

UML

<http://www.uml.org/#UML2.0>

Aula de Luiz Eduardo Guarino de Vasconcelos

Objetivos



- História
- Tecnologia OO
- UML
- Pacotes
- Diagrama de Classe
 - ▣ Atributos
 - ▣ Métodos
 - ▣ Visibilidade

História



- 60, 70
 - ▣ COBOL, FORTRAN, C
 - ▣ Métodos de Análise e Projeto Estruturado
- 80 – início 90's
 - ▣ Smalltalk, ADA, C++, Object Pascal
 - ▣ Geração dos métodos OO
- 90: Início de atração por OO
 - ▣ Java, UML, Unified Process
 - ▣ Proliferação de Métodos de Desenvolvimento OO

Tecnologia OO

- Mais do que um **Modo de Programar**
- Modo de pensar abstrato sobre um domínio de problemas
 - ▣ usa conceitos do mundo real ao invés de conceitos computacionais
- Baseada em construções chamadas objetos, proporciona um paradigma evolucionário para:
 - ▣ criar modelos do mundo real em computador
 - ▣ usar estes modelos para simular o mundo real

Tecnologia OO



- Transição difícil para algumas pessoas
 - ▣ Linguagens de programação convencional forçam pensar em termos computacionais ao invés de em termos da aplicação

Métodos Desenvolvimento OO

- **CRC (ClasseResponsabilidadeColaborador) 1989 WirfsBrock:**
 - ▣ *Designing ObjectOriented Software, PrenticeHall, 1990.*
- **Modelos OOA e OOD (1991) Coad/ Yourdon**
 - ▣ *Análise Baseada em Objetos, Editora Campus, 1991*
 - ▣ *Projeto Baseado em Objetos, Editora Campus, 1991*
- **OMT (Object Modeling Technique) (1991) – James Rumbaugh**
 - ▣ *Modelagem Baseada em Objetos Editora Campus, 1991*
- **BOOCH (1991) – Grady Booch**
 - ▣ *ObjectOriented Design with Applications, Benjamin Cummings, 1991*
- **OBJECTORY OOSE (1992) – Ivar Jacobson**
 - ▣ *ObjectOriented Software Engineering, AddisonWesley*
- **Fusion (Booch, OMT, CRC, Métodos Formais) 1994 Colemann**

Como surgiu OO?

- 94: Parceria de metodologistas:
 - ▣ Booch (Booch Method) e Rumbaugh (OMT)
 - ▣ Busca de Unified Method (UM)
- 95: Rational compra Objective Systems de Jacobson (Objectory OOSE)
 - ▣ parceria Booch/Rumbaugh estendida com Jacobson
 - ▣ Unified Modeling Language (UML) ao invés de UM)
- 98: Reengenharia de livros, métodos e cases OO para incluir UML

Criadores da UML



- **“os três amigos”**
- James Rumbaugh
 - ▣ Object Modeling Technique (OMT)
- Grady Booch
 - ▣ Booch Method
- Ivar Jacobson
 - ▣ Objectory OOSE Process

O que é UML?



- É um padrão aberto
 - ▣ versão 1.1 aprovada pelo OMG (Object Management Group) em Novembro de 1997
 - ▣ versão 1.3 aprovada em Junho de 1999
 - ▣ 1.4, 2.0, 2.2 (2009)
- Suporta todo o ciclo de vida do software
 - ▣ modelagem do negócio (processos e objetos do negócio)
 - ▣ modelagem de requisitos alocados ao software
 - ▣ modelagem da solução de software

Parceiros da UML



- Rational Software Corporation
- HewlettPackard
- Ilogix
- IBM
- ICON Computing
- Intellicorp
- MCI Systemhouse
- Microsoft
- ObjectTime
- Oracle
- Platinum Technology
- Taskon
- Texas Instruments/Sterling Software
- Unisys

Modelos e Diagramas

- Modelos (UML 1.3)

- Diagramas de Casos de Uso

- Diagramas de Classes

- Diagramas de Objetos

- Diagramas de Componentes

- Diagramas de Distribuição

- Diagramas de Sequência

- Diagramas de Colaboração

- Diagramas de Estados

- Diagramas de Atividades

- http://www.omg.org/technology/documents/modeling_spec_catalog.htm#UML

UML DIAGRAMA DE CASO DE USO

Aula de Luiz Eduardo Guarino de Vasconcelos

Objetivos



- História
- Atores
- Casos de Uso
- Diagramas
- Estruturação (Generalização, Inclusão, Extensão)

Diagramas de Caso de Uso



- Facilitam o entendimento de um sistema mostrando a sua “visão externa”
- São usados para modelar o contexto de um sistema, subsistema ou classe
- Uma das maneiras mais comuns de documentar os requisitos do sistema (**as funcionalidades**)
 - ▣ Delimitam o Sistema
 - ▣ Definem a funcionalidade do sistema
- Representa os **requisitos funcionais** do sistema

Utilidade dos Casos de Uso



- Equipe de clientes (**validação**)
 - ▣ aprovam o que o sistema deverá fazer
 - ▣ entendem o que o sistema deverá fazer
- Equipe de desenvolvedores
 - ▣ Ponto de partida para refinar requisitos de software.
 - ▣ Podem seguir um desenvolvimento dirigido a casos de uso.
 - ▣ Designer (projetista): encontrar classes
 - ▣ Testadores: usam como base para **casos de teste**

Atores



- ❑ O sistema será descrito através de vários Casos de Uso que são executados por um número de atores
- ❑ Atores constituem as entidades do ambiente do sistema
- ❑ São pessoas ou outros subsistemas que interagem com o sistema em desenvolvimento
- ❑ Atores são externos ao sistema
- ❑ Primeiro é preciso identificar os usuários do sistema, que serão chamados de atores.
- ❑ Um ator é um tipo ou categoria de usuário.

Atores

Exemplos



- ❑ *Cargos (Empregado, Cliente, Gerente, Almoxarife, Vendedor, etc);*
- ❑ *Organizações (Empresa Fornecedora, Agência de Impostos, Administradora de Cartões, etc);*
- ❑ *Outros sistemas (Sistema de Cobrança, Sistema de Estoque de Produtos, etc).*
- ❑ *Equipamentos (Leitora de Código de Barras, Sensor, etc.)*

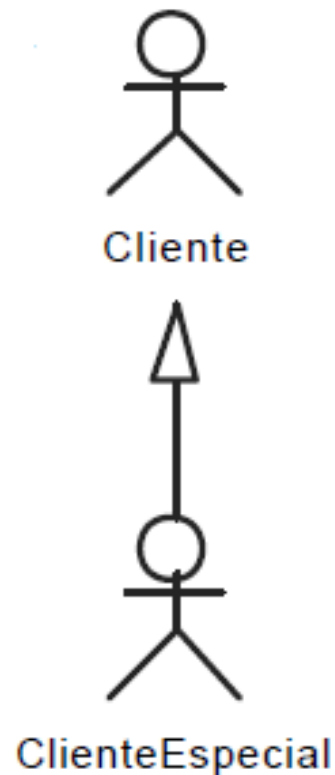
Atores

- Um ator é alguém ou algum outro sistema que deve interagir com o sistema em desenvolvimento



Especialização

- É possível definir tipos gerais de atores e especializá-los usando o relacionamento de especialização



Encontrando Atores

- Podem ser identificados pelas seguintes questões:
 - ▣ Quem usará a funcionalidade principal do sistema (atores primários)?
 - ▣ Quem precisará do auxílio do sistema para fazer suas tarefas diárias?
 - ▣ Quem precisará manter, administrar, conservar o sistema funcionando (atores secundários)?
 - ▣ Que dispositivos de hardware o sistema precisa para funcionar?
 - ▣ Com que outros sistemas o sistema precisa interagir?
 - ▣ Quem ou o que tem interesse nos resultados que o sistema produz?
- Não considere apenas os usuários que usam o sistema diretamente, mas todos os outros que precisam dos serviços do sistema

Casos de Uso

- Atores são examinados para determinar as suas necessidades
 - ▣ Gerente de Campanha - adiciona um novo cliente
 - ▣ Funcionário de Contato - Altera um contato do cliente
 - ▣ Contador - Registra o pagamento do cliente



Add new client



Change a client contact



Record client payment

Casos de Uso

Definição



- Uma unidade coerente de funcionalidade provida por um classificador (um sistema, subsistema ou classe) manifestado por uma sequência de mensagens trocadas entre o sistema e um ou mais usuários externos (representados como atores), junto com as ações executadas pelo sistema.

