Análise de Sistemas Orientados a Objetos

Prof. Tiago Eugenio de Melo
tiago@comunidadesol.org

www.tiagodemelo.info

Roteiro

- Conceitos de Orientação a Objetos (OO)
- Visão Geral da UML
- Diagrama de Classes
- Diagramas de Casos de Uso

Conceitos de Orientação a Objetos

- Histórico da OO
  - SmallTalk foi a primeira linguagem OO desenvolvida no início da década de 1970.
  - Outras linguagens:
    - C++
    - Object Pascal
    - Eiffel
    - Java

Conceitos de Orientação a Objetos

- Principais conceitos:
  - Objeto
  - Classes
  - Encapsulamento
  - Herança
  - Polimorfismo

Exercícios

- Identifique classes com seus atributos dos seguintes contextos:
  - Numa turma de um curso de especialização, temos disciplinas ministradas em salas diferentes.
  - Está passando na rede de cinemas ArtFilme o filme “Jogos2”, todos os dias, em três sessões por dia. Aos sábados e domingos existem em algumas sessões duas salas de exibição.
- Se dois desenvolvedores modelarem uma mesma classe X para sistemas distintos, obrigatoriamente as classes terão os mesmos atributos e operações? Por que?
- Qual é a diferença entre operações e métodos?

Visão Geral da UML

- Histórico da UML
  - Evolução das metodologias de desenvolvimento.
  - Análise e projeto estruturado (Yourdon, 1979).
  - Metodologias OO (Booch, 1991; OMT, Rumbaugh; Jacobson, 1992).
  - A UML é formada a partir das metodologias desses três autores.
  - O início da unificação foi em 1994.
  - A versão inicial foi a 0.9 em 1996.
  - Em janeiro de 1997 a Rational lançou a versão 1.0 da UML.
  - Ainda neste ano a proposta foi aceita pela OMG (Object Management Group).
  - No ano de 2000 foi lançada a UML 2.0.

Visão Geral da UML

- O que é UML?
  - Linguagem visual para modelar sistemas de software.
- Objetivo da UML?
  - Simplificar e consolidar métodos já conhecidos.
- Características?
  - Não é proprietária.
  - Tem propósito geral (especificar, visualizar, construir e documentar).
  - É independente do processo de desenvolvimento.

Visão Geral da UML

- Quais são os aspectos abordados pela UML?
  - Aspectos estáticos
    - Tipos de objetos e relacionamentos entre eles.
  - Aspectos dinâmicos
    - Evolução dos objetos no tempo e interação entre eles.
  - Aspectos do ambiente
  - Aspectos organizacionais
    - Particionamento de grandes sistemas.
    - Representação de decisões de implementação.
    - Implantação do sistema (organização em tempo de execução).

Exercícios

- O que é a UML?
- A UML pode ser empregada para documentação de sistemas?
- Quais são os propósitos da UML?

Diagrama de Classes

- Conceitos Gerais
  - Visibilidade
    - Identifica por quem uma propriedade (atributo ou operação) pode ser utilizada.
    - Resumo das visibilidades possíveis:

+ ou public (público)
ou protected (protegido)
- ou private (privado)
~ ou package (pacote)

    - Exemplo:

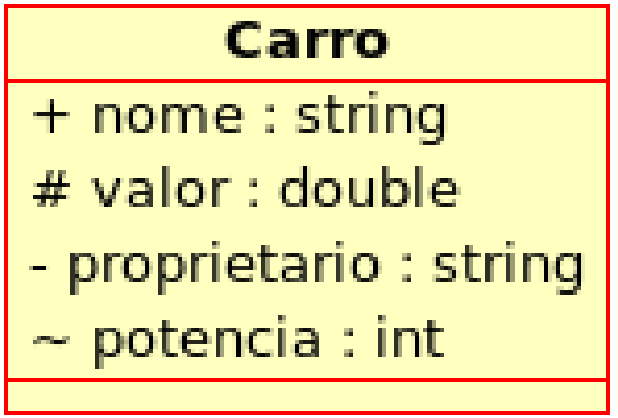


Diagrama de Classes

- Conceitos Gerais:
  - Multiplicidade
    - Indica a faixa de cardinalidade permitida a um elemento, isto é, a quantidade de instâncias possíveis em um relacionamento.
    - As multiplicidades mais comuns são:
      - 0..1 (valor opcional).
      - 1 ou 1..1 (exatamente um).
      - * ou 0..* (qualquer valor inteiro não-negativo).
      - 1..* (qualquer valor inteiro positivo).

Diagrama de Classes

- Conceitos Gerais:
  - Exemplo de multiplicidade

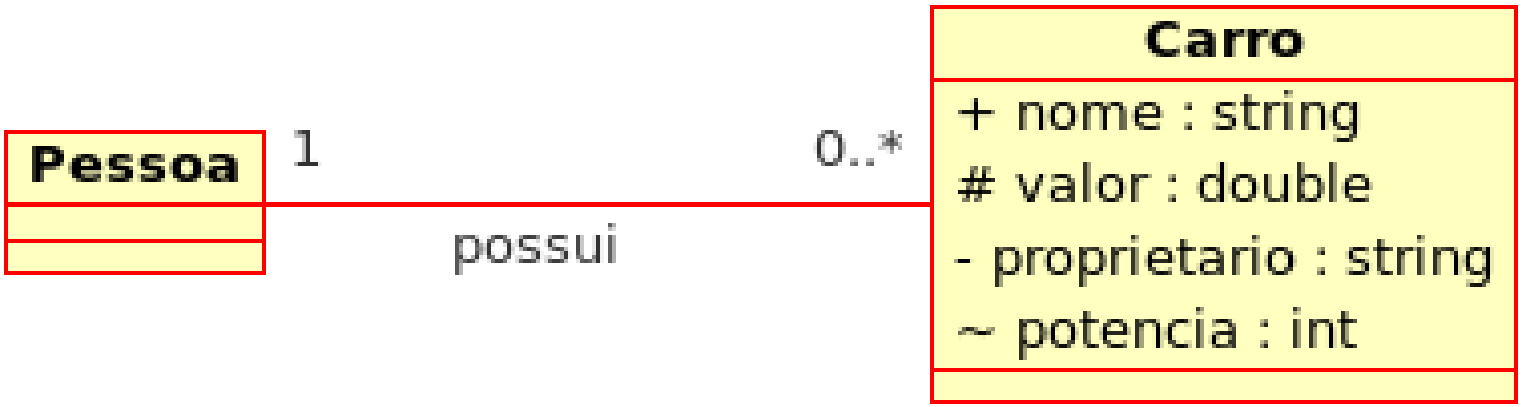


Diagrama de Classes

- Conceitos Gerais:
  - Estereótipo
    - É um mecanismo de extensibilidade da UML que representa uma subclasse de um elemento já existente com o mesmo formato porém com objetivos diferentes e bem definidos.
    - É usado geralmente quando se deseja distinguir um uso específico para elemento.
    - O estereótipo possui a mesma representação gráfica do elemento, sendo colocado seu nome entre guillemets (<< >>).
    - Exemplo:

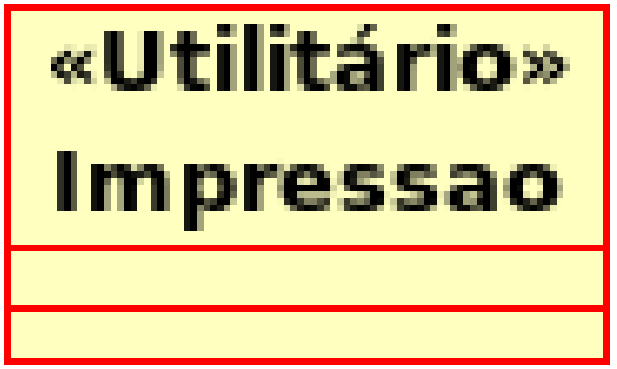


Diagrama de Classes

- Conceitos Gerais:
  - Notas
    - É um símbolo gráfico contendo informação textual.
    - Exemplo:

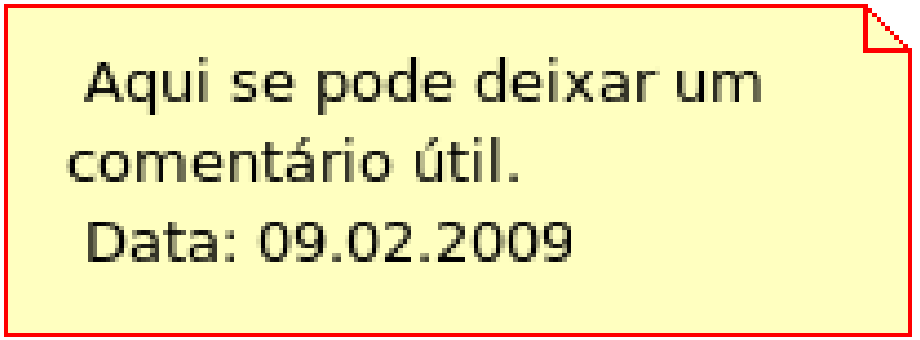


Diagrama de Classes

- Conceitos Gerais:
  - Restrições (*constraints*)
    - As restrições podem aparecer em diversos elementos da UML.
    - Uma restrição é mostrada como um texto entre chaves.
    - Exemplo:
      - `{valor > 1000,00}`

Diagrama de Classes

- Criando Diagramas de Classe
  - Exemplo

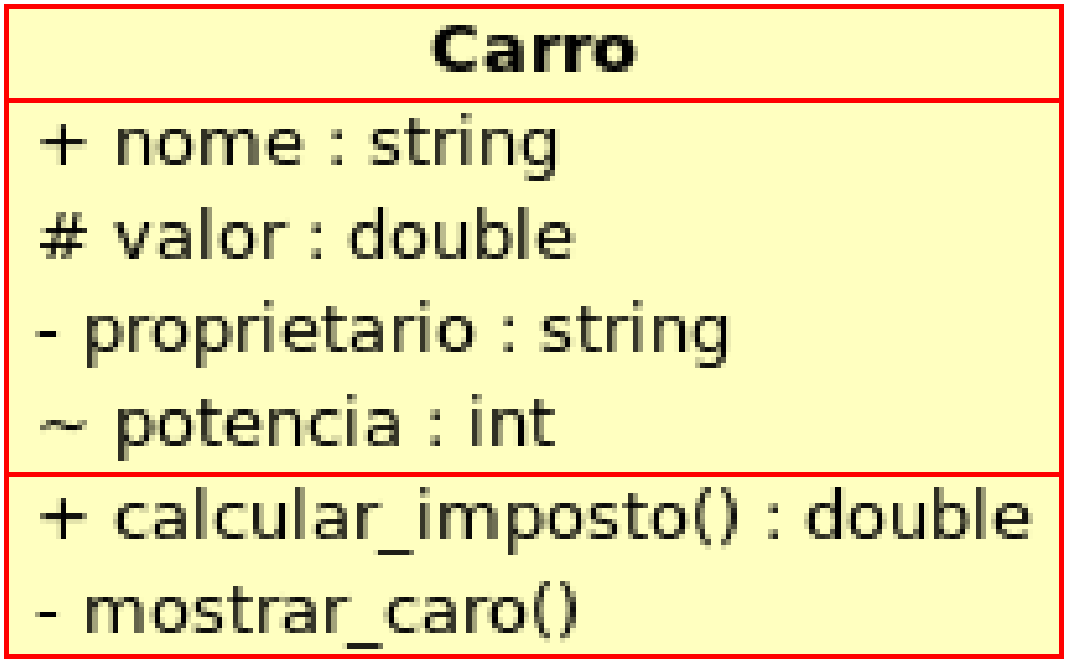


Diagrama de Classes

- Criando Diagramas de Classe
  - Atributos
    - Além do nome e do tipo, podemos também definir o valor inicial, a visibilidade e outras características dos atributos.
    - O tipo do atributo pode corresponder ao nome de uma classe ou ser um tipo dependente da linguagem de programação adotada.
    - Exemplo de um atributo com valor inicial igual a HP e com escopo estático.

