Diagrama de Classes

Profa. Lisane Brisolara

Revisão conceitos OO

- Objeto: é uma entidade independente que armazena dados, encapsula serviços, troca mensagens com outros objetos para executar as funções do sistema.
- Classe: é uma descrição de um conjunto de objetos que compartilham os mesmos atributos, operações, relacionamentos e semântica.
  - Atributos
  - Operação (método)
    - Argumentos (Parâmetros)

Diagramas de Classes

- Descreve:
  - Classes do sistema, com seus atributos e operações
  - Relacionamentos entre classes: herança, agregação, composição...

- Representa a **visão estática** do sistema

Diagramas de Classes

- Pode ter três perspectivas diferentes:
  - Modelo Conceitual
  - Modelo de projeto
  - Modelo de Implementação

Perspectivas do Diag de Classes

- O modelo conceitual (análise) representa as classes no domínio do negócio em questão. Não leva em consideração restrições inerentes à tecnologia a ser utilizada na solução de um problema.

- O modelo de classes de especificação (projeto) é obtido através da adição de detalhes ao modelo anterior conforme a solução de software escolhida.

- O modelo de classes de implementação corresponde à implementação das classes em alguma linguagem de programação.

O que é uma classe?

- Conceitual: são agrupamentos de objetos, são abstrações de um coletivo de entidades do modelo de negócio (ou domínio de aplicação)

- O modelo genérico desse coletivo contém atributos e comportamentos comuns.

- Usualmente capturadas **a partir das descrições de casos de uso**
  - Procure por substantivos importantes

O que é uma classe?

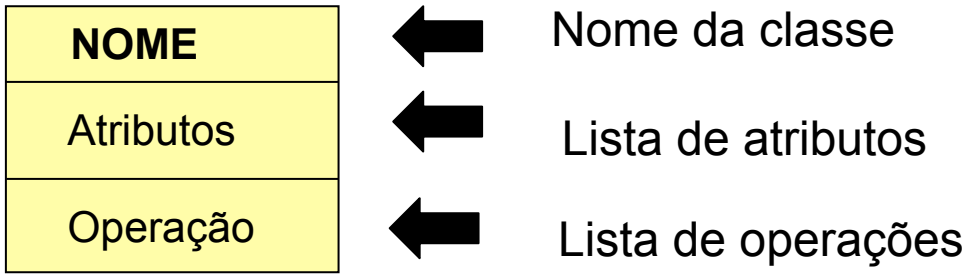
- De implementação: corresponde a um tipo de uma linguagem de programação

- Um modelo genérico para criar variáveis que armazenarão os objetos correspondentes.

