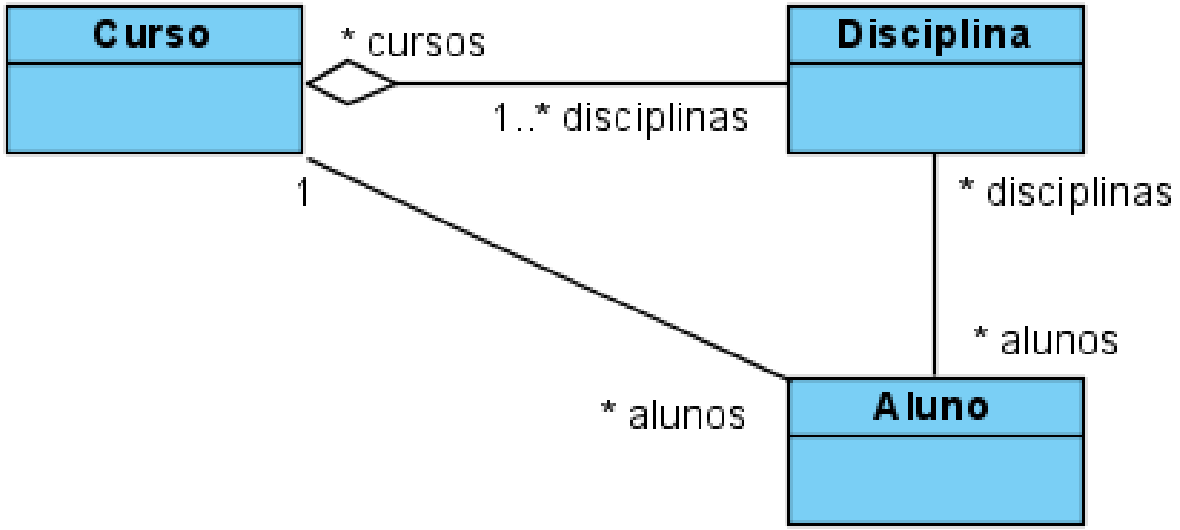



```
Context Venda inv:
  self.total <= 1000,00

Context Venda::total
  derive: self.itens->sum(x|x.subtotal)

Context Item::subtotal
derive: self.quantidade*self.obraLiteraria.preco
```

Uso de invariante para relacionar associações




```
Context Aluno inv:
  self.disciplinas→forAll(d|
    d.cursos→includes(self.curso)
)
```

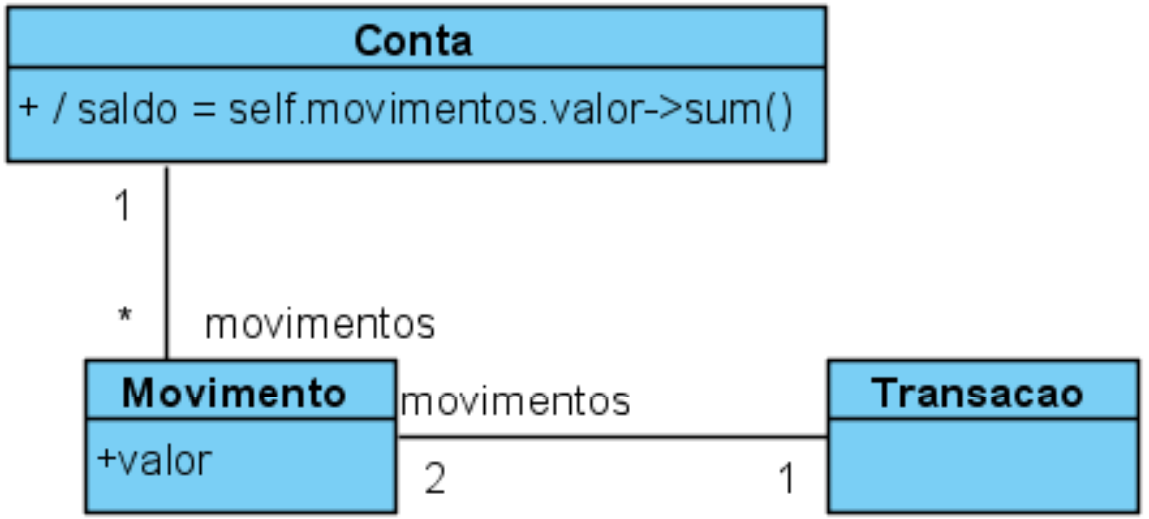
Padrões de Análise

- São um subcaso dos padrões de projeto.
- Aplicam-se ao modelo conceitual.
- São sugestões e boas práticas, não regras.