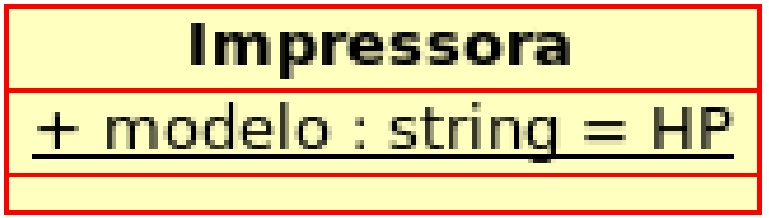


Diagrama de Classes

- Criando Diagramas de Classe
  - Atributos
    - Um atributo também pode ser derivado. Este é representado através de uma barra (/) à frente do nome.
    - Exemplo:

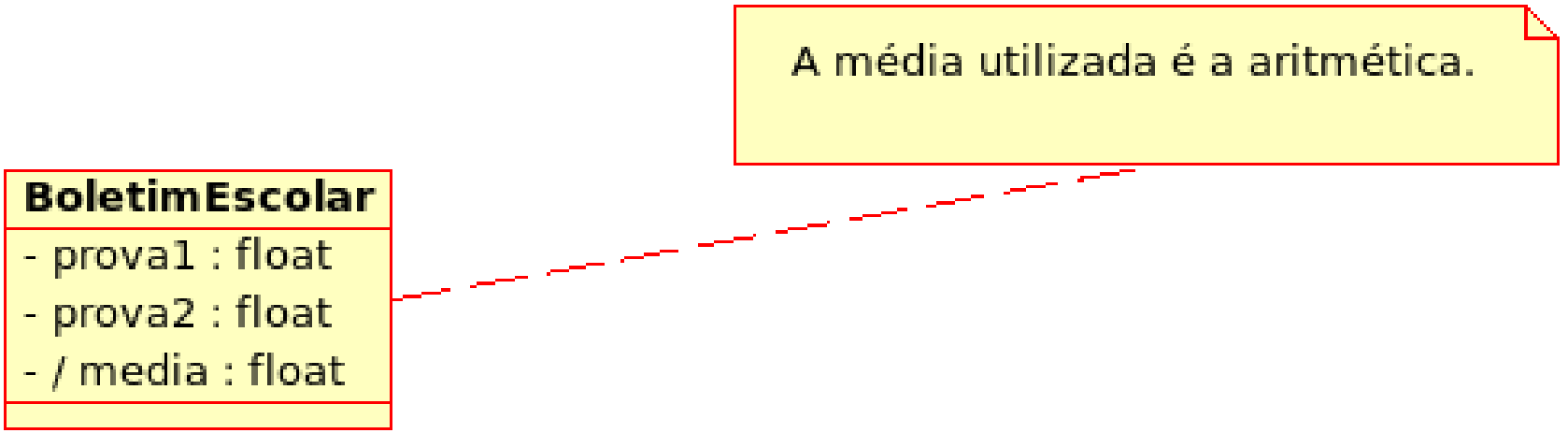


Diagrama de Classes

- Criando Diagramas de Classe
  - Operações
    - Assim como os atributos, a modelagem das operações não se limitam a seu nome e parâmetros.
    - A visibilidade pode ser representada pelas palavras-chaves *public*, *protected* ou *private*.
    - A lista de parâmetros corresponde a uma lista separada por vírgula, conforme o padrão:
      - escopo-parâmetro nome: tipo = valor-default
    - O escopo do parâmetro pode assumir os valores:
      - in: o parâmetro é apenas de entrada, não aceitando modificações.
      - out: o parâmetro é apenas de saída, não aceitando leitura.
      - inout: o parâmetro é de entrada e saída. Aceita leitura e modificação.

Diagrama de Classes

- Criando Diagramas de Classe
  - Operações
    - O parâmetro também pode ter um valor-default.
    - É necessário indicar se haverá ou não retorno da função.
    - A operação ainda pode ser ou não abstrata. Para indicar que a operação é abstrata, esta deve ser marcada como *{abstract}* ou a assinatura da operação em itálico.
    - Exemplo:

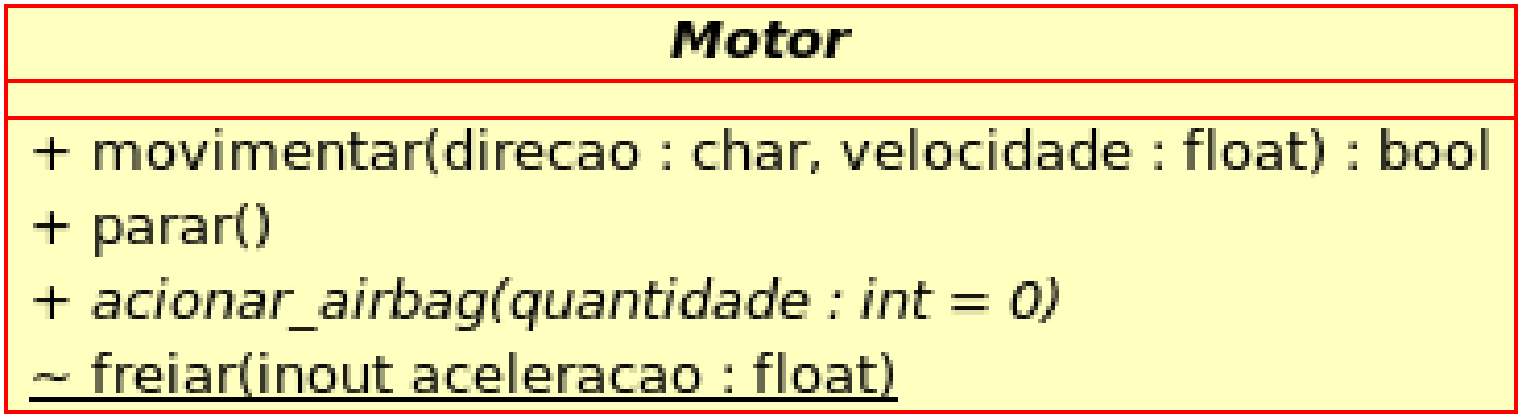


Diagrama de Classes

- Relacionamentos
  - As classes dentro do contexto da modelagem de um sistema, na sua maioria, não trabalham sozinhas.
  - No diagrama de classes temos os relacionamentos de **associação**, **generalização** e **especialização**.
  - Como variação do relacionamento de associação, ainda é possível modelar relacionamentos de **agregação** e **composição**.

Diagrama de Classes

- Relacionamentos
  - Associação
    - É um relacionamento que conecta duas (binária) ou mais classes (n-ária), demonstrando a colaboração entre as instâncias de classe.
    - Exemplos de associação binária

Diagrama de Classes

- Relacionamentos
  - Associação
    - Exemplo de associação ternária

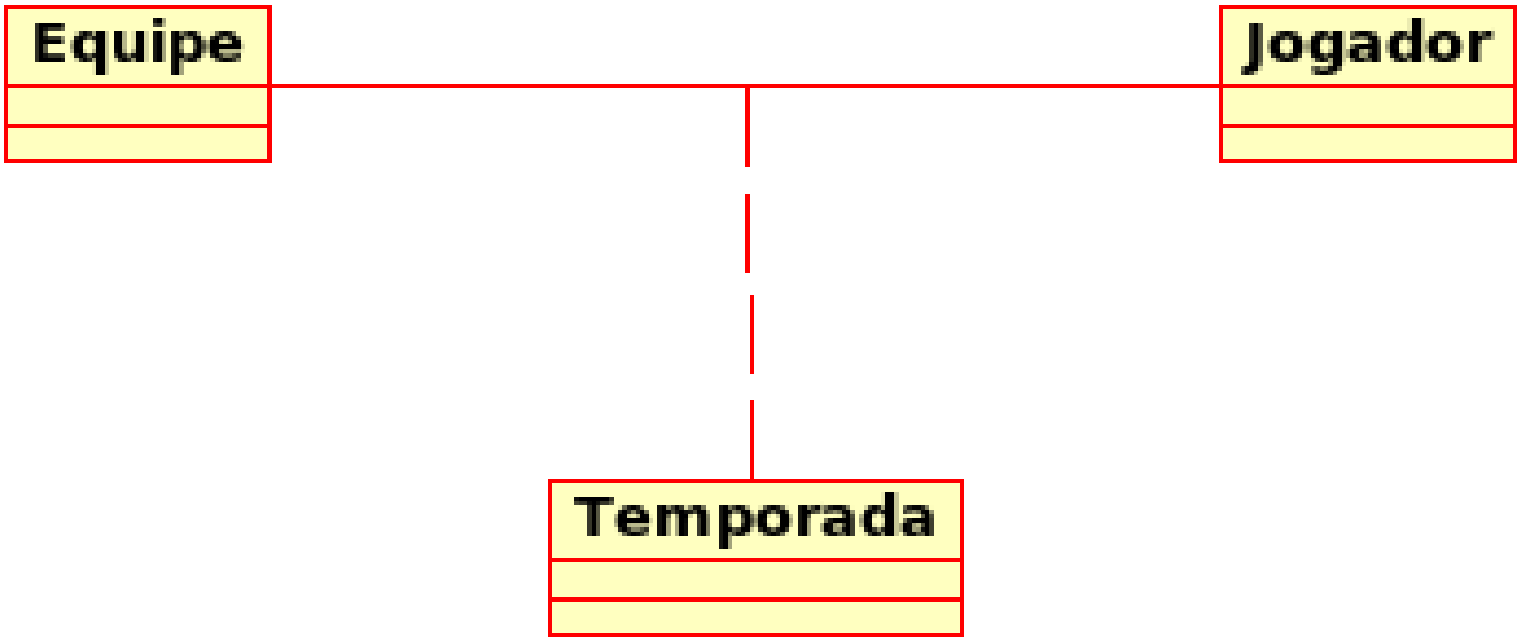


Diagrama de Classes

- Relacionamentos
  - Associação (adornos)
    - Nome da associação
    - Multiplicidade
    - Papel (role)
    - Navegabilidade
    - Qualificador

Diagrama de Classes

- Relacionamentos
  - Generalização
    - Exemplo:

