



# *Análise Orientada a Objetos*

## *Modelo Conceitual*

---

Engenharia de Software I (SCE-306)

Profa. Ellen Francine Barbosa

# Processo de Software

DEFINIÇÃO

Análise de Sistema  
Planejamento do Projeto

**Engenharia  
de Requisitos**

CONSTRUÇÃO

Codificação  
Teste

SOFTWARE PRODUTO

MANUTENÇÃO

Entendimento  
Modificação  
Revalidação

- Gerenciamento de Configuração
- Aplicação de Métricas
- Acompanhamento e Controle do Projeto
- Atividades de SQA
- Produção e Preparação de Documentos
- Gerenciamento de Risco

Atividades para  
Garantir a Qualidade



# ***Engenharia de Requisitos***

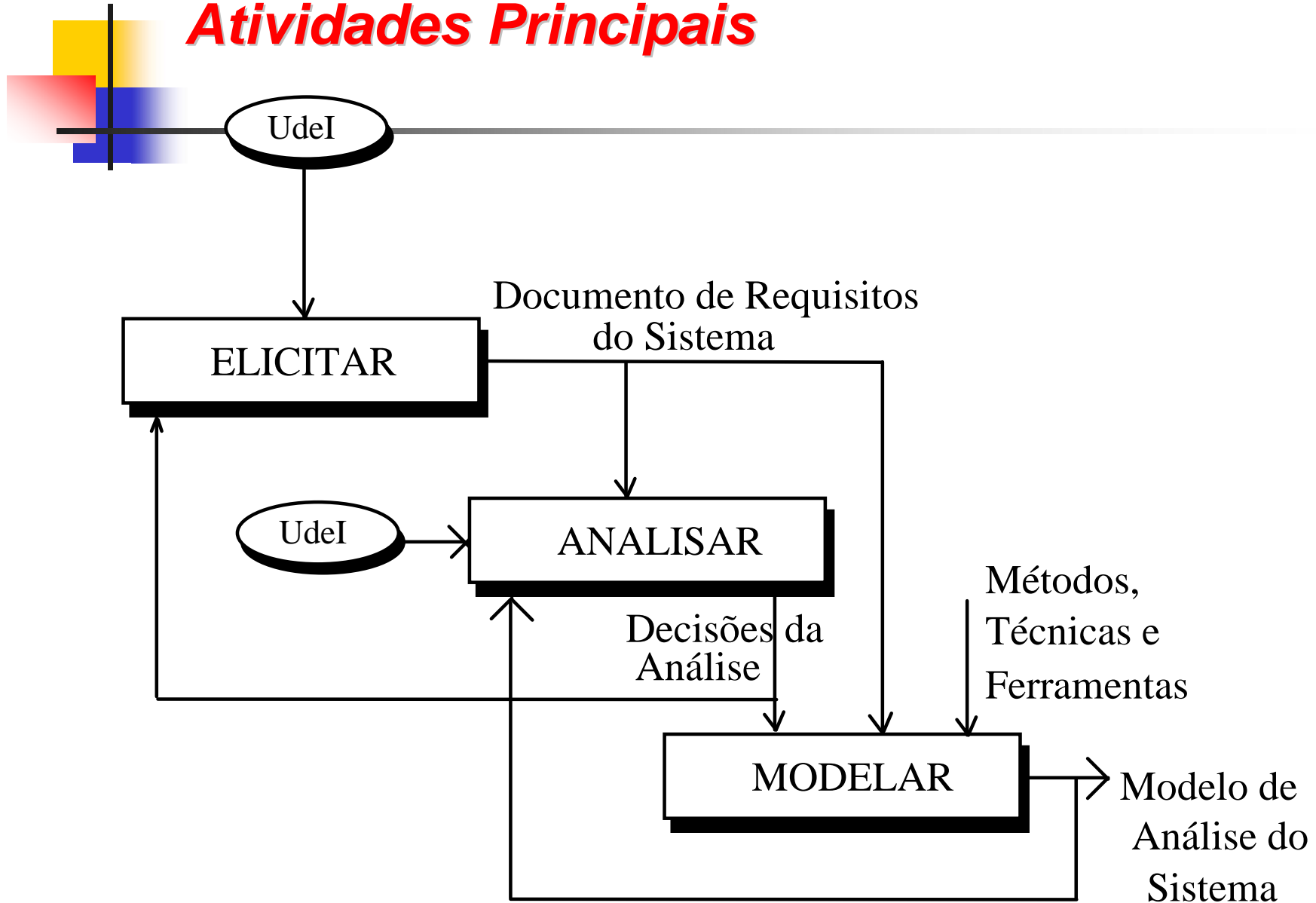
## ***Atividades Principais***

---

- **Elicitação**
- **Análise**
- **Modelagem**

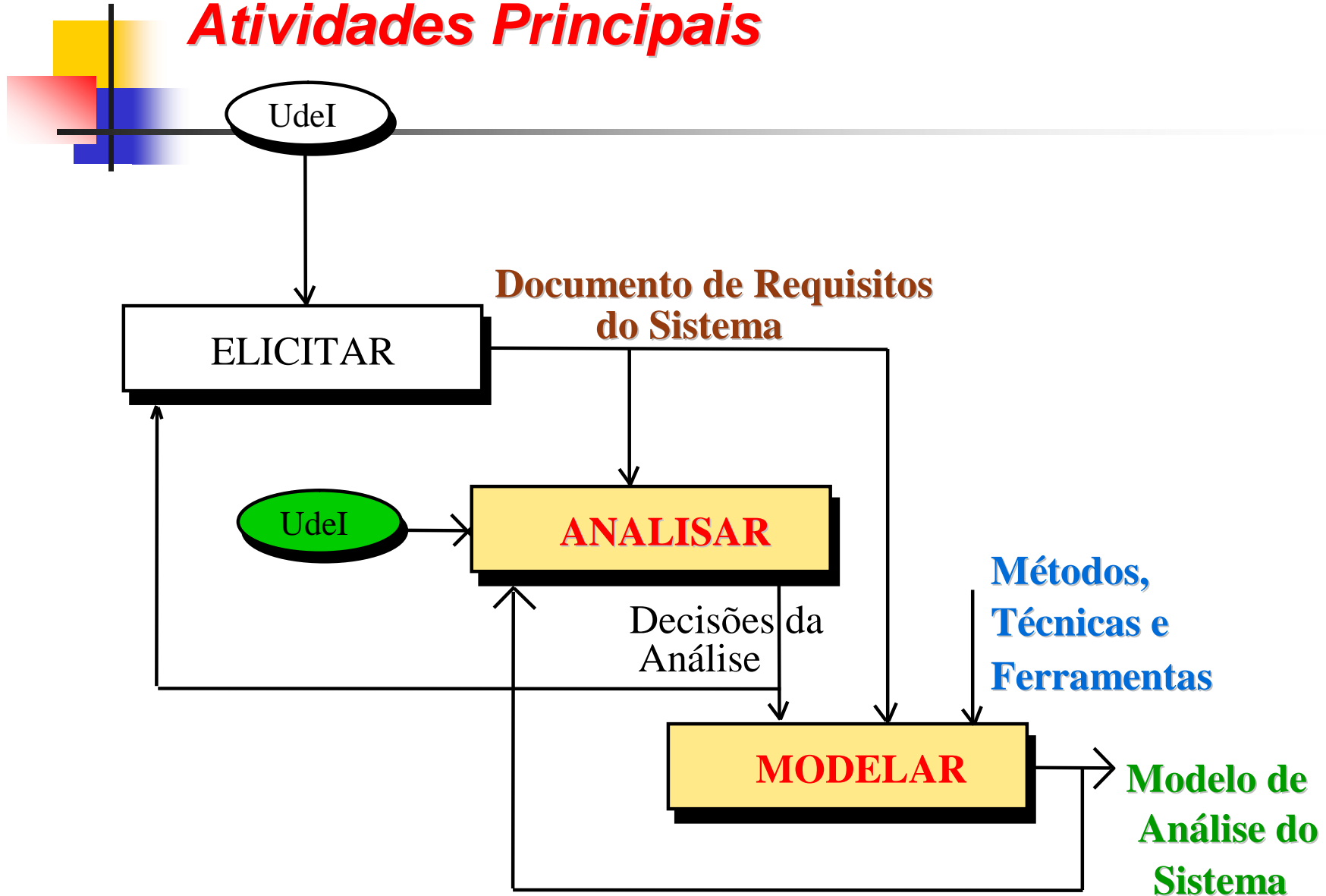
# ***Engenharia de Requisitos***

## ***Atividades Principais***



# ***Engenharia de Requisitos***

## ***Atividades Principais***





# ***Princípios de Análise***

---

- Existem vários **métodos de análise e modelagem** de software.
  - Cada um tem um ponto de vista singular.
  - Todos têm um conjunto fundamental de **princípios**.



# ***Princípios de Análise***

---

## ■ **Princípios Operacionais:**

- O **domínio de informação** de um problema precisa ser representado e entendido.
- As **funções** a serem desenvolvidas pelo software devem ser definidas.
- O **comportamento** do software (como consequência de eventos externos) precisa ser representado.
- Os **modelos** que mostram informação, função e comportamento devem ser **particionados** de modo que revele detalhes de modo hierárquico.
- O processo de análise deve ir da **informação essencial** até o **detalhe de implementação**.



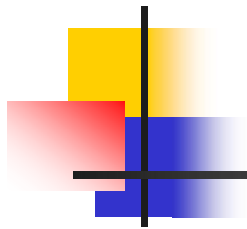
# ***Princípios de Análise***

---

## ■ **Princípios Diretivos:**

- Entenda o problema antes de criar modelos da análise.
- Desenvolva protótipos que permitam ao usuário entender como a interação homem/máquina vai ocorrer.
- Registre a origem e a razão para cada requisito.
- Use múltiplas visões dos requisitos.
- Ordene os requisitos.
- Trabalhe para eliminar a ambigüidade.



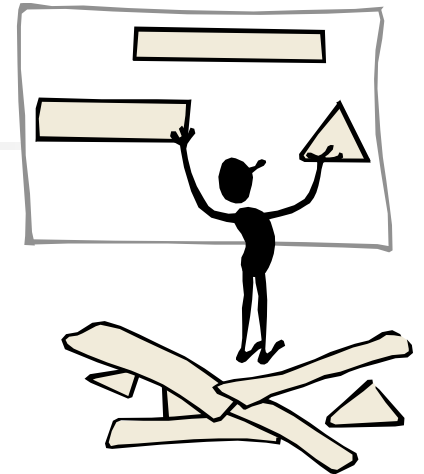


# *Análise*

---

- Combinação de formas **textuais** e **diagramáticas** para representar os requisitos (de dados, função e comportamento) do software.
  - Fácil de entender.
  - Mais direto para revisar.

# Análise



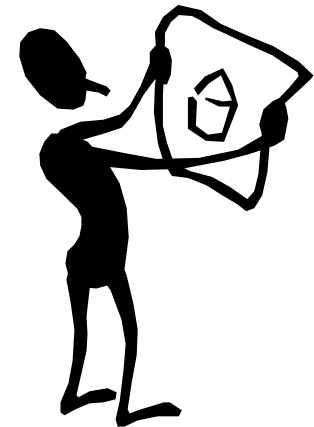
- Objetivos:
  - Descrever **o que** o cliente deseja.
  - Estabelecer a **base** para a criação de um **projeto de software**.
  - Definir um conjunto de **requisitos** que possa ser validado quando o software for construído.



# Análise

---

- O modelo de análise é a primeira representação técnica de um sistema
  - Duas técnicas de modelagem se destacam:
    - Análise Estruturada
    - Análise Orientada a Objeto
  - Técnicas alternativas de análise
    - DSSD
    - Método de Jackson
    - Técnica SADT
    - Técnicas Formais de Especificação





# Análise

---

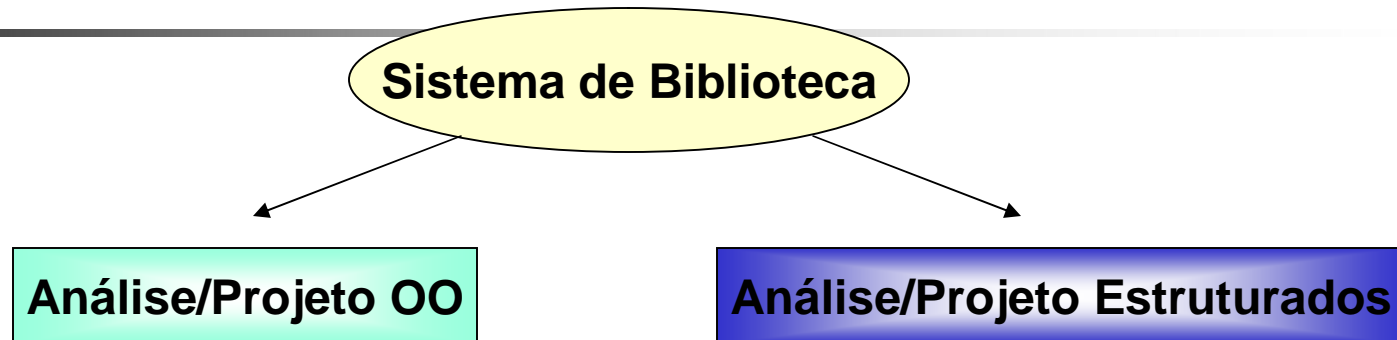
## ■ Análise Estruturada

- Foco nas **funcionalidades** do sistema: funções diferentes atuam sobre os dados de forma desordenada.
- Não existe uma junção lógica entre dados e funções.
  - Os dados são considerados separadamente dos processos que os transformam.
- Faz uso intenso da decomposição funcional.