Padrão "Conta/Transação"

- Como modelar um controle de estoque com:
  - Contas a pagar e a receber
  - Vendas e comissões
  - Entrada e saída de mercadoria
  - Pedidos de compra e venda pendentes
  - Devoluções
  - Etc.
- Com um único padrão?

Conta/Transação




```
Context Transacao inv:
  self.movimentos.valor→sum() = 0
```

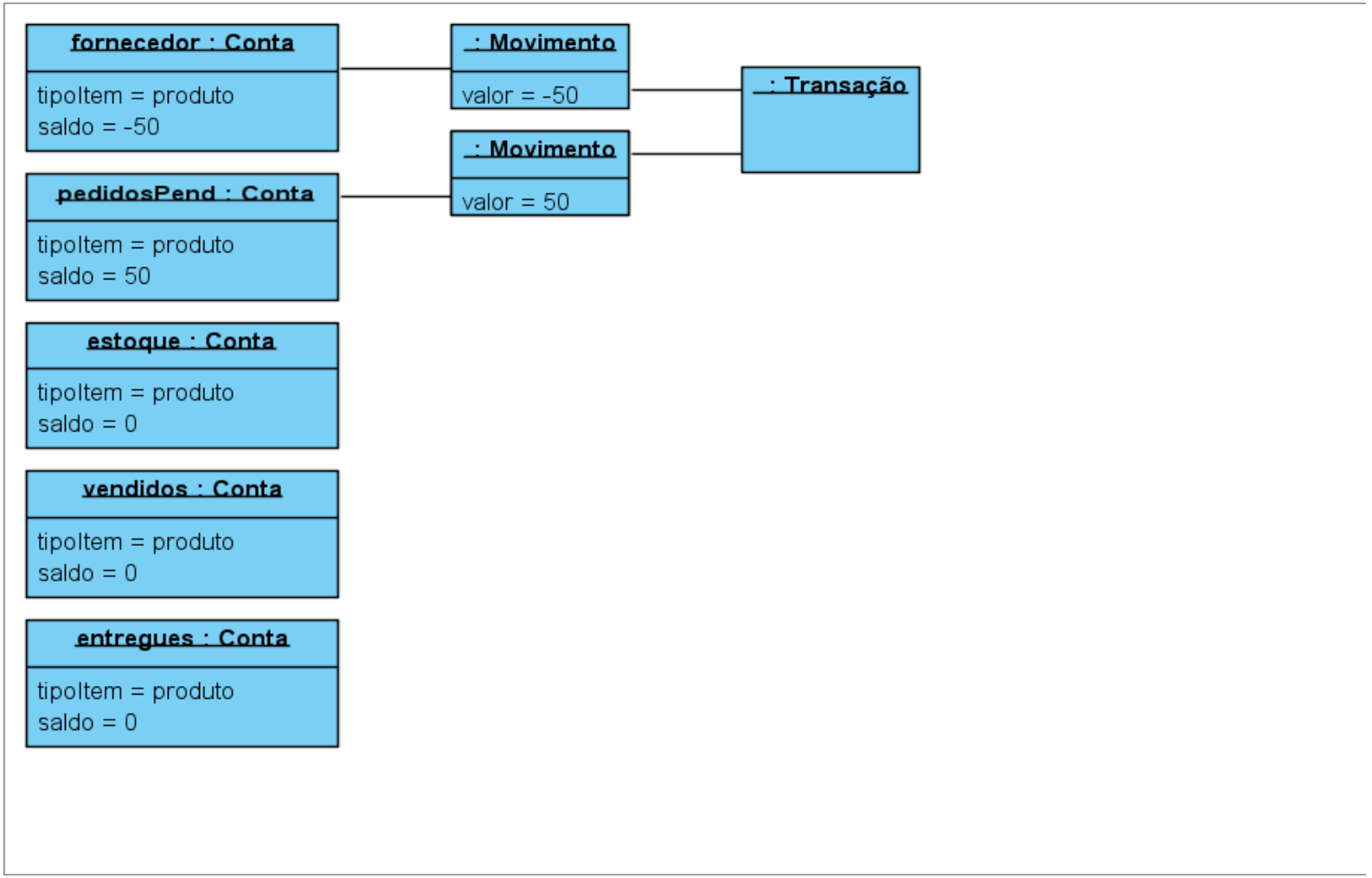
Esse padrão pode usar uma invariante

Todas as facetas do problema são instâncias de Conta

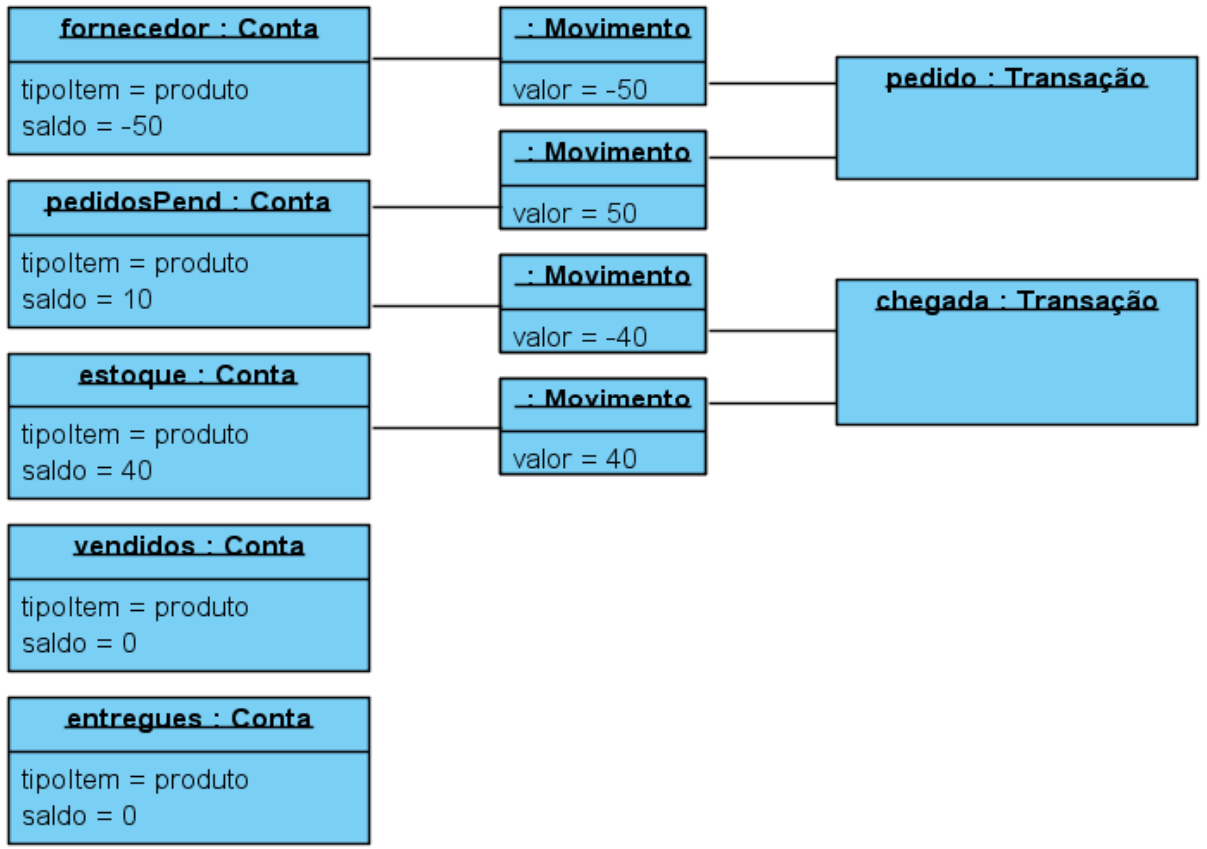
- Conta corrente
- Contas a pagar
- Contas a receber
- Estoque
- Produtos vendidos
- Produtos enviados
- Produtos comprados
- Produtos recebidos
- Produtos devolvidos
- Etc.