Diagrama de Classes: Elementos

- Classes
- Relacionamentos

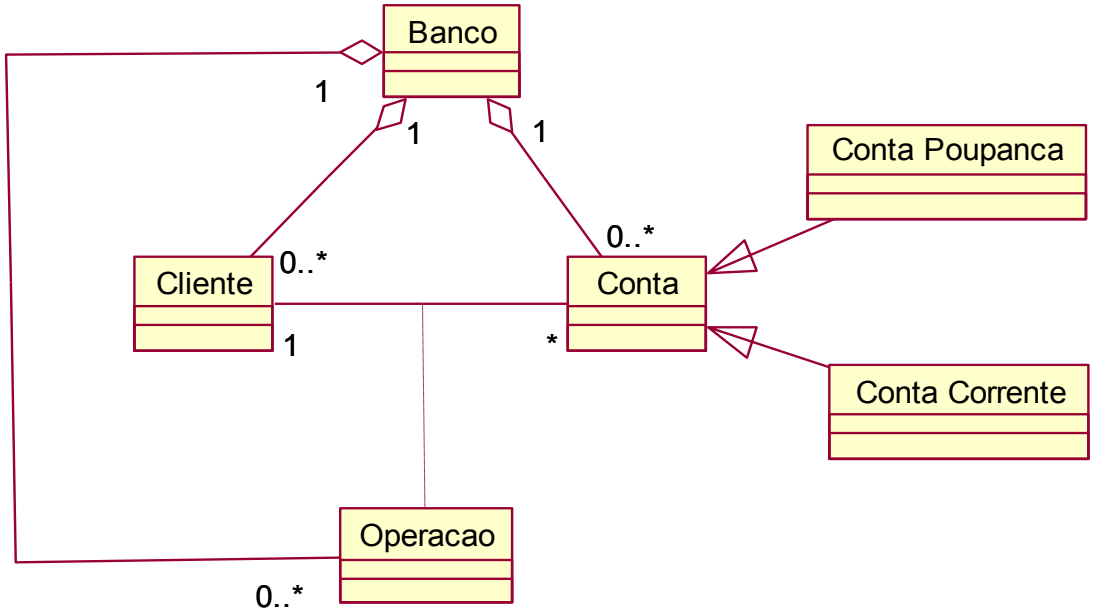
Representação de uma Classe



Relacionamentos

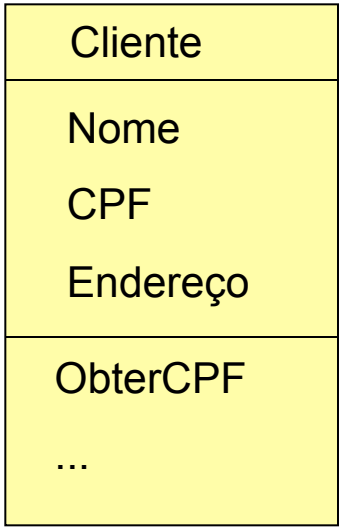
- Relacionamentos entre classes:
  - Associação
  - Agregação
  - Composicão
  - Herança

Relacionamento:
Representados por
linhas e conectores



Projeto das Classes

- Uma classe deve sempre versar sobre um mesmo assunto (alta coesão)
- Não faz sentido inserirmos em uma classe que modela um apartamento, coisas sobre a imobiliária ou atrasos de pagamento de hipoteca.
  - Assuntos diferentes -> classes diferentes



Atributo

- Característica, qualidade de um objeto ou classe
- Seus valores servem para diferenciar objetos (Instâncias)
- **Representação geral : nome e tipo**
- Representação detalhada:

[Visibili/d] Nome [Multiplici/d] : [Tipo] = [Valor] [{Proprie/ds}]

Operação

- Operação que pode ser realizada com objetos de uma dada classe

- Representação geral:
  - Nome da Operação
  - Argumentos (ou Parâmetros): dados de entrada e/ou saída para o método

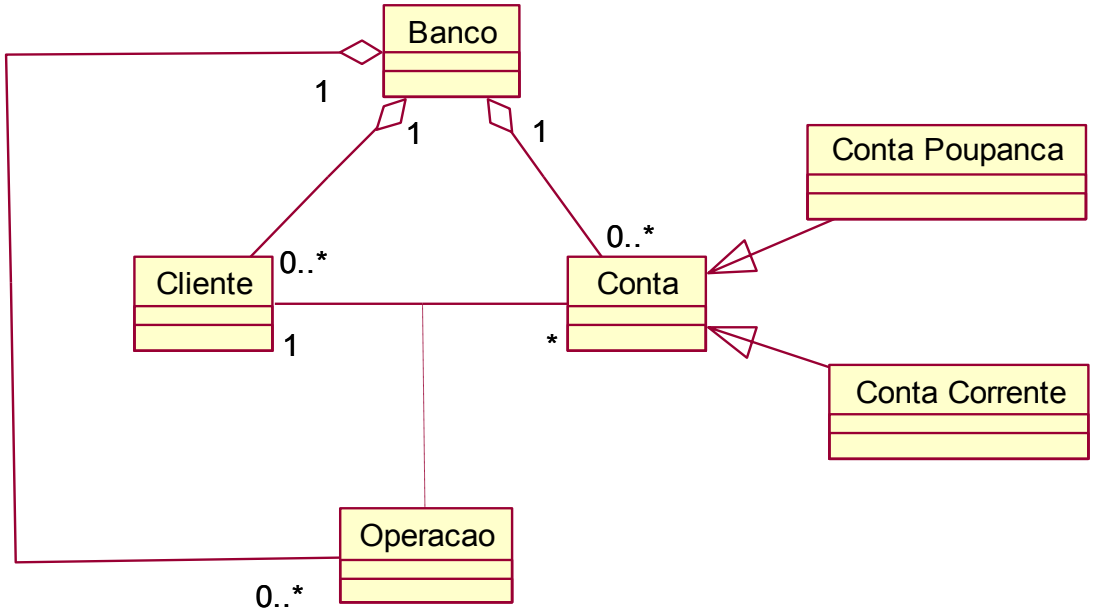
Ex: **desenhaRetangulo (largura, altura)**