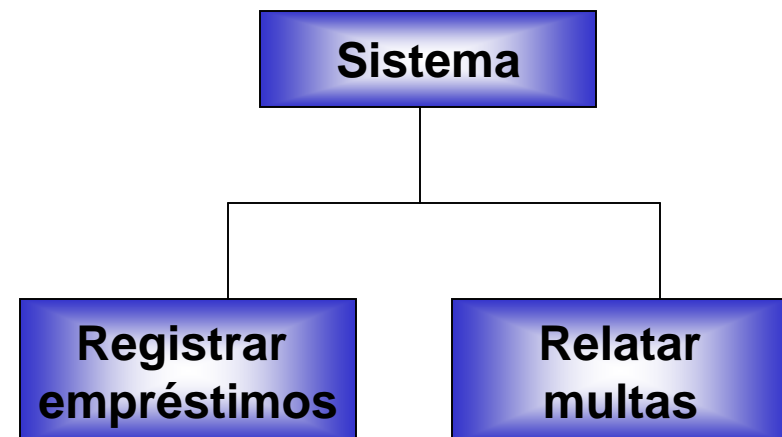
## ■ Análise OO

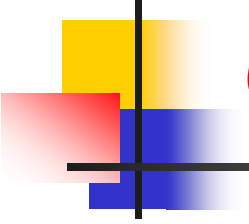
- Permite abstrair de uma forma mais real o “mundo” a ser modelado.
  - Utiliza abstrações do mundo real chamadas de **objetos**.
- Acoplamento entre dados e funcionalidade.

# Análise OO X Análise Estruturada



Decomposição: **objetos** ou **conceitos**.      Decomposição: **funções** ou **processos**.



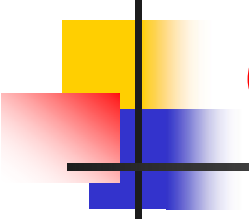


# ***Conceitos de Orientação a Objetos***

---

## ■ **Objeto**

- Componente do mundo real que é mapeado para o domínio do software.
- Representa uma entidade de natureza física ou conceitual.
- Possui limites bem definidos e significado bem conhecido dentro do escopo de uma aplicação.



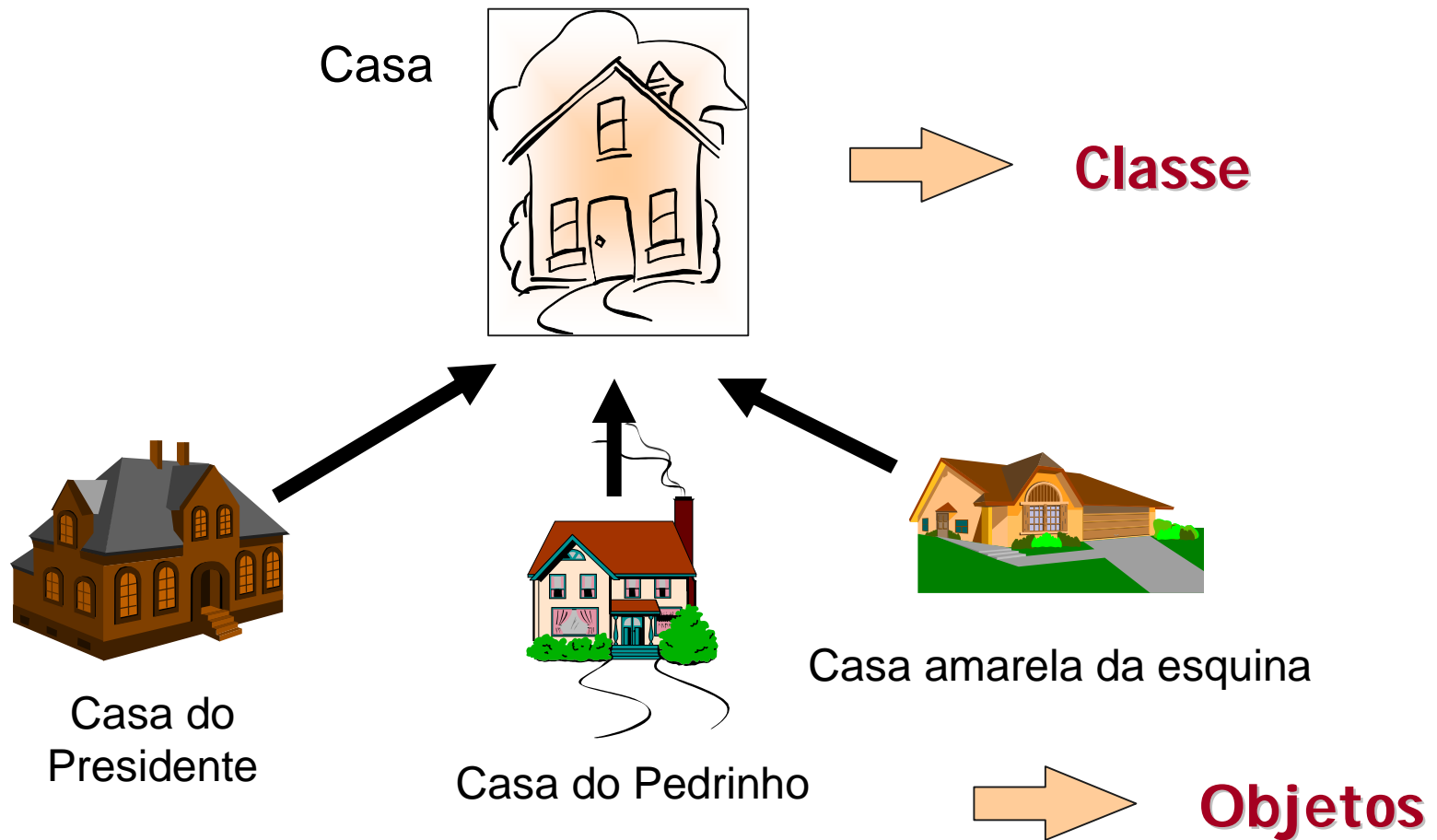
# ***Conceitos de Orientação a Objetos***

---

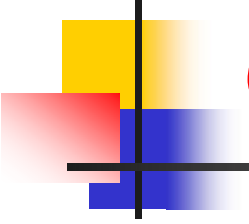
## ■ **Classe**

- Representação de um conjunto de objetos similares.
  - Objetos que compartilham a mesma estrutura de atributos, operações e relacionamentos, dentro de um mesmo contexto.
- Uma classe especifica a estrutura de um objeto sem informar quais serão seus valores.
  - Um objeto corresponde à ocorrência (**instância**) de uma classe num determinado momento.

# Conceitos de Orientação a Objetos



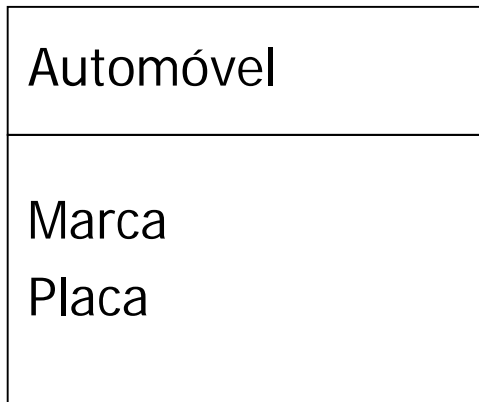




# *Conceitos de Orientação a Objetos*

---

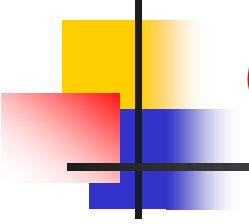
- Classes e Objetos



**Classe**



**Objetos**



# *Conceitos de Orientação a Objetos*

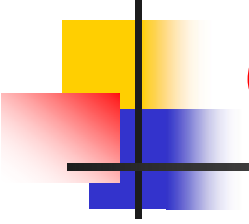
---

- **Objeto**

- Possui características ou propriedades que o definem.

- **Atributos.**

- Os atributos identificam o **estado** de um objeto.



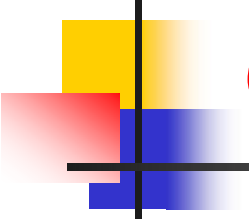
# ***Conceitos de Orientação a Objetos***

---

- **Objeto**

- Possui comportamentos que modificam seu estado (atributos) ou prestam serviços a outros objetos.

- **Operações**



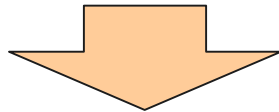
# *Conceitos de Orientação a Objetos*

---

## ■ Objeto

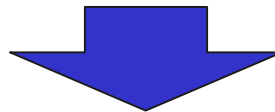
- As operações são implementadas pelos **métodos**.
- Os métodos de uma classe manipulam somente as estruturas de dados daquela classe, ou seja, não podem acessar diretamente os dados de outra classe.
  - Uma classe tem conhecimento dos dados de outra pela solicitação de serviços: **mensagem**.

