



Conteúdo

1. Introdução

2. Levantamento de Requisitos

3. Análise Orientada a Objetos

- Classe, Atributo e Associação

- Classe de Associação, Agregação, Herança e Pacote

4. Projeto Orientado a Objetos

5. UML

6. Métodos Ágeis



Análise Orientada a Objetos

Modelo Conceitual

(continuação)



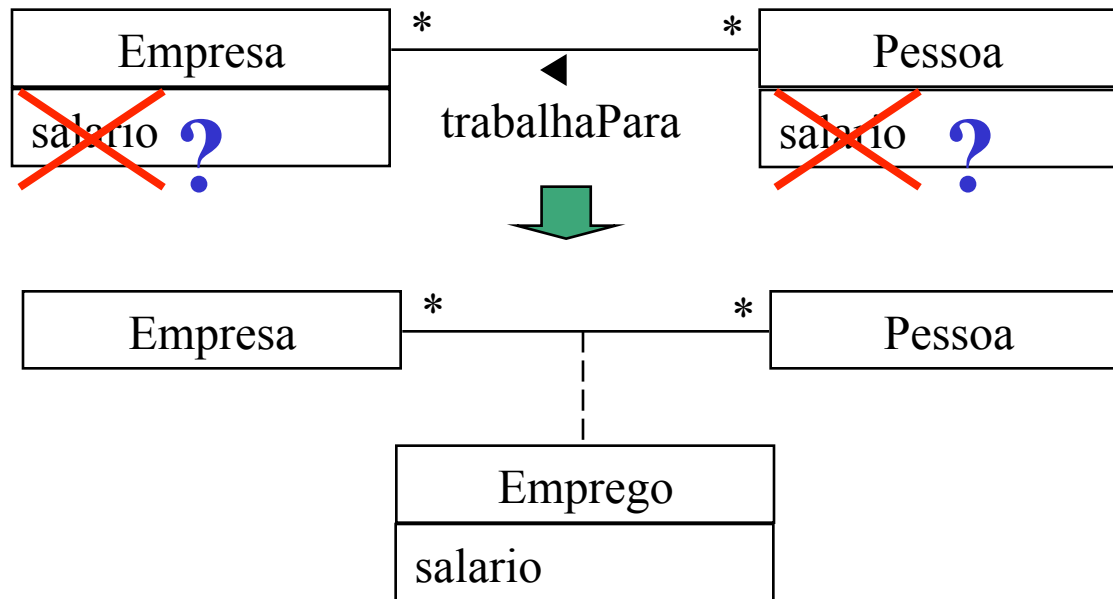


Diagrama de Classes

Classes de Associação

Classe de Associação

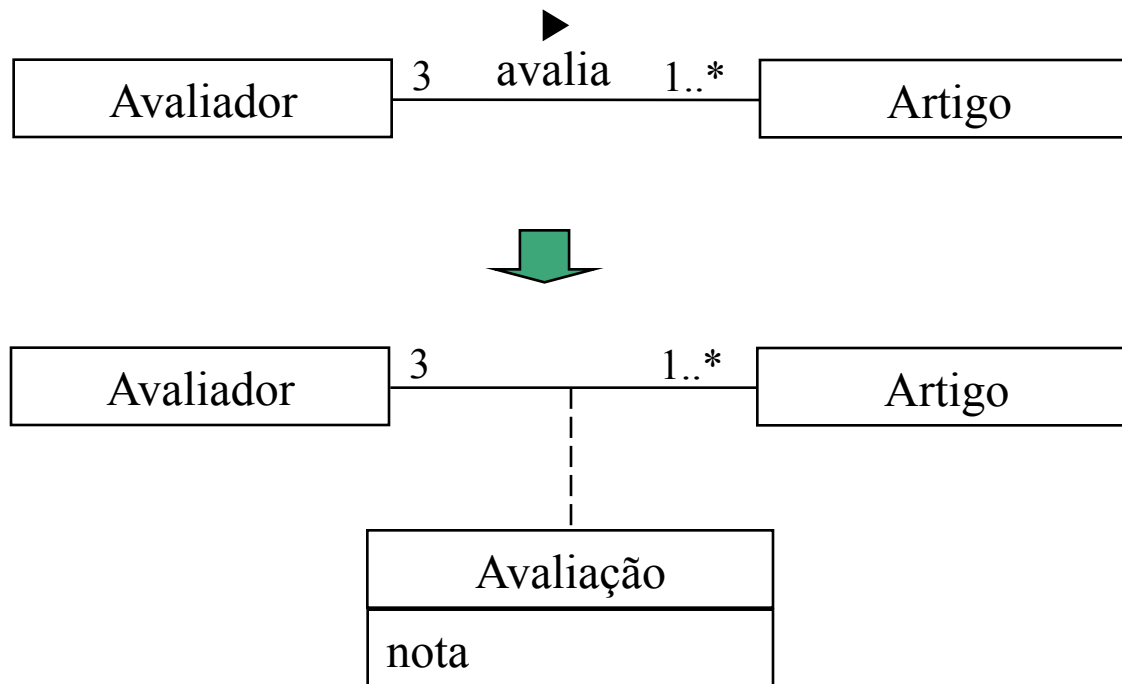
Exemplo: Considere que uma empresa tenha vários empregados e que um empregado pode trabalhar para várias empresas. Onde deve ser colocado o salário que o empregado recebe de uma empresa?



➔ Se um atributo depende da associação entre duas classes, ele é adicionado à própria associação.

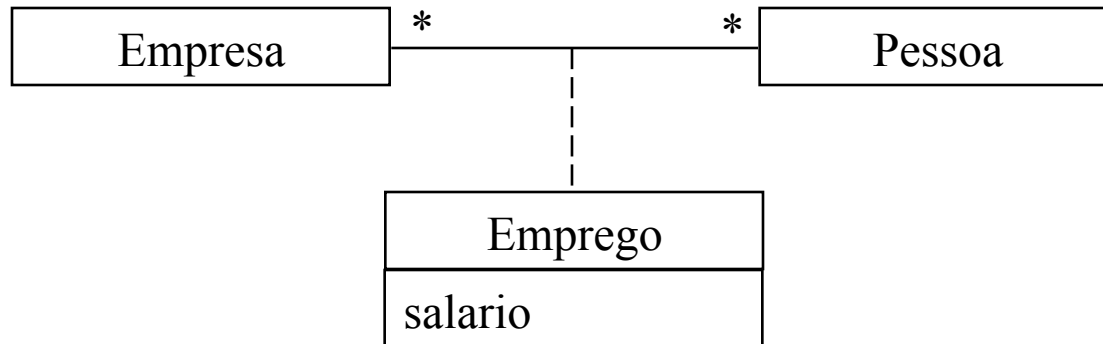
Classes de Associação

Exemplo: Considere que cada avaliador atribui uma nota para cada artigo avaliado por ele e que cada artigo recebe uma nota de cada avaliador que o avalia. Onde deve ficar a nota dada pelo avaliador para o artigo?



Notação de Classes de Associação

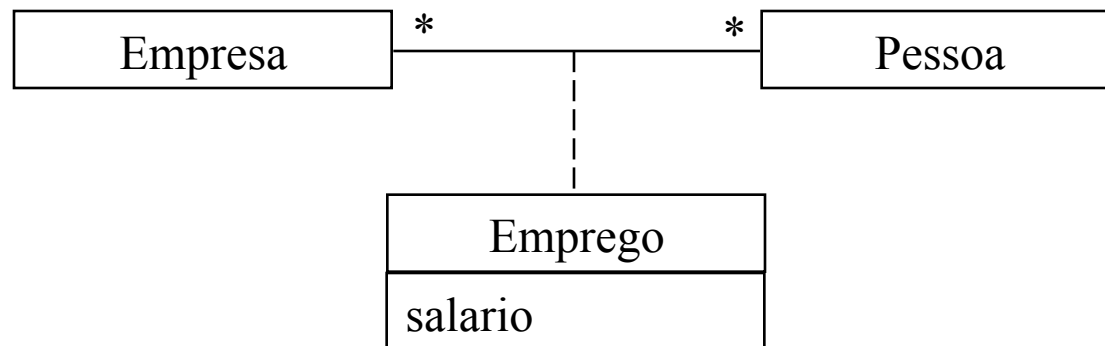
A representação de uma classe de associação é feita através de uma linha tracejada que conecta a associação à classe de associação.