Construção modelos de classes

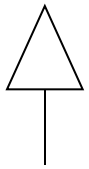
- **Na visão conceitual**: cada classe pode ser vista como um conceito ou um tipo, e os métodos são identificados numa fase posterior.
- **Na visão de implementação**: os métodos aparecem obrigatoriamente e consideramos aspectos de controle, estereótipos, pacotes, etc.
- Podem existir outras visões intermediárias (por exemplo: de domínio e de aplicação, (análise) de especificação(projeto)

Diagrama de Classes: Relacionamentos

- Relacionamentos entre classes:
  - Associação
  - Agregação
  - Composicão
  - Herança

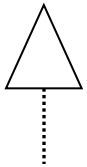


UML: Relacionamentos



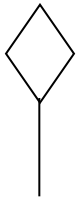
Generalização/Especialização (herança) : é um(a)

Associação



Realização de interfaces

Dependência



Agregação: parte de

Composição: agregação mais forte

Multiplicidade:

1 - exatamente 1 (default)

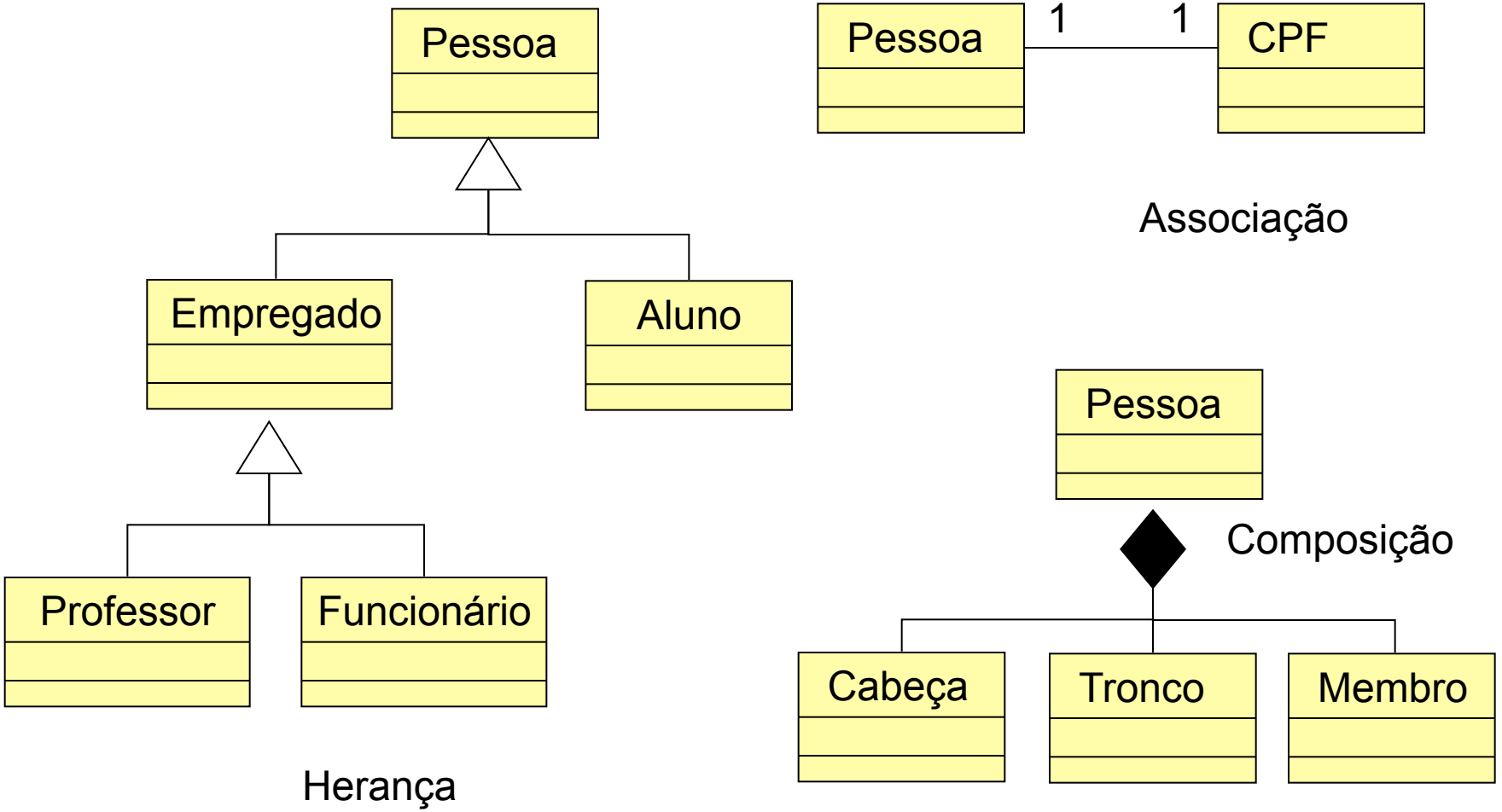
* ou 0..* - muitos (eventualmente nenhum)