Casos de Uso

Descrição

- ❑ Pode ser numa forma resumida ou numa forma mais detalhada na qual a interação entre o ator e o caso de uso é descrita passo a passo.
- ❑ Descreve interações assim como o usuário vê, e não é uma definição de processos internos do sistema ou algum tipo de especificação de programa.
- ❑ Um documento com o fluxo de eventos é criado para cada caso de uso
- ❑ Escrito sob o ponto de vista de um ator
- ❑ Detalha o que o sistema deve oferecer para o ator quando o Caso de Uso é executado

Caso de Uso detalhado



- Entretanto, a UML não define nada acerca de como essa descrição textual deve ser construída.
- Por conta disso, há várias dimensões independentes sob as quais a descrição textual de um caso de uso pode variar:
 - ▣ Grau de abstração (essencial ou real)
 - ▣ Formato (contínua, tabular, numerado)
 - ▣ Grau de detalhamento (sucinta ou expandida)

Exemplo de descrição contínua

- Este caso de uso inicia quando o Cliente chega ao caixa eletrônico e insere seu cartão. O Sistema requisita a senha do Cliente. Após o Cliente fornecer sua senha e esta ser validada, o Sistema exibe as opções de operações possíveis. O Cliente opta por realizar um saque. Então o Sistema requisita o total a ser sacado. O Cliente fornece o valor da quantidade que deseja sacar. O Sistema fornece a quantia desejada e imprime o recibo para o Cliente. O Cliente retira a quantia e o recibo, e o caso de uso termina.

Exemplo de descrição numerada

- 1) Cliente insere seu cartão no caixa eletrônico.
- 2) Sistema apresenta solicitação de senha.
- 3) Cliente digita senha.
- 4) Sistema valida a senha e exibe menu de operações disponíveis.
- 5) Cliente indica que deseja realizar um saque.
- 6) Sistema requisita o valor da quantia a ser sacada.
- 7) Cliente fornece o valor da quantia que deseja sacar.
- 8) Sistema fornece a quantia desejada e imprime o recibo para o Cliente
- 9) Cliente retira a quantia e o recibo, e o caso de uso termina.

Exemplo de descrição tabular

Cliente	Sistema
<p>Inserir seu cartão no caixa eletrônico.</p> <p>Digitar senha.</p> <p>Solicitar realização de saque.</p> <p>Fornecer o valor da quantia que deseja sacar.</p> <p>Retira a quantia e o recibo.</p>	<p>Apresenta solicitação de senha.</p> <p>Valida senha e exibe menu de operações disponíveis.</p> <p>Requisita quantia a ser sacada.</p> <p>Fornecer a quantia desejada e imprimir o recibo para o Cliente</p>

Caso de Uso detalhado

Conteúdo

- Nome
- Descrição
- Identificador
- Importância
- Sumário
- Fluxo Principal
- Fluxos Alternativos
- Fluxos de Exceção
- Pós-condições
- Ator Primário
- Atores Secundários
- Pré-condições
- Regras do Negócio
- Histórico
- Notas de Implementação

Exemplo - Descrição Textual

CSU001 – Atualizar informações do Professor

Sumário: Administrador do Sistema usa o sistema para atualizar as informações cadastrais sobre professores a partir do SRH.

Ator Primário: Administrador

Ator Secundário: Sistema de Recursos Humanos (SRH)

Pré-condições: O Administrador está identificado pelo sistema

Fluxo Principal:

- 1.O administrador solicita ao sistema que obtenha os dados atualizados sobre professores
- 2.O sistema se comunica com o SRH e obtém os dados a partir deste.
- 3.O sistema apresenta os dados obtidos e solicita a confirmação do Administrador para realizar a atualização
- 4.O administrador confirma a atualização
- 5.O sistema atualiza os dados cadastrais dos professores e o caso de uso termina.

Fluxo de Exceção (2): Houve uma falha na obtenção de dados

- a.O sistema não consegue obter os dados a partir do SRH.
- b.O sistema reporta o fato e o caso de uso termina.

Fluxo Alternativo (4): Desistência de atualização

O administrador declina da atualização e o caso de uso termina.

