Aula 5 – POO 1
Análise OO

Profa. Elaine Faria

UFU - 2021

Sobre o Material

- Agradecimentos
  - Aos professores José Gustavo e Fabiano, por gentilmente terem cedido seus materiais.

- Os slides consistem de adaptações e modificações dos slides dos professores José Gustavo e Fabiano

Introdução

- **Análise**
  - Detalha “o que deve ser feito”, detalha requisitos do sistema
- **Projeto**
  - Detalha “como será feito”, cria modelos de como o sistema será construído
- **Programação**
  - “Faz”, constrói o sistema

Introdução

- **Programação orientada a objetos**
  - **Método de implementação de software** no qual os programas são organizados na forma de coleções cooperativas de **objetos**
  - Cada um dos objetos representa uma instância de uma **classe**
- **Linguagem orientada a objetos**
  - Linguagem que suporta os mecanismos utilizados na programação orientada a objetos

Análise O. O.

- Método de análise que examina os *requisitos* a partir da perspectiva das classes e objetos encontrados no vocabulário do domínio do problema
- O produto da análise são modelos que servem como entrada para o projeto, que por sua vez produz os modelos que são utilizados para a programação

Análise O. O.

- Definir a missão do produto (para que ele servirá)
- Estabelecer os requisitos → ***o que*** o sistema vai fazer, sem se importar em como ele vai fazer
  - Isso envolve muita interação com o "cliente" do sistema
  - Os requisitos podem ser descritos por frases e por diagramas

Requisitos de Software

- A captura de requisitos do software é uma atividade fundamental para qualquer processo de desenvolvimento
- Um requisito é uma caraterística ou propriedade que deve ser implementada num sistema
- Seu objetivo é estabelecer os serviços que devem ser fornecidos pelo sistema e as restrições sob as quais ele deve operar

Análise O. O

- **Método**: linguagem de modelagem + processo
  - Linguagem de Modelagem → notação (principalmente gráfica) utilizada por métodos para expressar projetos
  - Processo → passos a serem seguidos na elaboração de um projeto

Unified Modeling Language (UML)

- **UML: linguagem de modelagem unificada**
  - Não é uma linguagem de programação
  - É uma linguagem de modelagem, utilizada pararepresentar o sistema de software sob os seguintes aspectos:
    - Requisitos
    - Comportamento
    - Estrutura lógica
    - Dinâmica de processos
    - Comunicação/Interface com os usuários

Universal Modeling Language (UML)

- A UML auxilia na construção do software, porque permite um mapeamento de seus modelos para as linguagens de programação

- Facilita a documentação, pois possui suporte para a criação e documentação de vários dos artefatos que são gerados durante o desenvolvimento de um sistema

Universal Modeling Language (UML)

- Por que modelar um sistema?
  - Um sistema computacional é, de modo geral, excessivamente complexo
  - Necessário decompô-lo em pedaços compreensíveis
  - Criação de diagramas auxiliam no entendimento do problema
  - Linguagem única que permite a todos os desenvolvedores entender quais objetos fazem parte do sistema e como eles se comunicam

Universal Modeling Language (UML)

- Diagramas

Diagramas Estáticos	• Diagrama de Classes • Diagrama de Objetos • Diagrama de Componentes • Diagrama de Implantação
Diagramas Dinâmicos	• Diagrama de Casos de Uso • Diagrama de Seqüência • Diagrama de Atividades • Diagrama de Colaborações • Diagrama de Gráficos de Estados

Diagrama de Classes

- O Diagrama de Classes
  - É responsável pela tarefa de modelar as classes e seus relacionamentos
  - Lista todos os conceitos do domínio que serão implementados no sistema e as relações entre os conceitos
  - É muito importante pois define a estrutura do sistema a desenvolver

Diagrama de Classes

- Passos
  1. Identificar as classes
  2. Identificar os atributos das classes
  3. Analisar os atributos, identificando que alguns deles são na realidade relacionamentos
  4. Analisar classes semelhantes, remodelando as com relacionamentos de herança
  5. Detalhar os atributos e métodos

Identificando Classes

- Recomendação prática: procurar substantivos na análise do sistema
  - Verbos: métodos
- Exemplo: processamento de pedidos
  - Item
  - Pedido
  - Endereço de remessa
  - Pagamento
  - Conta

Identificando Classes

- Métodos
  - "Itens são ***adicionados*** aos pedidos. Pedidos são ***encaminhados*** ou ***cancelados***. Pagamentos são ***aplicados*** aos pedidos"
  - Identificar, para cada método, o objeto responsável por executá-lo
    - Ex.: adicionar item
  - IMPORTANTE: experiência deve ajudar na decisão

Criando Diagramas de Classe

- Uma classe é representada por um retângulo dividido em 3 partes, cada uma armazenando
  - Nome da classe (no singular)
  - Lista de atributos
  - Lista de operações

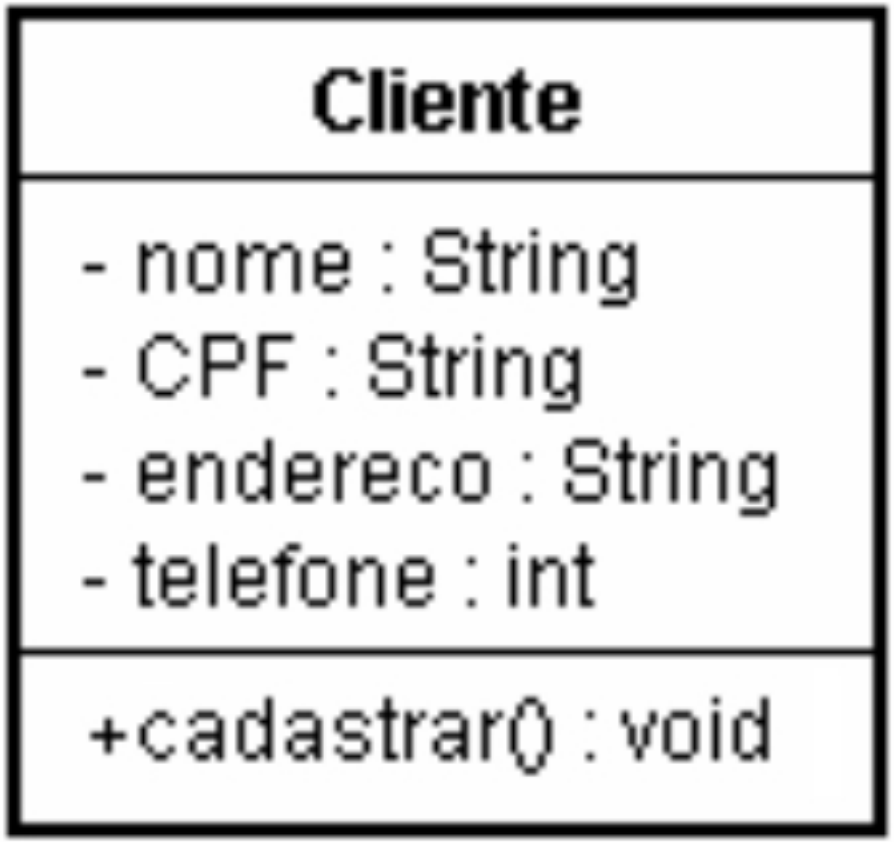
Criando Diagramas de Classe

- Nome da classe
  - Centralizado e negritado
  - Com suas letras iniciais maiúsculas, inclusive as primeiras letras de nomes compostos
- Atributos e operações
  - Formatação normal e alinhados à esquerda
  - Primeira letra deve ser minúscula, e as letras subseqüentes podem ser maiúsculas
  - Siglas são mantidas inalteradas

Criando Diagramas de Classe

- Operações
  - Geralmente contém um verbo e um complemento e terminam com um par de parênteses
  - Ex: criar(), bloquear(), creditar(), desbloquear(), ...