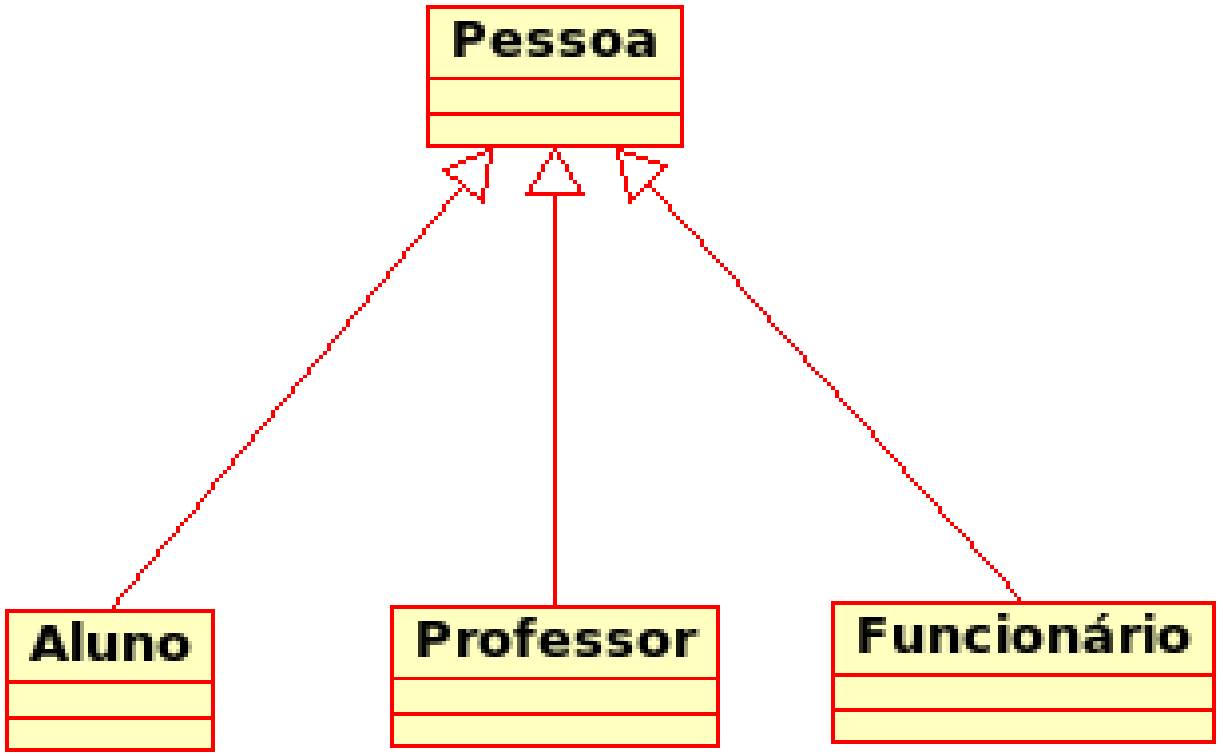


Diagrama de Classes

- Relacionamentos
  - Generalização
    - Restrições:
      - Sobreposição (overlapping).
      - Disjunção (disjoint).
      - Completo (complete).
      - Incompleto (incomplete).

Diagrama de Classes

- Relacionamentos
  - Generalização
    - Exemplo de uso de polimorfismo

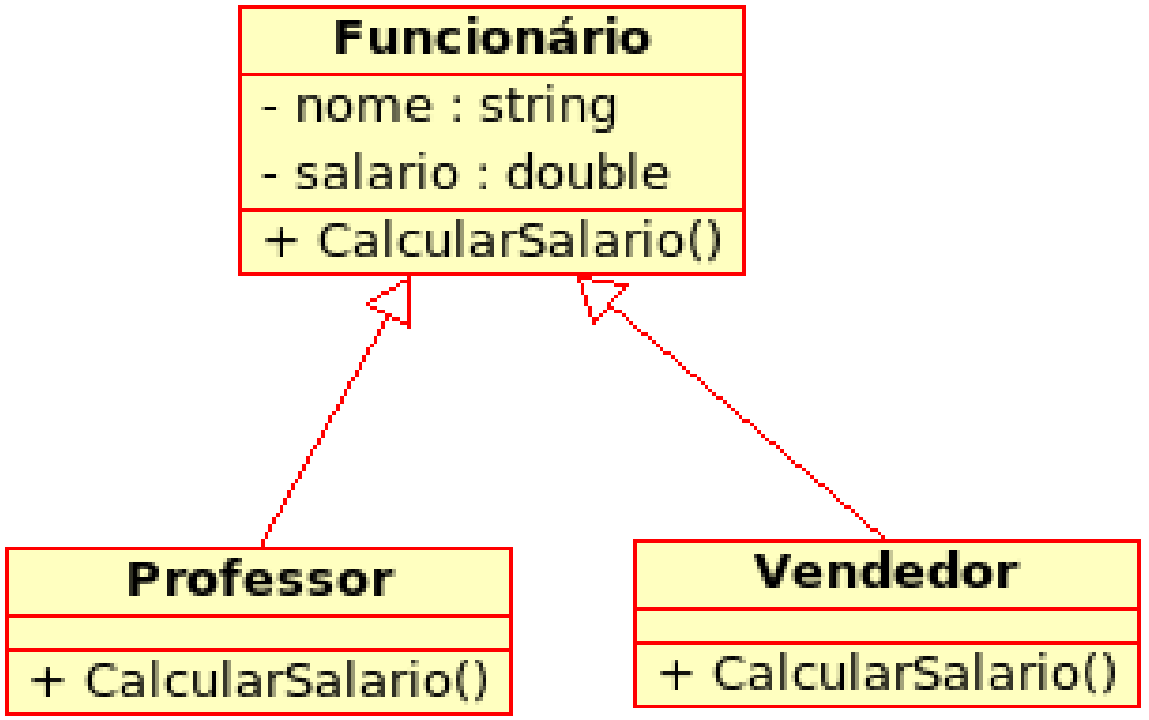


Diagrama de Classes

- Relacionamentos
  - Dependência
    - Indica que uma mudança na interface de uma delas pode causar mudanças na outra.
    - Por exemplo, troca de mensagens entre classes.
    - Exemplo:

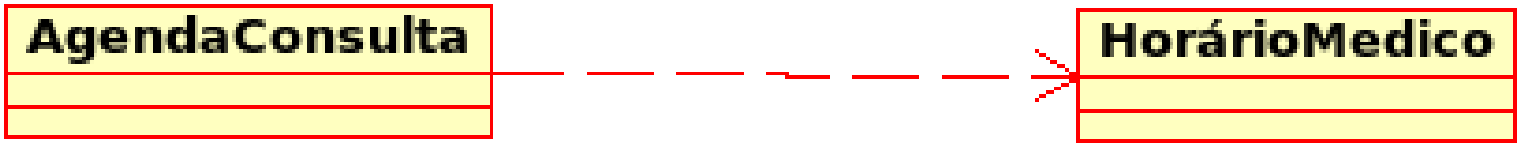


Diagrama de Classes

- Relacionamentos
  - Agregação
    - A agregação corresponde a um caso particular da associação, utilizada para expressar um relacionamento **todo-parte**.
    - Exemplo:

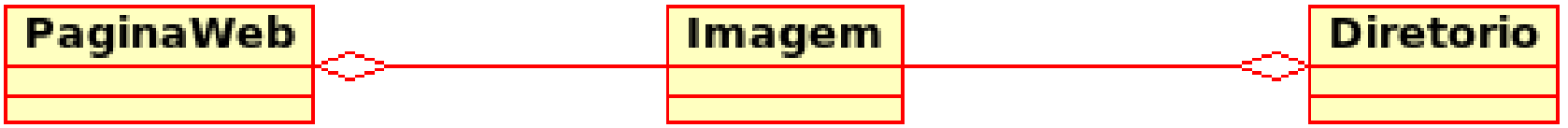


Diagrama de Classes

- Relacionamentos
  - Composição
    - É uma variação da agregação.
    - A diferença com a agregação consiste no fato de que a classe parte pertence só e somente à classe todo, em um determinado momento, não podendo fazer parte de outro relacionamento de composição.
    - A classe composta é responsável pela criação e destruição de suas partes.
    - Exemplo:

Diagrama de Classes

- Outros conceitos
  - Enumeração
    - É usada como tipo de dados nos diagramas de classes e pode ser modelada separadamente como uma classe esteriotipada como <<enumeration>>.
    - Exemplo:

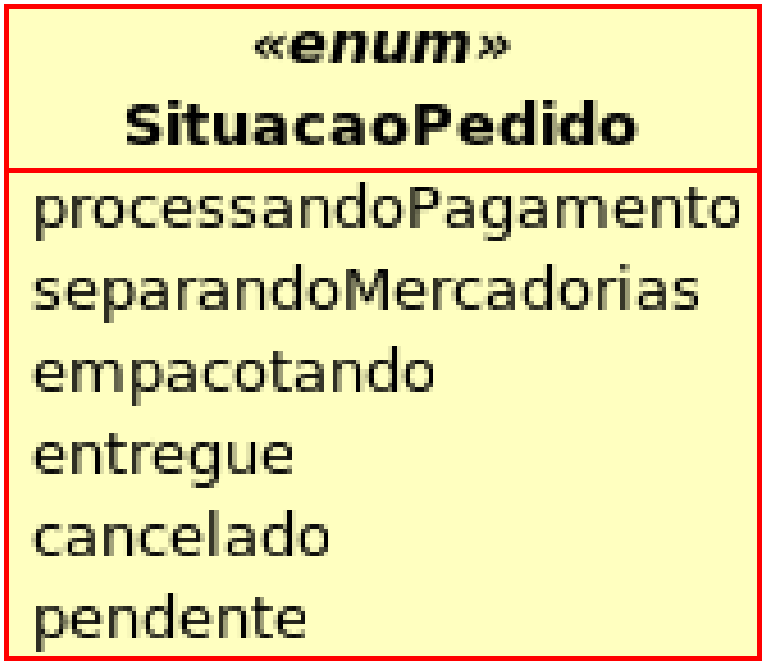


Diagrama de Classes

- Outros conceitos
  - Classe de associação
    - Representa uma associação que possui propriedades de classes como atributos, operações e outras associações.
    - Ela possui elementos próprios.
    - Exemplo:

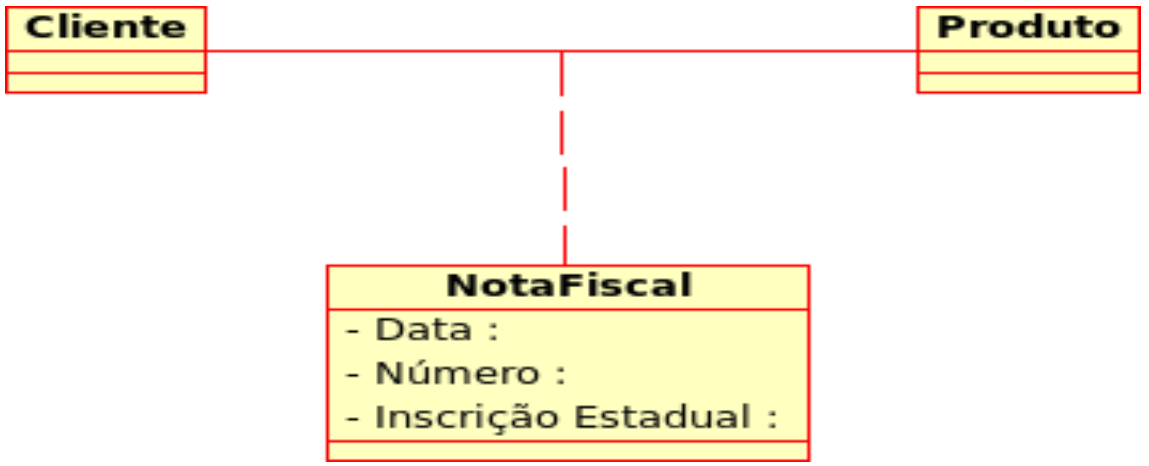


Diagrama de Classes

- Classes abstratas
  - É uma classe que não possui instâncias diretas, apenas suas classes descendentes.
  - Classes abstratas organizam características comuns a diversas classes.
  - Exemplo:

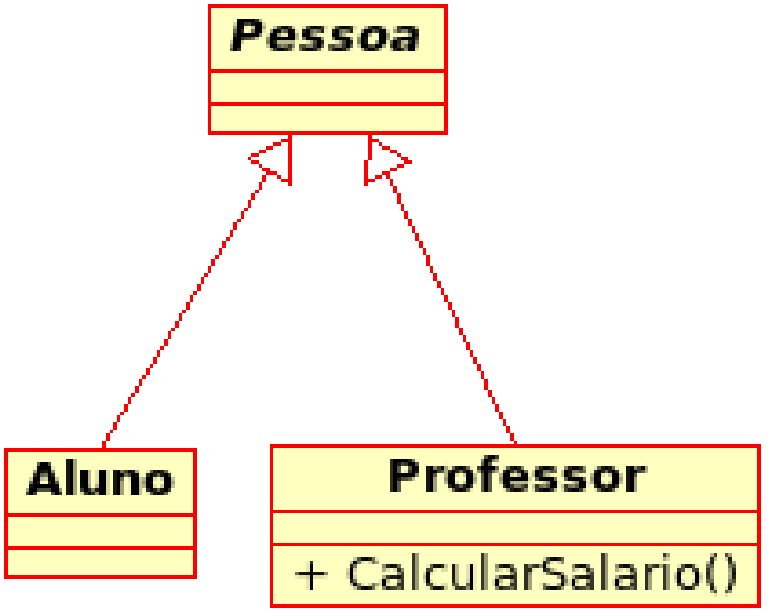


Diagrama de Classes

- Interface
  - Uma interface possui apenas operações.
  - O relacionamento entre interface e classe é feita através da realização.
  - Exemplo:

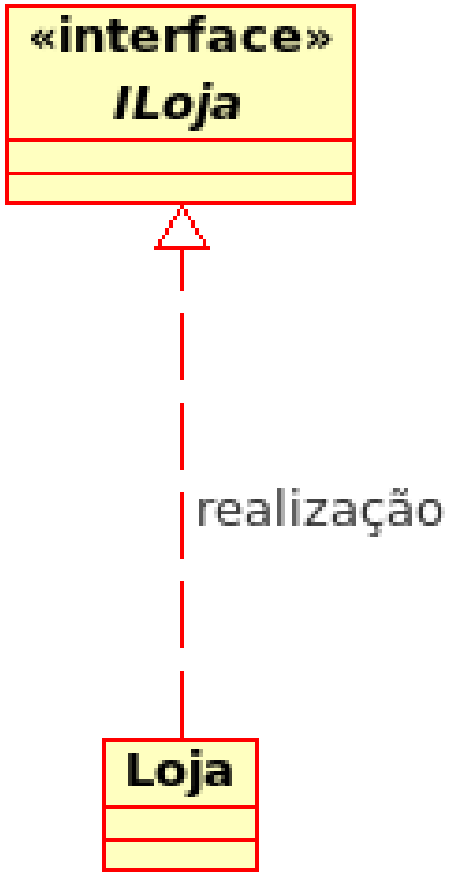
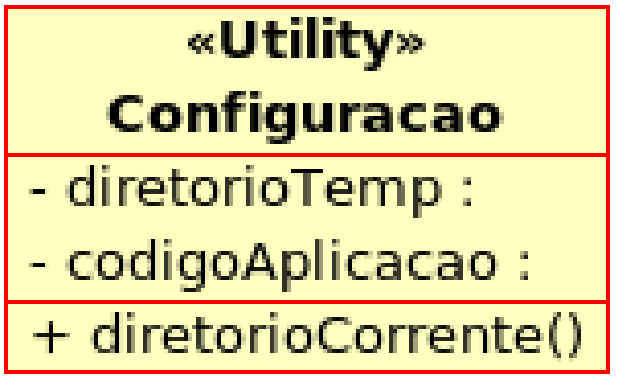


Diagrama de Classes

- Utilitário
  - É um agrupador de variáveis e procedimento globais representados na forma de uma classe.
  - É um recurso utilizado para fins de implementação, fazendo com que variáveis e procedimentos globais sejam enxergados como atributos e operações de uma classe.
  - Exemplo:



Exercícios

- Qual é a diferença entre composição e agregação? Dê um exemplo.
- O que caracteriza uma classe de associação? Dê um exemplo.
- Qual é a diferença entre classes abstratas e interfaces?

Casos de Uso

- O diagrama de casos de uso é empregado para representação e levantamento dos **requisitos** do usuário.
- O levantamento de requisitos é uma parte inicial do desenvolvimento e terá influência sobre todas as demais etapas.

Casos de Uso

- O que é um caso de uso?
  - Descreve uma sequência de ações que representam um cenário principal (perfeito) e cenários alternativos, com o objetivo de demonstrar o comportamento de um sistema (ou parte dele), através de interações com atores.
- Exemplo:
  - Cenário para emitir saldo em um terminal de caixa eletrônico.
  - Cenário principal e cenário alternativo.
- O caso de uso deve ser totalmente compreensível tanto para a equipe de desenvolvimento quanto para os clientes que detêm o conhecimento do sistema.

Casos de Uso

- As ações dos casos de uso são mantidas pelos atores.
- Um ator representa um conjunto de papéis exercido por um usuário do sistema ao interagir com um determinado caso de uso.

Casos de Uso

- Caso de Uso: Reserva em um Restaurante
- Cenário Principal
  - O cliente informa ao atendente a data da reserva, que é repassada ao sistema.
  - O sistema mostra o mapa do salão do restaurante, indicando as mesas já reservadas e as que estão livres. O sistema calcula e exibe o número de reservas ainda disponíveis.
  - Um ou vários lugares disponíveis é (são) escolhido (s) para reserva.
  - Um sistema solicita o CPF do cliente, para identificação do mesmo no sistema. O sistema pesquisa o cliente e mostra nome e telefones do contrato, para confirmação.
  - Após confirmação, a reserva é efetuada em nome do cliente.

Casos de Uso

- Cenários Alternativos
  - Alternativa: Data não disponível para reserva
    - O sistema verifica se para a data informada é possível efetuar reservas. Caso negativo, uma nova data deve ser solicitada.
  - Alternativa: Reservas esgotadas
    - O sistema verifica se para o dia informado, as reservas estão esgotadas. O sistema deve possibilitar que seja informada nova data ou que se encerre a solicitação de reserva.
  - Alternativa: Cliente não cadastrado
    - Se o cliente não for cadastrado: Extend Cadastrar Cliente de Reserva.

Casos de Uso

- Relacionamento entre casos de uso e atores
  - Entre casos de uso:
    - Generalização.
    - Extensão.
    - Inclusão.
  - Entre relacionamentos de atores:
    - Generalização.
  - Entre relacionamentos entre atores e casos de uso:
    - Associação.

Casos de Uso

- Associação
  - Representa a interação do ator com o caso de uso, ou seja, a comunicação entre atores e casos de uso, por meio do envio e recebimento de mensagens.
  - As associações são sempre binárias.
  - Exemplo:
    - O ator Correntista envia e recebe mensagens do Caso de Uso CalcularEmpréstimoPessoal, por um relacionamento de associação.

Casos de Uso

- Generalização
  - Ocorre entre casos de uso ou entre atores.
  - É considerado quando temos dois elementos semelhantes, mas com um deles realizando algo a mais.
  - Exemplo:
    - Podemos criar um ator genérico Aluno e especializá-lo nos atores **Aluno Matriculado** e **Aluno Ouvinte**.

Casos de Uso

- Extensão (*Extend*)
  - Um relacionamento de extensão entre casos de uso indica que um deles terá ser procedimento acrescido, em um ponto de extensão, de outro caso de uso, identificado como base.

Casos de Uso

- Inclusão (*Include*)
  - Indica que um deles terá seu procedimento copiado num local especificado no outro caso de uso, identificado como base.

Casos de Uso

- Modelando requisitos com casos de uso
  - É comum empregar os casos de uso na primeira etapa do desenvolvimento de software.
  - Todos os casos de uso possuem nomes que o identificam e diferenciam dos demais.
  - Normalmente são breves expressões representando a essência do que você está modelando.

Casos de Uso

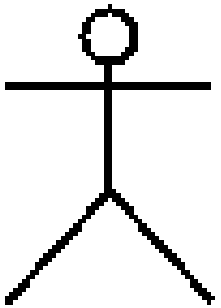
- Criando diagramas de casos de uso
  - Os diagramas de casos de uso são usados para expressar a fronteira do sistema, e/ou modelar os requisitos do mesmo.
  - Não é obrigatória a construção de diagramas de casos de uso.