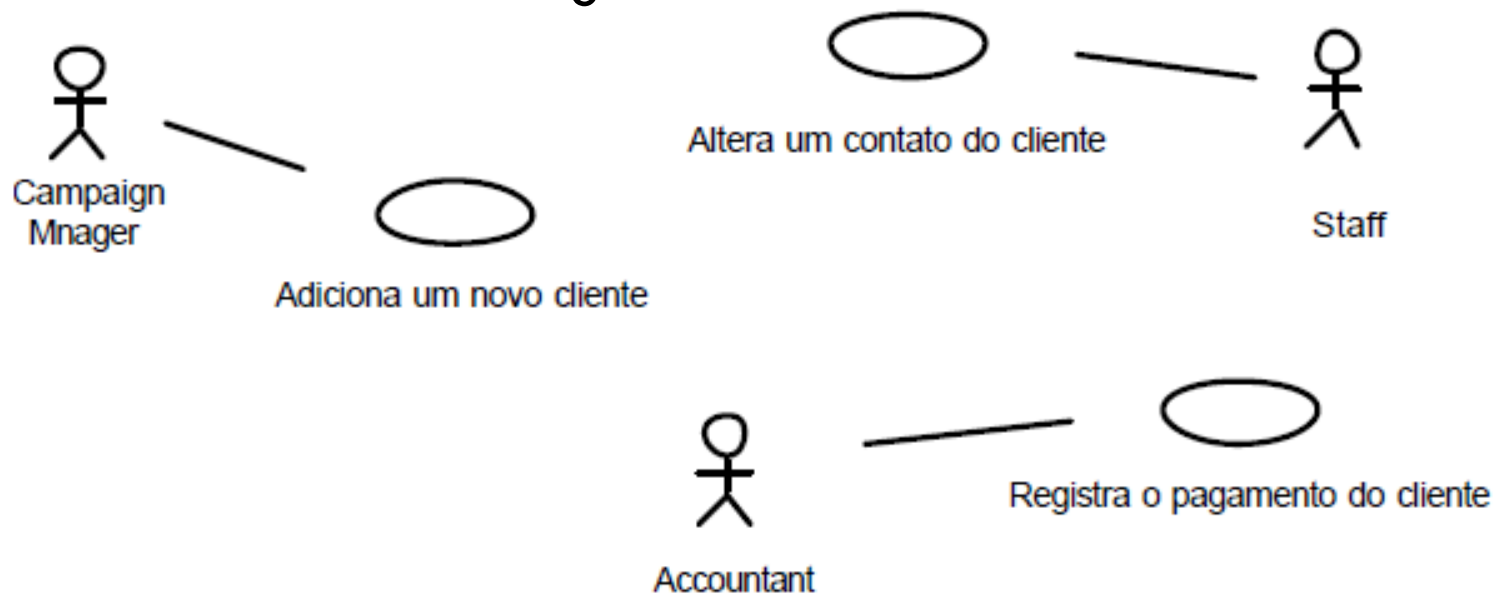
Caso de Uso detalhado

Dicas para documentar

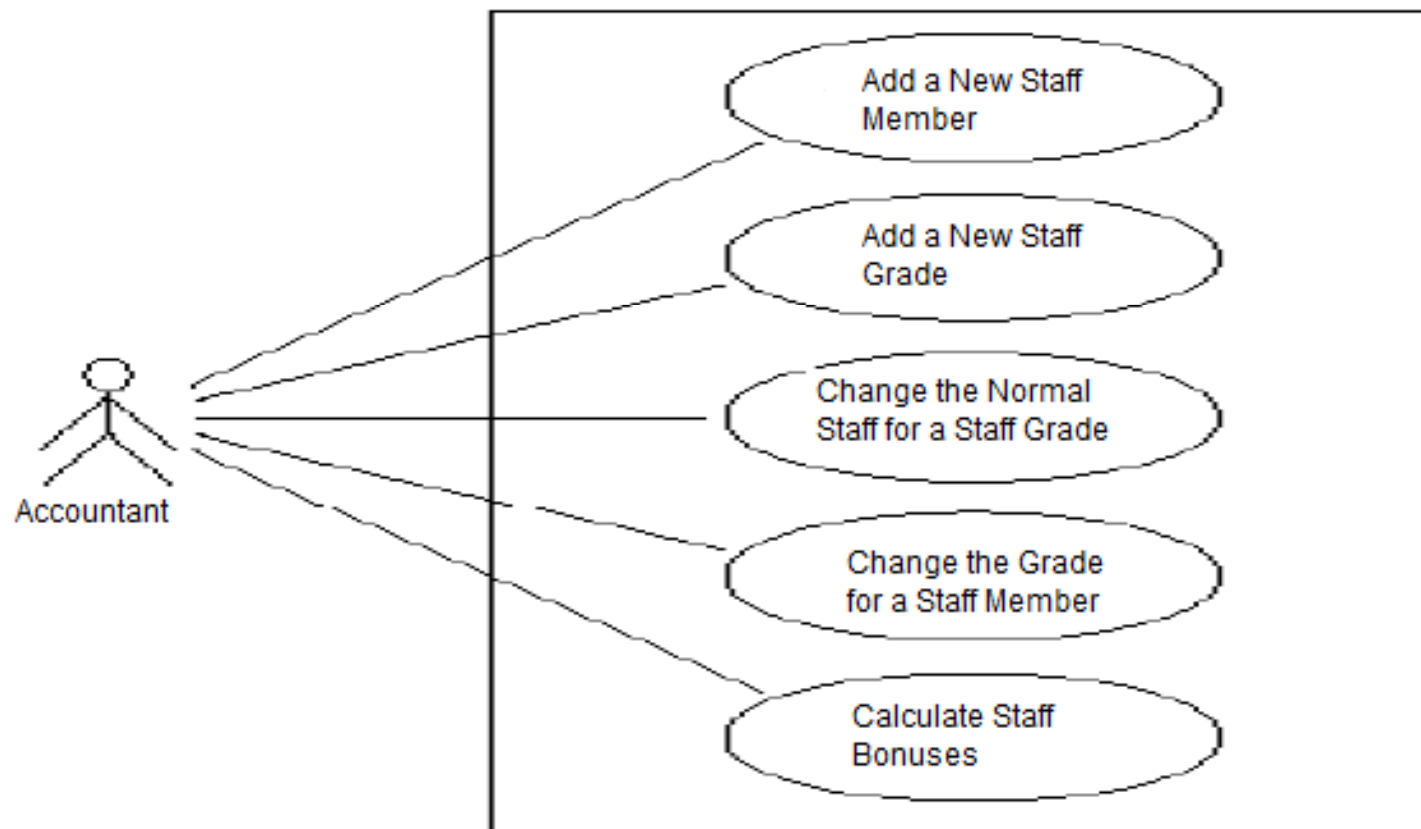
- ❑ Algumas boas práticas na documentação de casos de uso.
- ❑ Comece o nome do caso de uso com um verbo no infinitivo (para indicar um processo ou ação).
- ❑ Tente descrever os passos de caso de sempre na forma sujeito + predicado. Ou seja, deixe explícito quem é o agente da ação.
- ❑ Não descreva **como o sistema realiza internamente um passo de um** caso de uso.
- ❑ Tente dar nomes a casos de uso seguindo perspectiva do ator primário.
- ❑ Foque no **objetivo desse ator. Exemplos: Registrar Pedido, Abrir Ordem de Produção, Manter Referência, Alugar Filme, etc.**
- ❑ Tente manter a descrição de cada caso de uso no nível mais simples possível...

Diagrama de Caso de Uso

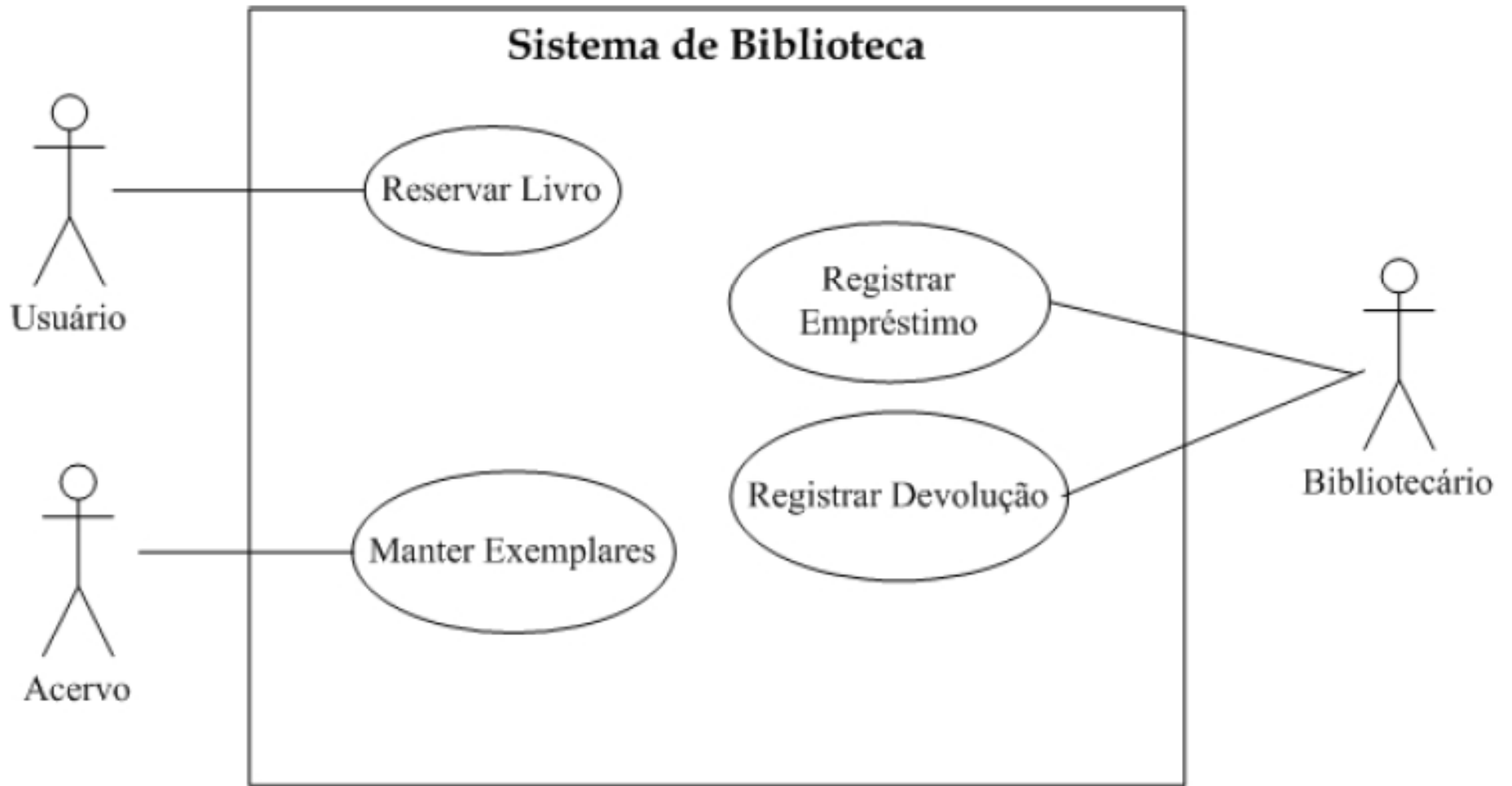
- Uma associação entre um ator e um use case indica que há uma comunicação, possivelmente com envio e recepção de mensagens
- São criados para visualizar as relações entre os atores e os Casos de Uso
- Permite dar uma visão global e de alto nível do sistema.