Exemplo de Classe



Conceitos

- Visibilidade → identifica por quem uma propriedade (atributo ou operação) pode ser utilizada
- Tipos de visibilidade
  - + ou *public*
    - A propriedade será vista e usada dentro da classe na qual foi declarada, em qualquer elemento externo e nas classes descendentes
  - # ou *protected*
    - A propriedade será vista e usada apenas dentro da classe na qual foi declarada, e pelas classes descendentes

Conceitos

- Tipos de Visibilidade (continuação)
  - - ou *private*
    - A propriedade será vista e utilizada apenas dentro da classe na qual foi declarada
  - ~ ou *package*
    - A propriedade poderá ser vista e utilizada por elementos que estejam declarados dentro do mesmo pacote no qual está inserida a classe que a declarou

Criando Diagramas de Classe

- Sintaxe padrão de atributos

visibilidade nome : tipo [multiplicidade *ordering*] = valor inicial {string-propriedade}

  - Visibilidade → informa quais elementos podem utilizar este atributo
  - Nome → nome do atributo
  - Tipo → tipo de dado representado pelo atributo (inteiro, string, outra classe,...)
  - Multiplicidade → informa quantos valores possíveis deste atributo a classe pode possuir

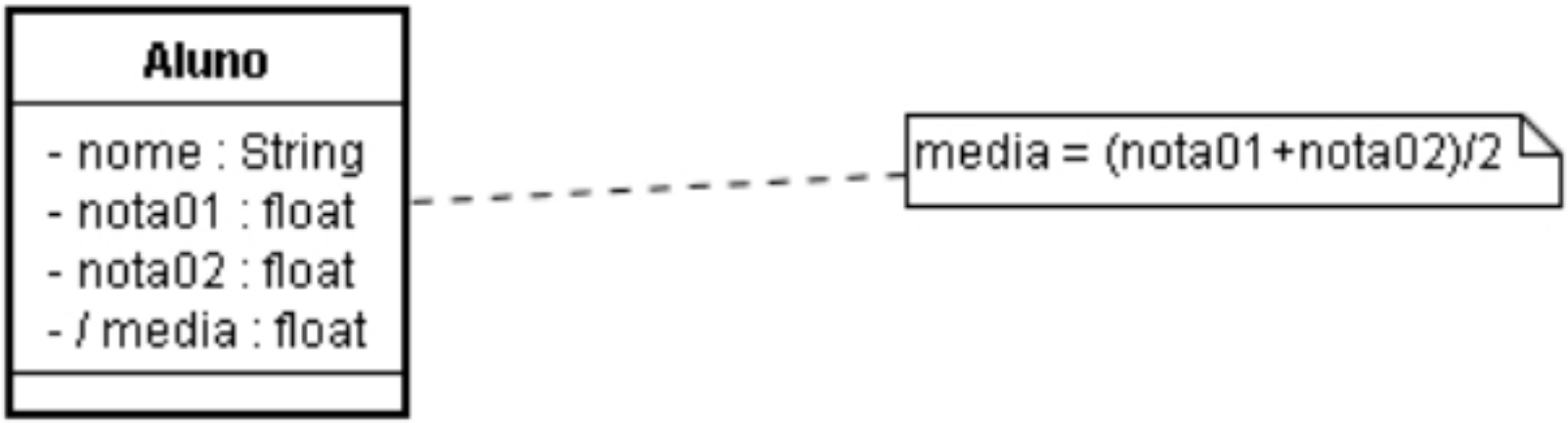
Criando Diagramas de Classe

- *Ordering* → Aplicado apenas quando a multiplicidade for maior que um. Indica se os valores dos atributos estarão em ordem ou não
- Valor inicial → valor do atributo no momento de inicialização de uma instância da classe
- String-propriedade → define regras sobre a utilização do atributo

Criando Diagramas de Classe

- O escopo padrão de um atributo é o de instância
- Para representar um atributo com escopo de classe, deve-se sublinhar o nome e o tipo do atributo
- Atributo derivado → seu valor é computado a partir do valor de outros atributos
  - É identificado graficamente com o símbolo “/” antes de seu nome
  - A forma do cálculo do valor de um atributo derivado pode ser representada por uma nota ligada ao nome do atributo

Criando Diagramas de Classe



Criando Diagramas de Classe

- Sintaxe padrão para operações

  visibilidade nome (lista de parâmetros): tipo de retorno {string-propriedade}

  – Visibilidade → informa quais elementos podem invocar esta operação
  – Nome → nome da operação
  – Tipo de retorno → tipo de dado retornado pela operação
  – String-propriedade → define regras para a utilização da operação. Pode indicar também detalhes sobre seu funcionamento

Criando Diagramas de Classe

- Para a lista de parâmetros, temos uma lista separada por vírgula, onde cada parâmetro possui a sintaxe padrão

  escopo-parâmetro nome : tipo = valor padrão

  – Escopo-parâmetro

    in → parâmetro de entrada, não é possível escrever nele

    out → parâmetro de saída, não é possível ler dele

    in-out → parâmetro de entrada e saída. É o tipo padrão

  – Nome → nome do parâmetro

  – Tipo → tipo de dado representado pelo parâmetro

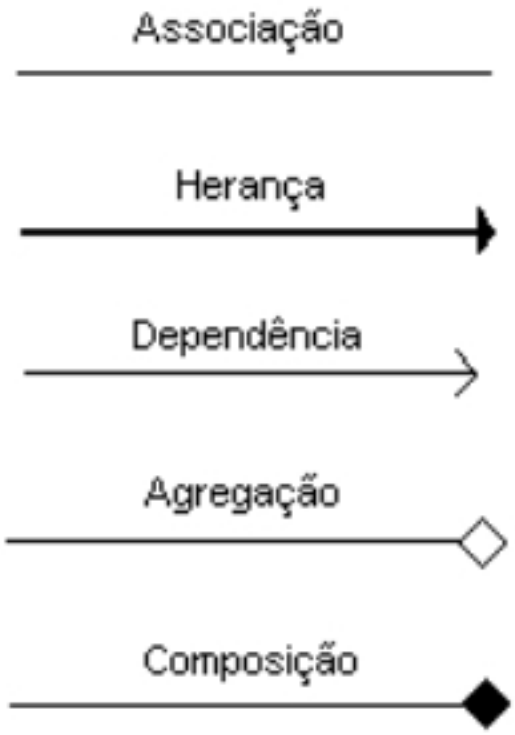
  – Valor padrão → valor inicial atribuído ao parâmetro, caso este seja passado como nulo para a operação

Método X Operação

- As operações são definições conceituais dos serviços de uma classe, enquanto os métodos são implementações das operações
- O corpo de um método pode ser mostrado no diagrama de classes por meio de uma nota ligada à operação

Relacionamento

- Os objetos de um sistema podem se relacionar uns com os outros
  - O relacionamento possibilita a troca de mensagens