1..* - muitos (pelo menos 1)

n - exatamente n

Diagrama de Classes: Relacionamentos

Relacionamento entre classes

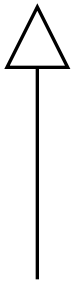


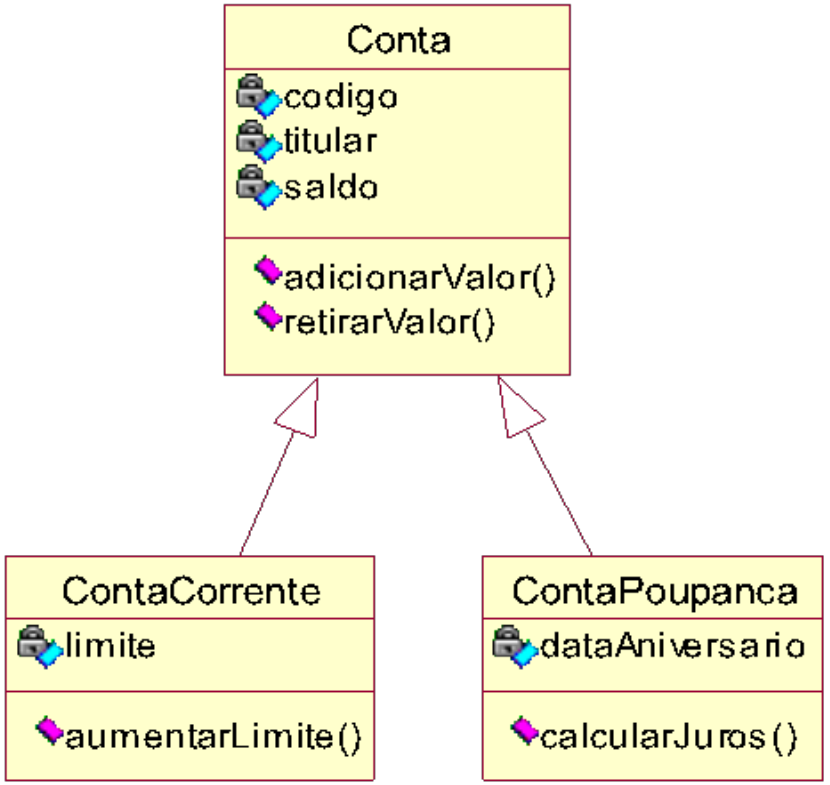
Herança

Herança

◆ É a capacidade de uma classe definir o seu comportamento e sua estrutura aproveitando definições de outra classe, normalmente conhecida como classe base ou classe mãe.