Exemplo 1



Exemplo 2



Encontrando Casos de Uso

- Faça as seguintes perguntas para cada ator
 - Que funções o ator requer do sistema? O que o ator precisa fazer?
 - O ator precisa ler, criar, destruir, modificar, ou armazenar alguns tipos de informações no sistema?
 - O ator tem que ser notificado sobre eventos no sistema? Ou o ator precisa notificar o sistema sobre alguma coisa? O que estes eventos representam em termos de funcionalidade?
 - O trabalho diário do ator poderia ser simplificado ou feito com mais eficiência através de novas funções no sistema?
- Sem considerar os atores atuais
 - Quais entradas/saídas o sistema precisa ? De onde as entradas vêm e para onde as saídas vão?
 - Quais são os maiores problemas com a implementação atual do sistema?

Organizando Casos de Uso



- Generalização
- Inclusão
- Extensão

Caso de Uso

Generalização



- ❑ Relaciona um Use Case especializado a um mais geral
- ❑ O filho herda os atributos, operações e sequências de comportamento dos pais
- ❑ O filho pode adicionar e redefinir o comportamento do pai
- ❑ O filho pode substituir o pai em qualquer lugar que ele aparece

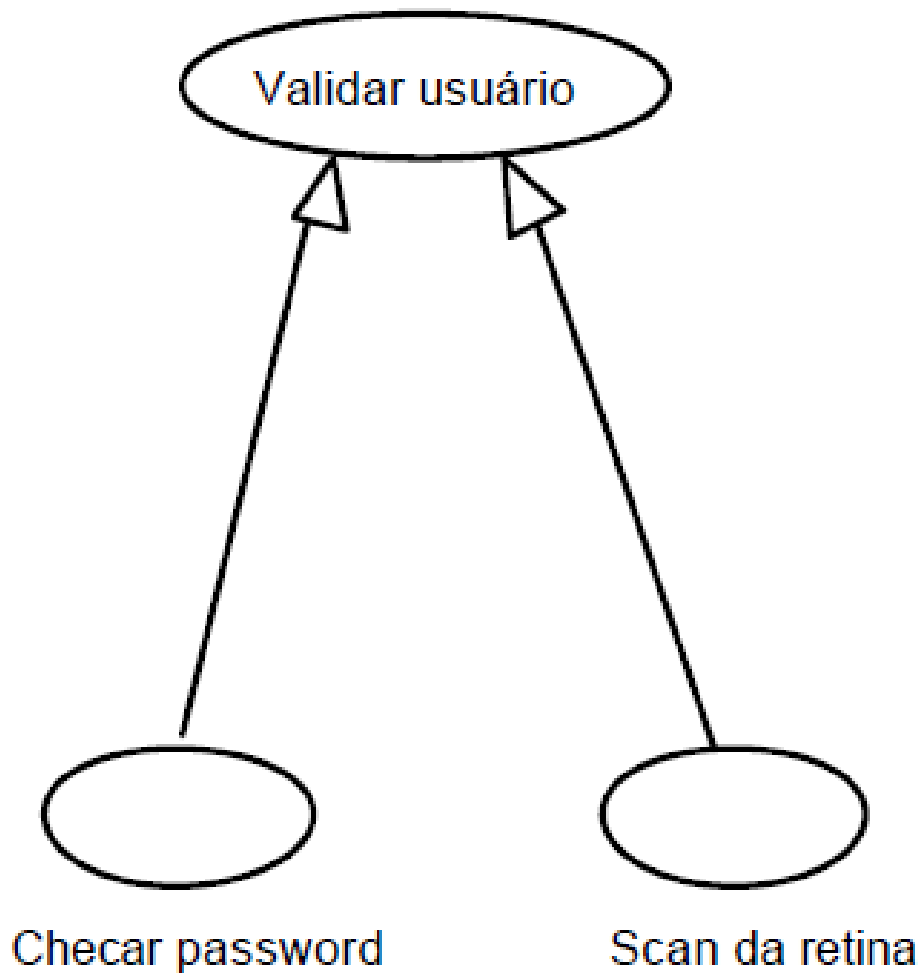
Caso de Uso

Generalização

- É possível abstrair comportamentos de use cases. Normalmente a similaridade entre use cases é identificada após a construção do use case.
- Os use cases Checar password e Scan de retina ambos servem para validar o usuário.
- Identificar um use case abstrato Validar usuário para realizar esta validação.