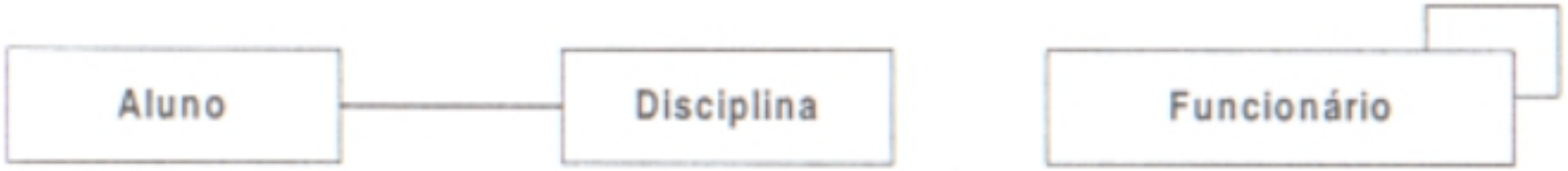
Relacionamentos - Associação

- É um relacionamento que conecta duas ou mais classes, demonstrando a colaboração entre as instâncias de classe
- Podem ser unárias, binárias ou n-árias
- É representada por uma linha sólida que liga a(s) classe(s)
- Exemplo:
  - Cliente compra produtos
  - Uma conta corrente possui um histórico de transações

Relacionamentos - Associação

- Associação ("faz um","tem um", etc)
  - Classes que possuem como atributos objetos de outras classes
  - Ex: Classe Pedido possui varios objetos da classe Item

Relacionamentos - Associação



Relacionamentos - Associação

- Existem adornos que ajudam a obter uma melhor compreensão dos objetivos e características dos relacionamentos
  - Nome: descrição do relacionamento
  - Sentido de leitura
  - Multiplicidade: 0..1, 0..*, 1, 1..*, 2, 3..7
  - Tipo
  - Papéis

Relacionamentos - Associação

- Nome da associação
  - Realiza uma pequena explicação sobre a definição da associação
  - Posicionado na linha da associação

Relacionamentos - Associação

- Direção da leitura ou Navegabilidade
  - Indica como a associação deve ser lida
  - Indica de onde e para onde vai a associação
  - Representada por uma seta, partindo da extremidade da classe relacionadora, em direção à classe relacionada
  - Quando omitidas, indicam bidirecionalidade, ou então desconhecimento
  - A classe para a qual o sentido aponta é aquela cujos objetos não possuem visibilidade dos objetos da outra classe

Relacionamentos - Associação

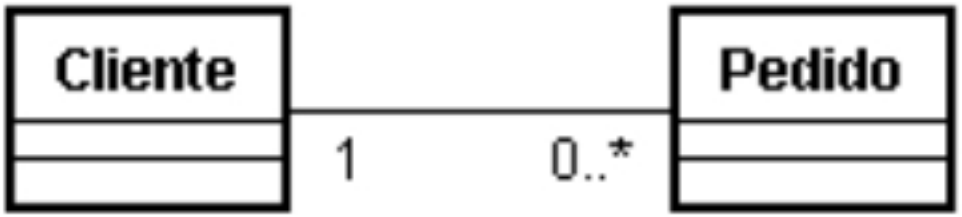
- Multiplicidade
  - Representar a informação dos limites inferior e superior da quantidade de objetos aos quais outro objeto pode estar associado
  - Cada associação possui 2 multiplicidades, uma em cada extremo da linha
  - É representada por uma string compreendida numa seqüência de intervalos inteiros, separados por vírgula, no qual um intervalo representa uma faixa de inteiros no formato: LIMITE INFERIOR .. LIMITE SUPERIOR

Relacionamentos - Associação

- Multiplicidade
  - Se houver um asterisco em uma multiplicidade, significa que podemos ter uma faixa infinita de valores
    - 0..1 → zero ou um
    - 1 ou 1..1 → apenas um
    - 0..* → zero ou muitos
    - 1..* → um ou muitos

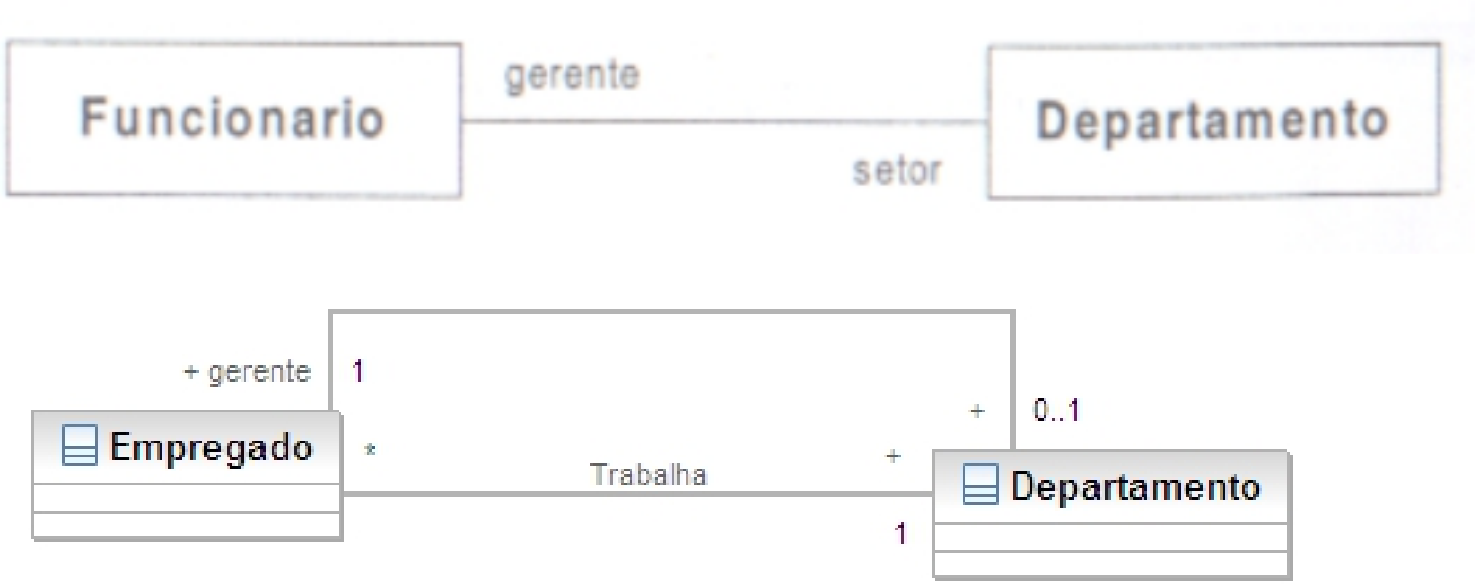
Relacionamentos - Associação

- Multiplicidade
  - As associações podem ser agrupadas em apenas três tipos
    - Muitos para muitos
    - Um para muitos
    - Um para um