Casos de Uso

- Criando diagramas de casos de uso
  - Representação gráfica:
    - O caso de uso é representado por uma elipse contendo o seu nome.
    - Exemplo:

Casos de Uso

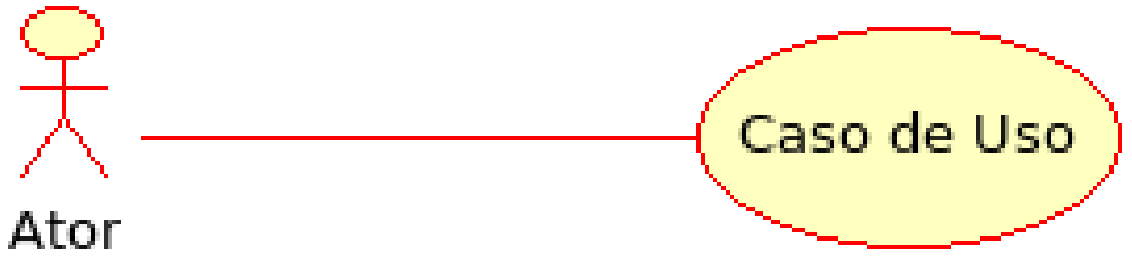
- Criando diagramas de casos de uso
  - Representação gráfica:
    - O ícone estereótipo padrão para um ator é a figura de um "*stick man*", contendo seu nome abaixo da figura.
    - Outra representação consiste num retângulo de classe, com o estereótipo <<*actor*>>.
    - Exemplo:



Cliente

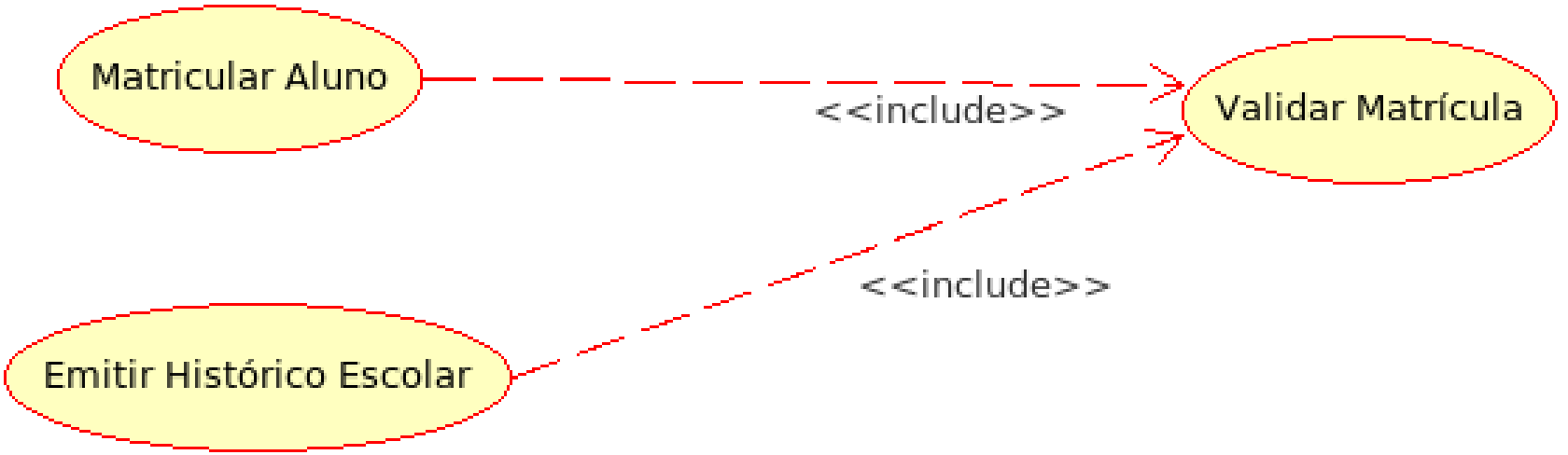
Casos de Uso

- Criando diagramas de casos de uso
  - Representação gráfica:
    - A representação gráfica de uma associação corresponde a uma linha sólida, ligando o caso de uso ao ator e vice-versa.
    - Exemplo:



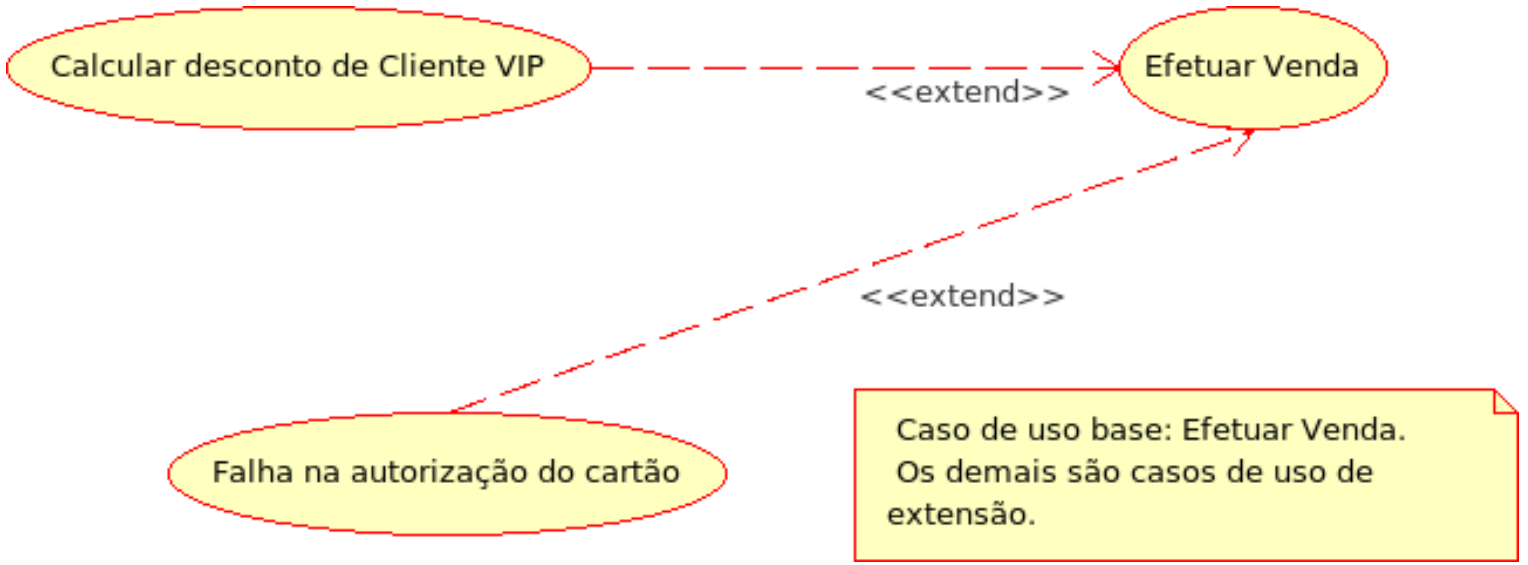
Casos de Uso

- Criando diagramas de casos de uso
  - Representação gráfica:
    - Um relacionamento de inclusão é representado graficamente por uma seta tracejada com a ponta aberta, que parte do caso de uso base e contém o estereótipo <<include>>.
    - Exemplo:



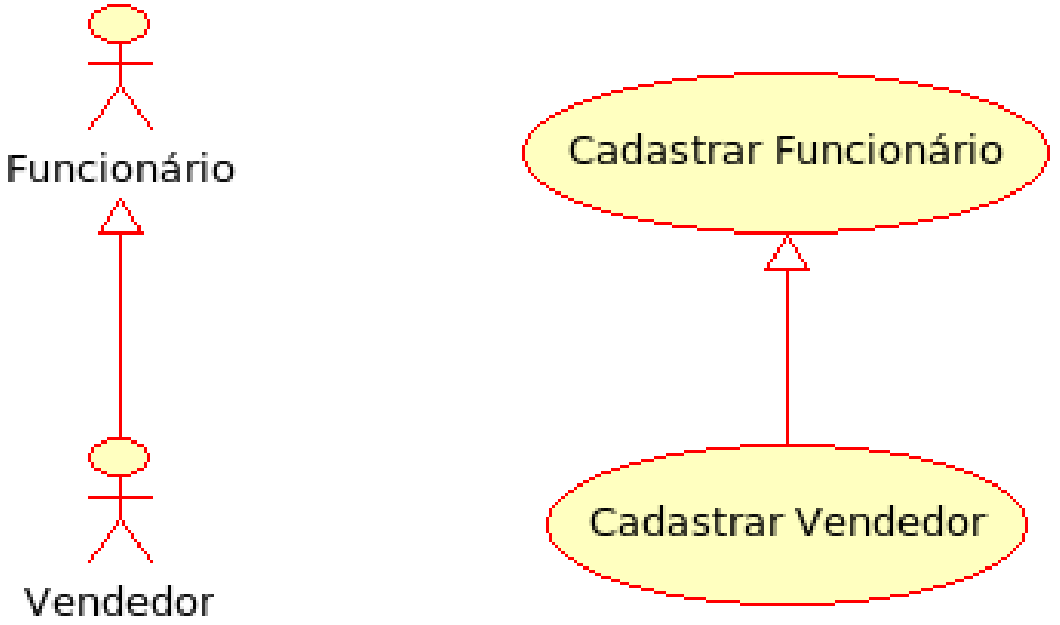
Casos de Uso

- Criando diagramas de casos de uso
  - Representação gráfica:
    - Um relacionamento de extensão é representado graficamente por uma seta tracejada com a ponta aberta, que parte do caso de uso estendido e contém o estereótipo <<extend>>.
    - Exemplo:



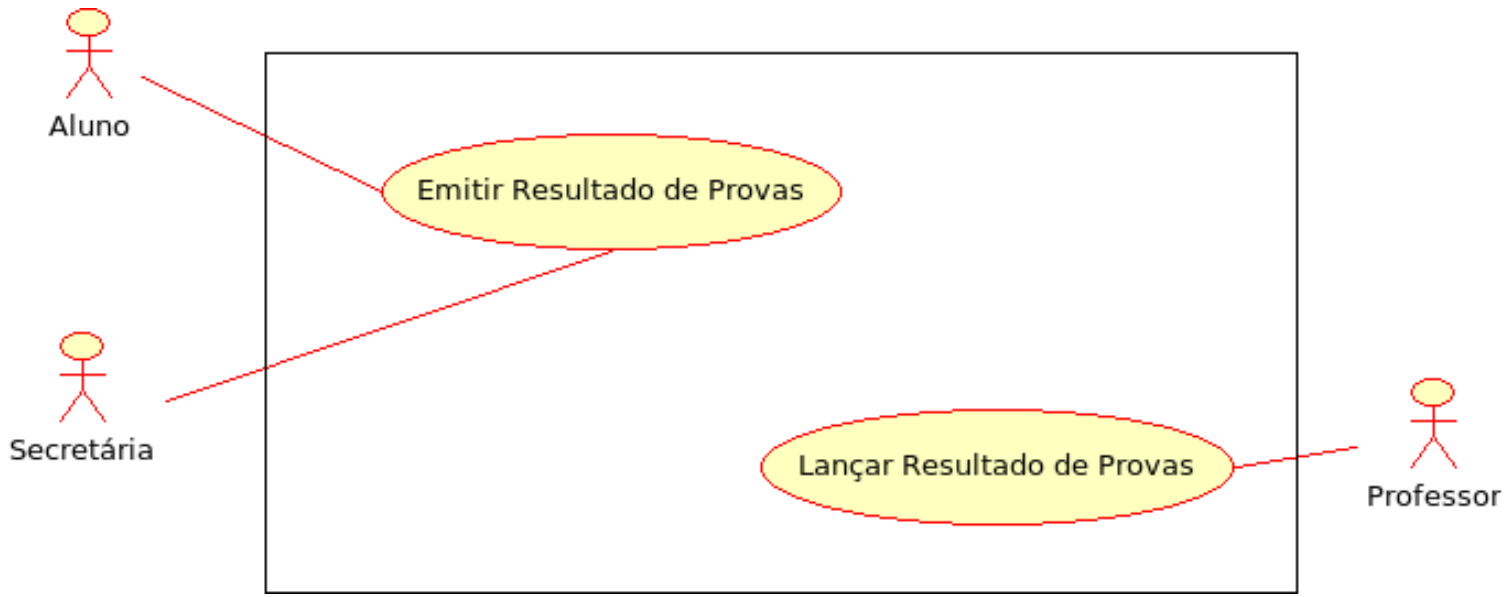
Casos de Uso

- Criando diagramas de casos de uso
  - Representação gráfica:
    - Um relacionamento de generalização é representado graficamente pela seta de generalização, que corresponde a uma linha sólida com uma única fechada, mas não preenchida em uma das pontas.
    - Exemplo:



Casos de Uso

- Criando diagramas de casos de uso
  - Representação gráfica:
    - O diagrama de caso de uso pode mostrar ainda a fronteira do sistema que é representada como um retângulo envolvendo os casos de uso.
    - Exemplo:



Casos de Uso

- Criando diagramas de casos de uso
  - Auxílio de pacotes
    - Em sistemas de média/alta complexidade é comum termos dezenas de caso de uso.
    - Nesse caso, a representação de todos eles em um único diagrama é uma tarefa impossível.
    - A fim de minimizar a visualização e, principalmente, organizar esses casos de uso considerando uma mesma abordagem conceitual, podemos trabalhar com pacotes.