Caso de Uso

Generalização



Caso de Uso

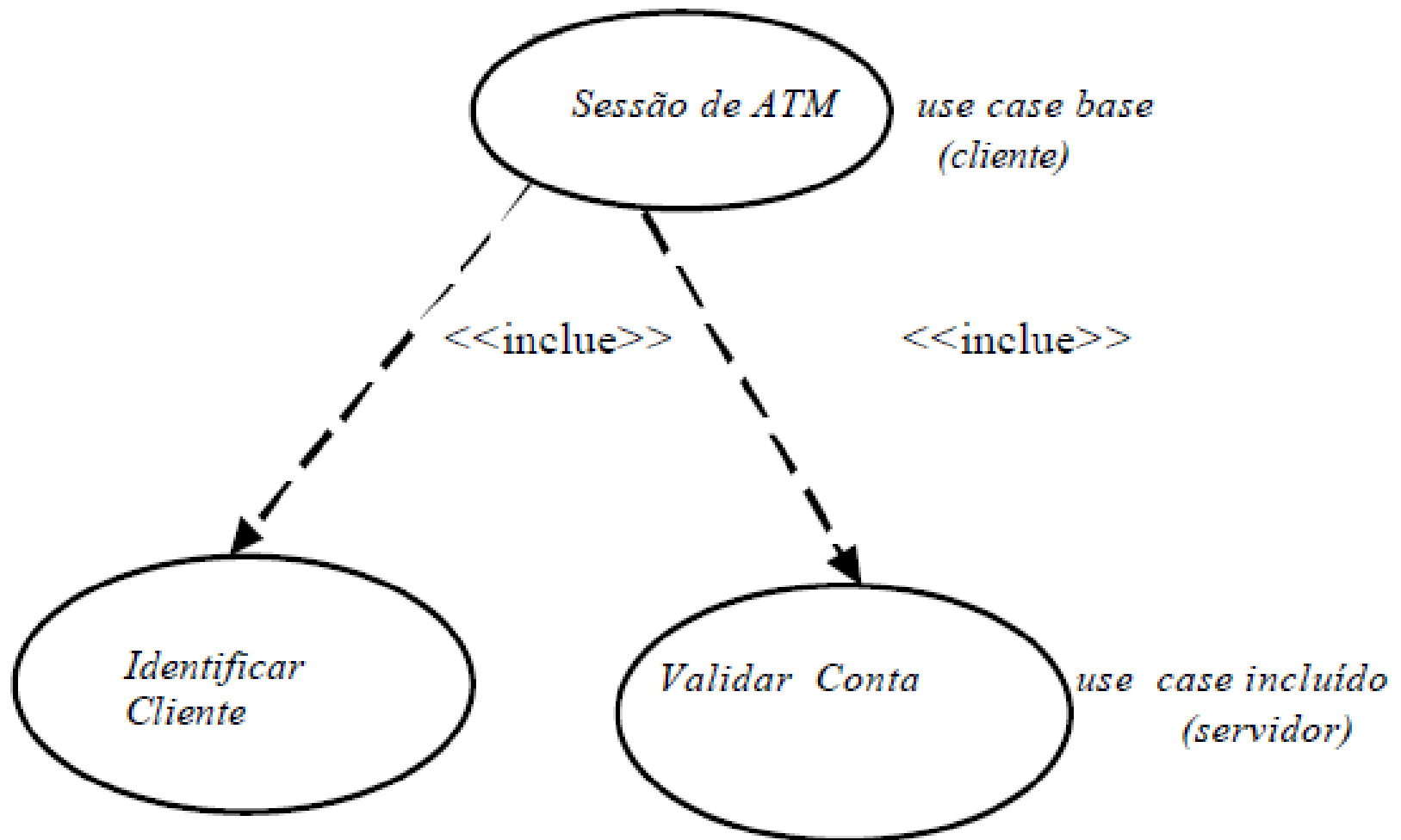
Inclusão



- ❑ O use case base incorpora explicitamente o comportamento de outro use case no local especificado na base.
- ❑ O use case incluído nunca estará sozinho, somente será instanciado de um use case base que o incluirá.
- ❑ Usado para evitar a descrição do mesmo fluxo de eventos várias vezes.

Caso de Uso

Inclusão



Caso de Uso

Inclusão

- Use Case: Sessão de ATM
- mostre anúncio do dia
- **inclue Identificar Cliente**
- **inclue Validar Conta**
- imprimir cabeçalho do recibo
- log out

- Use Case de Inclusão: Identificar Cliente
- pegue o nome do cliente
- **inclue Verificar Identidade**
- **if falha de verificação then abort a sessão**
- obtenha número da conta do cliente

- Use Case de Inclusão: Validar Conta
- estabeleça conexão com banco de dados de contas
- obtenha status e limite da conta

Caso de Uso

Inclusão



- ❑ Descreve uma sequência adicional de comportamento que será inserida na instância de use case que está executando o use base base
- ❑ O mesmo use case de inclusão pode ser inserido em múltiplos use case base
- ❑ A inclusão representa comportamento encapsulado que potencialmente poderá ser reusado em outros use case base

Caso de Uso

Extensão

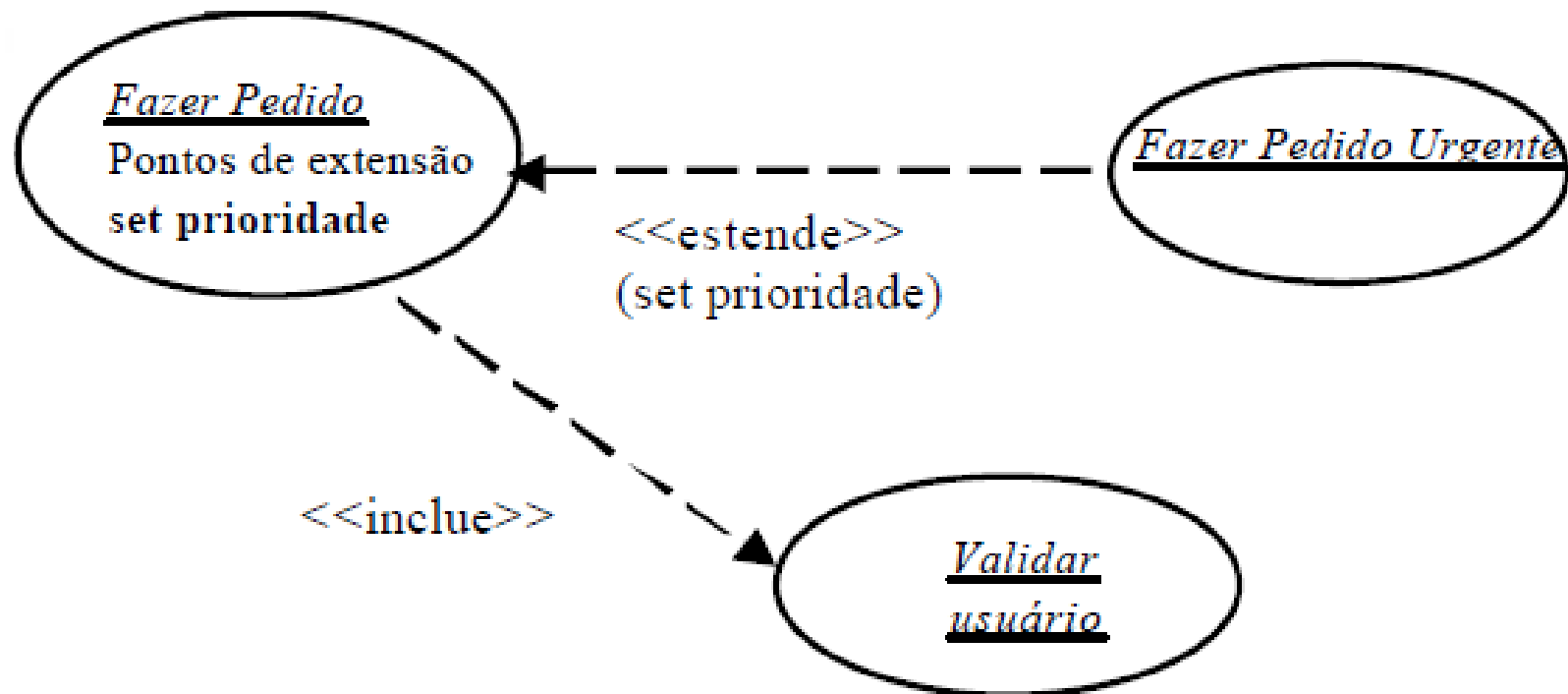


- ❑ Usado para:
- ❑ Para modelar partes opcionais de use cases
- ❑ Para modelar cursos alternativos e complexos que raramente ocorrem, como Entrega Urgente
- ❑ Para modelar sub-cursos que são executados somente em certos casos

Caso de Uso

Extensão

- Use Case *Fazer Pedido*
- Fluxo principal de eventos: *inclue (Validar usuário)*.
- Receber do usuário os itens do pedido. (*set prioridade*).
- Submeter o pedido para processamento.



Caso de Uso

Extensão



- Cada ponto de extensão precisa ser definido no use case base.
- Quando a execução da instância do use case alcança o local do use case referenciado pelo ponto de extensão e a condição da extensão é satisfeita, a execução é transferida para a sequência do segmento de extensão. Quando terminada, o controle volta para o use case original

Documentação Associada



- O modelo de casos de uso força o desenvolvedor a pensar em como os agentes externos interagem com o sistema.
- No entanto, este modelo corresponde somente aos requisitos funcionais.
- Outros tipos de requisitos (desempenho, interface, segurança, regras do negócio, etc.) também devem ser identificados e modelados.
- Esses outros requisitos fazem parte da documentação associada ao MCU.
- Dois itens importantes dessa documentação associada são o ***modelo de regras do negócio e os requisitos de desempenho***

Regras do Negócio

- São políticas, condições ou restrições que devem ser consideradas na execução dos processos de uma organização.
 - ▣ Descrevem a maneira pela qual a organização funciona.
- Estas regras são identificadas e documentadas no chamado **modelo de regras do negócio (MRN)**.
 - ▣ A descrição do modelo de regras do negócio pode ser feita utilizando-se texto informal, ou através de alguma forma de estruturação.
- Regras do negócio normalmente influenciam o comportamento de determinados casos de uso.
 - ▣ Quando isso ocorre, os identificadores das regras do negócio devem ser adicionados à descrição dos casos de uso em questão.
 - ▣ Uso da seção “regras do negócio” da descrição do caso de uso.