- As associações que possuem classes de associação não precisam apresentar nomes.

Diretrizes no Uso de Classes de Associação

- Um atributo está relacionado com a associação;
- Instâncias da classe de associação dependem da associação durante toda a sua existência;
- Existe uma associação muitos-para-muitos entre os dois conceitos e existe informações relacionadas com a própria associação.



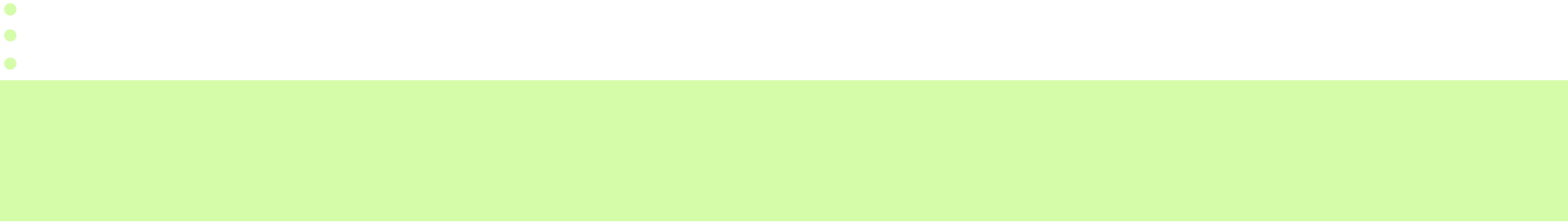


Diagrama de Classes

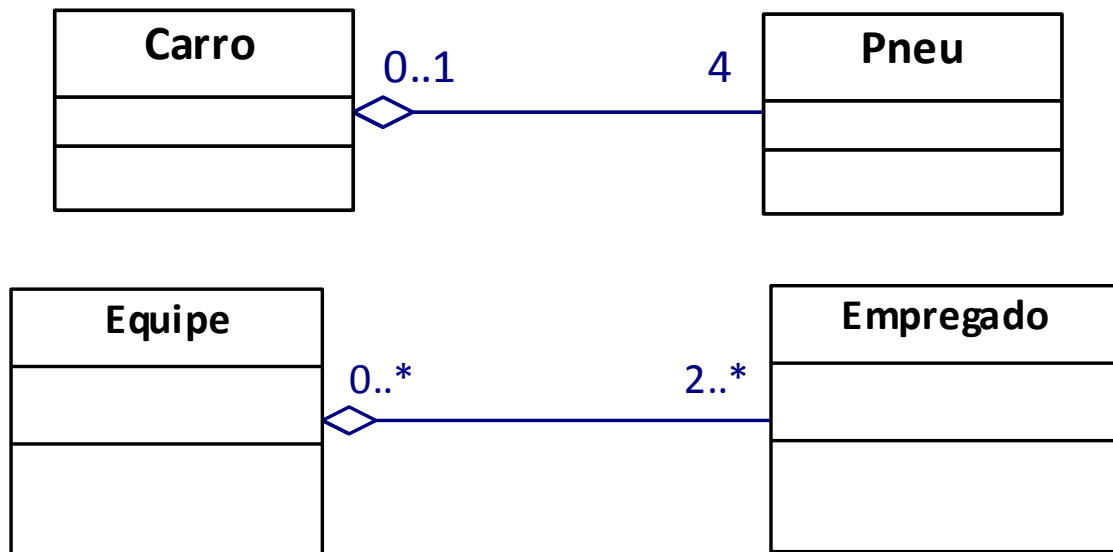
Agregação



Agregação

É uma forma de associação usada para modelar os relacionamentos todo-parte entre as classes.

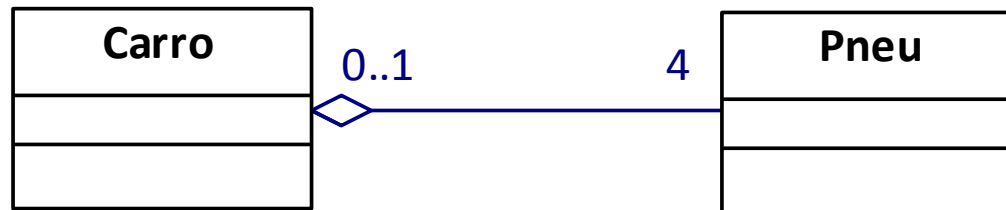
É usado quando um objeto é composto por outros objetos.



➔ Não apresenta diferença semântica significativa em relação à associação.

Notação de Agregação

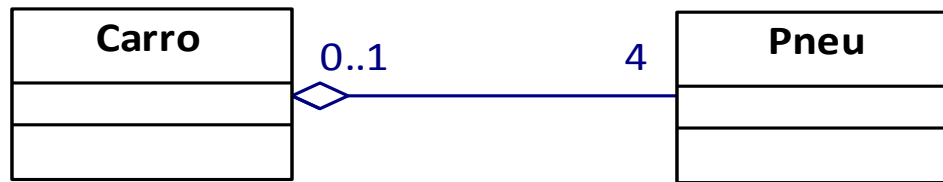
→ É utilizado um losango no lado do composto.



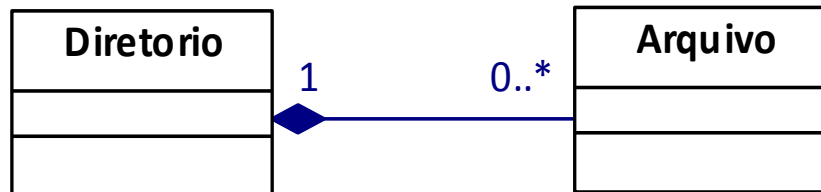
- O nome da associação é frequentemente excluído no relacionamento de agregação, pois ele é tipicamente do tipo **tem-parte**.
- O losango pode ser preenchido (composição) ou vazio (agregação).

Tipos de Agregação

→ Agregação Compartilhada (Agregação)

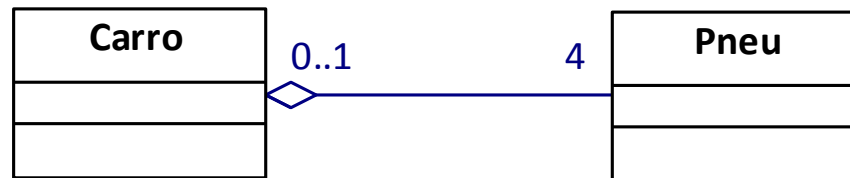


→ Agregação Composta (Composição)

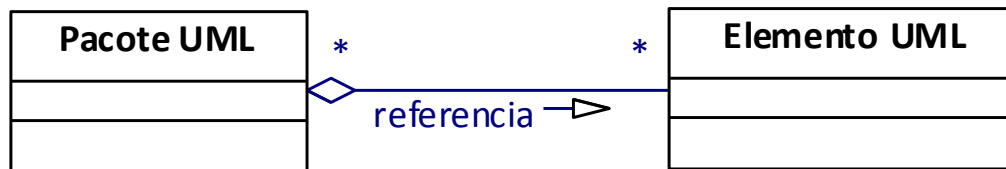


Agregação

Agregação (ou agregação compartilhada): a parte não depende da existência do composto e pode ser membro de vários objetos compostos.