Casos de Uso

- A importância dos protótipos
  - Uso atual de ferramentas RAD.

Exercícios

- O que você entende por requisitos?
- Quais são os tipos de requisitos?
- Qual é a diferença entre extensão e inclusão nos casos de uso?
- Os atores dos casos representam pessoas? Explique a sua resposta.
- O que você entende por fronteira do sistema?
- Qual é a vantagem no uso dos protótipos no desenvolvimento de software?

Diagrama de Objetos

- Diagrama de objetos consiste numa instância do diagrama de classes, no qual para cada classe temos um objeto em um determinado ponto do tempo.
- Representa uma “fotografia” do sistema em determinado momento.
- O diagrama de objetos pode ser empregado para:
  - Facilitar a modelagem de estruturas complexas de dados.
  - Auxiliar o desenvolvedor no momento de identificar problemas na execução de uma aplicação.
- A representação gráfica de um objeto é similar à de uma classe, pois consiste num retângulo com um ou dois compartimentos.
- Exemplo:

Diagrama de Objetos

- As instâncias das classes, chamadas de objetos, são representadas como classes, só que com o nome do objeto sublinhado.

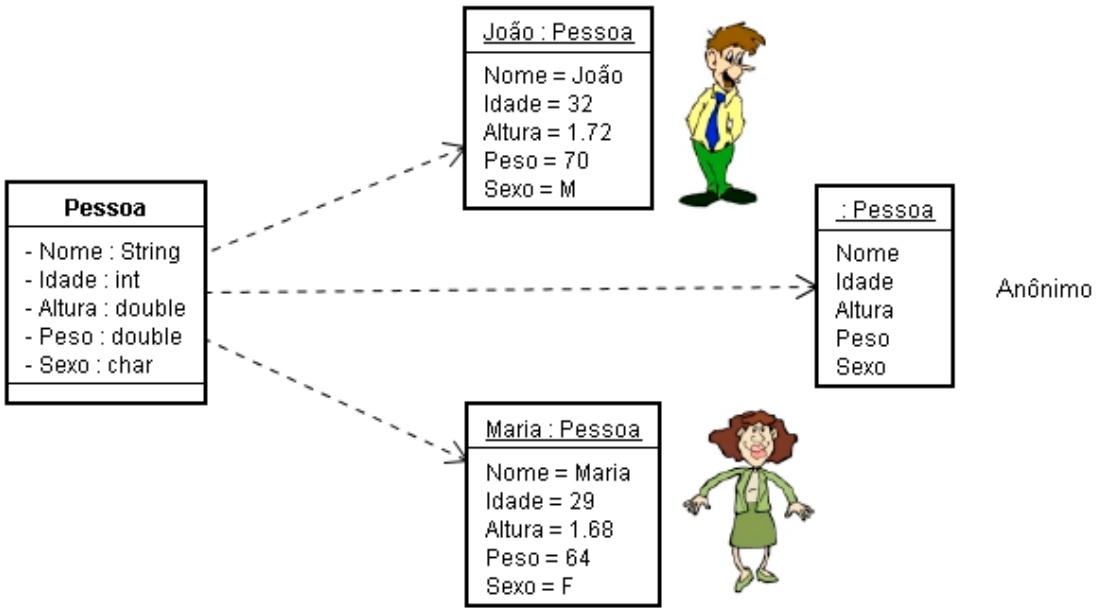
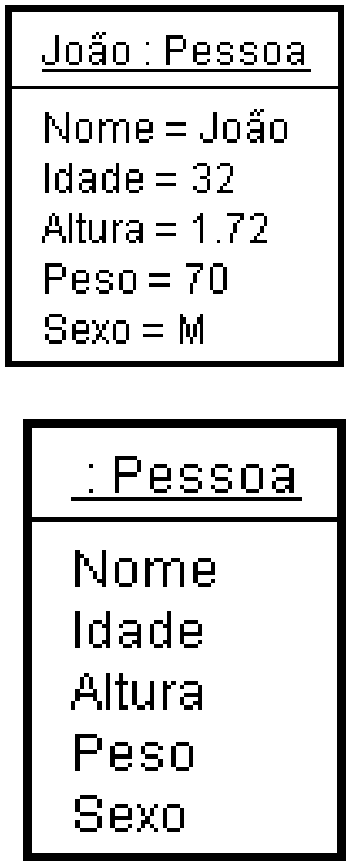


Figura 2 – Representação de classes e objetos em UML

Diagrama de Objetos



- No primeiro caso, temos um objeto específico do mundo real, representando o objeto João, da classe Pessoa.
- Também é possível usar a forma abreviada :nome da classe sem o nome do objeto. Esta forma é conhecida como objeto anônimo (isto é usado quando todos os objetos da classe têm o mesmo comportamento).

Diagrama de Objetos

- Comparação entre os diagramas de classes e de objetos

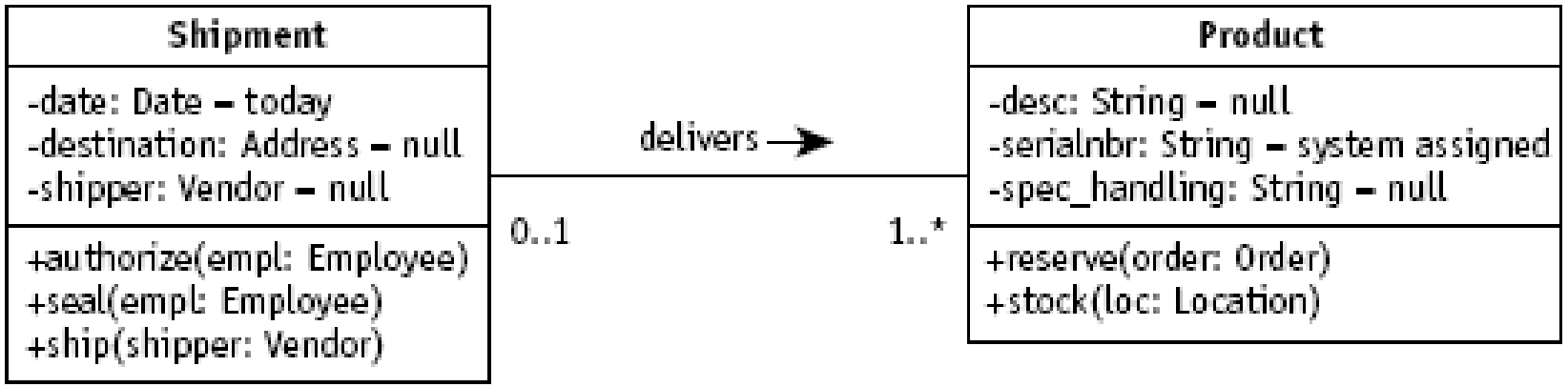


Diagrama de Objetos

- Comparação entre os diagramas de classes e de objetos

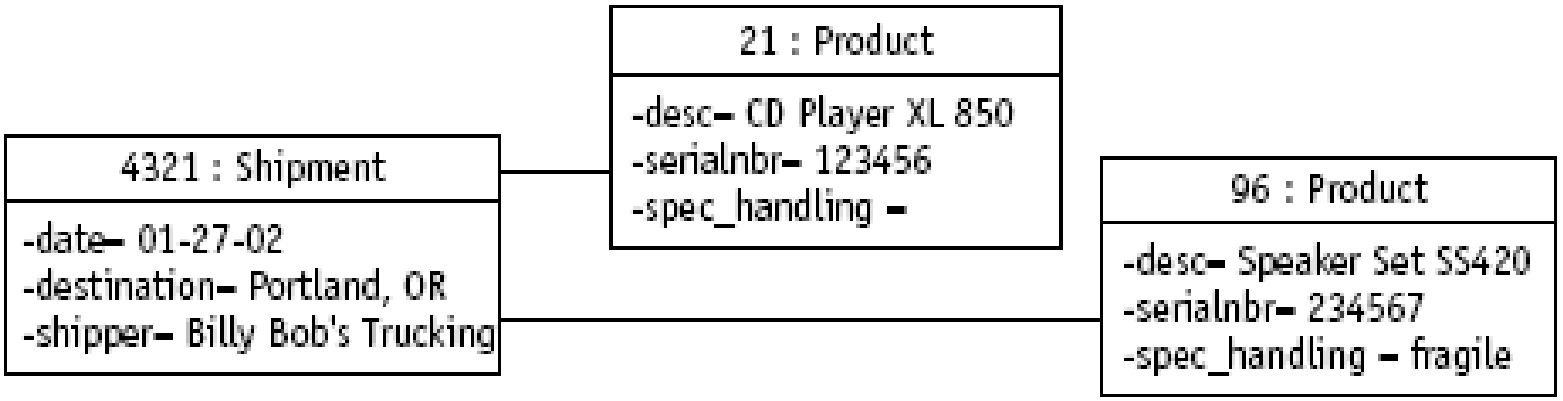


Diagrama de Objetos

- Exemplo de diagrama de objetos.