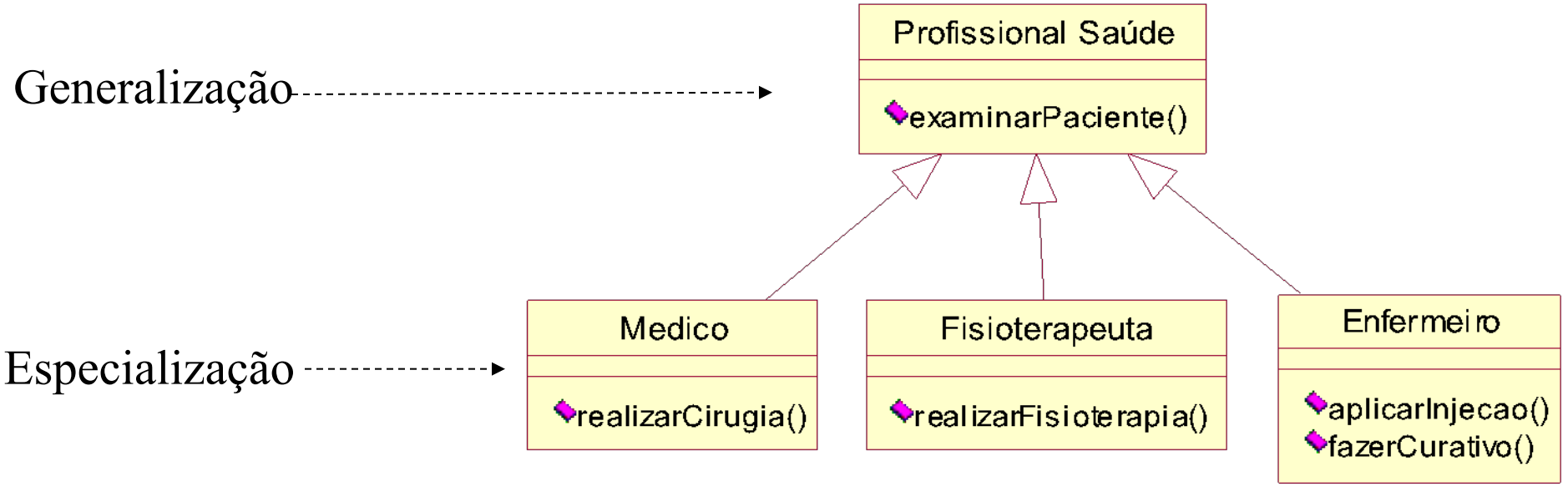
Note que as subclasses herdam tudo o que a classe pai possui e acrescenta as suas características particulares.





Especialização e Generalização

- Através do mecanismo de **herança** é possível definir classes genéricas que agreguem um conjunto de definições comuns a um grande número de objetos (Generalização).
- A partir destas especificações genéricas pode-se construir novas classes, mais específicas, que acrescentem novas características e comportamentos aos já existentes (Especialização).

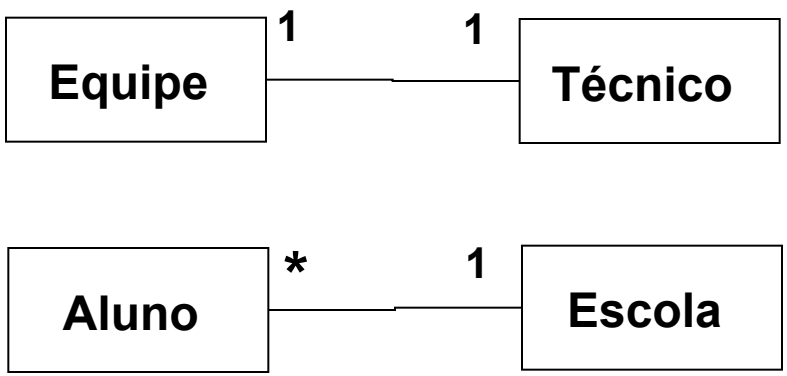


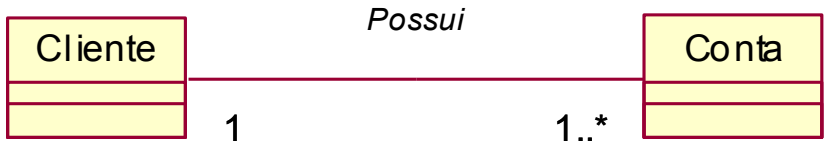
Relacionamento entre Classes

Associação

- A associação entre classes indica que existe um link entre seus objetos, ou seja, uma conexão entre eles.