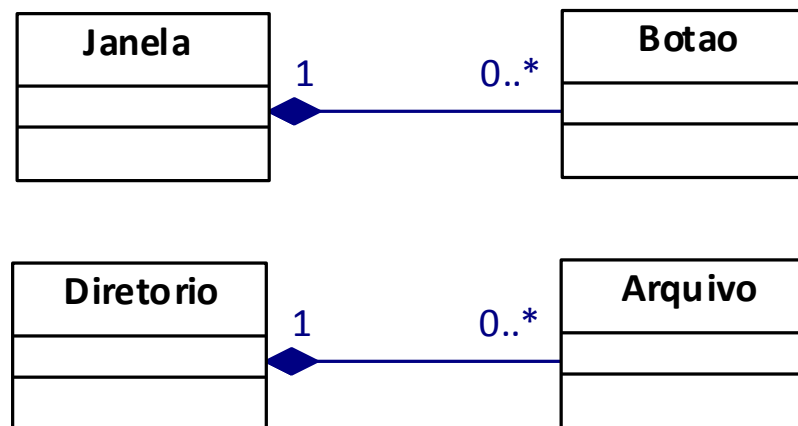
➔ A multiplicidade no lado do composto pode ser mais do que 1.



➔ O losango é vazio.

Composição

Composição (ou agregação composta): a parte é um membro de um único objeto composto e existe uma dependência de existência da parte em relação ao composto.



- ➔ A multiplicidade no lado do composto é exatamente 1.
- ➔ O losango é preenchido.

Diretrizes no Uso de Composição

- Existe uma combinação física ou lógica de todo-parte;
 - A existência da parte está contida dentro da existência do composto, ou seja, existe uma dependência criação-deleção da parte no todo;
 - As operações aplicadas ao composto são propagadas para as partes, tais como destruição e cópia.
-
- No projeto, o objeto composto cria os objetos que são partes dele.
- ➔ Na dúvida, não use composição ou agregação, use associação.

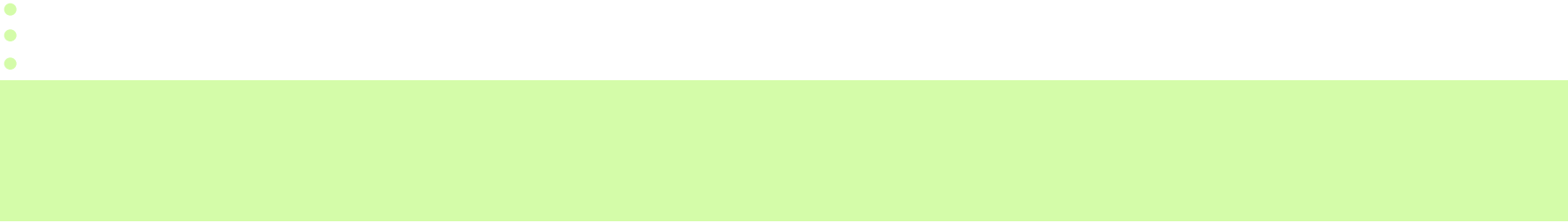


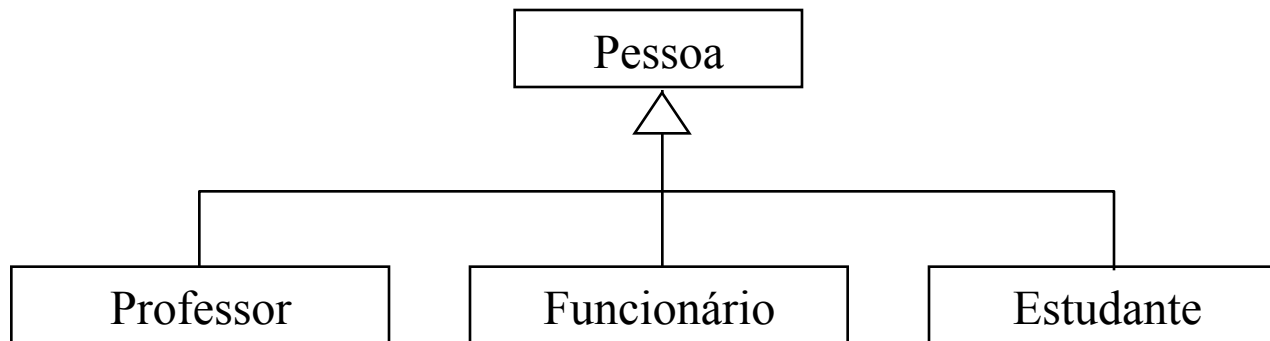
Diagrama de Classes

Generalização/Especialização

Generalização / Especialização

Atividade de identificar as partes comuns entre classes e definir relacionamentos entre uma superclasse (conceito geral) e uma subclasse (conceito especializado).

Exemplo:

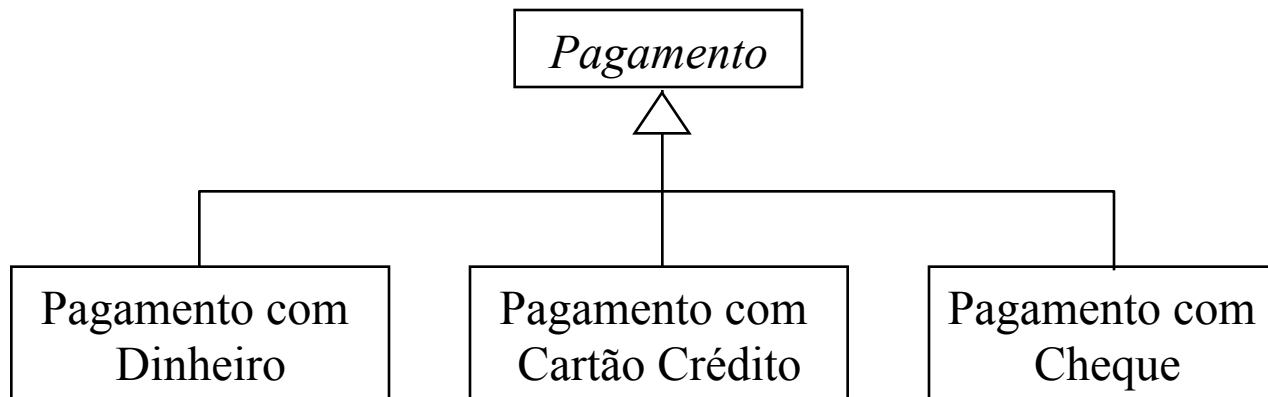


Generalização / Especialização

➔ Todos os membros do conjunto de uma subclasse devem ser membros do conjunto da sua superclasse.