# Conceitos de Orientação a Objetos



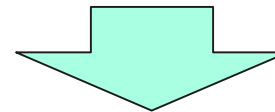
**Classe**

Portas  
Quartos  
Salas  
Localização  
Cozinha  
Telhado

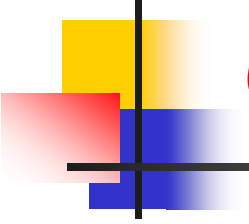


**Atributos**

Reformar  
Limpar  
Pintar  
Mobiliar



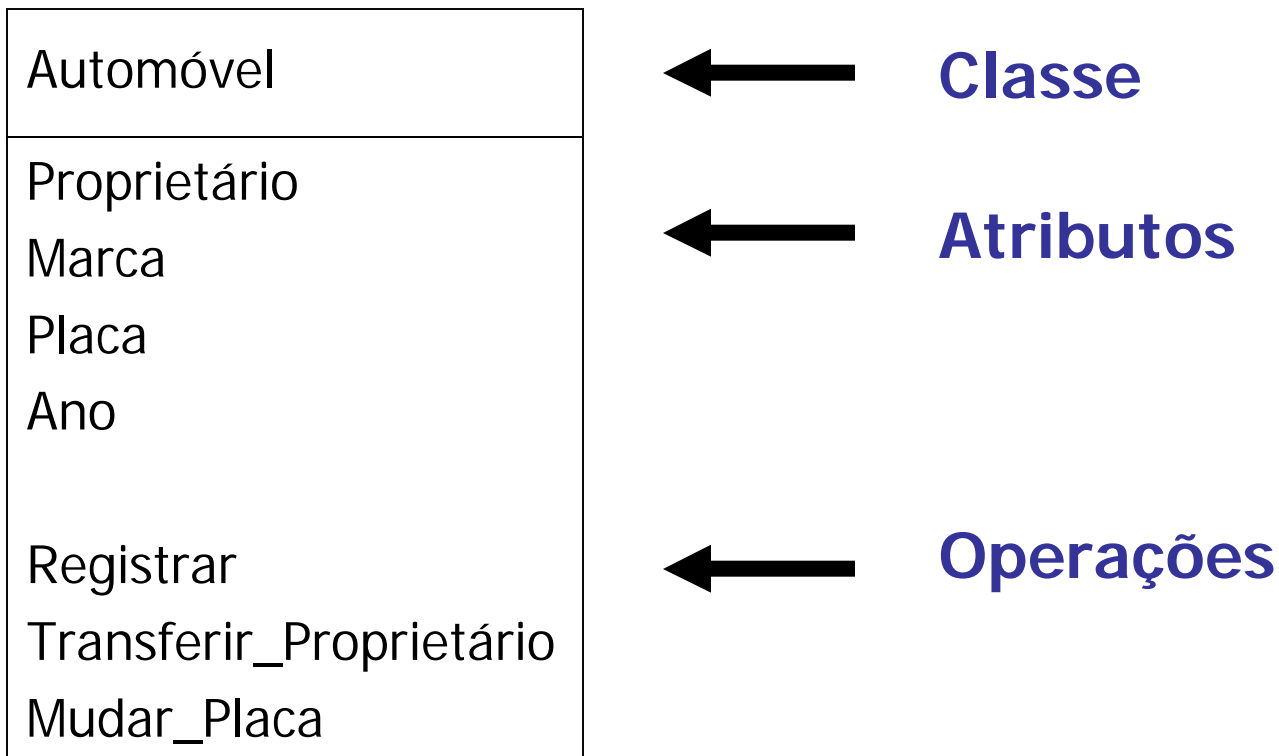
**Operações**

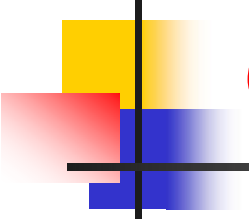


# ***Conceitos de Orientação a Objetos***

---

- **Classe, Atributos e Operações**

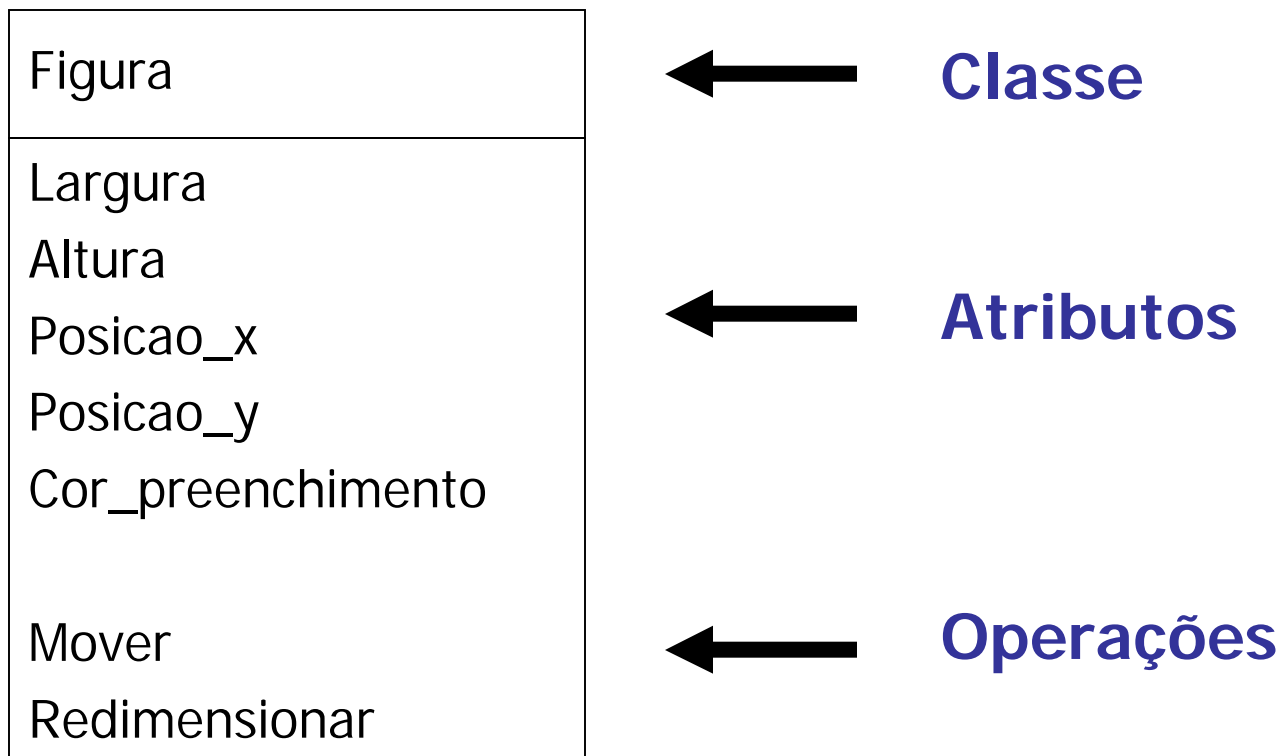


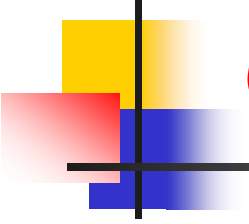


# ***Conceitos de Orientação a Objetos***

---

- Classe, Atributos e Operações



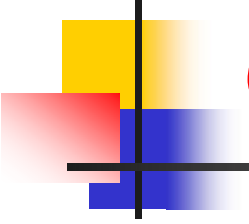


# ***Conceitos de Orientação a Objetos***

---

- Elementos-chave de OO:
  - Encapsulamento
  - Herança
  - Polimorfismo





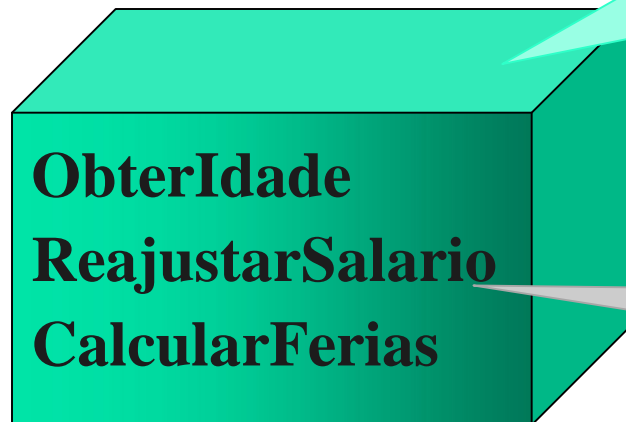
# *Conceitos de Orientação a Objetos*

---

- Encapsulamento
  - Objetos **encapsulam** seus atributos.
    - Os atributos de uma classe são acessíveis apenas pelos métodos da própria classe.
    - Outras classes só podem ter acesso aos atributos de uma classe invocando os métodos públicos.
  - Restringe a visibilidade do objeto mas facilita o **reúso**.
    - Os dados e os métodos são empacotados sob um nome e podem ser reusados como uma especificação ou componente de programa.

# Conceitos de Orientação a Objetos

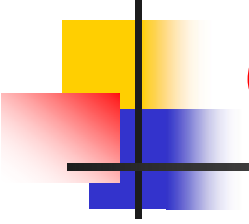
- Encapsulamento



Classe como  
uma caixa preta

Interface da  
classe

**Classe Funcionário**



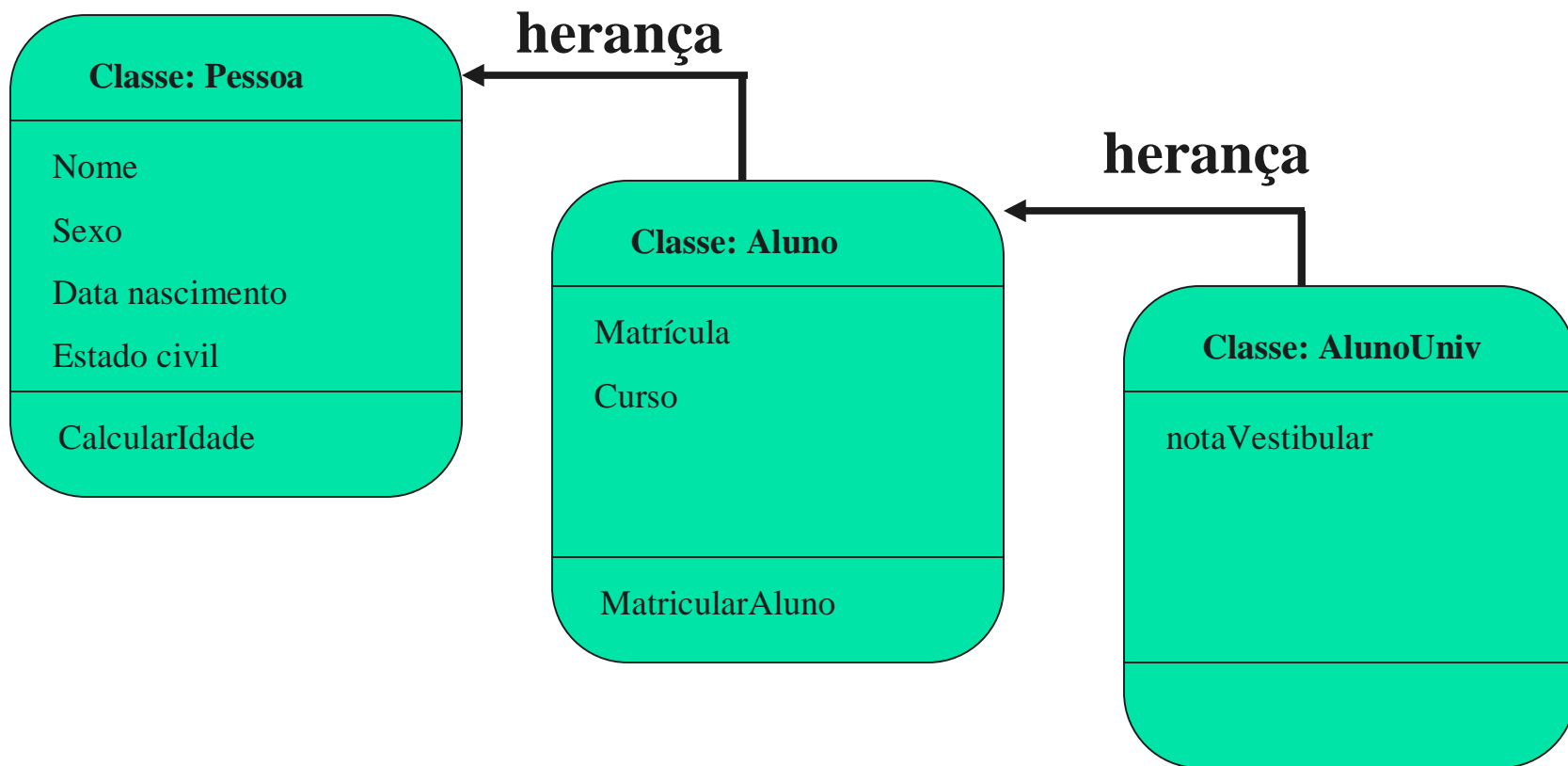
# ***Conceitos de Orientação a Objetos***

---

- Herança
  - Mecanismo pelo qual uma subclasse **herda** todas as propriedades da superclasse e acrescenta suas próprias e exclusivas características.
    - As propriedades da superclasse não precisam ser repetidas em cada subclasse.
  - Possibilita o reuso sem esforço (modificações na superclasse são propagadas nas subclasses relacionadas).

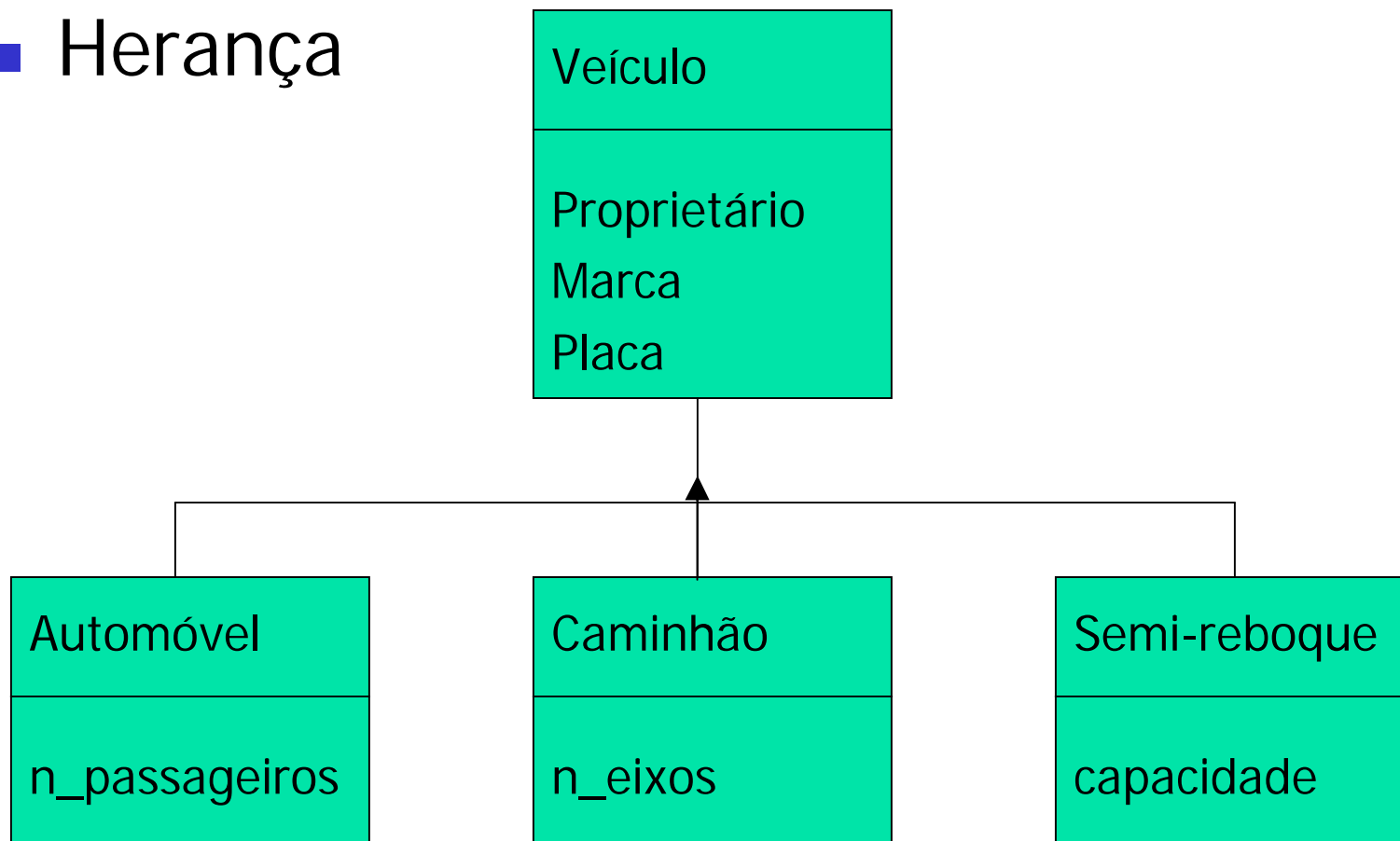
# Conceitos de Orientação a Objetos

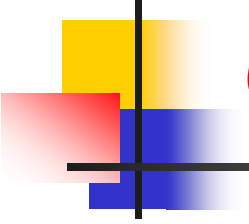
## ■ Herança



# Conceitos da Orientação a Objetos

## ■ Herança





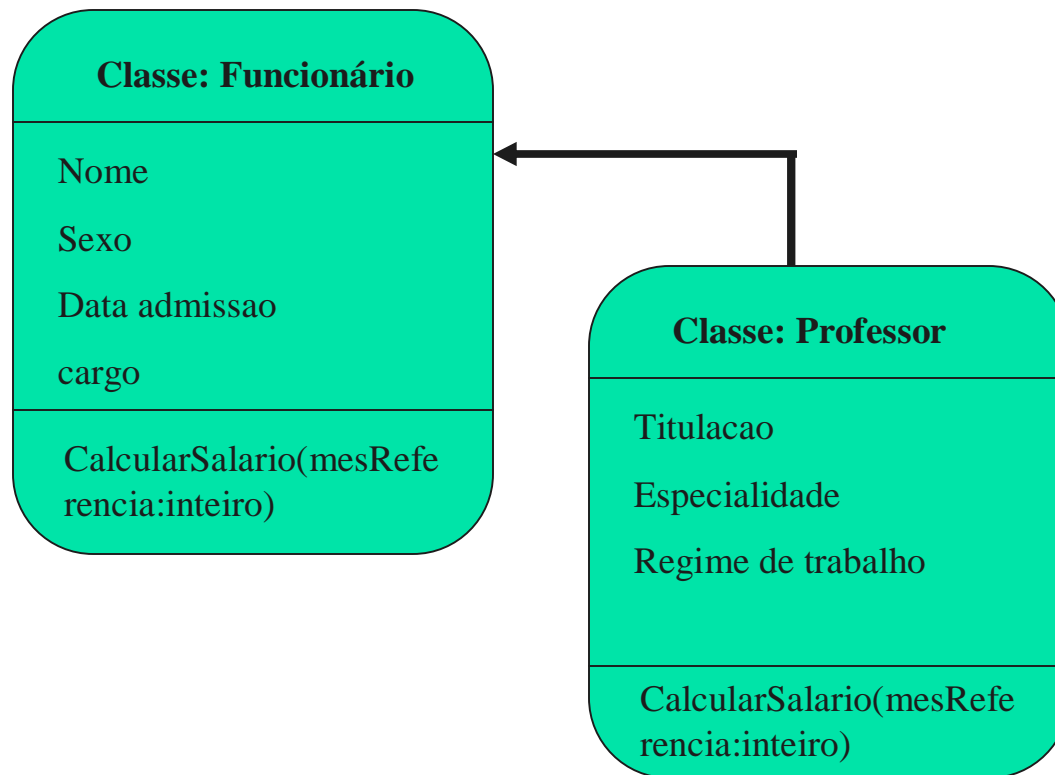
# ***Conceitos de Orientação a Objetos***

---

- Polimorfismo
  - Representa a capacidade de um objeto para assumir diferentes formas.
  - É a propriedade segundo a qual vários métodos podem existir com o mesmo nome.
    - Ao receber uma mensagem para efetuar uma operação, é o objeto quem determina como a operação deve ser efetuada.
    - Permite a criação de várias classes com interfaces idênticas, porém objetos e implementações diferentes.