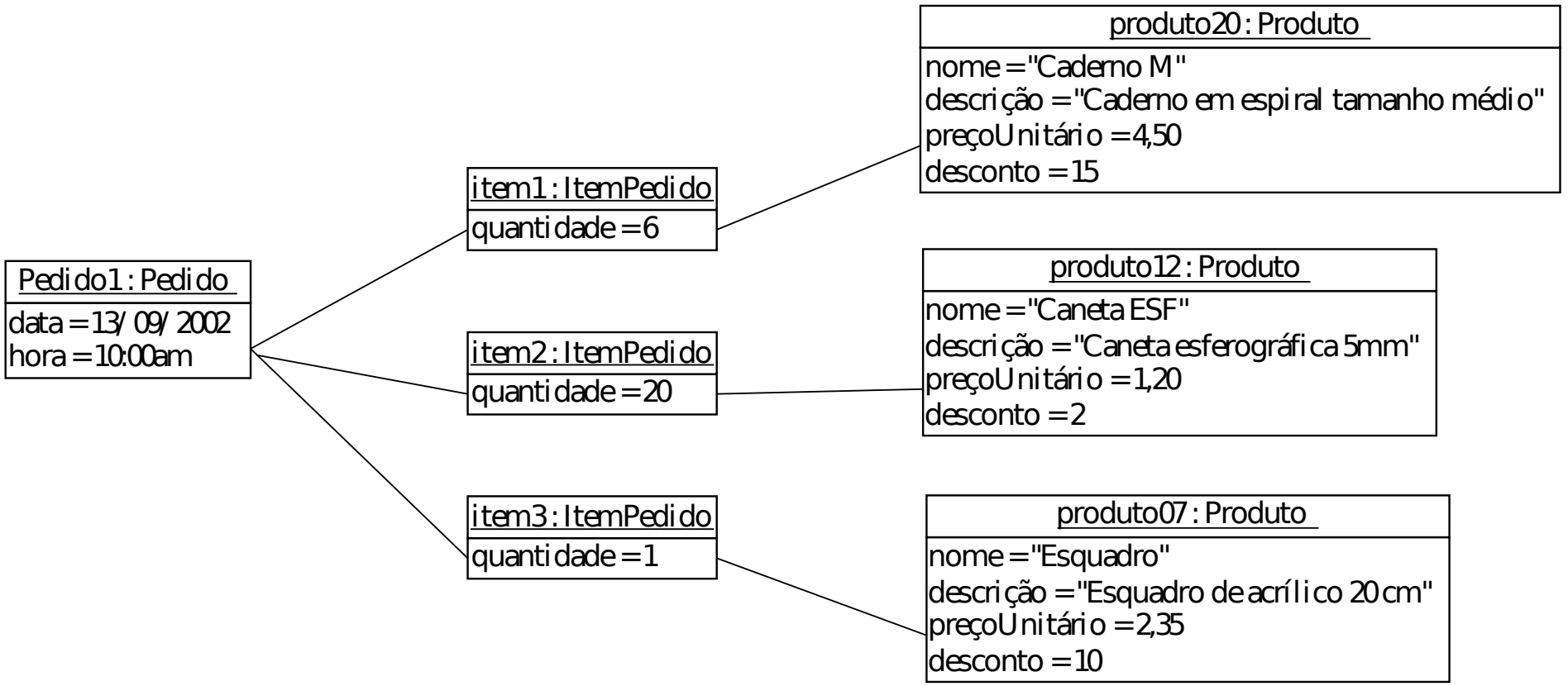
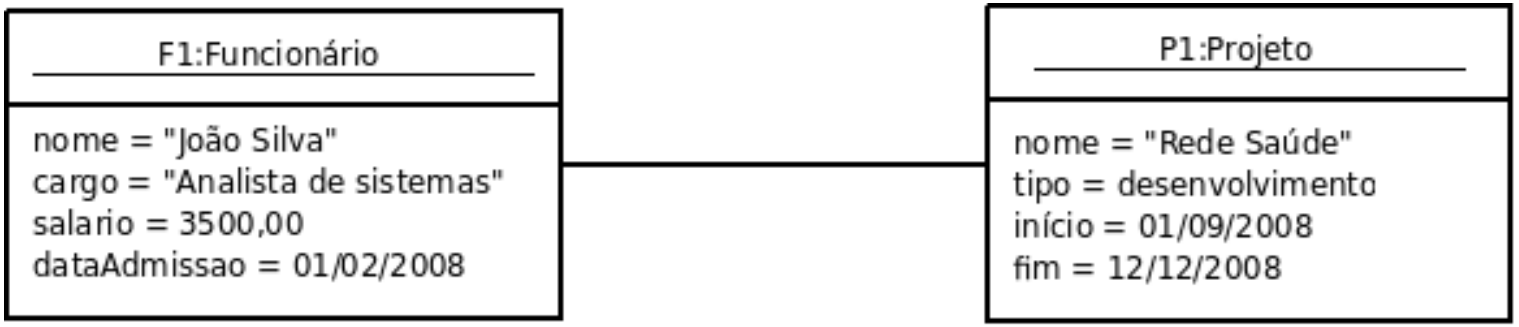


Diagrama de Objetos

- É possível também representar os atributos.
- A notação é **nome_do_atributo: tipo = valor**
- Atributos cujos valores não sejam relevantes para a modelagem podem ser omitidos.
- O relacionamento entre objetos é feito através de *links*.
- Um link é uma instância de uma associação.

Diagrama de Objetos

- Exemplo:



Exercícios

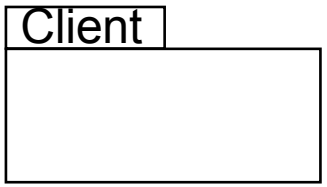
- Qual é a importância do diagrama de objetos na análise de sistemas?
- É possível representar as operações das classes nos diagramas de objetos? Justifique a sua resposta.
- Desenhe um diagrama de objetos para uma classe chamada Aluno.

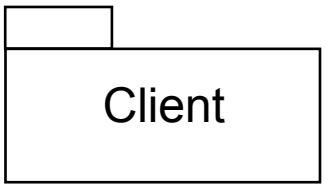
Diagrama de Pacotes

- Os pacotes são empregados para organizar seus elementos de modelagem em conjuntos maiores que possam ser manipulados como grupos.
- Os pacotes bem estruturados são fracamente acoplados em forte coesão, com acesso altamente controlado ao conteúdo do pacote.
- O uso de pacotes favorece a modularidade.
- Um diagrama de pacotes mostra pacotes e relações entre estes.

Diagrama de Pacotes

- A notação de pacote em UML é uma pasta com o nome no interior ou na aba.
- No caso de um pacote dentro de outro, pode ser representado pelo nome completo do pacote contido que inclui o nome do seu contenedor.





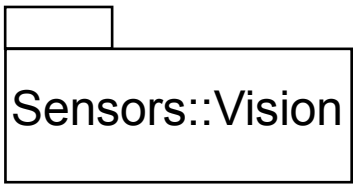


Diagrama de Pacotes

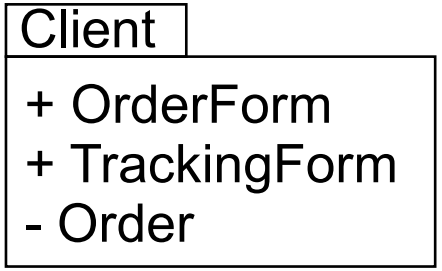
- Existem os seguintes tipos de pacotes:
  - Pacotes de classes (pacotes lógicos) – em diagramas de classes.
  - Pacotes de componentes – em diagrama de componentes.
  - Pacotes de nós – em diagramas de distribuição.
  - Pacotes de casos de utilização – em diagramas de casos de utilização.

Diagrama de Pacotes

- Pacotes lógicos
  - Um pacote lógico (ou módulo lógico) é um agrupamento lógico de classes e relações entre essas classes.
  - Corresponde ao conceito de `package` em Java ou de `namespace` em C++.
  - Um pacote lógico pode estar distribuído em vários arquivos.
  - Os diagramas de pacotes lógicos são empregados para modelar a arquitetura lógica de um sistema de software (organização em módulos lógicos e especificação de interfaces e dependência entre módulos).

Diagrama de Pacotes

- Conteúdo de pacote
  - Uma vez que representa um agrupamento, um pacote é formado por diversos elementos: classes, interfaces, componentes, nós, colaborações, casos de uso, diagramas e até outros pacotes.
  - Esses elementos podem ser indicados no interior do pacote, na forma de uma lista de nomes ou diagrama.



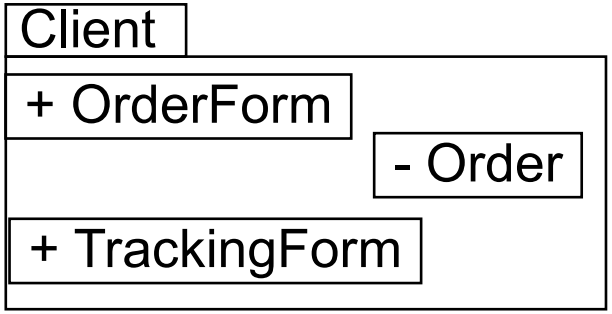


Diagrama de Pacotes

- A visibilidade dos elementos contidos num pacote:
  - Pode-se indicar a visibilidade dos elementos:

+ (público) : visível por todos que *importam* ou *acedem* ao pacote (nomes sem :: no 1º caso, com :: no 2º caso)

(protegido): visível só pelos pacotes-filhos (por relação de generalização - ver adiante)

- (privado): visível só por outros elementos do pacote

Diagrama de Pacotes

- Dependência entre pacotes
  - Dependência simples: uma alteração no pacote de destino afeta o pacote de origem (dependente) (informação útil para controle de informações).
  - Dependência com estereótipo <<access>>: o pacote de origem (dependente) acessa a elementos exportados pelo pacote de destino (precisa de :: nos nomes).
  - Dependência com estereótipo <<import>>: o pacote de origem (dependente) importa os elementos exportados pelo pacote de destino (não precisa de :: nos nomes).

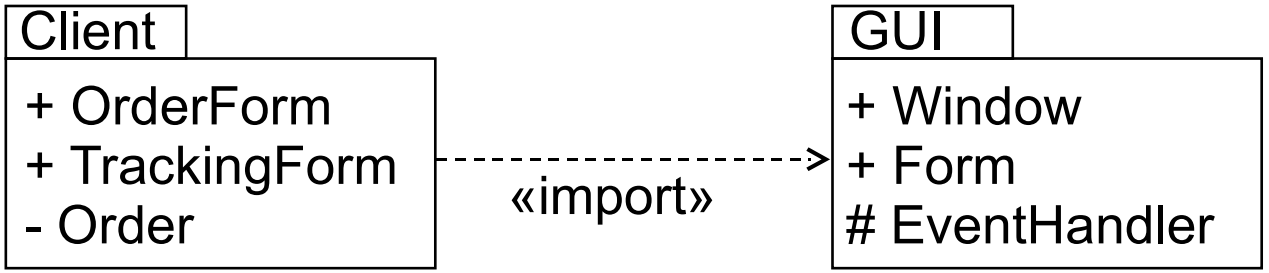


Diagrama de Pacotes

- Generalização de pacotes
  - Usada para especificar famílias de pacotes relacionados por herança