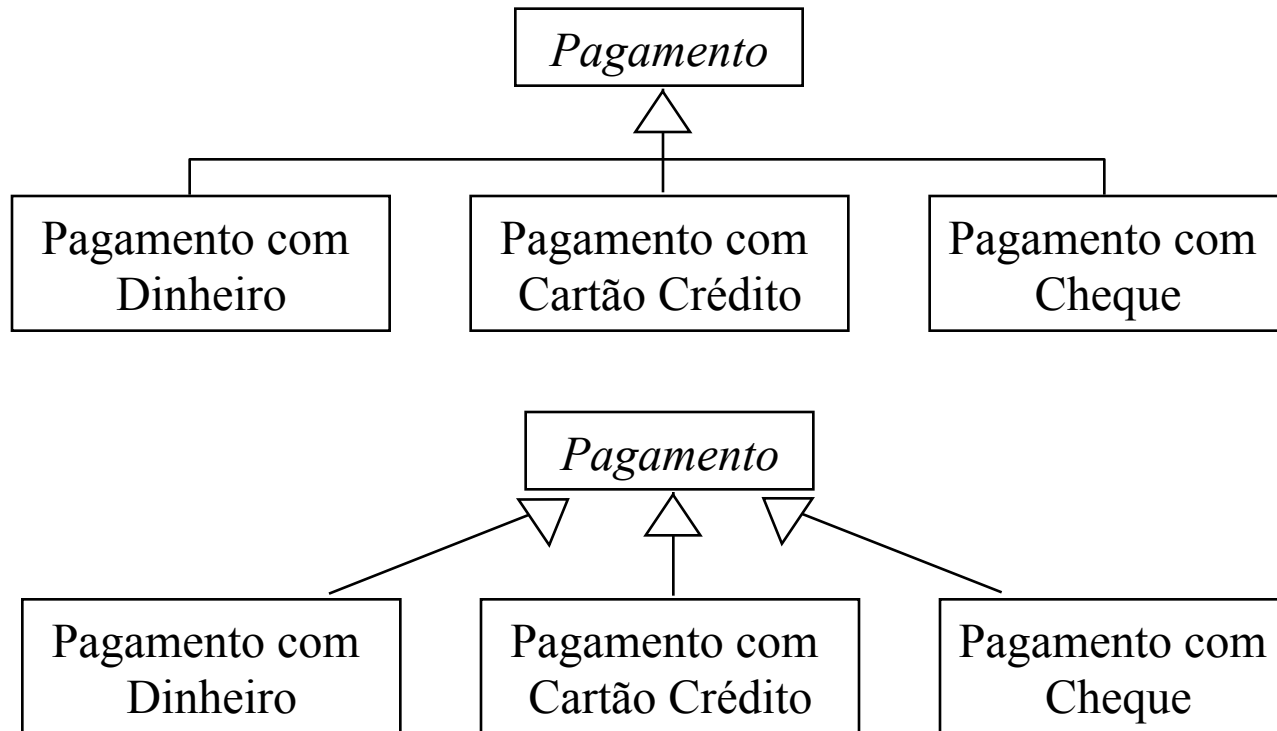
Subclasse **é uma** Superclasse

Exemplo: Pagamento com Cartão de Crédito é um Pagamento



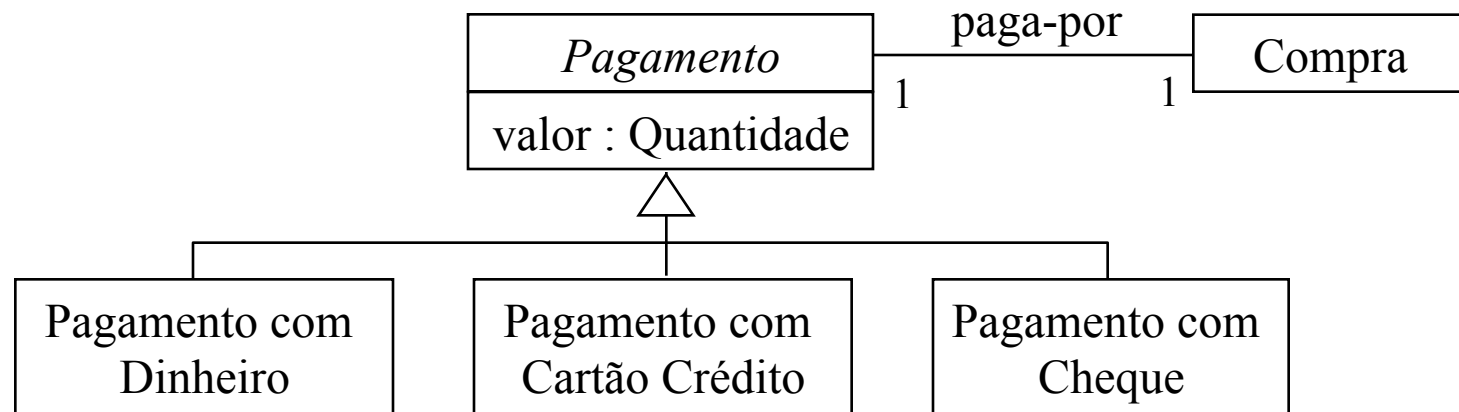
Notação

Notação da Generalização/Especialização na UML: O relacionamento de generalização entre classes é indicado com um triângulo vazado que aponta para a classe mais geral.



Generalização / Especialização

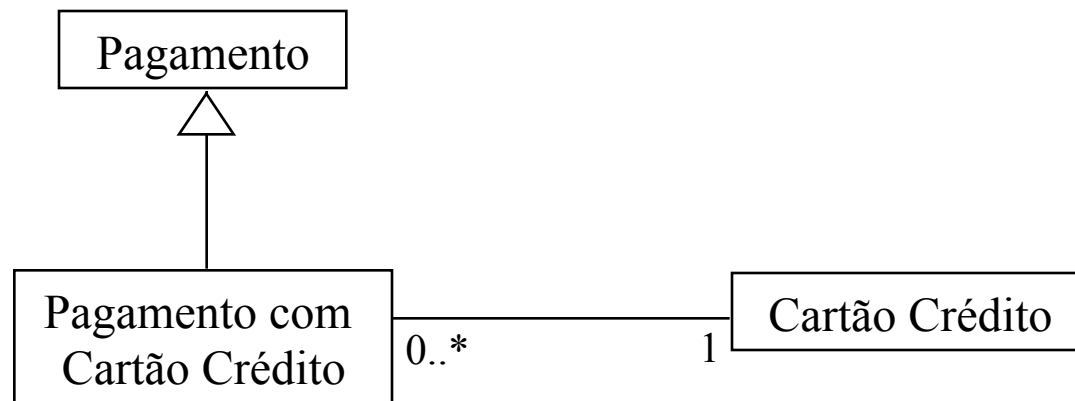
➔ Toda a definição da superclasse conceitual deve ser aplicável à subclasse, tanto em relação aos atributos como em relação às associações.



Especialização

Defina uma subclasse de uma superclasse quando:

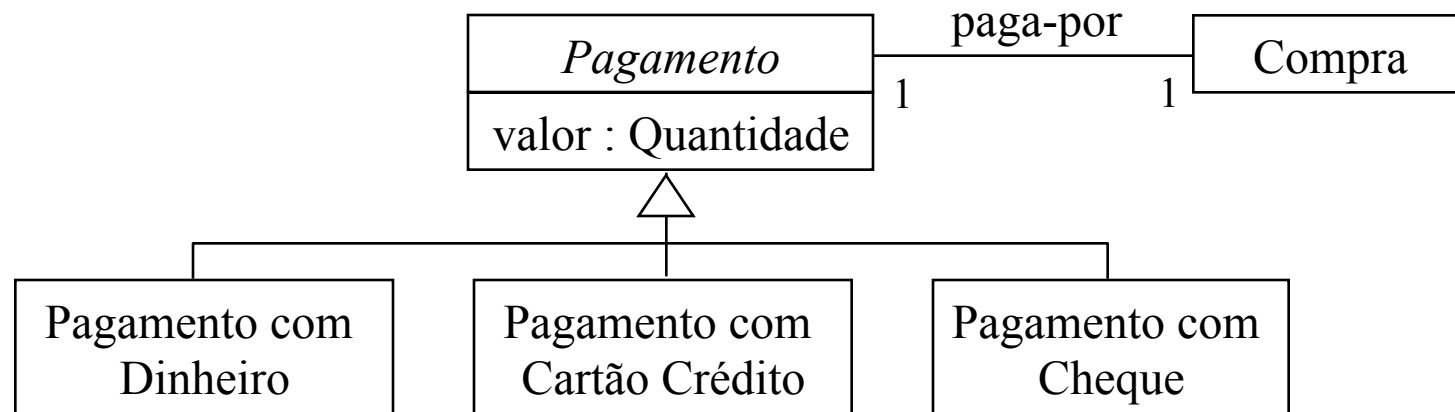
1. A subclasse tem atributos adicionais.
2. A subclasse tem associações adicionais.
3. O conceito da subclasse é operado, tratado, manipulado ou comporta-se de maneira diferente da superclasse ou de outras subclasses.