# Conceitos de Orientação a Objetos

## ■ Polimorfismo



O operador "+"  
pode ser usado com  
inteiros, pontos-  
flutuantes ou  
strings.



# *Análise Orientada a Objetos*

- Métodos de Análise OO

Permitem modelar um problema pela representação das características, tanto **estáticas** quanto **dinâmicas**, das **classes** e seus **relacionamentos** como os principais componentes de modelagem.

- Características de modelo de análise OO
  - Representação de classes e hierarquias.
  - Criação de modelos objeto-relacionamento.
  - Derivação de modelos objeto-comportamento.





# ***Análise Orientada a Objetos***

---

- Fim dos anos 80 e durante os anos 90...
  - Proliferação de métodos de análise e projeto OO.
    - Método de Coad-Yourdon (1990).
    - Método de Wirfs-Brock (1990).
    - Método de Booch (1991).
    - Método de Rumbaugh (1991).
      - OMT – *Object Modeling Technique*.
    - Método de Jacobson (1992).
      - OOSE – *Object-Oriented Software Engineering*.
    - Método Fusion, Coleman (1994).



# ***Análise Orientada a Objetos***

---

- Cada um introduz a sua própria notação, heurística e filosofia.
- Todos usam o mesmo conceito de orientação a objeto.
- Os processos gerais de AOO são bastante semelhantes.



# *Análise Orientada a Objetos*

---

- Uma abordagem unificada.
  - Combinar as melhores características dos métodos individuais de análise e projeto OO em um **método unificado**.
  - UML (Unified Modeling Language), 1997.
    - Expressar um modelo de análise usando uma notação de modelagem, regulada por um conjunto de **regras** sintáticas, semânticas e pragmáticas.



# *Análise Orientada a Objetos*

---

- Passos genéricos para conduzir análise OO:
  1. Deduzir os requisitos do cliente para o sistema.
  2. Identificar cenários ou casos de uso.
  3. Selecionar classes e objetos usando os requisitos básicos como diretriz.
  4. Identificar atributos e operações para cada objeto do sistema.
  5. Definir estruturas e hierarquias que organizem as classes.
  6. Construir um modelo objeto-relacionamento.
  7. Construir um modelo de comportamento de objeto.
  8. Revisar o modelo de análise OO com base nos casos de uso ou cenários.



# ***Análise Orientada a Objetos***

---

- Foco na compreensão dos **requisitos do sistema**.
  - “Fazer a Coisa Certa” – compreender objetivos, conceitos e características do domínio do problema.
- Artefatos ...

## **Artefato da Análise**

## **Questões Respondidas**

Casos de Uso .....	Quais são os processos do domínio?
Modelo Conceitual .....	Quais são os conceitos (objetos)?
Diagramas de Seqüência do Sistema .....	Quais são os eventos e operações?
Contratos de Operação .....	Qual é o comportamento da operação?



# ***Modelo Conceitual***

---

- Consiste em uma representação dos conceitos, pertencentes ao domínio do problema (mundo real).
- É exibido por um conjunto de diagramas de estrutura **estática**, no qual **não** se definem operações.
- Pode mostrar: **conceitos**, **associações** entre conceitos e **atributos** de conceitos.
- Pode ser tratado como um “dicionário visual” das abstrações significativas do domínio.
  - Ajuda a compreender a terminologia e o vocabulário do domínio.



# **Modelo Conceitual**

## **Conceito**

---

- Informal: idéia ou objeto do mundo real no domínio de interesse.
  - Algo digno de ser documentado, de importância para o domínio.
- Formal: Um conceito pode ser considerado em termos de seu:
  - **Símbolo**: palavra ou imagem representando um conceito.
    - Ex.: *Aeronave*
  - **Intenção**: a definição de um conceito.
    - Ex.: *Aeronave representa uma aeronave, ou seja, um meio de transporte aéreo que possui categoria, dimensões, número de lugares, ...*
  - **Extensão**: o conjunto de exemplos (instâncias) ao qual o conceito se aplica.
    - Ex.: *AirBus PT999, Boing747 PX111, ...*



# **Modelo Conceitual**

## **Conceito**

---

- **Símbolo**

- Ex.: *Venda*

- **Intenção**

- Ex.: (o conceito) *Uma venda representa uma transação de compra e possui data e hora.*

- **Extensão**

- Ex.: *Venda1, Venda2, Venda3, ...*

- Como identificar conceitos em um sistema ?





# ***Estratégias para Identificar Conceitos***

---

- **É melhor especificar em excesso um modelo conceitual com muitos conceitos do que subespecificá-lo.**
  - Menos conceitos não implicam em um modelo melhor.
  - Não exclua um conceito só porque sua necessidade não está óbvia nos requisitos.
  - Não exclua um conceito só porque não tem atributos – ele pode possuir um papel de comportamento e não de informação.
- Usar uma Lista de **Categorias de Conceitos**.
- Identificar **Substantivos**.



# ***Categorias de Conceitos***

## ***Exemplos***

---

- **Objetos físicos ou tangíveis:** *TPV, Aeronave*
- **Lugares:** *Loja, Aeroporto*
- **Transações:** *Venda, Pagamento, Reserva*
- **Regras e Políticas:** *PolíticaReembolso*
- **Itens de linha de transação:** *ItemLinhaVendas*
- **Especificações ou Descrições:**  
*EspecificaçãoProduto, ListaVerificação*



# ***Categorias de Conceitos***

## ***Exemplos***

---

- **Papéis desempenhados por pessoas:** *Caixa*
- **Contêineres:** *Depósito, Aeronave*
- **Coisas em um contêiner:** *Item, Passageiro*
- **Catálogos:** *CatálogoProdutos, CatálogoPeças*
- **Organizações:** *DepartamentoVendas*
- **Sistema externo:** *SistemaAutorizaçãoCartãoCrédito*
- ...

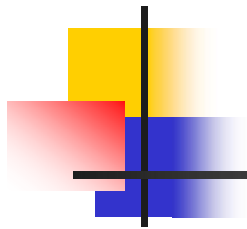


# ***Identificação de Substantivos***

## ***Domínio TPV – caso de uso Comprar Itens***

---

1. Este caso de uso começa quando um **Cliente** chega a um **ponto de pagamento** equipado com um **TPV** com vários **itens** que deseja comprar.
2. O **caixa** registra o **código universal do produto** (UPC) de cada **item**.  
Se houver mais de um exemplar do **item** o **caixa** também pode entrar a **quantidade**.
3. Determina o **preço** do **item** e acrescenta informação sobre o **item** à **transação de vendas** em andamento.  
A **descrição** e o **preço** do **item** corrente são apresentados



# ***Identificação de Substantivos***

---

Lembre-se:

1. Nem todos os substantivos são conceitos – linguagem natural pode ser ambígua.

Ex: substantivos diferentes podem representar o mesmo conceito – (*Consumidor* e *Cliente*)

2. Alguns dos substantivos são **candidatos a conceitos** e outros são **candidatos a atributos**.
3. Alguns **verbos** podem ser transformados em substantivos.



# ***Conceitos Candidatos***

## ***Domínio TPV – caso de uso Comprar Itens***

---

⇒ Ideal: Combinar as estratégias para identificar uma lista de candidatos a conceito.

- TPV
- Caixa
- Cliente
- Item
- Loja
- Venda
- CatálogoProdutos
- EspecificaçãoProduto
- ItemLinhaVenda
- Pagamento
- Gerente

# ***Modelo Conceitual***

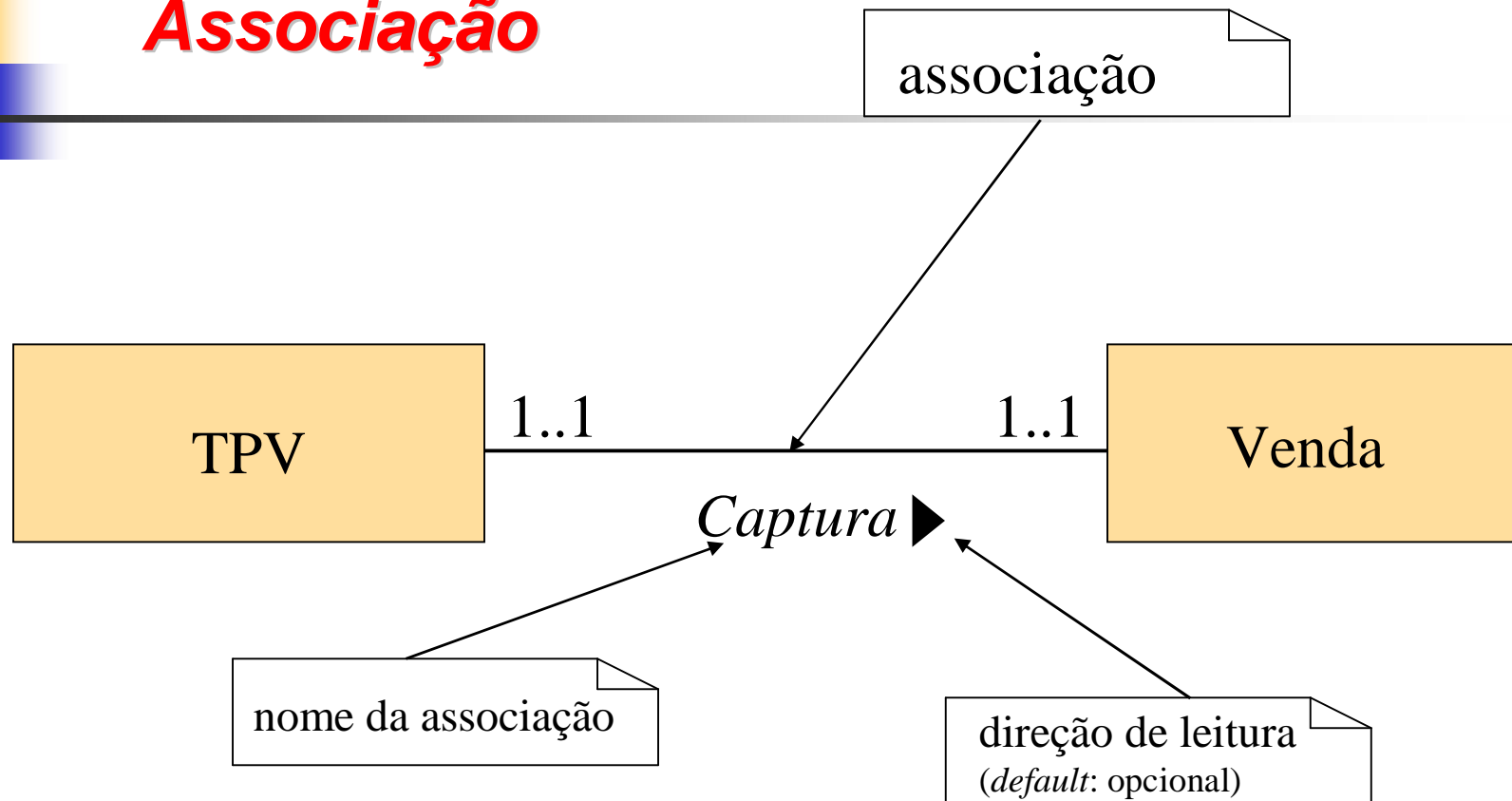
## ***Associação***

---

- **Associação** é um relacionamento entre conceitos.
  - Indica uma conexão com significado e interesse.
- Em UML são descritas como “relacionamentos semânticos entre objetos diferentes”.

# Modelo Conceitual

## Associação



OBS: o símbolo ▷ SOMENTE indica direção de leitura – não tem significado no modelo.



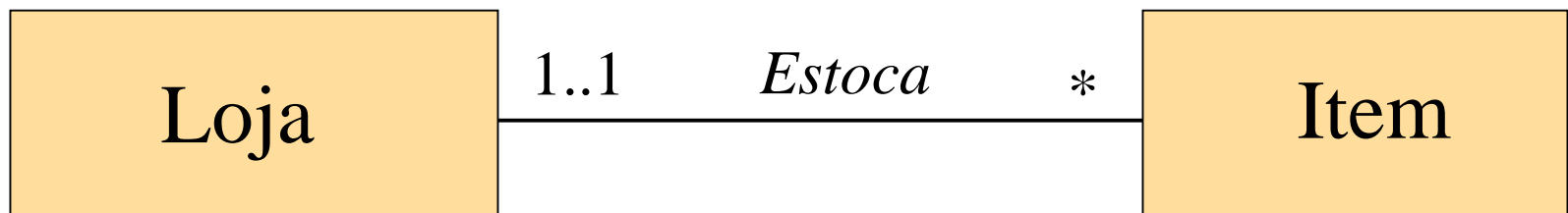
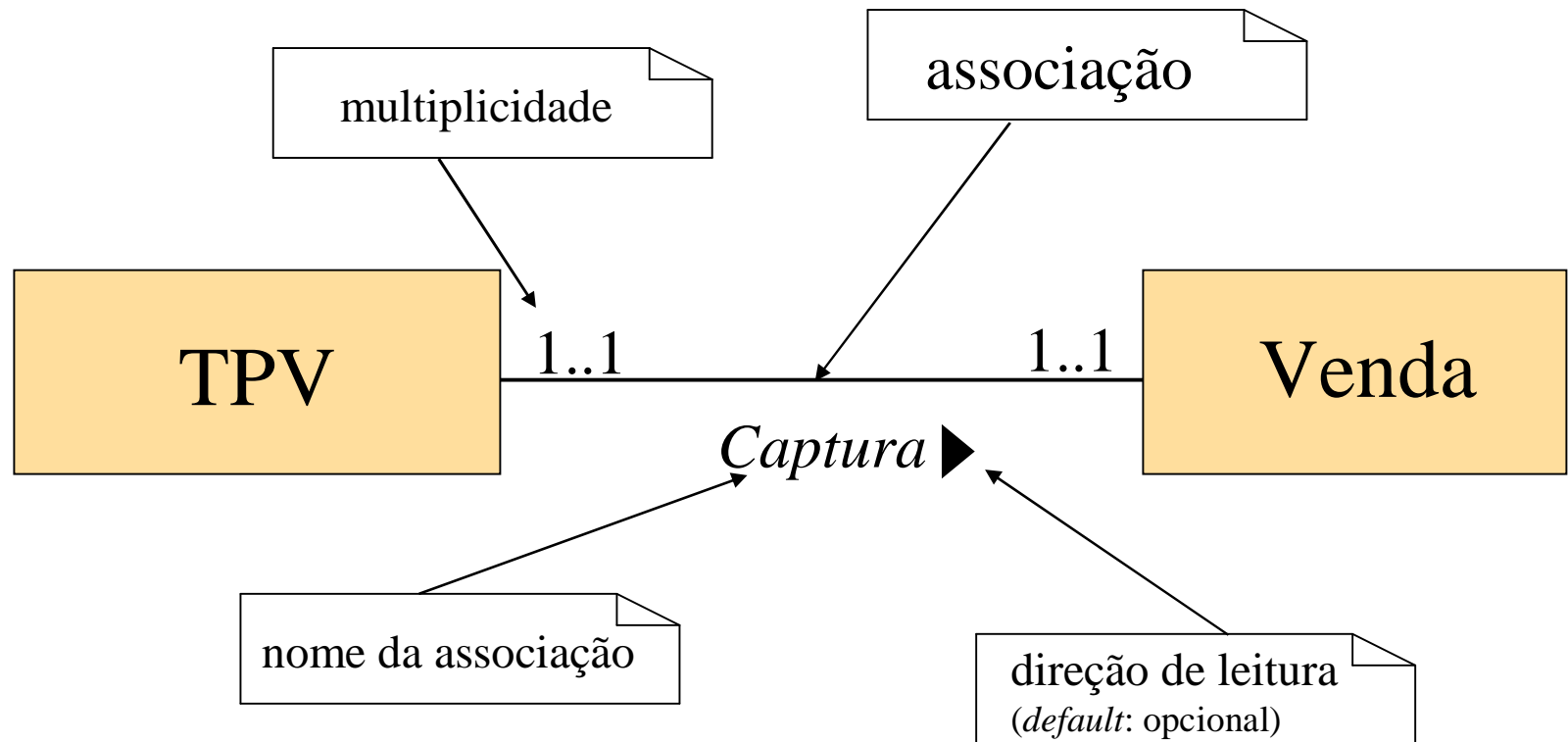


# ***Multiplicidade***

---

- A **multiplicidade** define quantas instâncias de um conceito A podem ser associadas a cada instância do conceito B.