Exemplo: situação inicial

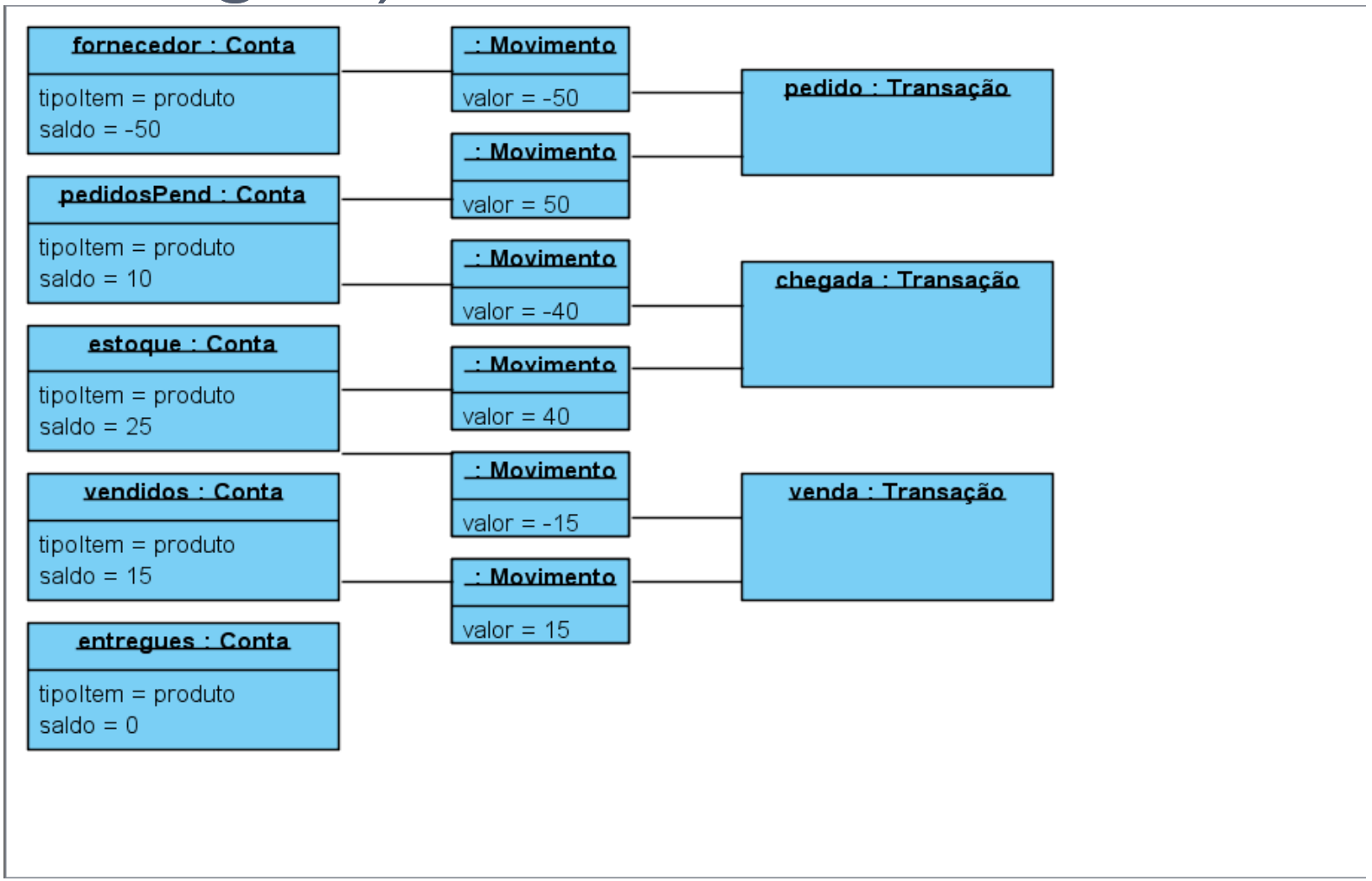
Após fazer um pedido de 50 itens



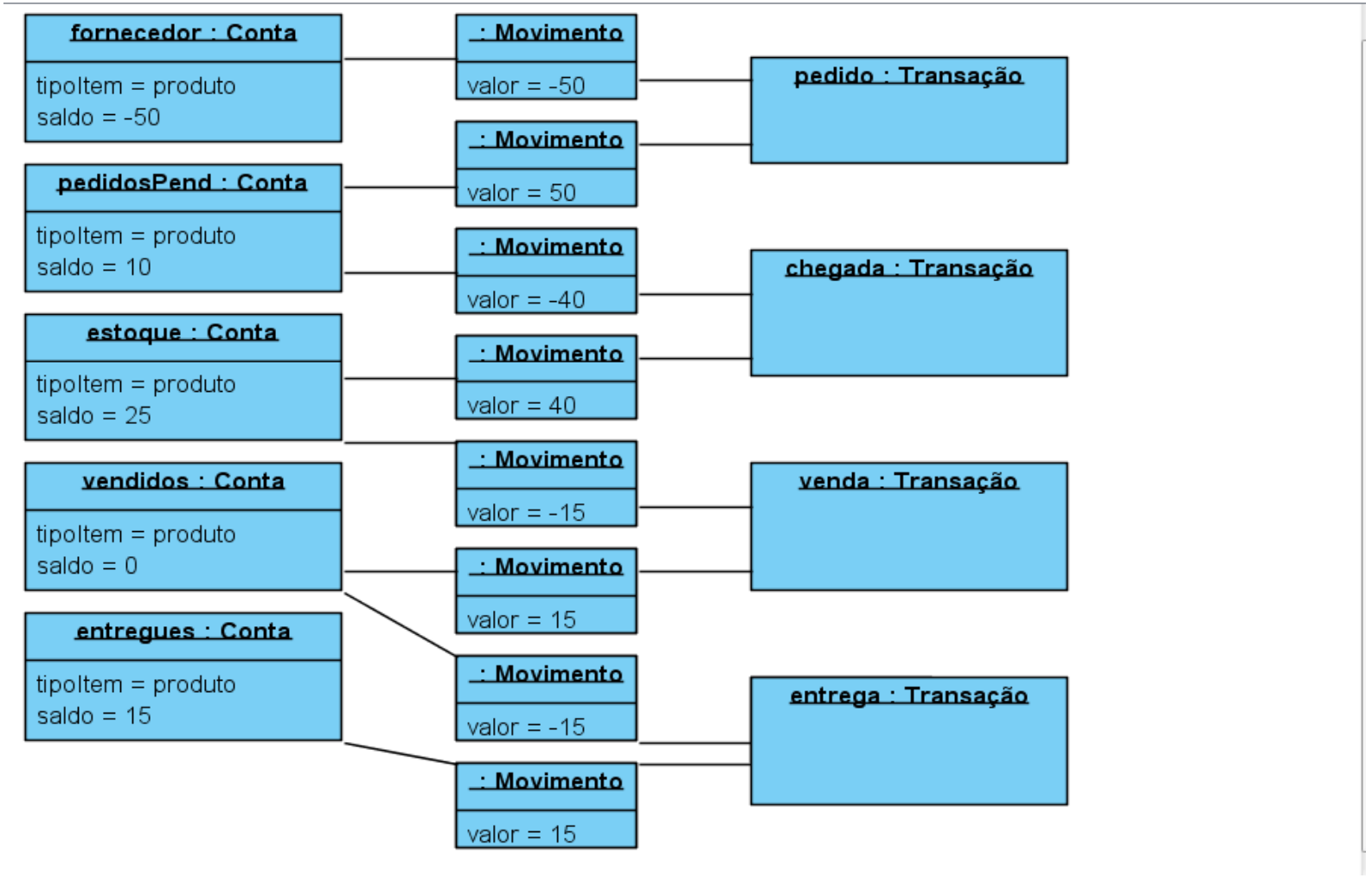
Apenas 40 itens chegaram



15 foram vendidos (mas ainda não entregues)



15 produtos são entregues



Discussão

- Um bom modelo conceitual produz um banco de dados organizado e normalizado.
- Um bom modelo conceitual incorpora regras estruturais que impedem que a informação seja representada de forma inconsistente.

Discussão

- Um bom modelo conceitual vai simplificar o código gerado porque não será necessário fazer várias verificações de consistência que a própria estrutura do modelo já garante.
- O uso de padrões corretos nos casos necessários simplifica o modelo conceitual e torna o sistema mais flexível e, portanto, lhe dá maior qualidade.

Discussão

- Apenas é necessário sempre ter em mente que só vale a pena criar um padrão quando os benefícios deste compensam o esforço de registrar sua existência.