Generalização

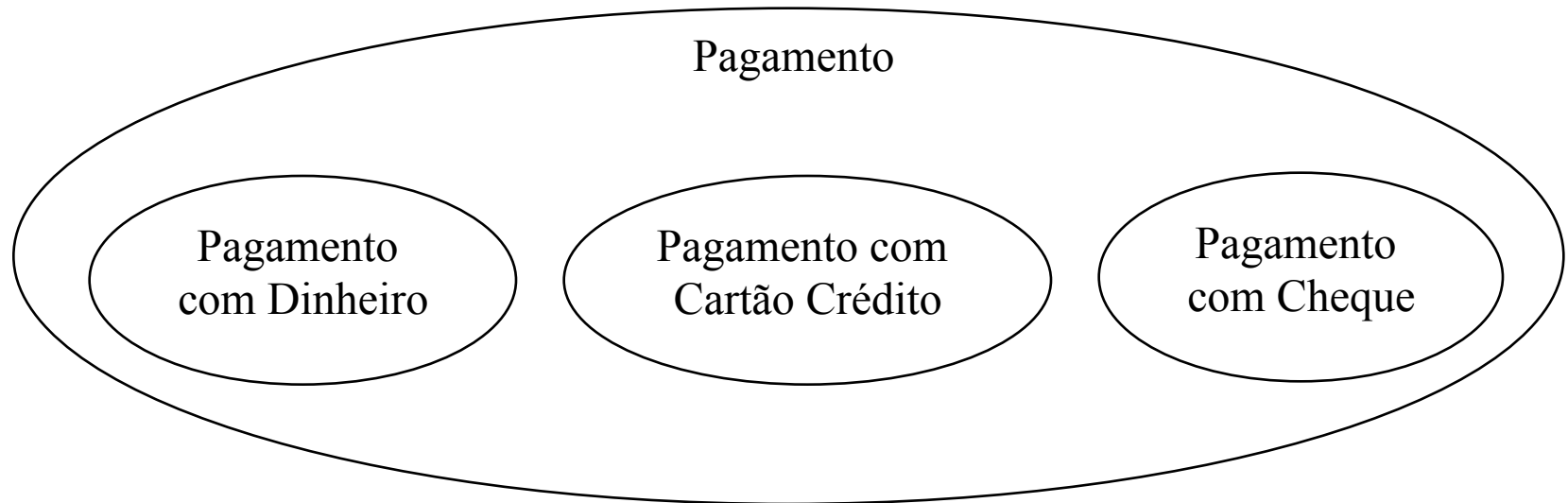
Defina uma superclasse quando:

1. As subclasses representam variações de um conceito similar.
2. As subclasses podem ser vistas como “Subclasse **é-uma** Superclasse”.
3. Todas as subclasses possuem os mesmos atributos que podem ser fatorados e incluídos na superclasse.
4. Todas as subclasses possuem as mesmas associações que podem ser fatoradas e incluídas na superclasse.



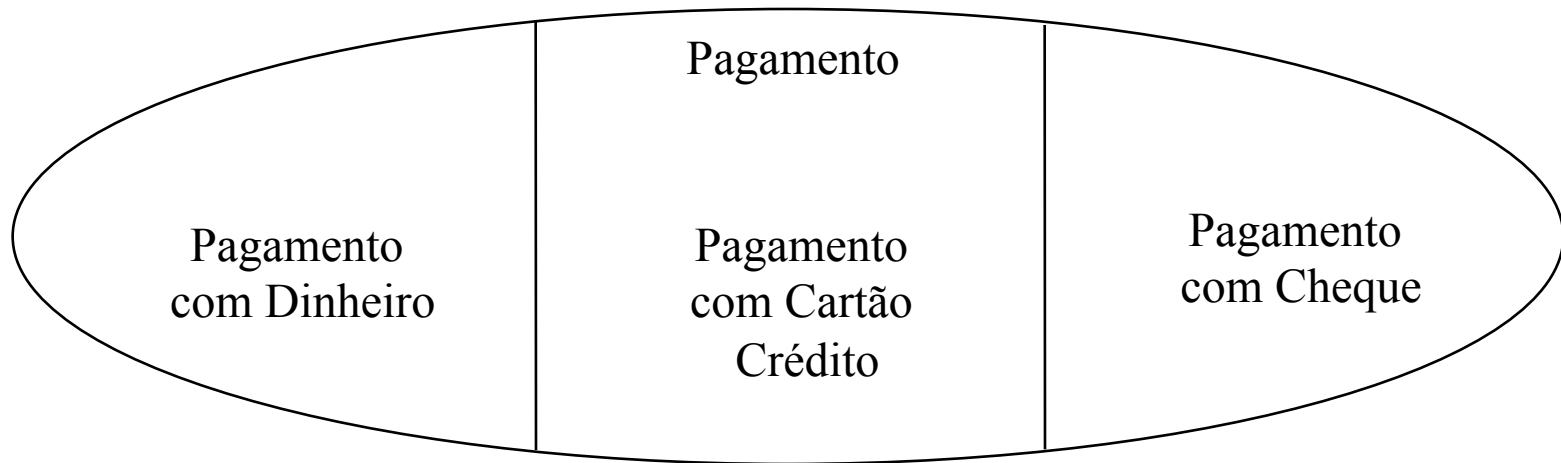
Generalização / Especialização

➔ Todos os membros do conjunto de uma subclasse conceitual são membros do conjunto da sua superclasse.



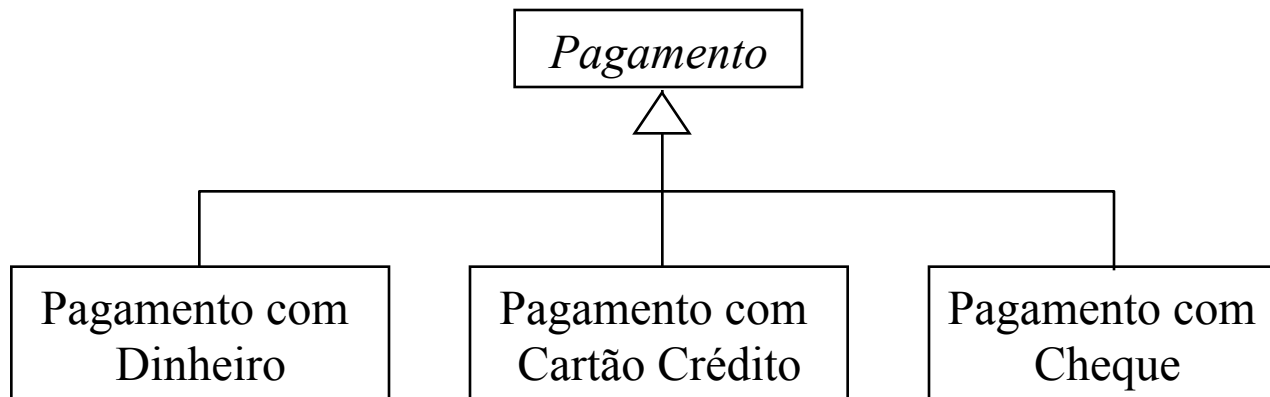
Classe Abstrata

Se cada membro de uma classe C deve ser também um membro de uma subclasse, então a classe C é chamada de **classe abstrata**.