<u>          *</u>	B	zero ou mais – muitos (as)
<u>      1..*</u>	B	um ou mais
<u>      1..40</u>	B	um a quarenta
<u>      5</u>	B	exatamente cinco
<u>      3,5,8</u>	B	exatamente três, cinco ou oito





# ***Critérios para Incluir Associações***

---

- Quando o conhecimento associado necessita ser preservado por algum tempo.
  - “necessário-ser-conhecida” – **requisitos** indicam essa necessidade.
  - Ex: associação entre *Venda* e *Pagamento*
- Evite associações cuja necessidade não é sugerida nos requisitos.
  - Ex: associação entre *Venda* e *Gerente*
- É mais importante identificar conceitos do que associações.
- Excesso de associações pode tornar o modelo conceitual confuso.
- Evite mostrar associações redundantes.



# Associações Comuns

---

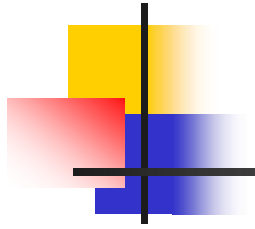
- **A é uma parte física de B**
  - *Gaveta – TPV*
  - *Asa – Aeronave*
- **A é uma parte lógica de B**
  - *ItemLinhaVenda – Venda*
  - *PernaVôo (Flight Leg) – RotaVôo*
- **A está fisicamente contida em/sobre B**
  - *Item – Prateleira*
  - *Passageiro – Aeronave*
- **A está logicamente contida em B**
  - *DescriçãoItem – Catálogo*
  - *Vôo – ProgramaçãoVôo*



# Associações Comuns

---

- **A é registrada em B**
  - *Venda – TPV*
  - *Reserva – ManifestoVôo*
- **A é uma descrição para B**
  - *DescriçãoItem – Item*
  - *DescriçãoVôo – Vôo*
- **A é um item de linha de uma transação ou relatório B**
  - *ItemLinhaVenda – Venda*
  - *ServiçoManutenção – LogManutenção*
- **A é uma transação relacionada a outra transação B**
  - *Pagamento – Venda*
  - *Reserva – Cancelamento*
- ...



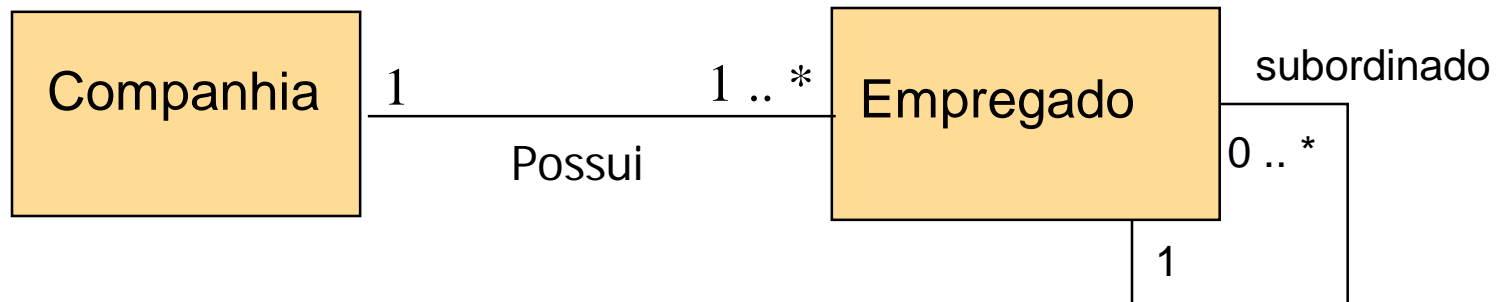
# ***Associações com Papéis***

---

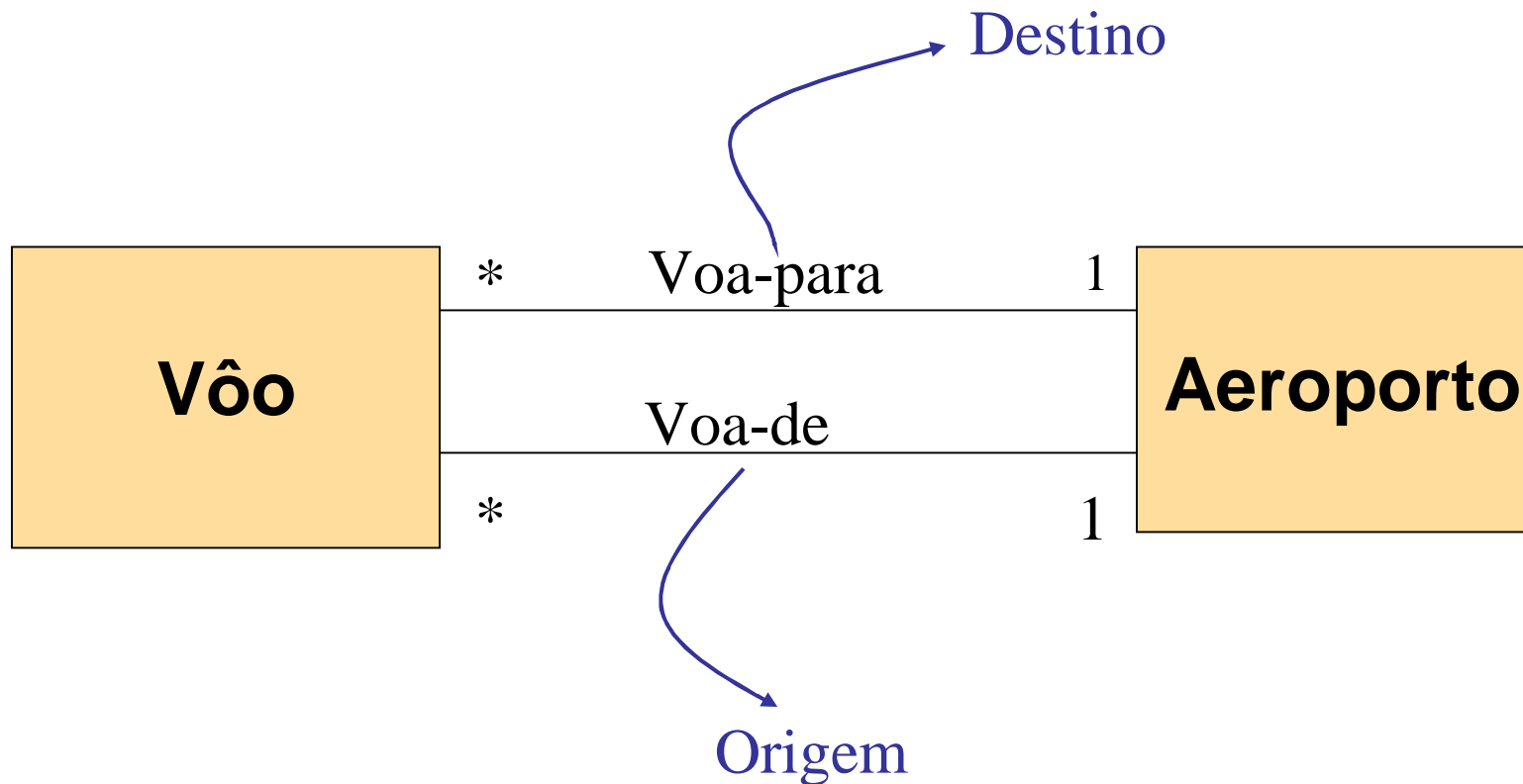
- Cada extremo de uma associação é chamado de **papel**.
- Os papéis podem ter, opcionalmente, as seguintes propriedades:
  - Nome
  - Expressão de multiplicidade
  - Navegabilidade

# Associações com Papéis

- Nomes de papéis são necessários, principalmente, para associação entre dois objetos de **mesma classe**.



# ***Associações Múltiplas entre Conceitos***



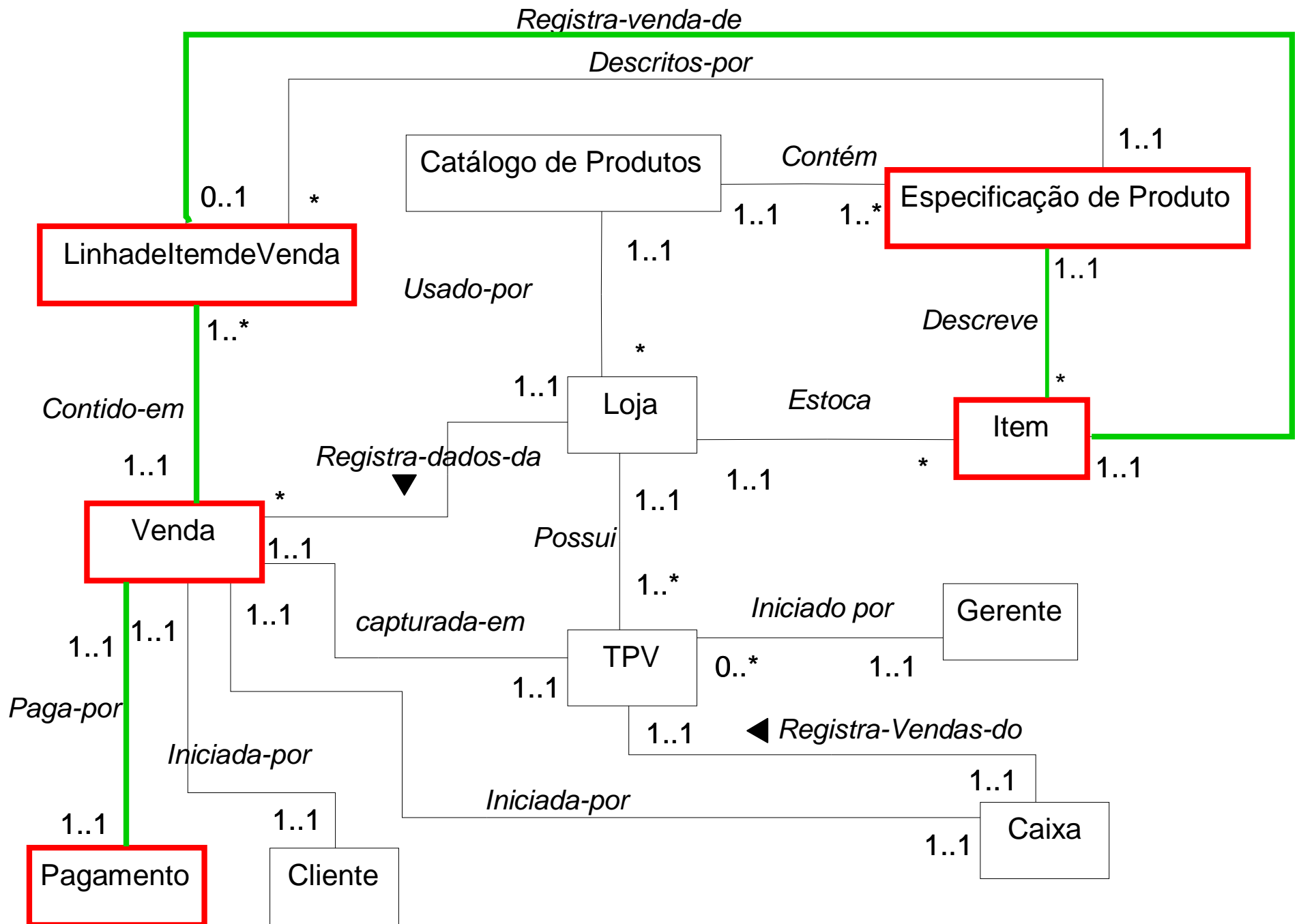




# ***Associações e Implementação***

---

- Uma associação indica um relacionamento significativo apenas sob a **perspectiva conceitual**.
  - Uma associação não implica em uma conexão entre objetos em uma solução de software.
  - Algumas associações do modelo conceitual podem não ser necessárias na implementação.
  - Durante a implementação podem ser descobertas associações entre objetos de software que foram esquecidas durante a modelagem conceitual.