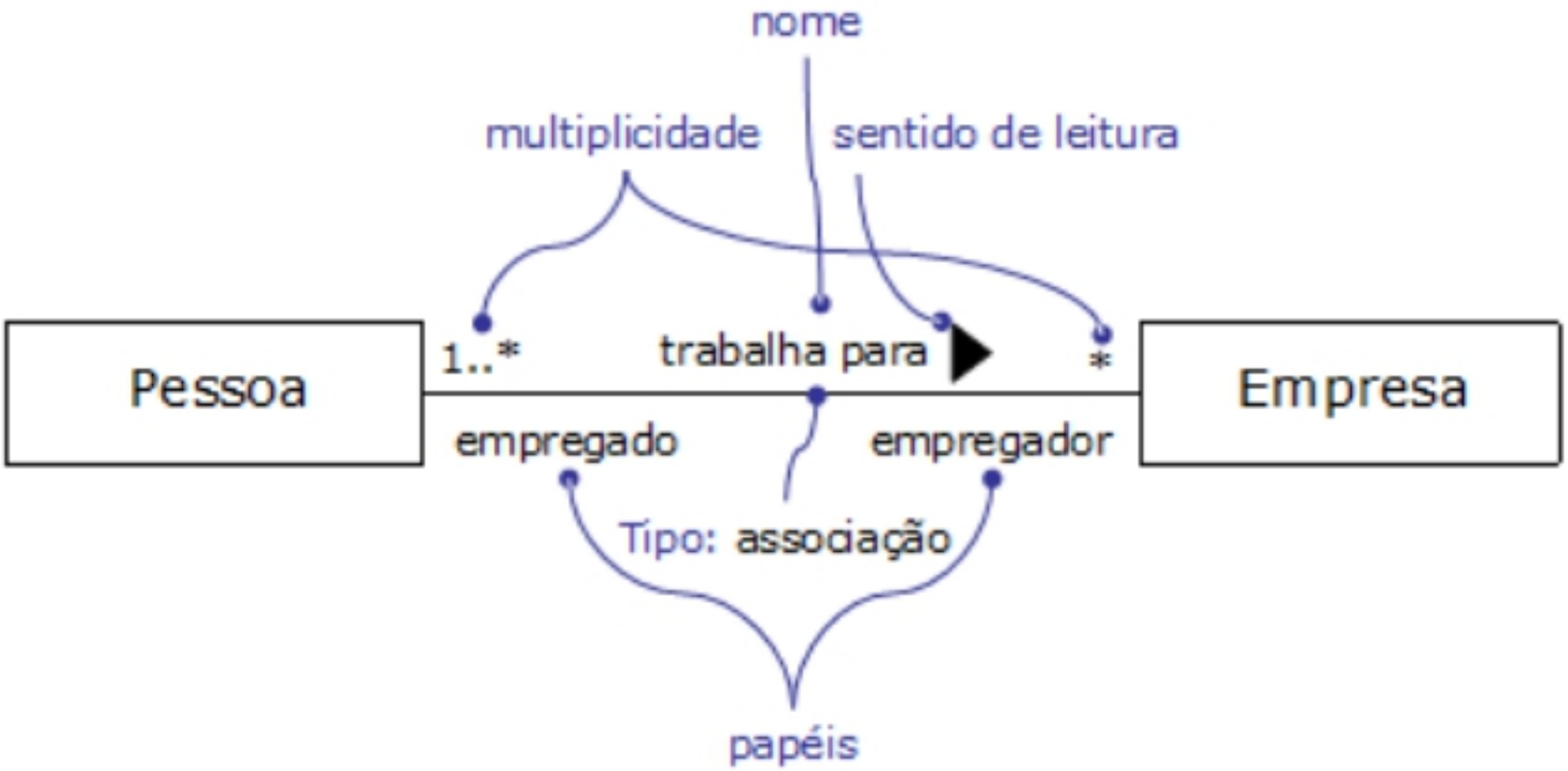


Relacionamentos - Associação

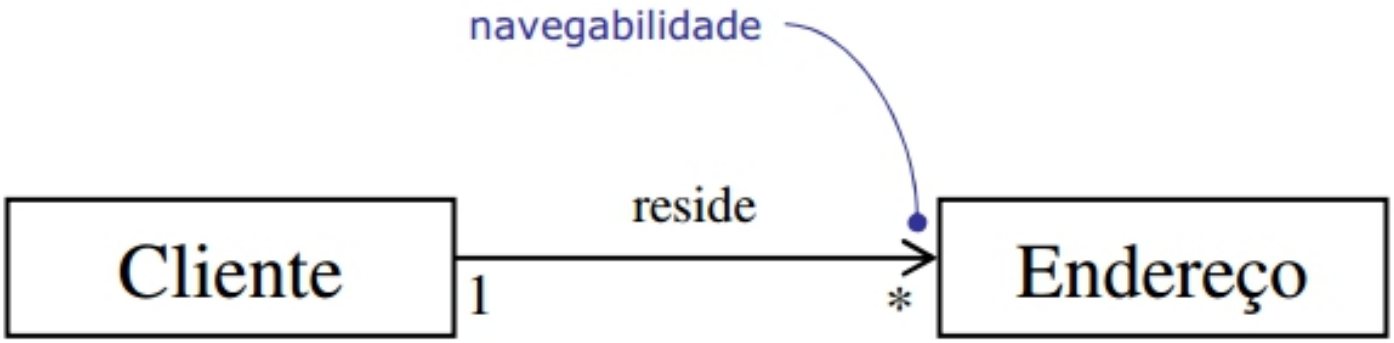
– Papel
  - Indica o papel representado pela classe na associação



Relacionamentos entre Classes



Relacionamentos entre Classes



O cliente sabe quais são seus endereços, mas o endereço não sabe a quais clientes pertence

Relacionamentos - Associação

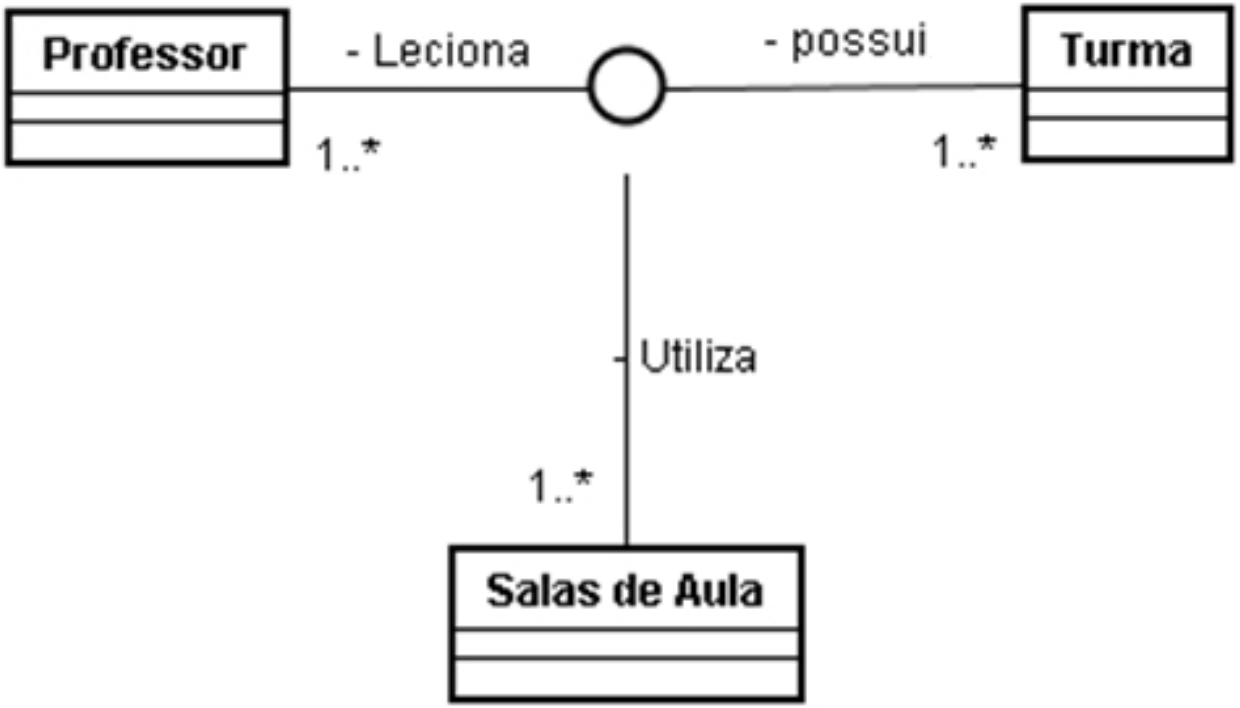
- A classe funcionário possui, cada qual, uma referência para um objeto Departamento
- Um objeto departamento não tem referência para os objetos associados a Funcionário