Classe Abstrata

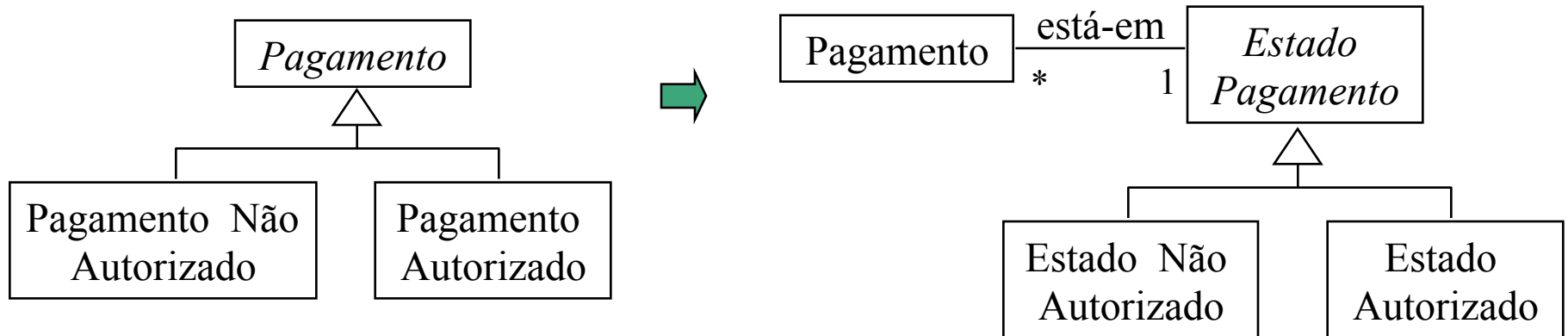
Notação da UML para Classes Abstratas: o nome da classe é escrito em itálico.



Modelagem dos Estados

Não modele os estados de um conceito X como subclasses de X:

- Defina uma hierarquia de estados e associe os estados com X OU
- Mostre os estados em diagramas de estado.





Vantagens de Generalização / Especialização

Vantagens da utilização da generalização/especialização no modelo de domínio:

- Proporciona uma maior expressividade e compreensão;
- Permite a redução da informação repetida;
- Durante o projeto e implementação, as classes de software que utilizam herança dão origem a um software melhor.



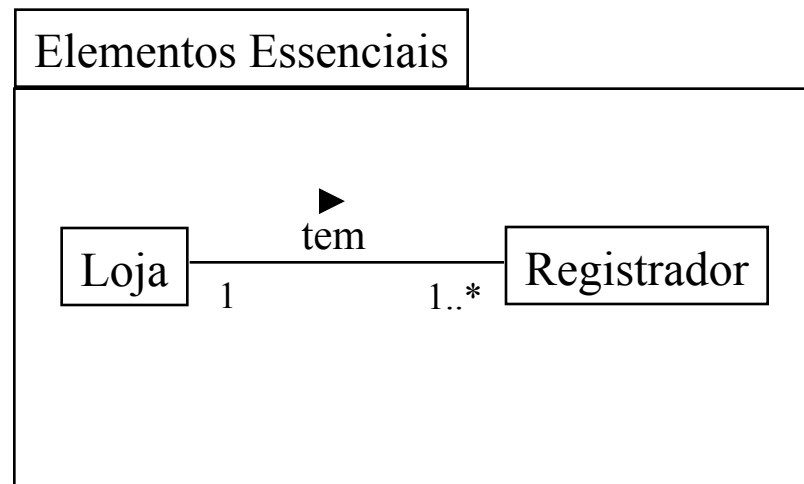


Diagrama de Classes

Pacotes

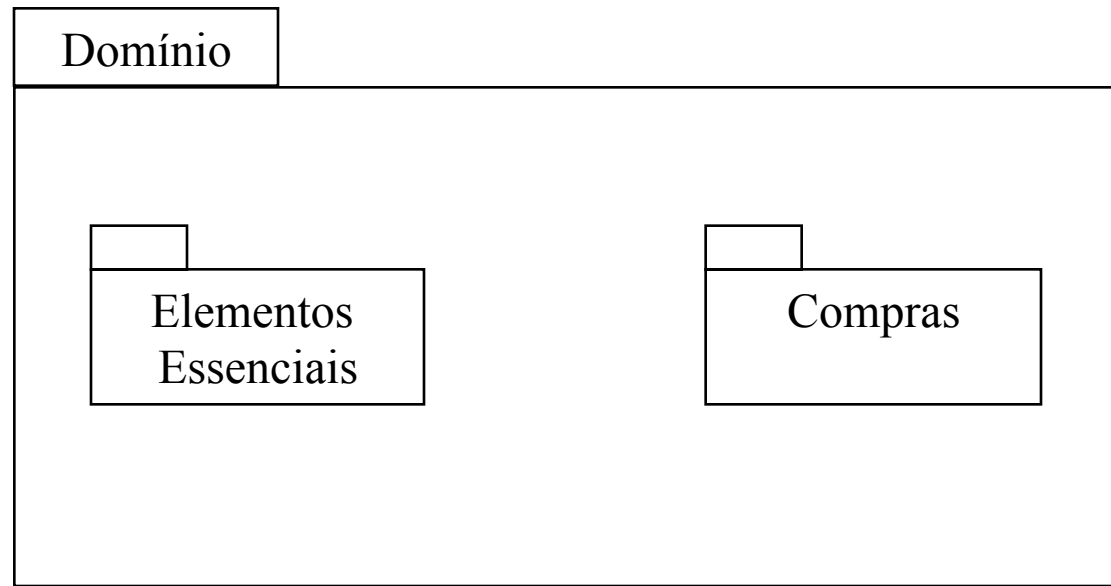
Pacotes

Um modelo de domínio pode ser distribuído em pacotes de conceitos fortemente relacionados.



Notação dos Pacotes

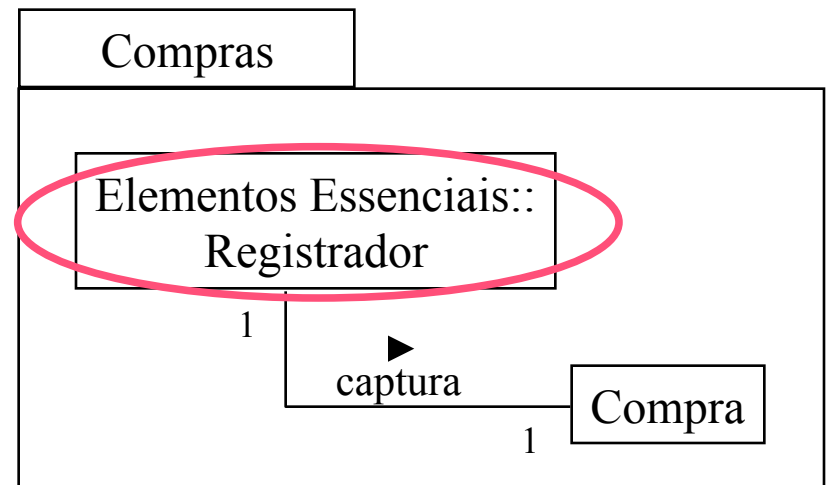
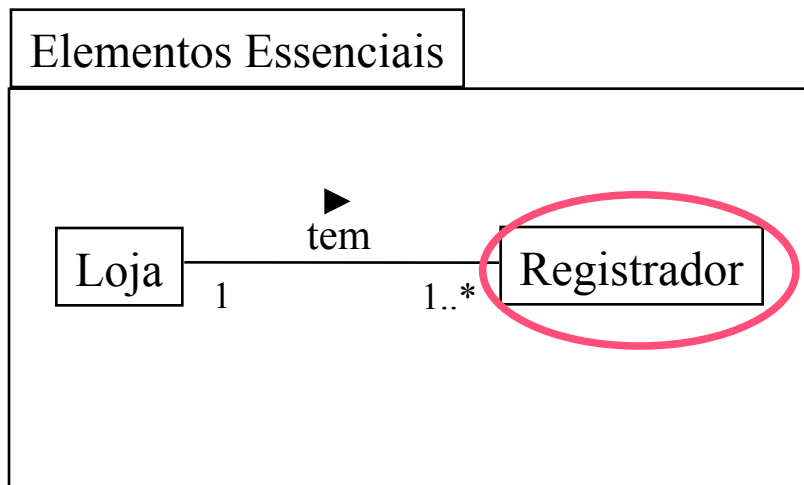
- O nome do pacote é colocado dentro do retângulo menor, se o seu conteúdo é mostrado, ou no retângulo maior.
- O conteúdo do pacote pode ser um outro pacote ou um conjunto de classes.