Multiplicidade: número de instâncias de uma classe que é associado a uma única instância de outra classe.





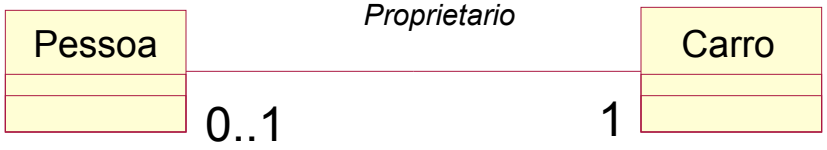
Relacionamento entre Classes

Associação

- Associações podem ser bidirecionais ou unidirecionais por

Unidirecionais: uma seta indica a direção na qual a associação pode ser percorrida.

unidirecional



bidirecional

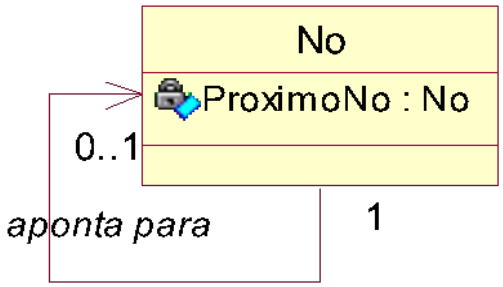
Seta dupla ou sem linha sem seta

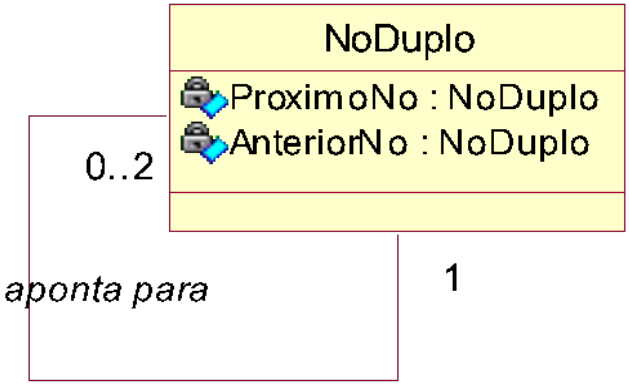
As setas indicam a navegabilidade da associação

Relacionamento entre Classes

Associação recursiva

- Uma associação recursiva representa uma associação entre objetos da mesma classe.



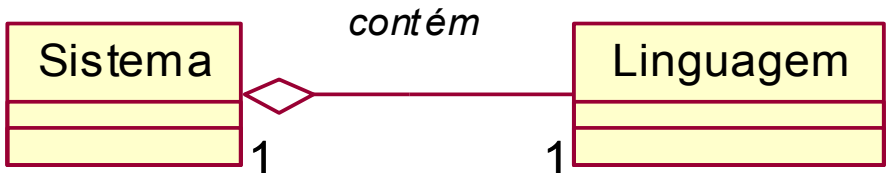


Relacionamento entre Classes

Agregação (relacionamento "todo-parte")

- conceito mais abstrato: composto (todo)
- conceito mais concreto: componente (parte)

- Relacionamento do tipo "parte de", nos quais objetos representando os componentes são associados com objetos representando uma montagem.
- Agregação é uma forma de associação com alguma semântica adicional. É transitiva (se A é parte de B e B parte de C, então A é parte de C)