# ***Modelo Conceitual***

## ***Atributo***

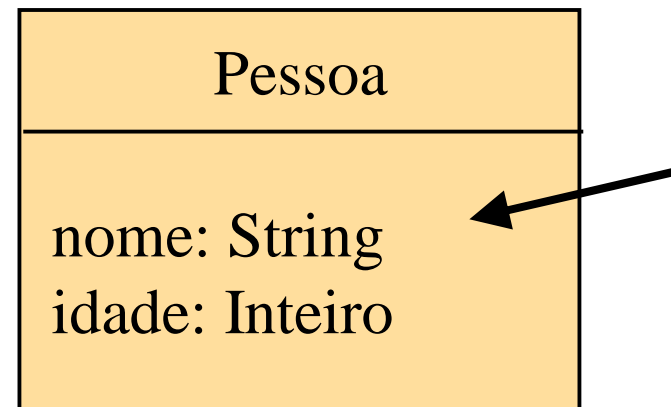
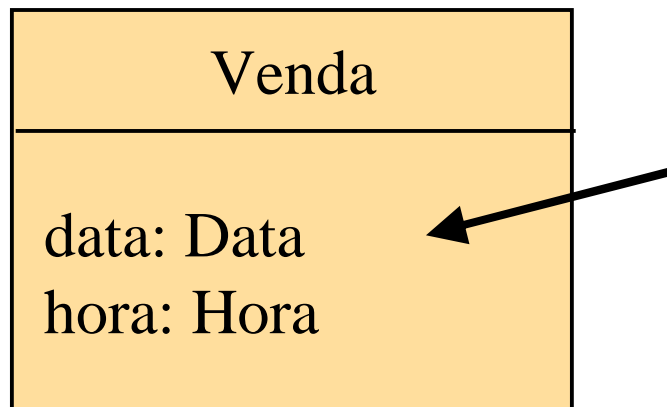
---

- Um **atributo** é um valor de dados lógico de um objeto. Descreve uma característica do objeto.
- Inclua no modelo conceitual apenas os atributos para os quais os requisitos sugerem ou implicam uma necessidade de memorizar a informação.
  - Ex: preço de item, quantidade, descrição, CUP, valor da compra, ...
- Preferivelmente, no modelo conceitual, os tipos de atributos devem ser **simples**, como:
  - **tipos de dados primitivos** - booleano, inteiro, real, cadeia de caracteres,...
  - data, hora, cor, endereço, número de telefone, CEP, ...

# Modelo Conceitual

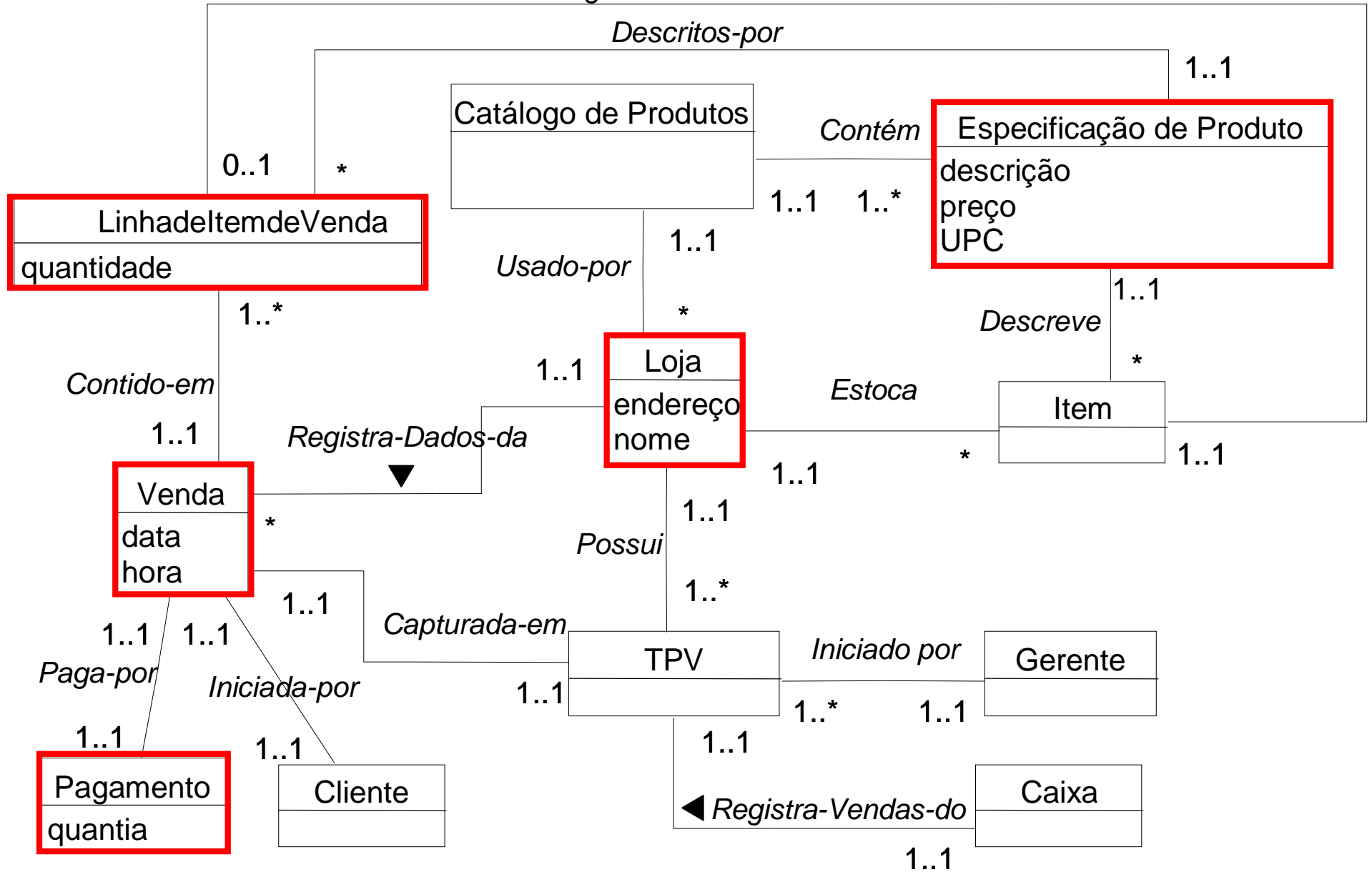
## Atributo

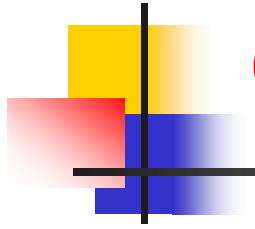
- Os atributos são descritos na segunda seção da caixa de conceito.
- O tipo do atributo é opcional.



*Registra-venda-de*

*Descritos-por*

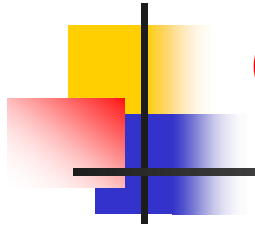




# Generalização

---

- No sistema TPV – caso de uso *ComprarItens* :
  - Os conceitos de *PagamentoComDinheiro*, *PagamentoComCartãoCrédito* e *PagamentoComCheque* são muitos semelhantes.
  - Podem ser organizados em uma hierarquia de tipos (ou conceitos).
    - Hierarquia “generalização/especialização”.



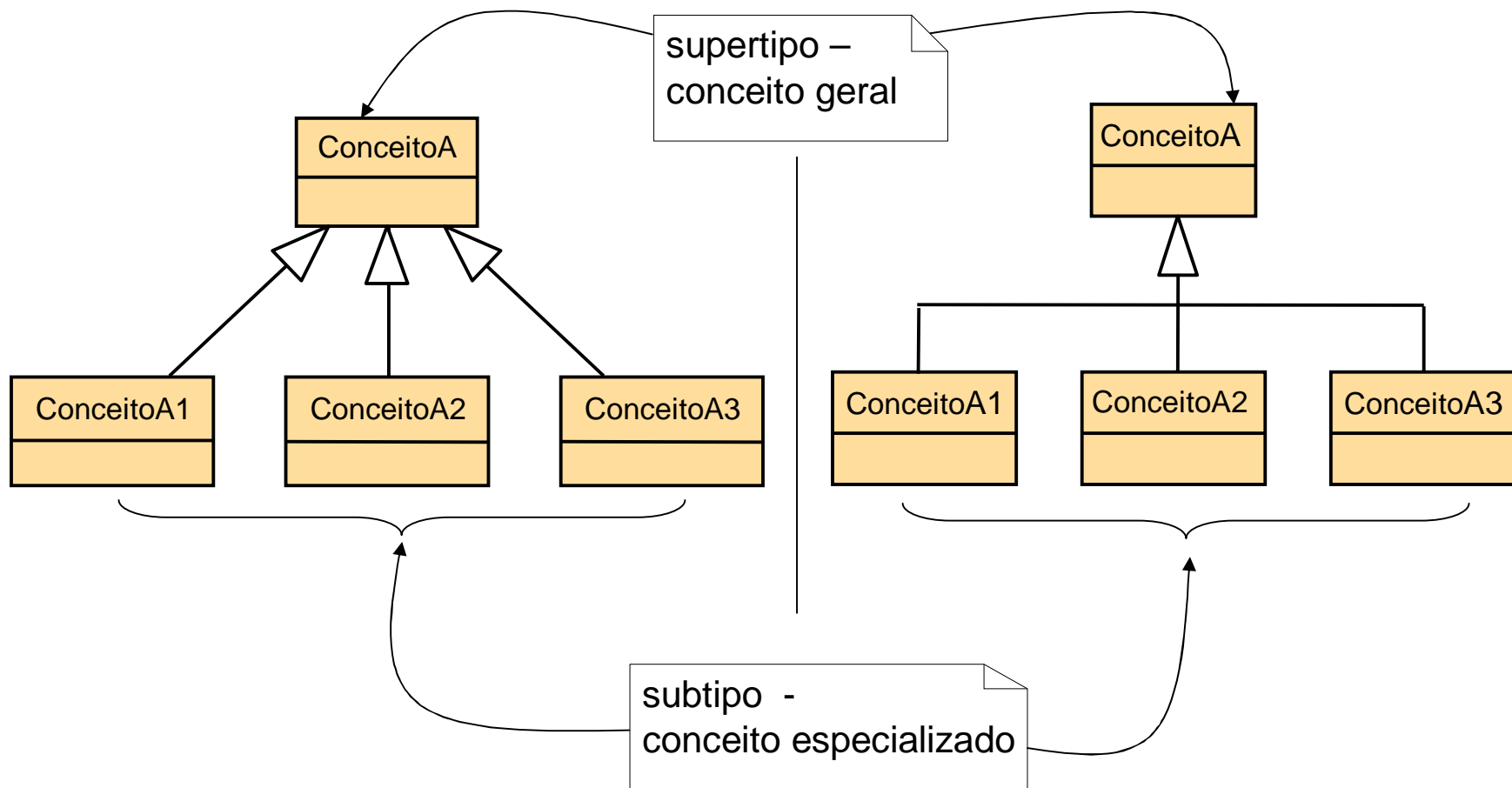
# ***Generalização***

---

- Identifica o que há em comum entre conceitos.
- Permite:
  - Construir classificações taxonômicas – hierarquias de tipos.
  - Compreender os conceitos em termos mais gerais e abstratos, ou mais refinados.
- Conduz a uma notação mais econômica
  - Evita repetição de informação.
- Na implementação, pode ser feita com classes e herança.

# Generalização

## Notação UML







# Generalização e Tipo

---

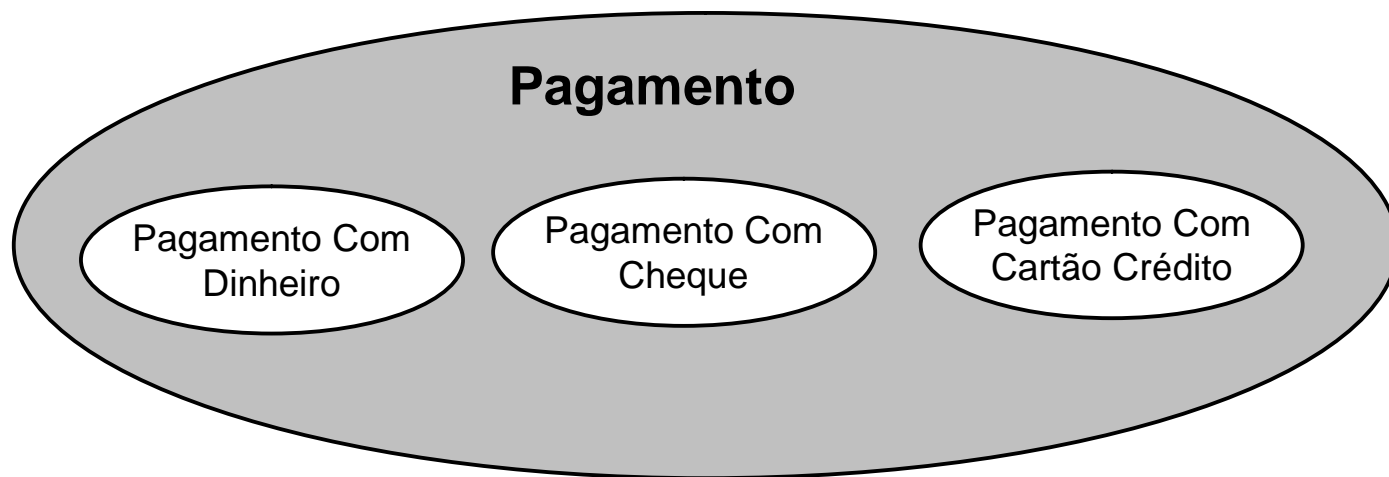
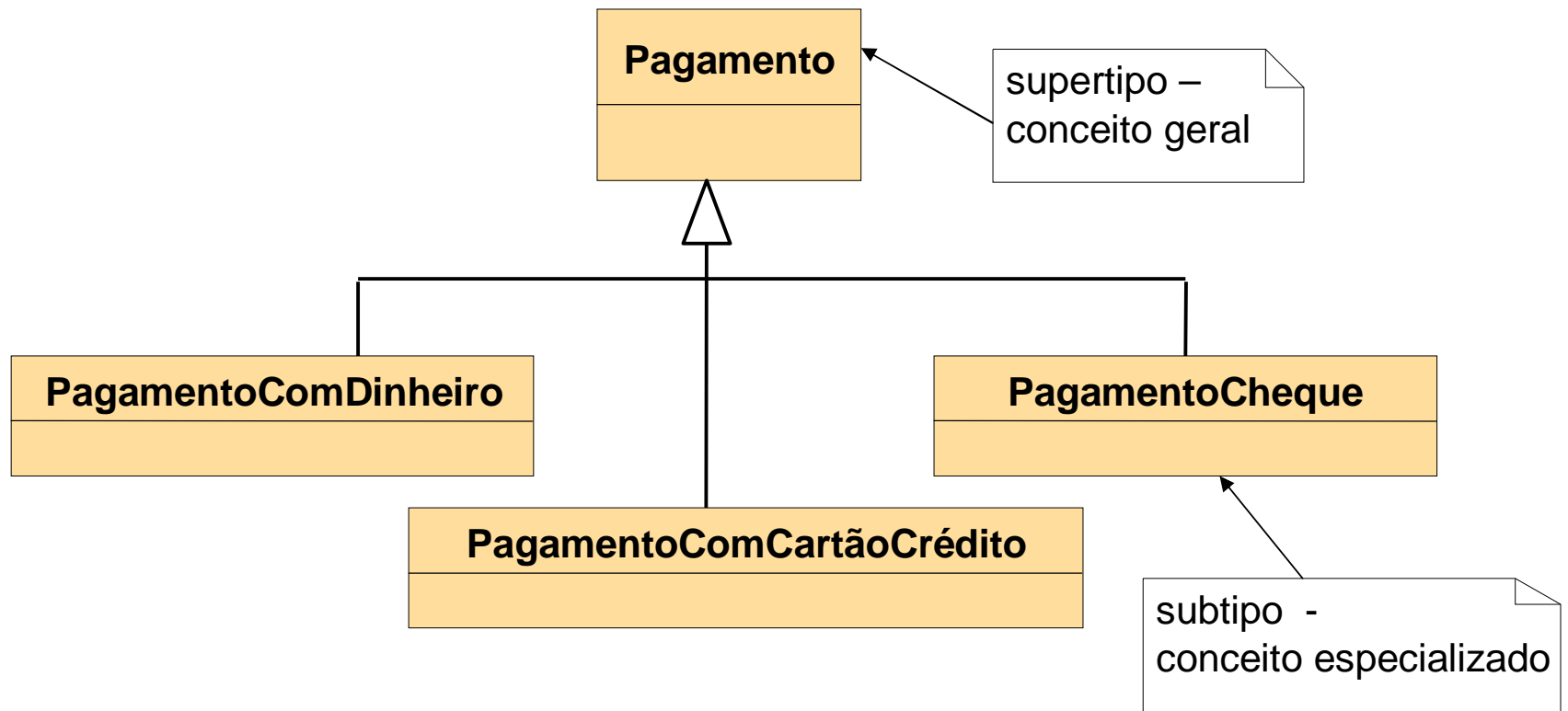
- A definição de um supertipo é mais geral e mais abrangente que a definição de um subtipo.
  - Pagamento: uma transação de transferência de dinheiro (não necessariamente em espécie) de um comprador para um vendedor.
  - PagamentoComCartãoCrédito: transferência de dinheiro, via uma instituição de crédito, que necessita ser autorizada.
- Propriedade **pertinência ao conjunto**: todos os membros de um subtipo são membros do supertipo.
  - ex: PagamentosComCartãoCrédito estão dentro do conjunto Pagamento.