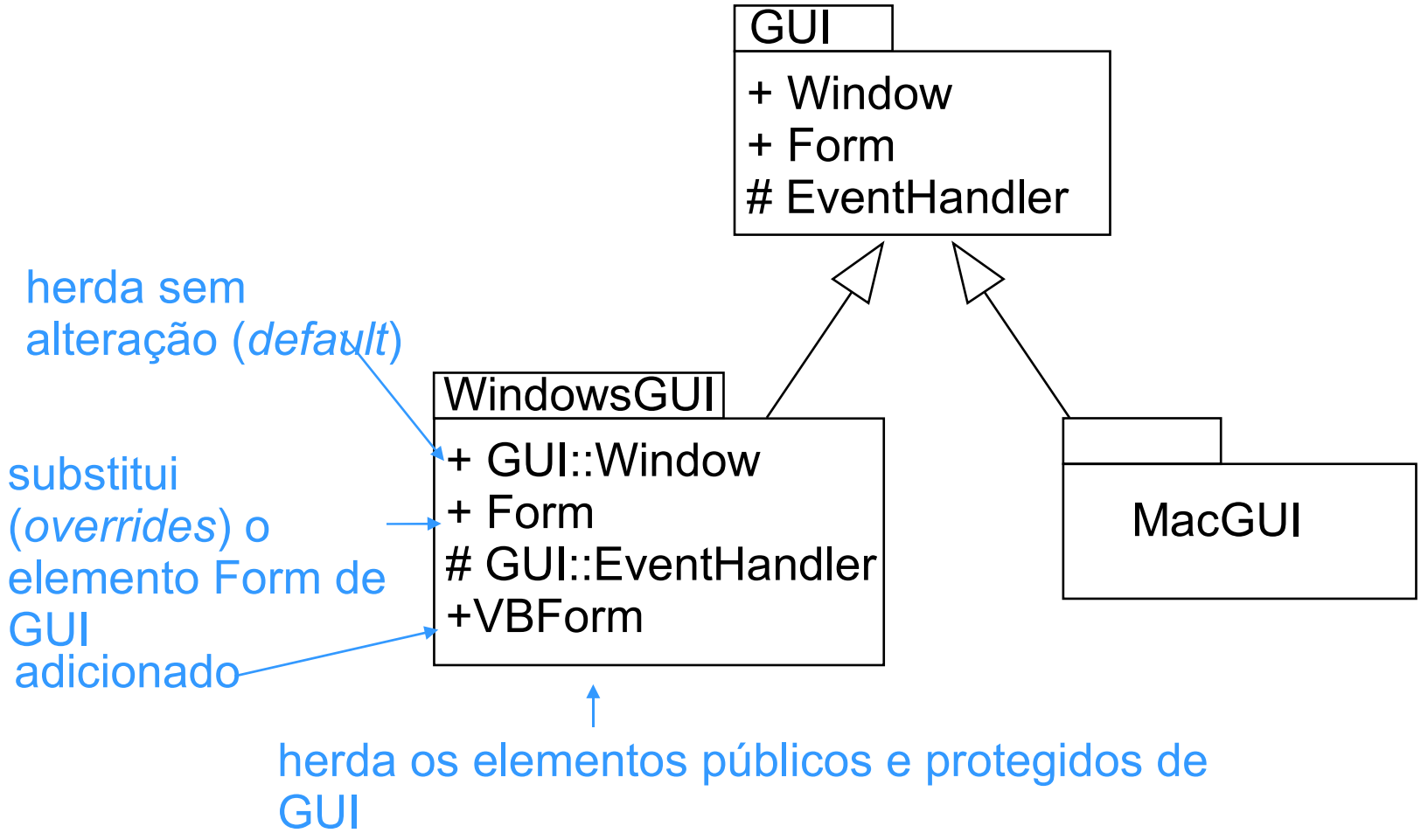
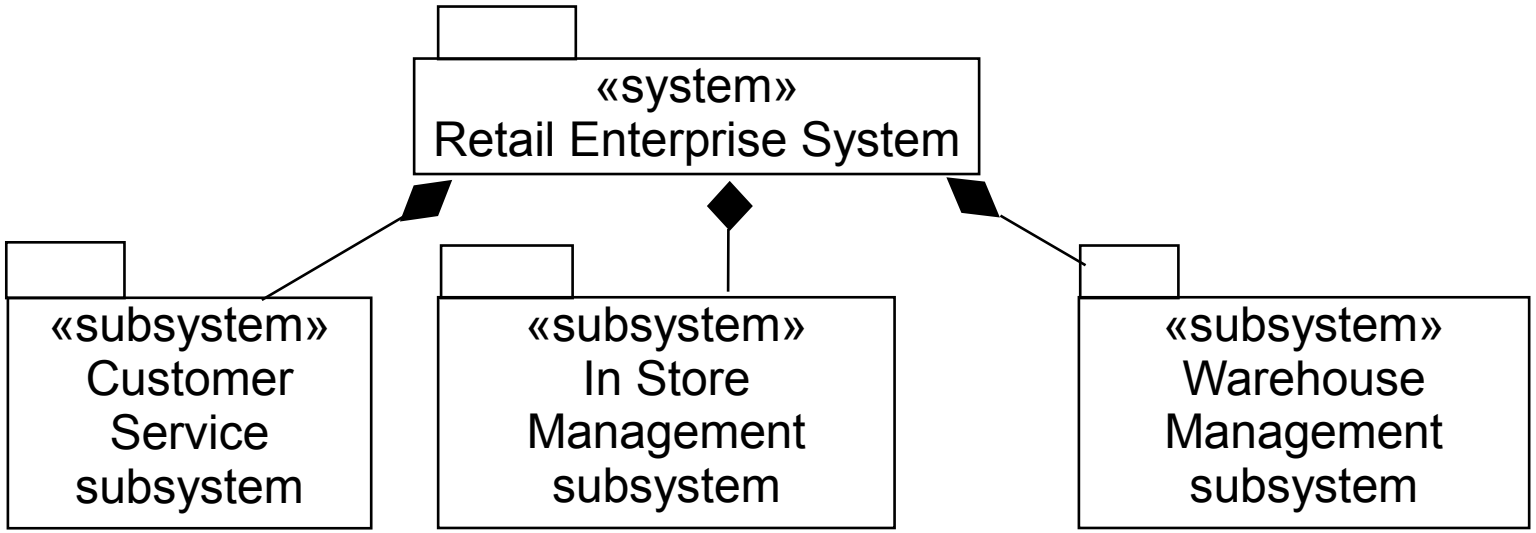


Diagrama de Pacotes

- «`system`» - pacote que representa o sistema completo que está a ser modelado (incluindo todos os modelos e elementos dos modelos)
- «`subsystem`» - pacote que representa uma parte independente de sistema completo que está a ser modelado; corresponde normalmente a um corte "vertical"
- «`facade`» (fachada) - pacote que constitui uma vista sobre outro pacote (não acrescenta funcionalidades, apenas apresenta de forma diferente)
- «`framework`» (infra-estrutura aplicacional) - pacote que representa um conjunto de classes abstractas e concretas concebido para ser estendido, implementando a funcionalidade típica de um determinado domínio de aplicação
- «`stub`» - pacote que serve como *proxy* para o conteúdo público de outro pacote
- «`layer`» - pacote que representa uma camada horizontal de um sistema

Diagrama de Pacotes

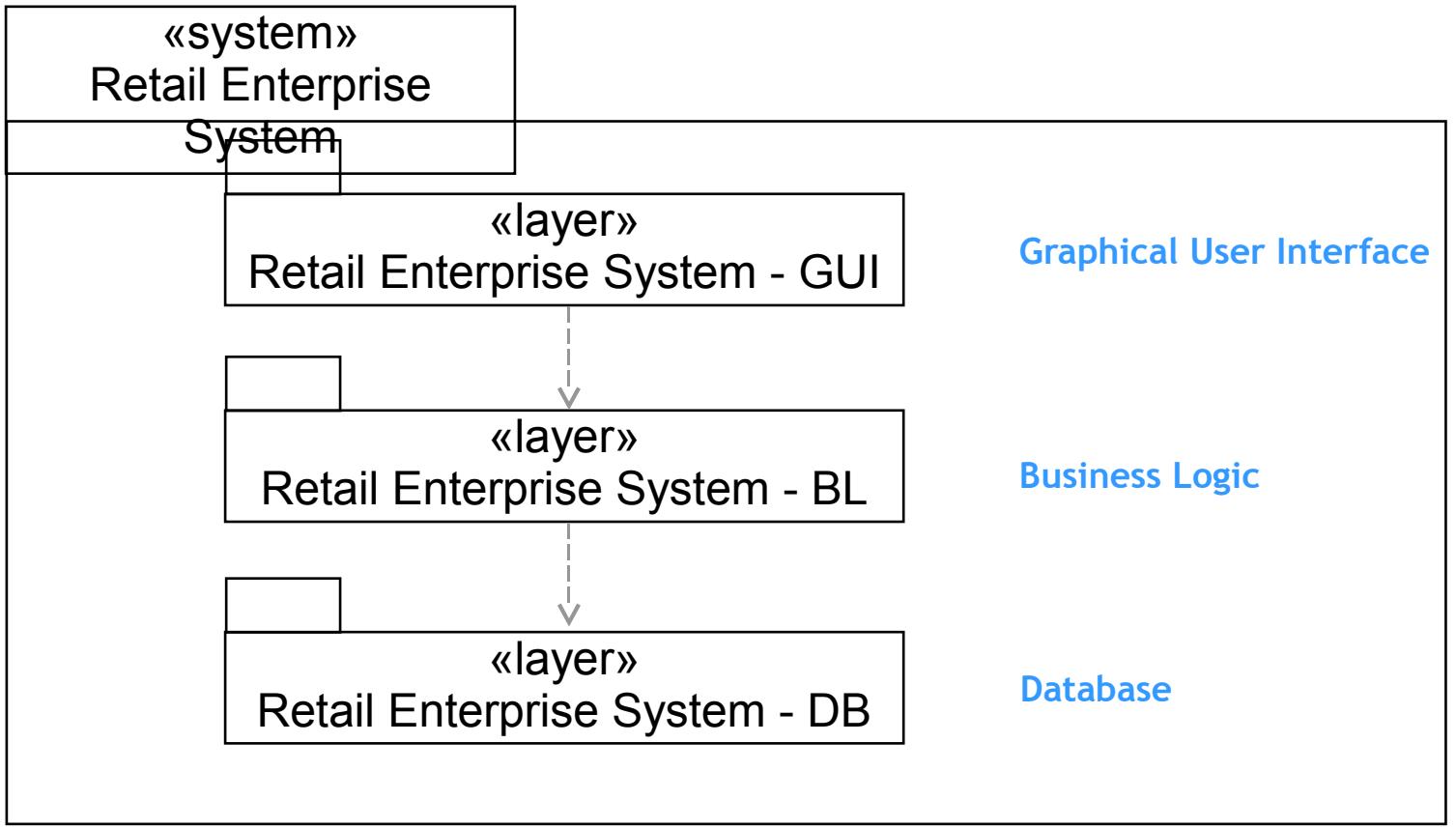
- Composição de pacotes
  - Sub-pacotes podem ser indicados dentro do pacote-dono ou com relações de composição.



Neste exemplo segue-se uma divisão vertical, por subsistemas!

Diagrama de Pacotes

- Composição de pacotes



Neste exemplo segue-se uma divisão horizontal, por camadas!

Exercícios

- Qual é a diferença entre pacotes e classes?
- Qual a importância do uso dos diagramas de pacotes?

Referências

- Booch, Grady & Rumbaugh, James & Jacobson, Ivar. **UML – Guia do Usuário**. Rio de Janeiro: Campus, 2000.
- Melo, Ana Cristina. **Desenvolvendo Aplicações com UML 2.0**. Rio de Janeiro: Brasport, 2004.