Exemplos de Regras de Negócio

- ❑ O valor total de um pedido é igual à soma dos totais dos itens do pedido acrescido de 10% de taxa de entrega.
- ❑ Um professor só pode estar lecionando disciplinas para as quais esteja habilitado.
- ❑ Um cliente de uma das agências do banco não pode retirar mais do que R\$ 1.000 por dia de sua conta. Após as 18:00h, esse limite cai para R\$ 100,00.
- ❑ Os pedidos para um cliente não especial devem ser pagos antecipadamente.

Regras do Negócio

- Possível formato para documentação de uma regra de negócio no MRN.

Nome	<code>Quantidade de inscrições possíveis (RN01)</code>
Descrição	<code>Um aluno não pode ser inscrever em mais de seis disciplinas por semestre letivo.</code>
Fonte	<code>Coordenador da escola de informática</code>
Histórico	<code>Data de identificação: 12/07/2002</code>

Requisitos de Desempenho

- Conexão de casos de uso a requisitos de desempenho.

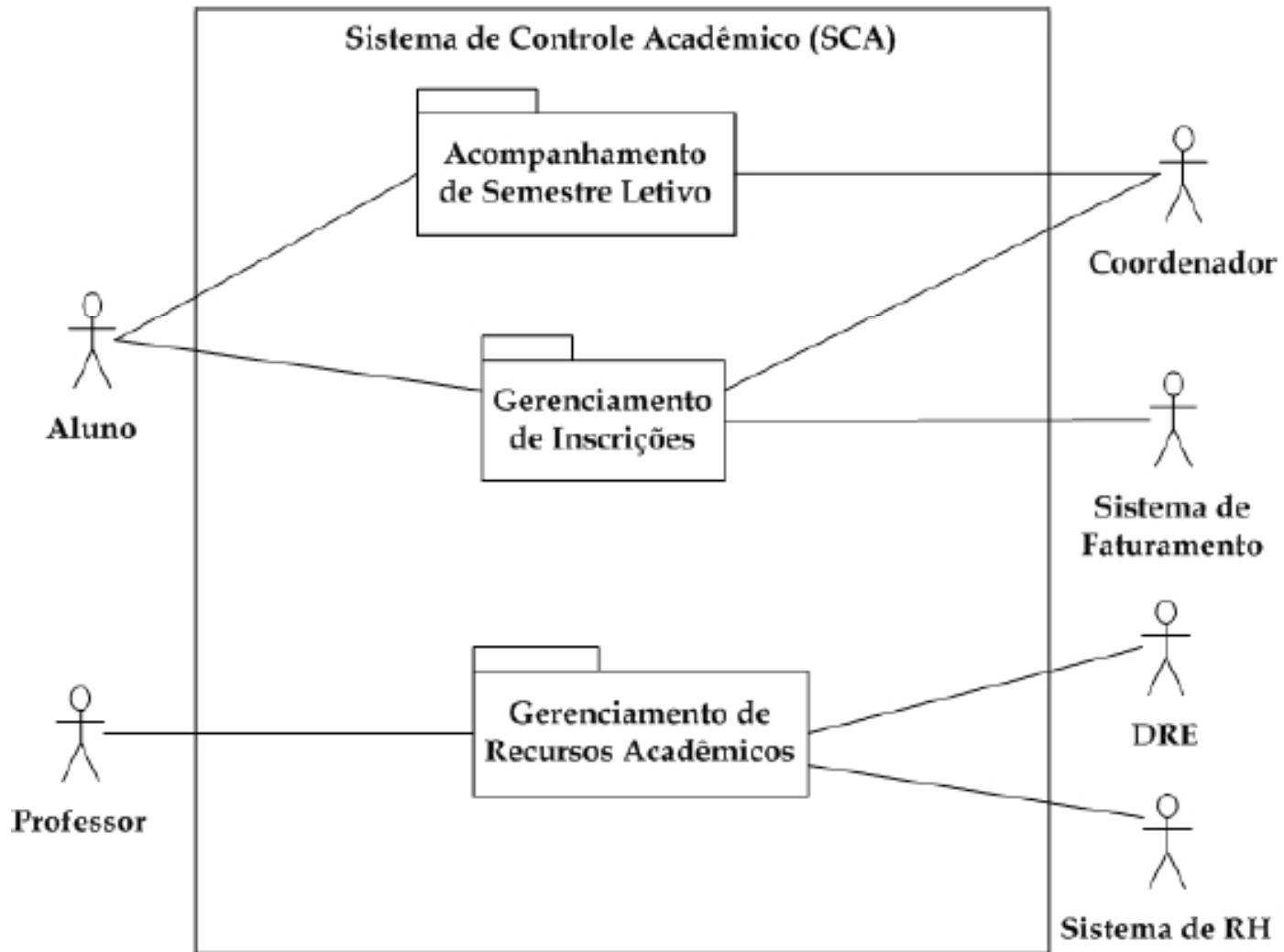
Identificador do caso de uso	Frequência da utilização	Tempo máximo esperado	...
CSU01	5/mês	Interativo	...
CSU02	15/dia	1 segundo	...
CSU03	60/dia	Interativo	...
CSU04	180/dia	3 segundos	...
CSU05	600/mês	10 segundos	...
CSU07	500/dia durante 10 dias seguidos.	10 segundos	...

Modelando o DCU



- Em sistemas complexos, representar todos os casos de uso do sistema em um único DCU talvez o torne um tanto ilegível.
- Alternativa: criar vários diagramas (de acordo com as necessidades de visualização) e agrupá-los em pacotes.
 - Todos os casos de uso para um ator;
 - Todos os casos de uso a serem implementados em um ciclo de desenvolvimento.
 - Todos os casos de uso de uma área (departamento, seção) específica da empresa.

Modelando o DCU



Modelando o Contexto do Sistema



- Identifique os atores que cercam o sistema
- ? Quais grupos precisam de ajuda do sistema para executarem suas tarefas
- ? Quais os grupos necessários para executarem as funções do sistema
- ? Quais grupos interagem com hardware externo ou outros sistemas de software
- ? Quais grupos executam funções secundárias de administração e manutenção