## Regra É-Um

Todos os membros de um conjunto subtipo devem ser membros de seu conjunto supertipo.

O Subtipo é **um** Supertipo.

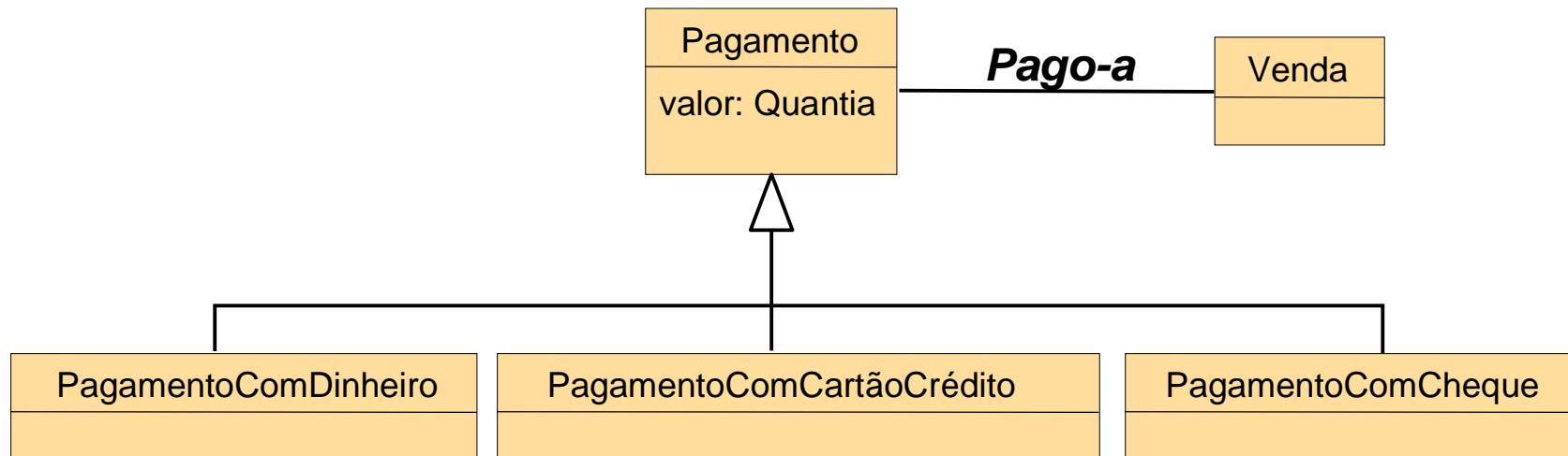


## Regra dos 100%

100% da definição do supertipo dever ser aplicada ao subtipo.

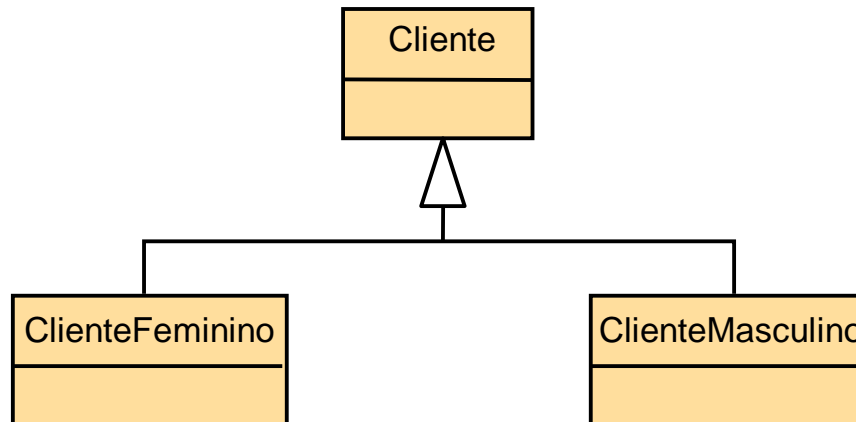
O subtipo deve estar em conformidade com 100% dos seguintes elementos do supertipo:

- Atributos
- Associação



# Quando definir um subtipo ?

- Criar subtipos significa **particionar** um tipo.
  - Dividir um tipo em subtipos disjuntos.
- Quando mostrar a partição de um tipo?
  - Depende da relevância da partição para o domínio do problema.
- Ex: No sistema TPV seria útil definir a seguinte hierarquia??



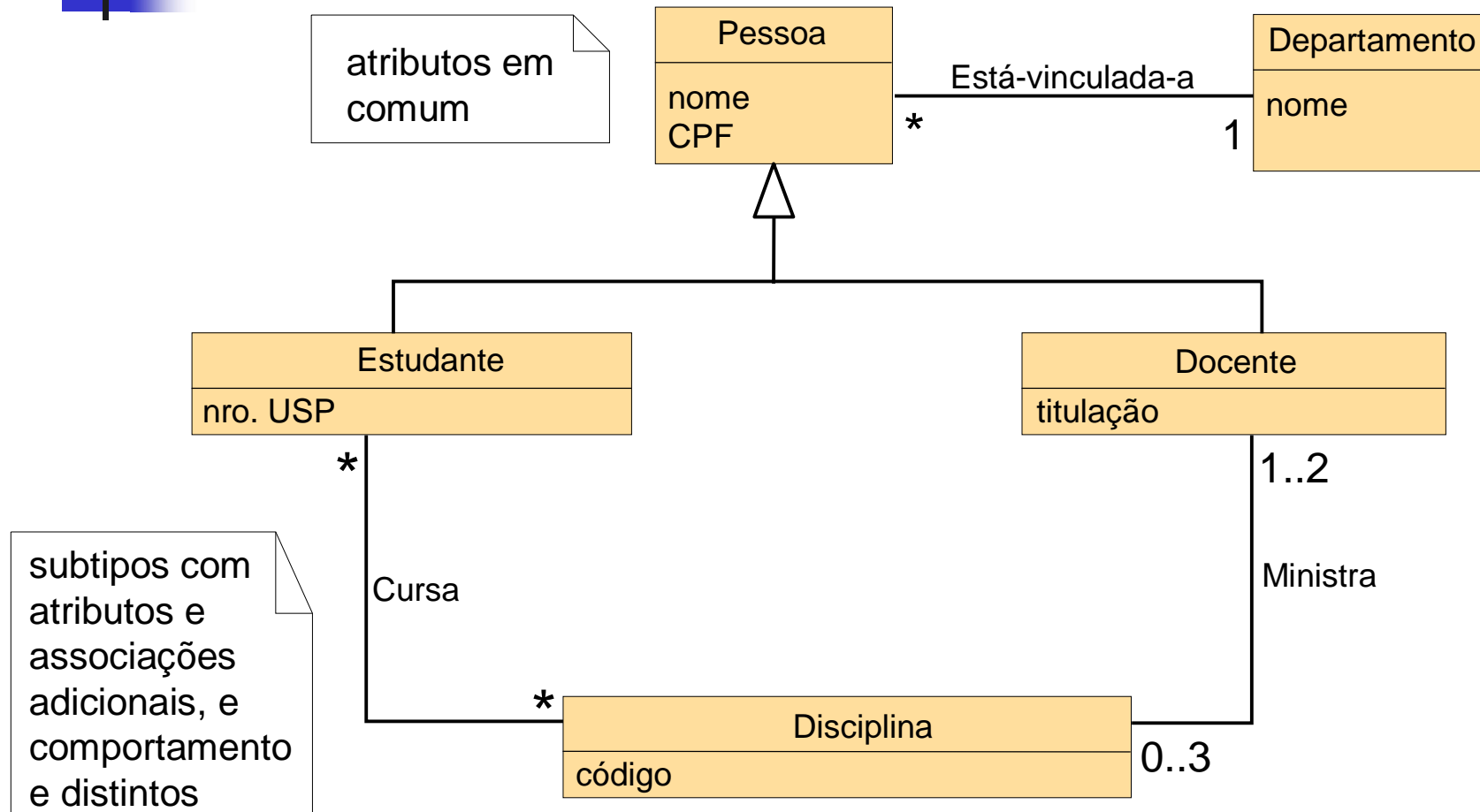


## ***Dicas de quando particionar...***

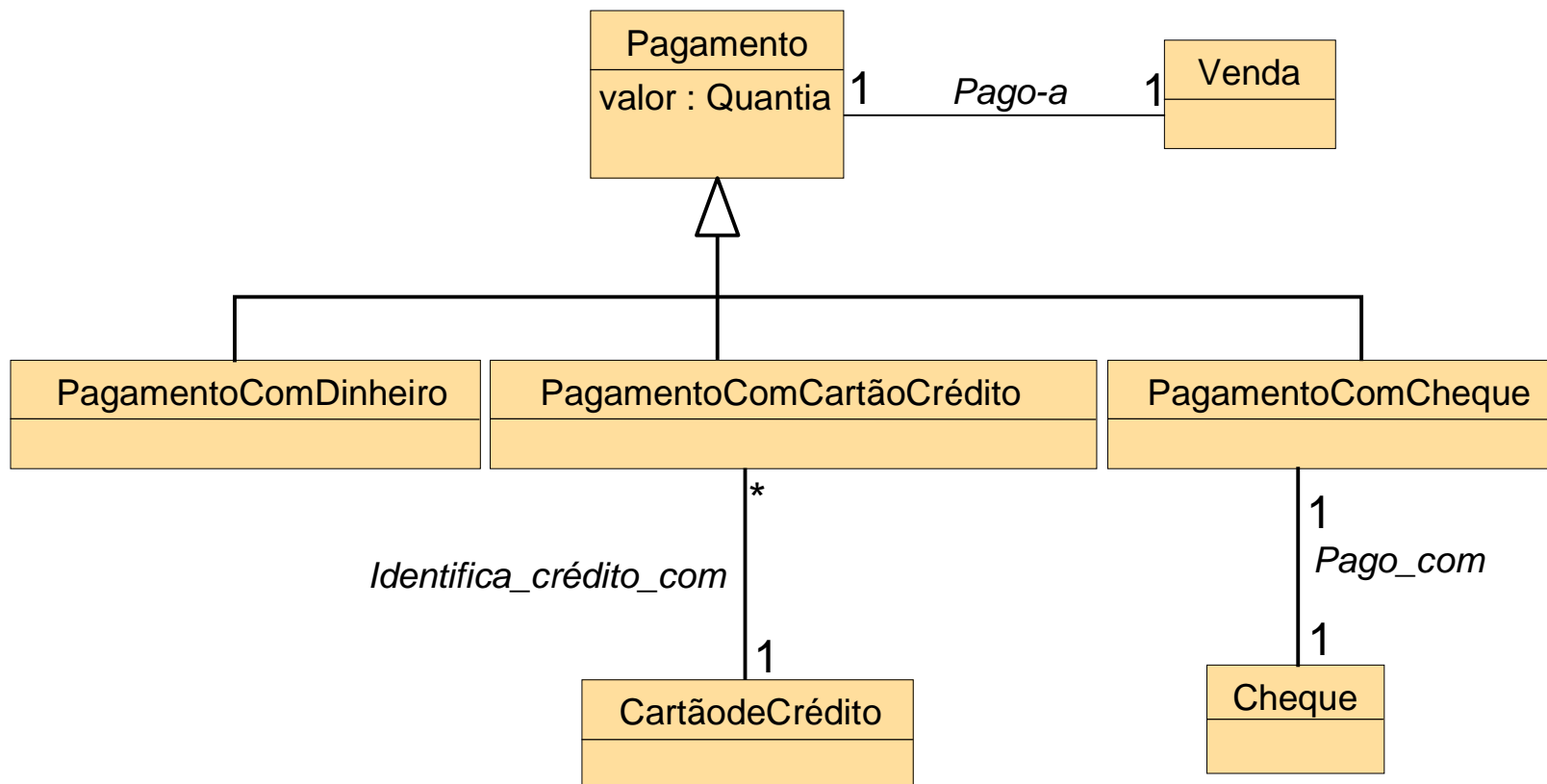
---

1. Um subtipo tem **atributos adicionais** de interesse.
2. O subtipo tem **associações adicionais** de interesse.
3. O conceito do subtipo é **tratado, operado ou manipulado de maneira diferente** que o supertipo ou outros subtipos, segundo formas que são de interesse considerar.
4. O conceito do subtipo representa algo que se **comporta de maneira diferente** do supertipo ou de outros subtipos, segundo formas que são de interesse considerar.