Relacionamentos - Associação

- Grau de uma associação: quantidade de classes envolvidas em uma associação
  - Na maioria dos casos as associações são binárias → entre 2 classes
  - Para associações n-árias, insere-se um diamante (ou um círculo), que faz a junção destas conexões

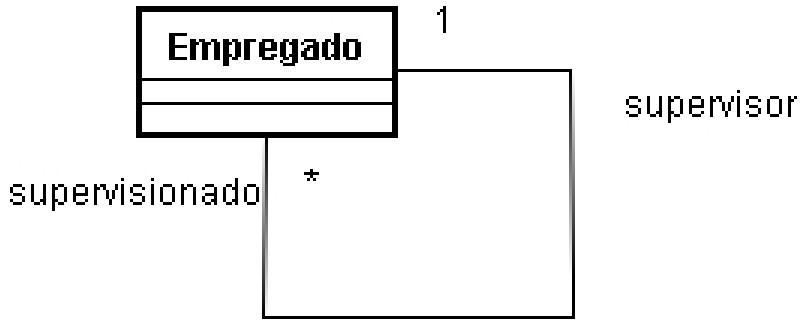
Relacionamentos - Associação

Relacionamentos - Associação



Relacionamentos - Associação

- Associações Reflexivas
  - Também denominada auto-associação
  - Associa objetos da mesma classe
  - Cada objeto tem um papel distinto nessa associação



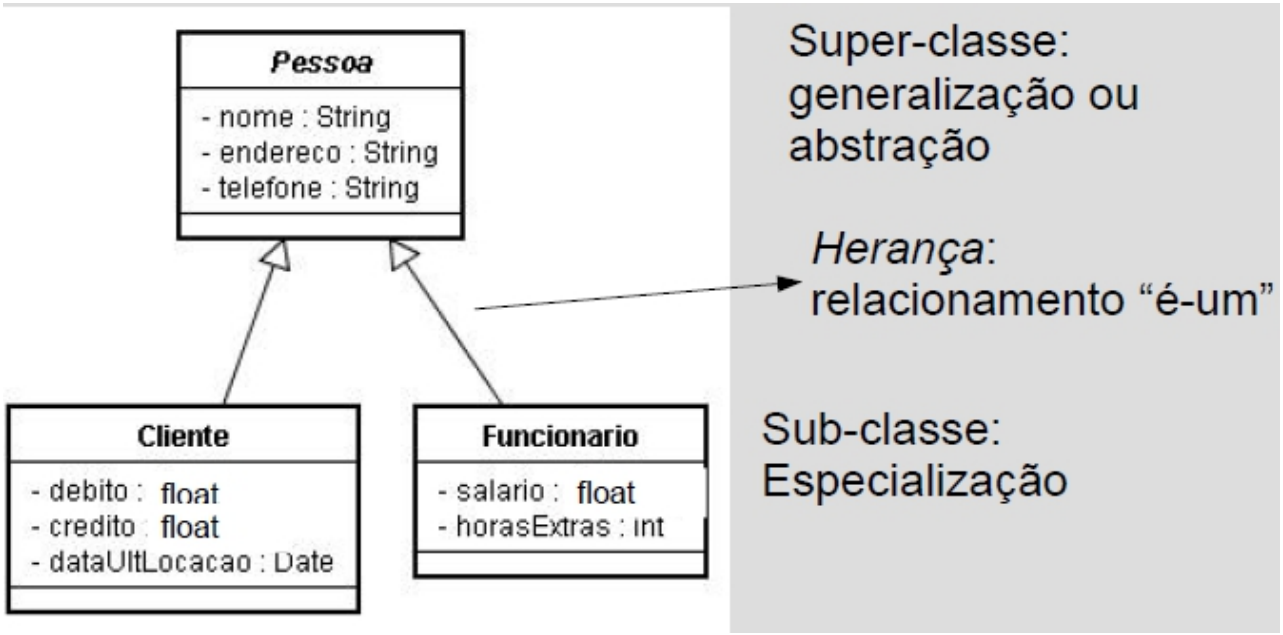
Relacionamentos - Herança

- Representa relacionamentos do tipo “é um”
  - Ex: um cachorro é um mamífero
- Generalização/especialização
  - A partir de duas ou mais classes, abstrai-se uma classe mais genérica
  - De uma classe geral, deriva-se uma mais específica
  - Sub-classes possuem todas as propriedades das superclasses
  - Deve existir pelo menos uma propriedade que distingue duas classes especializadas
    - Caso contrário, não há necessidade

Retirado de: https://edisciplinas.usp.br/pluginfile.php/4247716/mod_resource/content/1/Aula3_UML.pdf

Relacionamentos - Herança

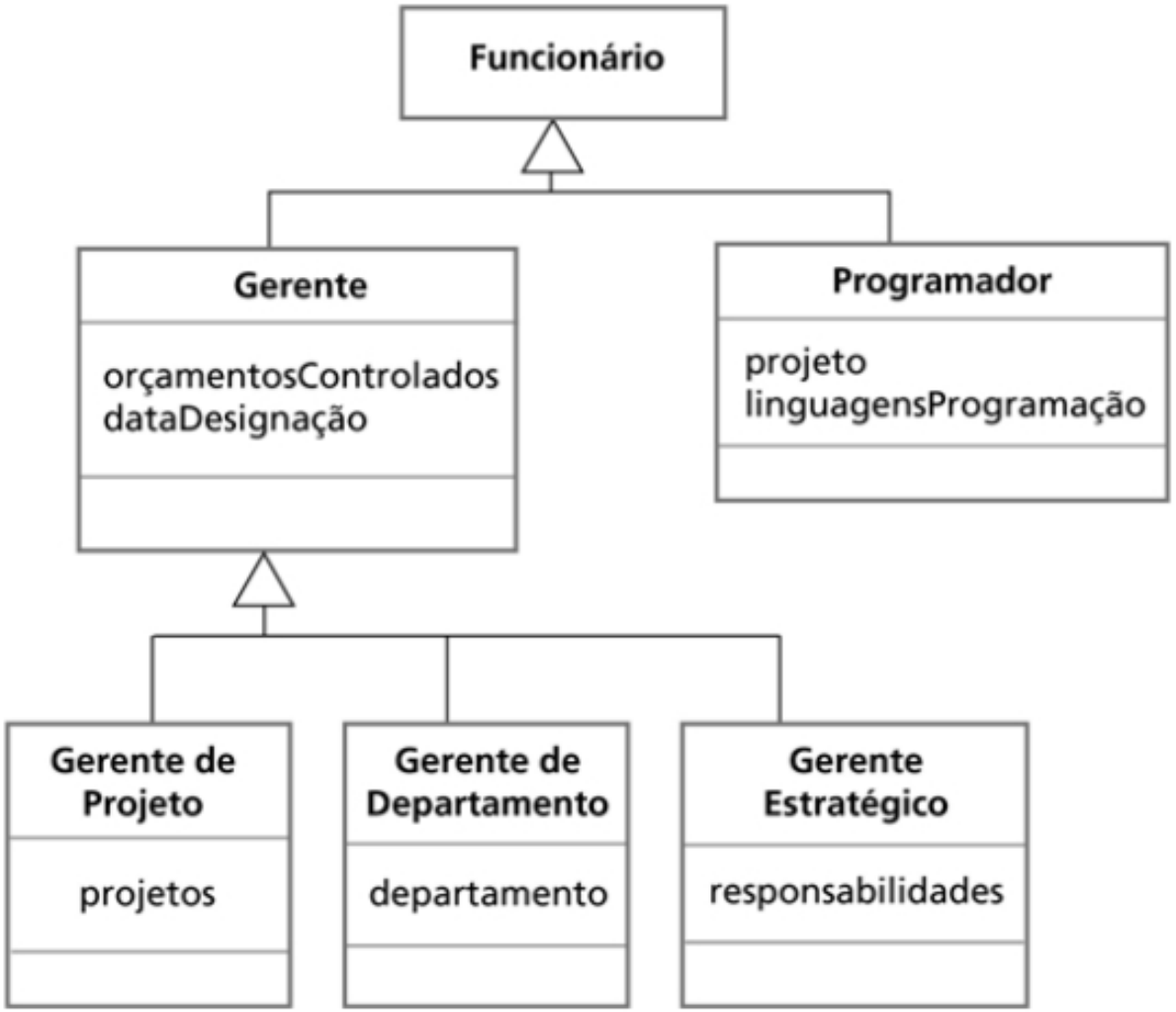
- No diagrama de classes
  - A generalização é representada por uma seta do lado da classe mais geral (classe base)