Modelando o Contexto do Sistema



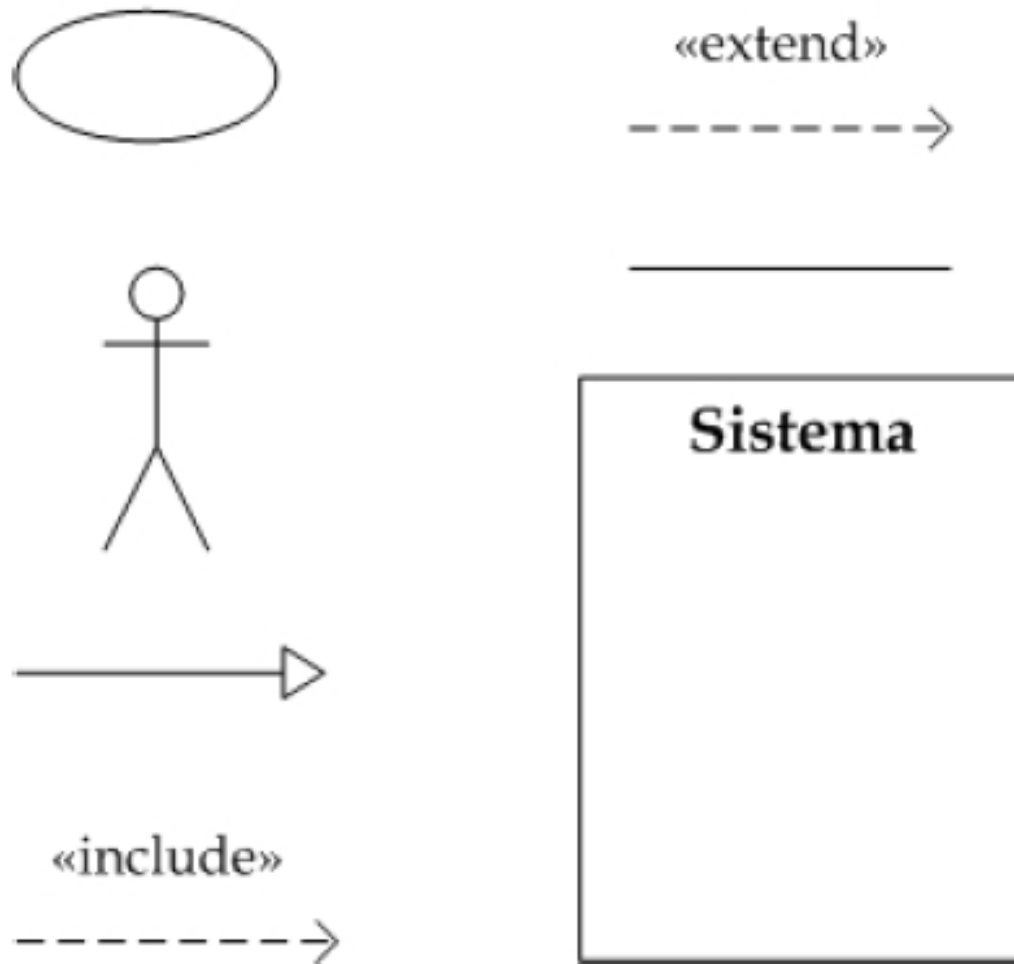
- ❑ ? Organize os atores que são similares em uma hierarquia de generalização / especialização.
- ❑ ? Quando ajudar a compreensão, faça um esteriótipo para o ator
- ❑ ? Use os atores no diagrama de use cases e especifique os caminhos de comunicação entre atores e use cases do sistema.

Modelando Requisitos do Sistema



- ❑ Estabeleça o contexto do sistema através da identificação dos atores que o cercam
- ❑ Para cada ator, considere o comportamento que eles esperam e requerem que o sistema produza
- ❑ De um nome aos comportamentos comuns (Caso de Uso)
- ❑ Fatore comportamentos comuns em novos use cases que serão usados por outros
- ❑ Fatore comportamento variante em novos use cases que estendem a fluxo principal de eventos
- ❑ Modele os use cases, atores e seus relacionamentos através de diagramas de use case

Resumo da Notação



Exercício 1



- Construa um modelo de casos de uso para a seguinte situação fictícia: *"Estamos criando um serviço de entregas. Nossos clientes podem nos requisitar a entrega de volumes. Alguns volumes são considerados de maior valor por nossos clientes, e, portanto, eles querem ter tais volumes segurados durante o transporte. Contratamos uma companhia de seguro para segurar volumes de valor"*.

Exercício 2

- Uma loja de Cds possui discos para venda e locação. Um cliente pode comprar ou locar uma quantidade ilimitada de discos. Para locar é obrigatório que o cliente esteja cadastrado na loja. A loja possui um funcionário cuja função é atender os clientes durante a venda e locação dos discos. Suas principais tarefas são: conferir o pagamento efetuado pelos clientes; emitir recibo de venda(emissão obrigatória) e locação (emissão obrigatória); ao final do dia, anotar em uma caderneta o valor de cada venda.

Exercício 3

- O vendedor de uma loja de eletrodomésticos, ao efetuar uma venda, encaminha o cliente para o caixa para a realização do pagamento do eletrodoméstico adquirido. Independentemente da forma de pagamento, o caixa deve verificar, se o cliente não consta do SPC. Após ter recebido o pagamento do cliente, o caixa deve emitir a nota fiscal ao consumidor. É efetuada a baixa no estoque ao final do dia. No final do mês o vendedor deve emitir um relatório de vendas realizadas para o gerente da loja .

Exercício 4

- Uma prefeitura municipal, através do Secretário Municipal de Saúde, cadastra todos os médicos que se dispõem a trabalhar no serviço público de saúde do município. A prefeitura possui diversas unidades de atendimento (hospitais, postos de saúde, ambulatórios médicos, etc) e o Secretário Municipal de Saúde também mantém o cadastro destas unidades. Os cidadãos que desejam ter acesso ao atendimento do sistema público do município são cadastrados, pelos funcionários das unidades de atendimento, previamente ou no momento de algum atendimento. O cidadão pode agendar consultas médicas em qualquer unidade de atendimento, através dos funcionários da unidade de atendimento. Mensalmente, o Secretário Municipal de Saúde estabelece uma escala de médicos para cada unidade de atendimento. A qualquer momento, o Secretário Municipal de Saúde pode extrair relatórios com indicadores do funcionamento do sistema. Diariamente, os funcionários das unidades de atendimento listam as consultas médicas agendadas para cada médico, para acompanhamento.

Exercício 5

- Uma empresa de organização de eventos gerencia seus compromissos da forma descrita a seguir. Os clientes são cadastrados pelos representantes da empresa, juntamente com o evento que deseja que seja organizado. Se um cliente já existir no momento de cadastrar um evento, é verificado se seus dados estão atualizados e, se não estiver, as alterações de cadastro são realizadas. Uma vez definido o evento, inicia-se um processo de divulgação do evento, pelo representante da empresa, aos potenciais participantes do evento, através de mala direta, utilizando-se um banco de dados. A qualquer momento, o representante da empresa pode acrescentar nomes neste banco de dados. O representante da empresa também pode emitir relatórios de providências a serem tomadas, a medida em que se aproxima o evento. Após a realização do evento, o representante faz o balanço, para sua apuração de custos e lucro.

UML

DIAGRAMA DE CLASSE

Aula de Luiz Eduardo Guarino de Vasconcelos

Diagrama de Classe



- Um diagrama de classes serve para modelar o vocabulário de um sistema, do ponto de vista do utilizador/problema ou do implementador/solução
 - ▣ Ponto de vista do utilizador/problema – na fase de captura e análise de requisitos, em paralelo com a identificação dos casos de utilização
 - ▣ Vocabulário do implementador/solução – na fase de projeto
- Construído e refinado ao longo das várias fases do desenvolvimento do software

Objetivos



- Também serve para:
 - ▣ Especificar colaborações (no âmbito de um caso de utilização ou mecanismo)
 - ▣ Especificar esquemas lógicos de bases de dados
 - ▣ Especificar visões (estrutura de dados de formulários, relatórios, etc.)
- Modelos de objetos de domínio, negócio, análise e design

Pacotes



- Organiza as classes de objetos em grupos.
- Melhorar a organização do sistema subsistemas
- Estrutura hierarquicamente o projeto
- Estrutura física dos arquivos do projeto
- Nomenclatura
 - ▣ Minúsculo
 - ▣ Endereços de web

Pacotes

br.edu.fatec.academico

br.com.uol.internet

org.kernel.drives

Pacotes

br.edu.fatec.academico

graduacao

posgraduacao

Mundo Real x Computacional



- No desenvolvimento de software orientado por objetos, procura-se imitar no computador o mundo real visto como um conjunto de objetos que interagem entre si
- Muitos objetos computacionais são imagens de objetos do mundo real
- Exemplos de objetos do mundo real:
 - ▣ o Sr. João
 - ▣ a aula de ES no dia 11/10/2000 às 11 horas