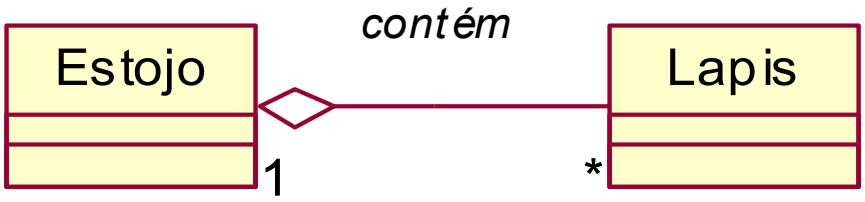


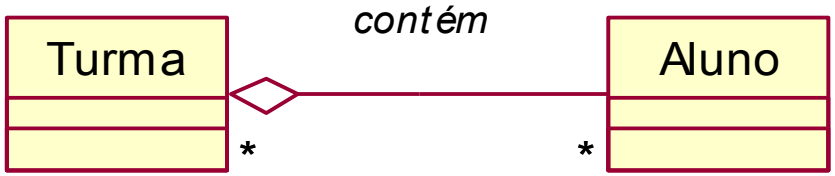
1 sistema contém 1 linguagem

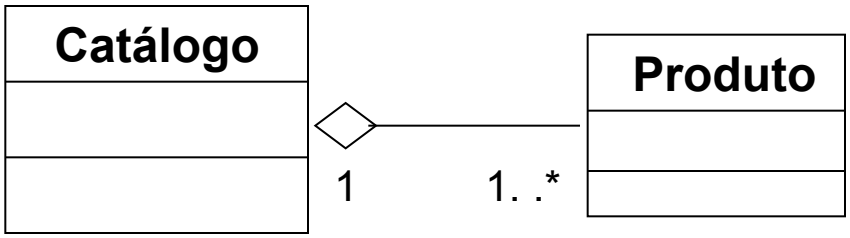
Relacionamento entre Classes

Agregação

- Agregação Compartilhada é um tipo especial de agregação que ocorre quando uma das classes é uma parte, ou está contida na outra, mas esta parte pode estar contida na outra várias vezes em um mesmo momento.