Relacionamentos - Herança

- Permite organizar as classes hierarquicamente
  - Técnica de reutilização de software
  - Novas classes são criadas a partir de classes existentes, absorvendo seus atributos e comportamentos (métodos)
  - Recebe novos recursos posteriormente

Relacionamentos - Herança



Relacionamentos Todo-Parte

- Correspondem a um caso particular da associação
- Indica que um dos objetos está contido no outro
- A UML define dois tipos de relacionamento todo-parte (casos especiais de associação)
  - **Agregação**
  - **Composição**

Relacionamentos Todo-Parte

- Características
  - Se um objeto A é parte de um objeto B, o objeto B não pode ser parte do objeto A
  - Um comportamento que se aplica a um todo automaticamente se aplica às suas partes
  - As partes são normalmente criadas e destruídas pelo todo

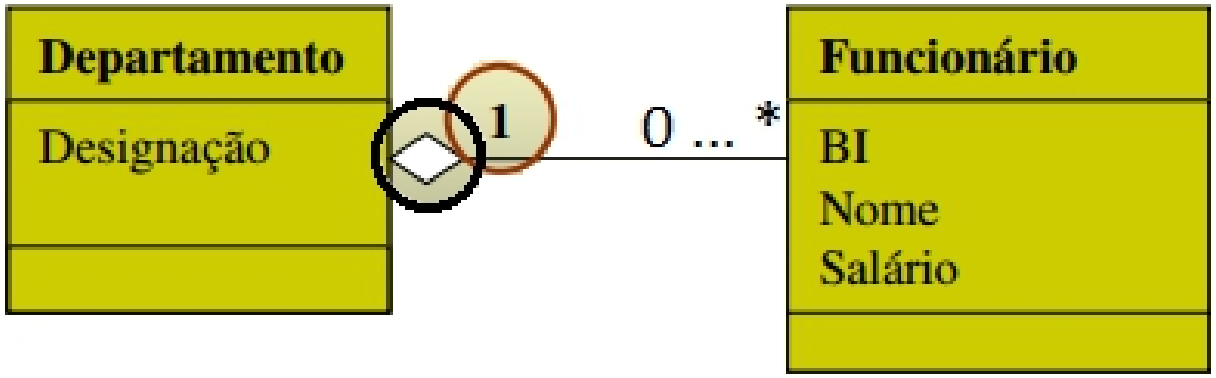
Relacionamentos - Agregação

- Tipo de associação onde um todo (que é uma classe) é formado ou composto por uma ou diversas partes (que são outras classes)
- Tanto o todo quanto as partes podem existir independentemente, mas ganham um novo significado quando estão juntas
- Uma parte pode fazer parte de diversos todos

Relacionamentos - Agregação

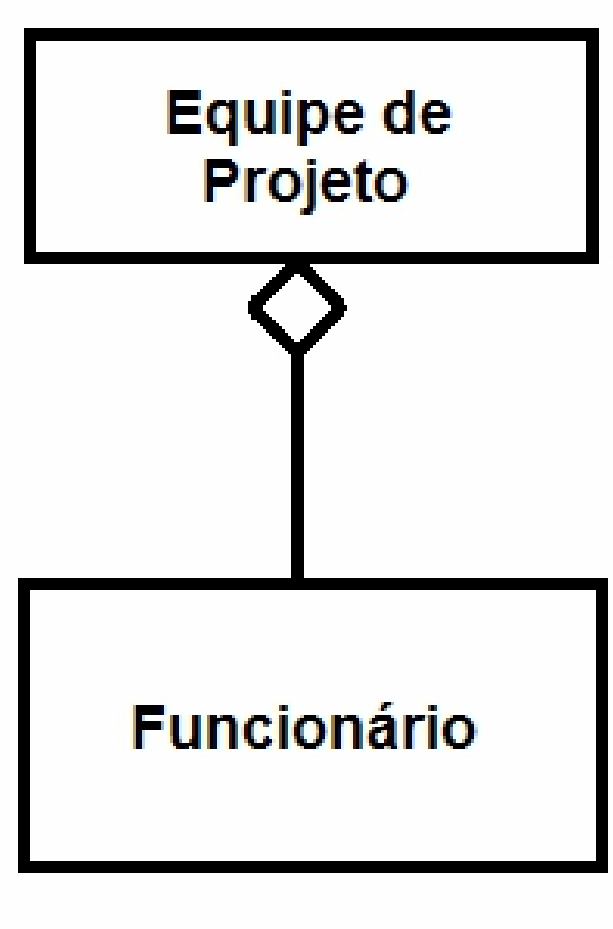
- Consiste de um objeto contendo referências para outros objetos, de tal forma que o primeiro seja o todo, e que os objetos referenciados sejam as partes do todo
- Exemplo
  - A associação entre uma universidade e seus departamentos é uma agregação

Relacionamentos - Agregação



- Um funcionário que não trabalhe num departamento não é relevante para o domínio em causa (se o seu departamento for removido ele terá que ser excluído ou reposicionado em outro departamento - mas isso PODERÁ ser feito)
- O funcionário existe per si, e não necessita estar associado a um departamento para ser referido/identificado
- O funcionário pode se associar com vários departamentos simultaneamente, dependendo do exemplo (não é o caso aqui)