Referência entre Pacotes

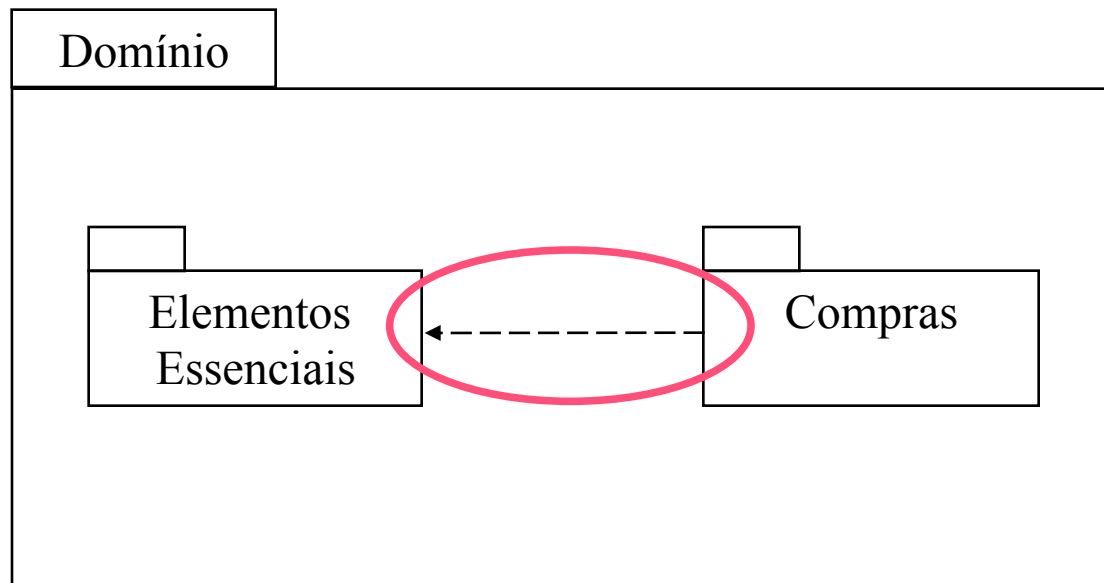
Um elemento é possuído pelo pacote na qual ele é definido.

Mas um elemento pode ser referenciado em outros pacotes.



Dependências entre Pacotes

A dependência entre pacotes indica que elementos do pacote dependente conhece elementos do pacote destino.





Pacotes

Como organizar o modelo de domínio em pacotes?

Coloque junto os elementos que:

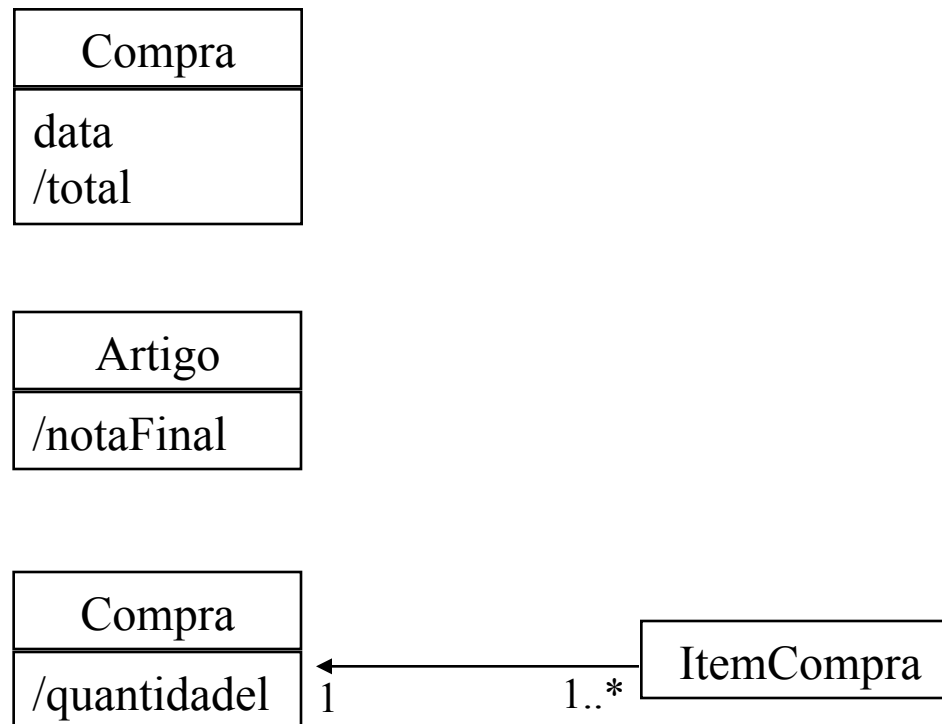
- estão relacionados através de algum conceito ou finalidade;
- estão dentro da mesma hierarquia de classes;
- participam dos mesmos casos de uso;
- estão relacionados através de várias associações.

Para que usar pacotes?

- para ajudar na compreensão do modelo de domínio;
- para ajudar em uma análise paralela, onde diferentes analistas fazem a análise de diferentes sub-domínios.

Elementos Derivados

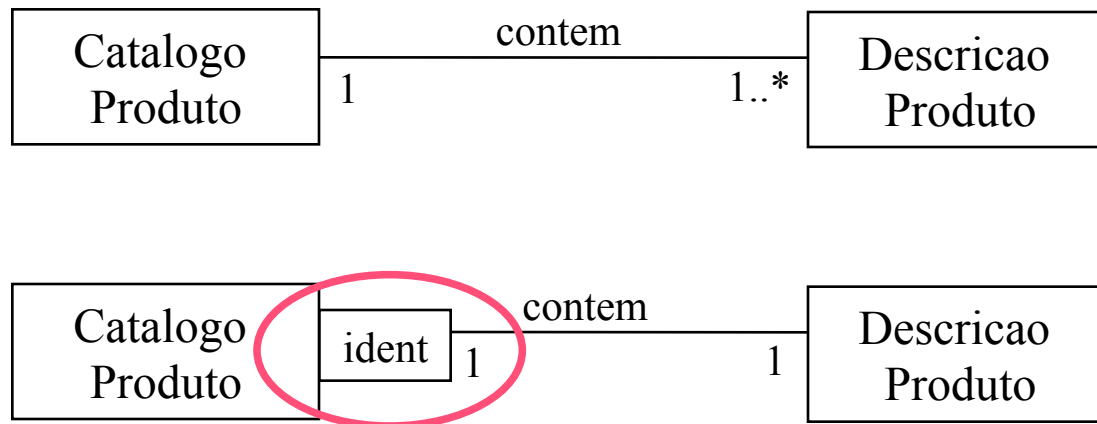
Um elemento derivado pode ser determinado a partir de outros.