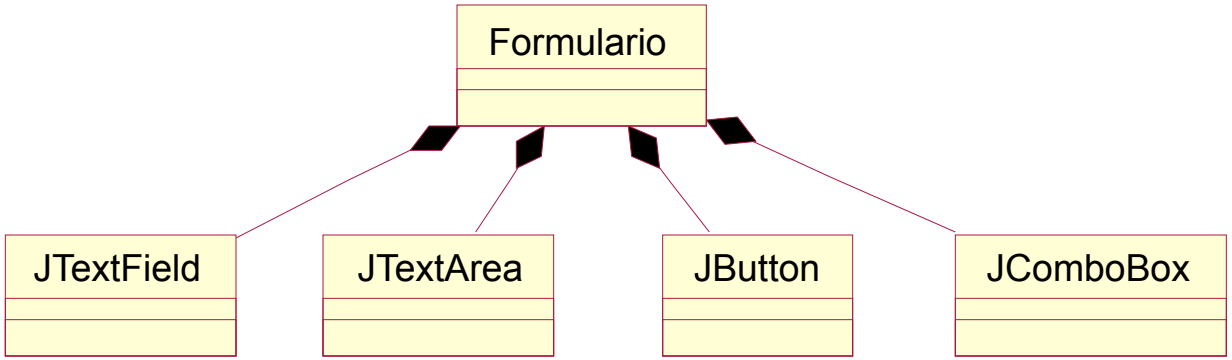


Relacionamento entre Classes

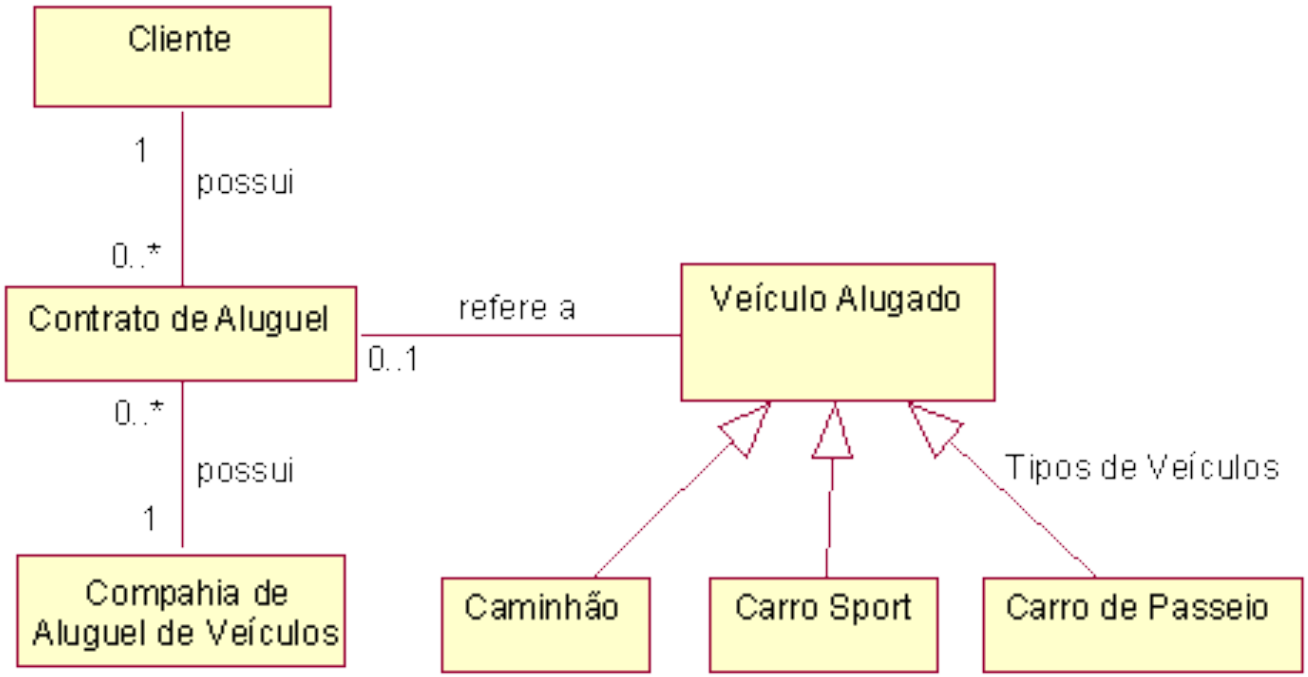
Composição

- É uma forma mais forte de agregação. Na composição, o objeto parte pode pertencer somente a um todo e espera-se que as partes vivam e morram com o todo.

- Se o objeto da classe que contém for destruído, as classes da composição serão destruídas juntamente.



Exemplo sistema de Aluguel de Veículos



Acoplamento e Coesão

- Boa prática: alta coesão e baixo acoplamento
- Coesão: A classe só versa sobre um assunto?

- Acoplamento trata de relacionamentos
  - Quantos mais ligações maior acoplamento
  - Muito cuidado ao definir relacionamentos
  - Classes muito relacionadas devem estar no mesmo pacote

Notações adicionais

- Visibilidade de atributos
- Representação detalhada de operações
- Modificadores especiais

Visibilidade de Atributos

+ : visibilidade **pública**: o atributo é visível no exterior da classe.

- : visibilidade **privada**: o atributo é visível somente por membros da classe.

: visibilidade **protegida**: o atributo é visível também por membros de classes derivadas (ou subclasses)

Encapsulamento: Visibilidade de seus atributos e de operações

Notações para operações

*[Visibili/d]**Nome(Parâmetros)**]:[Retorno][{Proprie/ds}]*

- Visão conceitual

  ImprimirData (data:TipoData)

- Visão de implementação

  ArmazenarDados (*nome:char[30],salario:float=0.0*)

Modificadores

- *Static*: usado em atributos e operações
  - Atributos estáticos
    - São da classe e não da instância
    - Único para todos os objetos da mesma classe
  - Operação estática é da classe
    - Para ser invocada não se usa objeto.operaçao()
    - **Se usa Classe.operação()**

Modificadores

- Modificadores de uma classe
  - *Abstract*: classe abstrata, nunca será instanciada (itálico no nome da classe)
    - Também pode ser usado para operação (só protótipo sem implementação)
  - *Final*: classe completa que não terá subclasses
    - Também pode ser usado para atributos constantes

Modificadores

- *Interface*: elemento que define uma interface (métodos abstratos)
  - Classes ligadas a esta interface precisam implementar estes métodos definidos na interface

Referências

- Fowler, M. UML Essencial.
- Medeiros, E. Desenvolvendo software com UML2.0

- Complementar:

http://www.inf.ufpr.br/silvia/ESNovo/UML/pdf/ModeloConceitualAI.pdf