# Exemplo



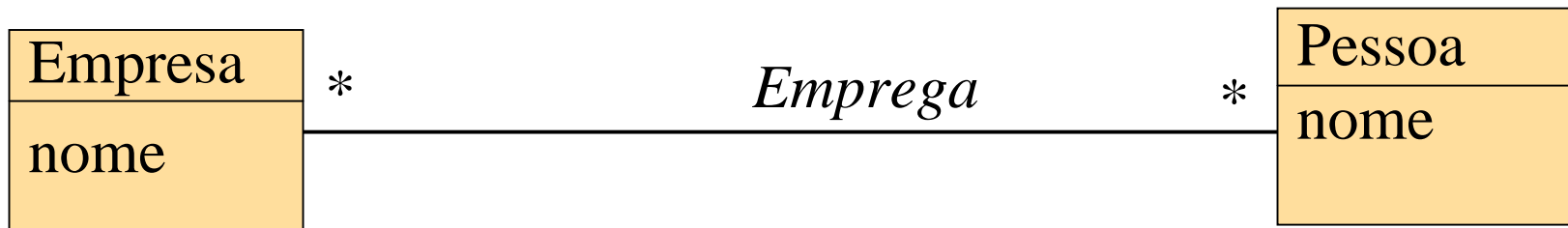
# Exemplo – Sistema TPV



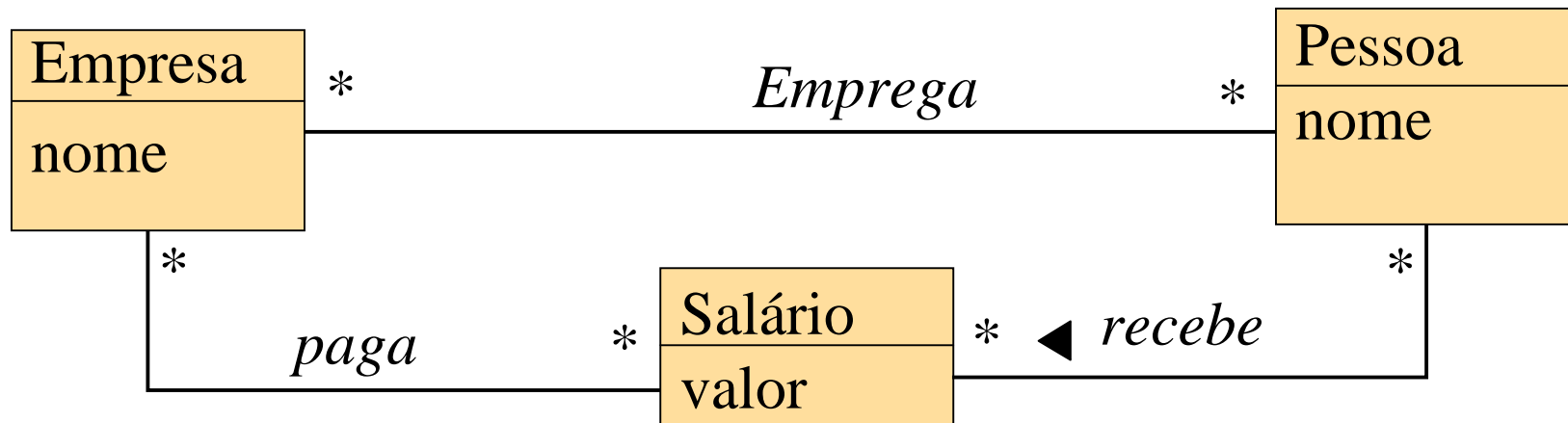


# O que fazer???

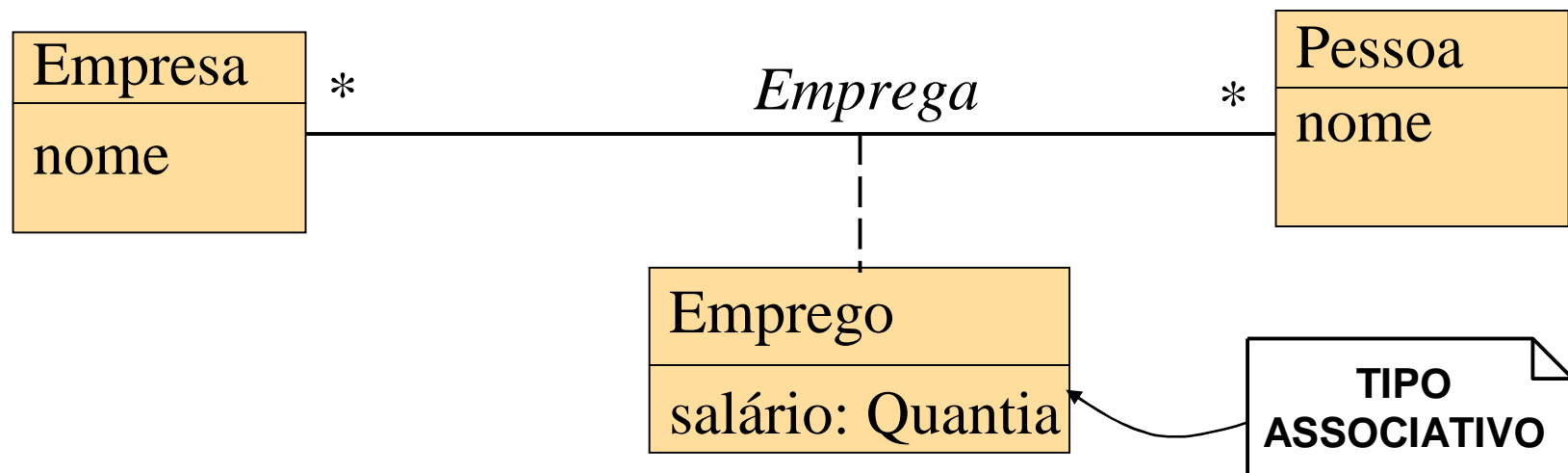
- Se uma pessoa pode ter mais de um emprego em empresas diferentes, onde colocar a informação de **salário**???



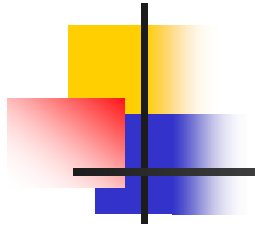
- Uma opção:



## Um opção MELHOR: Tipos Associativos



- **Tipo associativo:** seus atributos estão relacionados a uma **associação** e não a um dos conceitos envolvidos na associação.



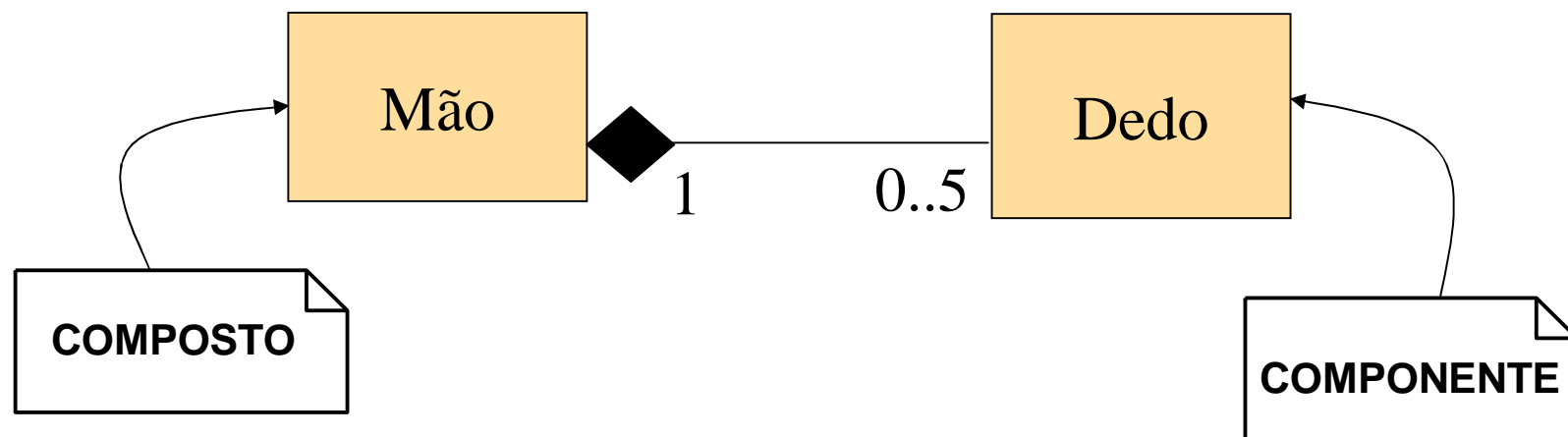
# ***Tipos Associativos***

---

- Indícios da existência de tipos associativos:
  - Um atributo está relacionado a uma associação.
  - As instâncias do tipo associativo têm tempo de vida dependente do tempo de vida da associação.
  - Existe uma associação  **muitos-para-muitos**  entre dois conceitos, bem como informações relacionadas à associação propriamente dita.

# Agregação

- É um tipo de associação usado para modelar relacionamentos **todo-parte** entre coisas.
- O **todo** é geralmente chamado **composto**, as **partes** podem ser chamadas **componentes**.
- Notação em UML: losango vazio ou preenchido.





# **Agregação Composta**

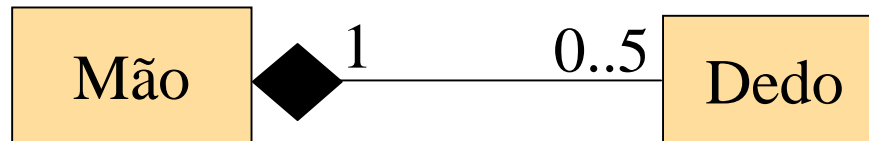
## **(Losango Preenchido)**

---

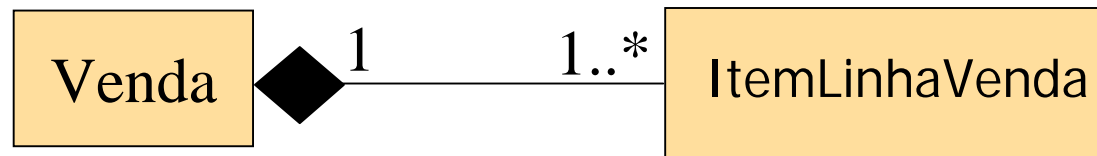
- Agregação **composta** ou **composição** significa que:
  - A multiplicidade na extremidade do composto pode ser **no máximo 1**.
  - Uma instância do componente pode ser parte de **apenas uma** instância do composto (simultaneamente).
- Existe uma **dependência de existência** entre o componente e o composto.
  - A existência de uma instância do composto implica na existência de instâncias dos componentes.
  - A destruição de uma instância do composto implica na destruição das instâncias dos componentes agregados.

## Exemplos

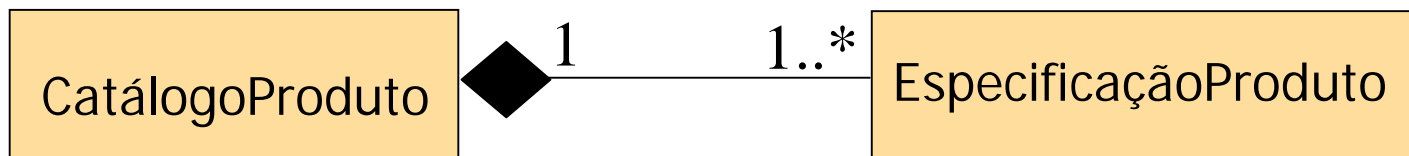
- Um dedo só pode fazer parte de uma mão.



- Um item de linha de venda só pode fazer parte de uma venda.



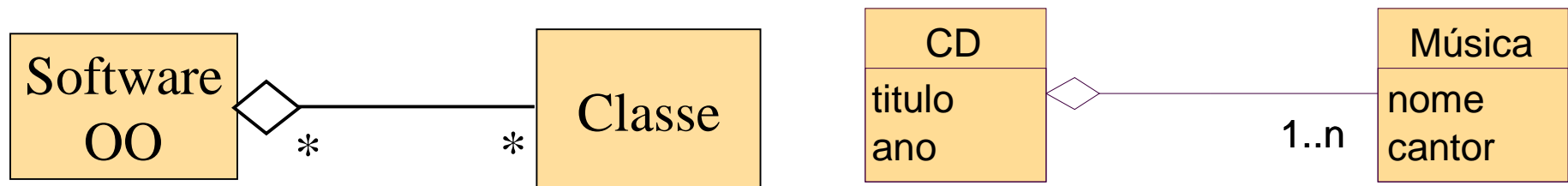
- Uma especificação de produto só pode ser parte de um catálogo.



# Agregação Compartilhada

## (Losango Vazio)

- Agregação **compartilhada** significa que:
  - A multiplicidade na extremidade do composto **pode ser maior que 1**.
  - Uma instância do componente pode estar **simultaneamente em muitas instâncias** do composto.
- Esse tipo de agregação é raro em agregados físicos, mas aparece em conceitos não-físicos.
  - Exemplo:





## ***Modelo Conceitual: Diretrizes para Construção***

---

- Liste os conceitos candidatos relacionados aos requisitos considerados.
    - Use a Lista de Categorias de Conceitos e a Identificação de Substantivos.
  - Desenhe os conceitos em um modelo conceitual.
  - Registre as associações entre conceitos.
- 
- Acrescente os atributos necessários para completar os requisitos.
  - Identifique possíveis agregações, generalizações e tipos associativos.