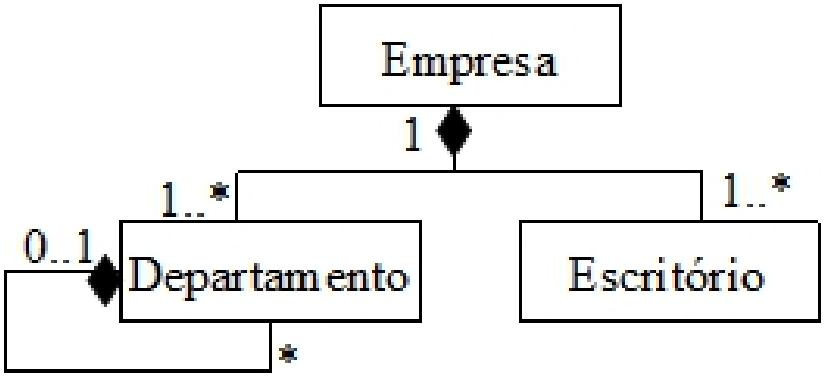
Relacionamentos - Agregação

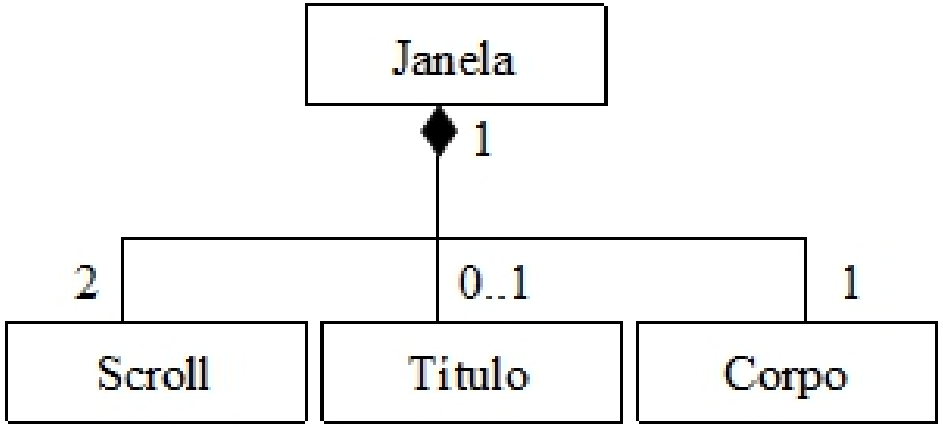


Relacionamentos - Composição

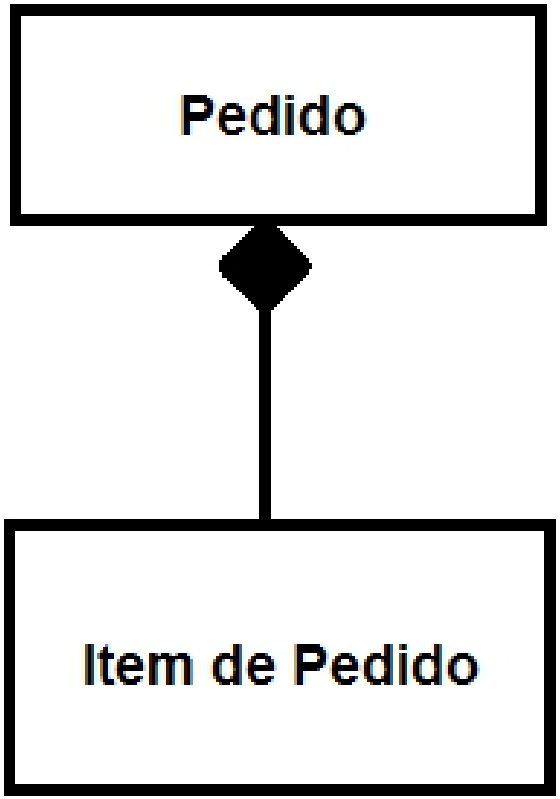
- Relacionamento mais forte, mais físico entre o todo e suas partes
- A classe parte pertence só e somente só à uma determinada classe todo, em um determinado momento, não podendo fazer parte de outro relacionamento de composição
- Parte não existe sem o todo; Seu tempo de vida coincide com o todo
- Se a classe todo deixa de existir, também todas as suas partes deixam de existir também

Exemplos de Composição

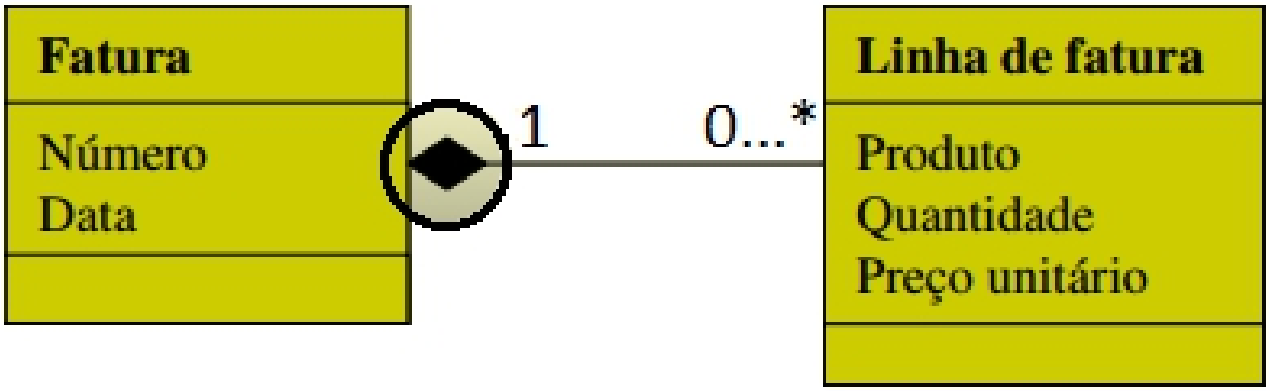




Relacionamentos - Composição

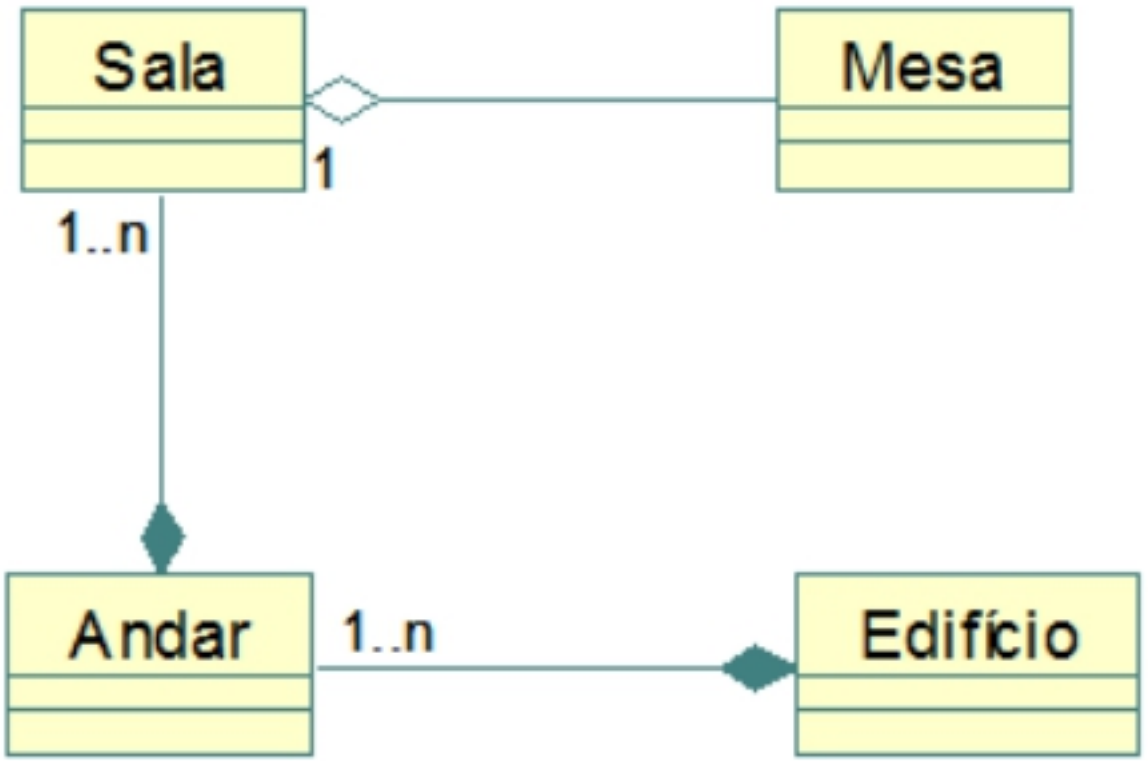


Exemplos de Composição



- A Linha da fatura só pode ser referida (distinguida das restantes) se for indicada a fatura correspondente