Associações Qualificadas

Associação Qualificada - Associação com um qualificador.

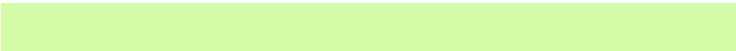
Qualificador - distingue o conjunto de objetos baseado no valor do qualificador.



➔ O qualificador comunica como os objetos de uma classe são distinguidos em relação a outra classe.



Contratos



Contratos

Casos de Uso descrevem o comportamento do sistema.

↳ geralmente são suficientes como entrada para o projeto (design)

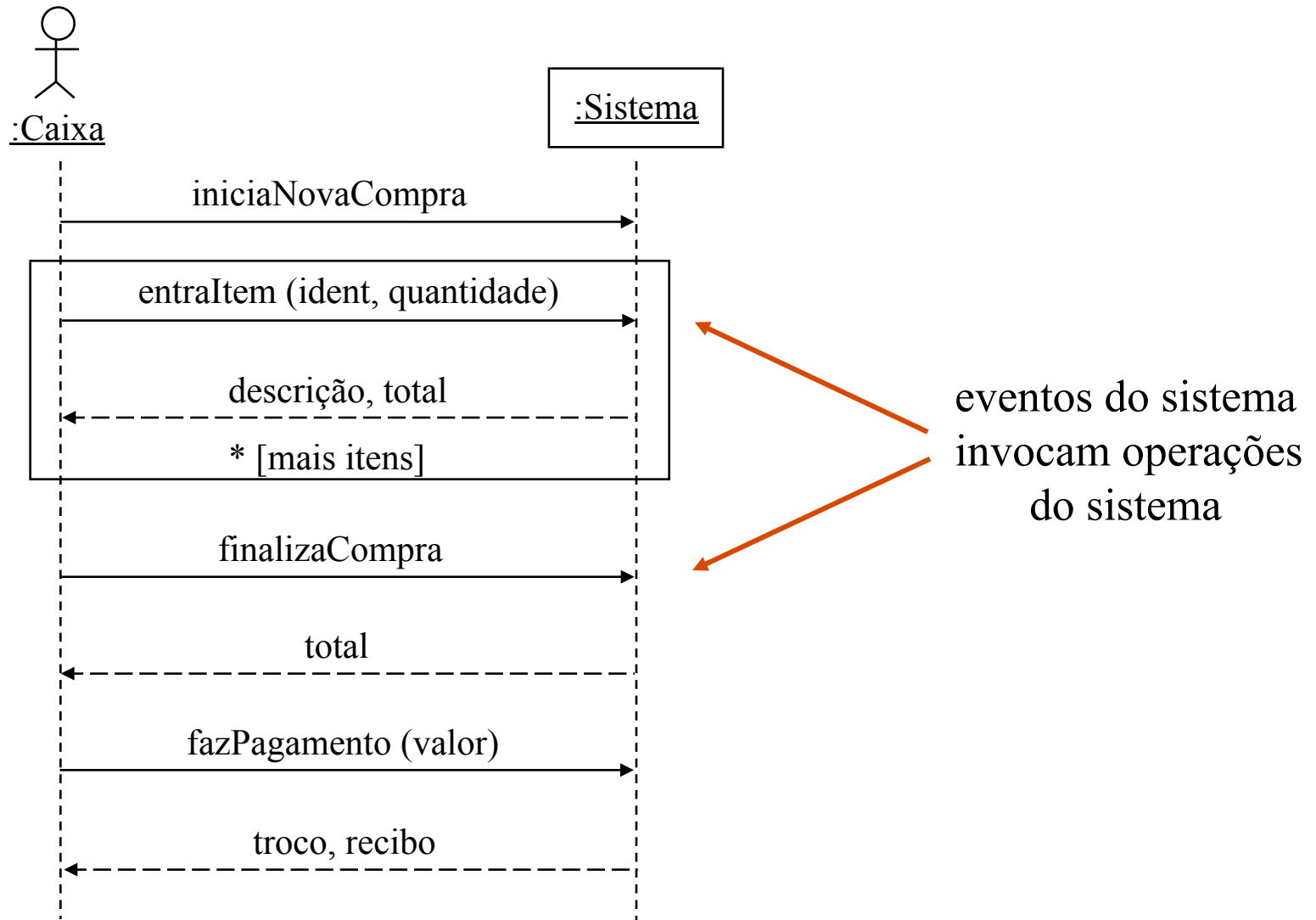
Contratos descrevem o comportamento do sistema em termos das mudanças dos estados dos **objetos do modelo conceitual**, como resultado de uma operação do sistema.

Operação do sistema: operação que o sistema (visto como uma caixa preta) oferece para manipular os eventos recebidos.

Exemplo: entra item, faz pagamento

➔ Os detalhes descritos nos contratos podem estar descritos nos casos de uso. Neste caso, os contratos não são necessários.

Contratos



Exemplo de Contrato

Contrato: entraltem

Operação: entraltem (ident, quantidade)

Referências cruzadas: caso de uso Processar Venda

Pré-condições:

- existe uma compra acontecendo

Pós-condições:

- uma instância de ItemCompra it foi criada
- it foi associada com a compra corrente
- it.quantidade recebeu quantidade
- it foi associado com uma EspecificacaoProduto baseado no ident



Contratos

Operação: nome da operação e seus parâmetros.

Referências cruzadas: (opcional) casos de uso nos quais a operação pode ocorrer.

Pré-condições: premissas sobre o estado do sistema ou objetos do modelo conceitual antes da execução da operação. Não precisarão ser testadas dentro da lógica desta operação.

Pós-condições: o estado dos objetos do modelo conceitual após a operação ter sido completada.

Pós-condições

Categorias das pós-condições:

- Criação e remoção de instâncias
- Modificação de atributos
- Associações formadas e desfeitas

Exemplos de pós-condições:

- Associação desfeita: exclusão de um item em uma compra desfaz a associação entre o item e a EspecificaçãoProduto.
- Remoção de instância: 7 anos após a falência de uma pessoa, todos os registros de declaração da falência devem ser destruídos.

↳ perspectiva conceitual e não de implementação



Contratos

Diretrizes para definição dos contratos:

1. Identifique as operações do sistema a partir dos diagramas de sequência do sistema.
2. Para as operações do sistema que são complexas ou que não estão bem especificadas no caso de uso, construa um contrato.
3. Descreva as pós-condições.

Exemplos de Contrato

Contrato: iniciaNovaCompra

Operação: iniciaNovaCompra ()

Referências cruzadas: caso de uso Processar Venda

Pré-condições:

- nenhuma

Pós-condições:

- uma instância de Compra co foi criada
- atributos de co foram inicializados

➔ Nem todos estes exemplos de contratos são necessários.

Exemplos de Contrato

Contrato: entraltem

Operação: entraltem (ident, quantidade)

Referências cruzadas: caso de uso Processar Venda

Pré-condições:

- existe uma compra acontecendo

Pós-condições:

- uma instância de ItemCompra it foi criada
- it foi associado com a compra corrente
- it.quantidade recebeu quantidade
- it foi associado com uma EspecificacaoProduto baseado no ident

Exemplos de Contrato

Contrato: finalizaCompra

Operação: finalizaCompra ()

Referências cruzadas: caso de uso Processar Venda

Pré-condições:

- existe uma compra acontecendo

Pós-condições:

- atributo compra.estaCompleta recebeu true

Exemplos de Contrato

Contrato: fazPagamento

Operação: fazPagamento (valor)

Referências cruzadas: caso de uso Processar Venda

Pré-condições:

- existe uma compra acontecendo

Pós-condições:

- uma instância de Pagamento pgto foi criada
- pgto foi associado com a compra corrente
- pgto.valor recebeu valor