Agregação X Composição



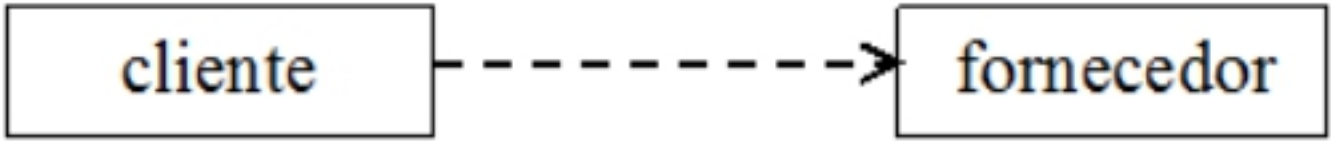
Diferenças entre agregação e composição

- Na agregação a destruição de um objeto todo não implica a destruição do objeto parte
- Na composição os objetos parte pertencem a um único todo

Dependência

- Dependência ("usa um")
  - Um objeto precisa acessar dados de outro
  - Métodos de uma classe utilizam ou manipulam objetos de outra classe
    - Classe **Pedido** necessita da classe **Conta** para verificar status de crédito
- Minimização de classes dependentes (acoplamento)

Dependência



- A classe cliente depende de algum serviço da classe fornecedor
- A mudança de estado do fornecedor pode afetar o objeto cliente
- A classe cliente não declara nos seus atributos um objeto do tipo fornecedor
- Fornecedor é recebido por parâmetro de método

Relacionamentos - Dependência

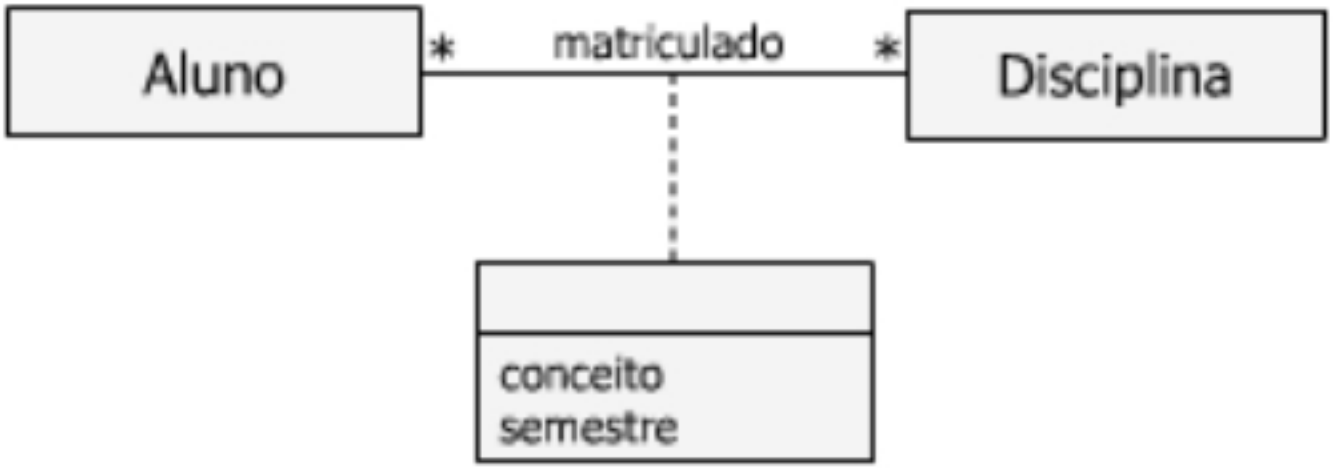
- Relacionamento que indica que mudanças na interface de um dos relacionados pode causar mudanças no outro relacionado
- Demonstra um certo grau de dependência

Figura: https://homepages.dcc.ufmg.br/~amendes/GlossarioUML/glossario/conteudo/classe/relacionamentos_entre_classes/dependencia_entre_classes.htm

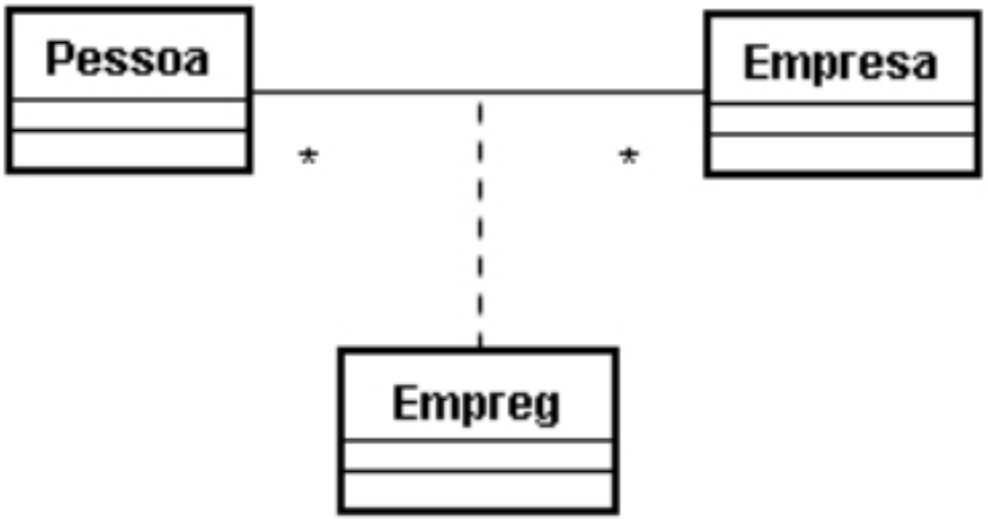
Classe Associativa

- Produzidas quando da ocorrência de associações que possuem multiplicidade muitos (*) em todas as suas extremidades
- Classe para armazenar os atributos transmitidos pela associação

Classe Associativa

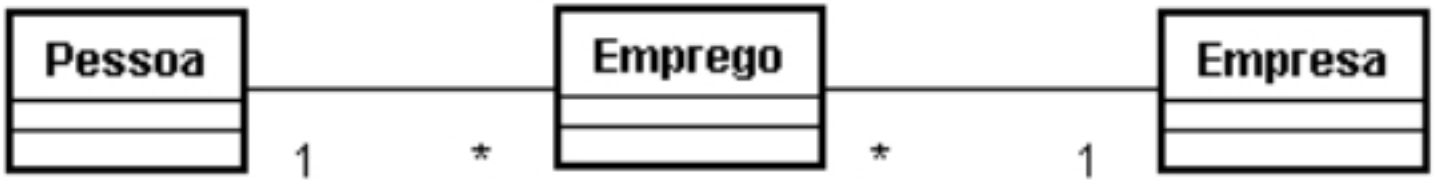


Classe Associativa



Classe Associativa

- O diagrama pode ser modificado para retirar a classe associativa sem perda de informação no modelo



Modelando Um Diagrama De Classes

1. Identificar as classes
2. Identificar os atributos das classes
3. Analisar os atributos, identificando que alguns deles são na realidade relacionamentos
4. Analisar classes semelhantes, remodelando as com relacionamentos de herança
5. Lançar detalhes dos atributos

Exercício

- Imagine um sistema desenvolvido para uma Locadora de automóveis
  - Faça a análise dos possíveis requisitos para esse sistema
  - Escreva as classes e relacionamentos possíveis

Bibliografia

- Bezerra E.; Princípios de Análise e Projeto de Sistemas com UML; 2ª edição; Editora Campus, 2007.