Objetos

- Um objeto é algo com fronteiras bem definidas, relevante para o problema em causa, com estado, modelado por valores de **atributos** (tamanho, forma, peso, etc.) e por **ligações** que num dado momento tem com outros objetos
- **Comportamento**
 - ▣ um objeto exhibe comportamentos invocáveis (por resposta a chamadas de operações) ou reativos (por resposta a eventos)
- **Identidade**
 - ▣ no espaço: é possível distinguir dois objetos mesmo que tenham o mesmo estado
 - ▣ no tempo: é possível saber que se trata do mesmo objeto mesmo que o seu estado mude

Classes



- Uma classe é um descritor de um conjunto de objetos que partilham as mesmas propriedades (semântica, atributos, operações e relações)
- Um objeto de uma classe é uma instância da classe
- A extensão de uma classe é o conjunto de instâncias da classe

Classes

- Em UML, uma classe é representada por um retângulo com o nome da classe
- Habitualmente escreve-se o nome da classe no singular (nome de uma instância), com a 1ª letra em maiúscula

Aluno

Triangulo

ItemAgenda

Disciplina

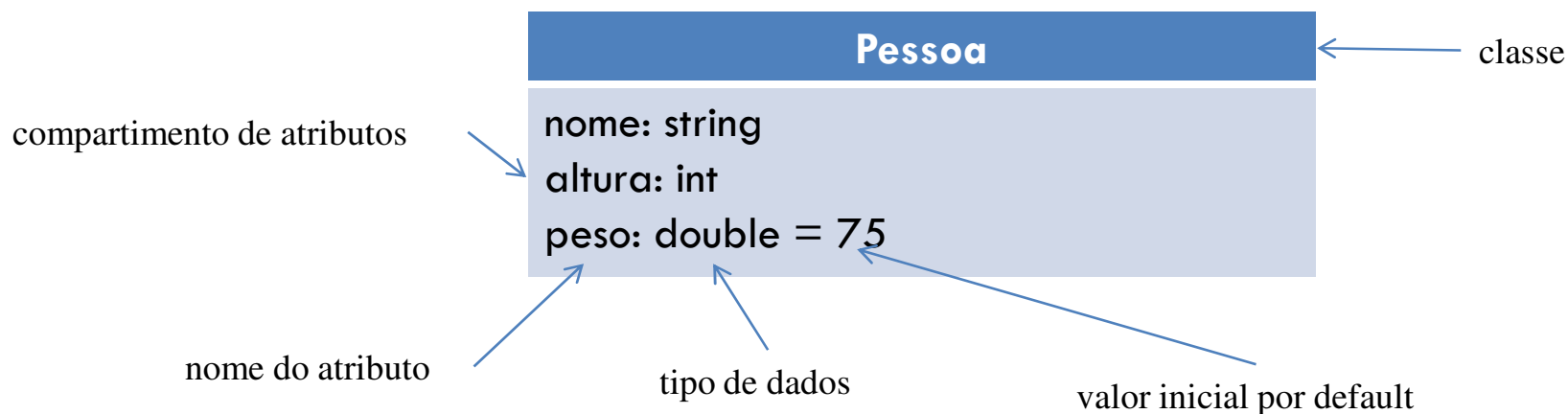
Equacao2Grau

Atributos

- O estado de um objeto é dados por valores de atributos (e por ligações que tem com outros objetos)
- Todos os objetos de uma classe são caracterizadas pelos mesmos atributos (ou variáveis de instância)
 - ▣ o mesmo atributo pode ter valores diferentes de objeto para objeto
- Atributos são definidos ao nível da classe, enquanto que os valores dos atributos são definidos ao nível do objeto
- Exemplos:
 - ▣ uma pessoa (classe) tem os atributos nome, data de nascimento e peso
 - ▣ João (objeto) é uma pessoa com nome “João Silva”, data de nascimento “18/3/1973” e peso “68 Kg”

Atributos

- ❑ Atributos são listados num compartimento de atributos (opcional) a seguir ao compartimento com o nome da classe
- ❑ Uma classe não deve ter dois atributos com o mesmo nome
- ❑ Os nomes dos tipos não estão pré-definidos em UML, podendo-se usar os da linguagem de implementação



Exemplo

- Nome do atributo em minúsculo e a primeira letra da concatenação de palavra em maiúscula

Expressao2Grau

a: double
b: double
c: double

Triangulo

ladoA: double
ladoB: double
ladoC: double
anguloAB: double
anguloBC: double
anguloCA: double

Atributos estáticos

- Atributo estático: tem um único valor para todas as instâncias (objetos) da classe
 - ▣ valor está definido ao nível da classe e não ao nível das instâncias
- Sublinha o atributo estático

Prova
<u>inicio: Date</u>
termino: Date
nota: Real

Operações (Métodos)



- Comportamento invocável de objetos é modelado por operações
 - ▣ uma operação é algo que se pode pedir para fazer a um objeto de uma classe
 - ▣ objetos da mesma classe têm as mesmas operações
- Operações são definidos ao nível da classe, enquanto que a invocação de uma operação é definida ao nível do objeto

Operações (Métodos)

- ❑ **Princípio do encapsulamento:**
acesso e alteração do estado interno do objeto (valores de atributos e ligações) controlado por **operações** (Padrão POJO - POCO)
- ❑ Nas classes que representam objetos do mundo real é mais comum definir responsabilidades em vez de operações
- ❑ Nome do atributo em minúsculo e a primeira letra da concatenação de palavra em maiúscula
- ❑ **Método Construtor e Destrutor**

Pessoa

nome: string

endereco: string

Pessoa()

getNome(): string

setNome (v:string): void

getEndereco(): string

setEndereco(v:string): void

~Pessoa()

Exemplo

Circulo

centroX: double

centroY: double

raio: double

getCentroX(): double

setCentroX (v:double): void

getCentroY(): double

setCentroY (v:double): void

getRaio(): double

setRaio(v:double): void

area(): double

perimetro(): double

Aluno

nome: string

prova: double

trabalho: double

getNome(): string

setNome (v:string): void

getProva(): double

setProva (v:double): void

getTrabalho(): double

setTrabalho(v:double): void

media(): double

aprovado(): boolean

Métodos estáticos

- ❑ Operação estática: não é invocada para um objeto específico da classe
- ❑ Não tem uma instância da classe para invocar o método.
- ❑ Sublinha o método estático

Triangulo

ladoA: double

ladoB: double

ladoC: double

hipPitagoras(catA: double, catB: double): double

Visibilidade de atributos e operações

- Princípio do **encapsulamento**: esconder todos os detalhes de implementação que não interessam aos clientes (utilizadores) da classe
 - ▣ permite alterar representação do estado sem afetar clientes
 - ▣ permite validar alterações de estado
- Visibilidade
 - + (public) : visível por todos
 - (private) : visível só por operações da própria classe
 - # (protected): visível por operações da própria classe e descendentes (subclasses)

Exemplo

Circulo

-centroX: double

-centroY: double

-raio: double

+getCentroX(): double

+setCentroX (v:double): void

+getCentroY(): double

+setCentroY (v:double): void

+getRaio(): double

+setRaio(v:double): void

+area(): double

+perimetro(): double

Aluno

-nome: string

-prova: double

-trabalho: double

+getNome(): string

+setNome (v:string): void

+getProva(): double

+setProva (v:double): void

+getTrabalho(): double

+setTrabalho(v:double): void

+media(): double

+aprovado(): boolean

UML DIAGRAMA DE CLASSE RELACIONAMENTO

Aula de Luiz Eduardo Guarino de Vasconcelos

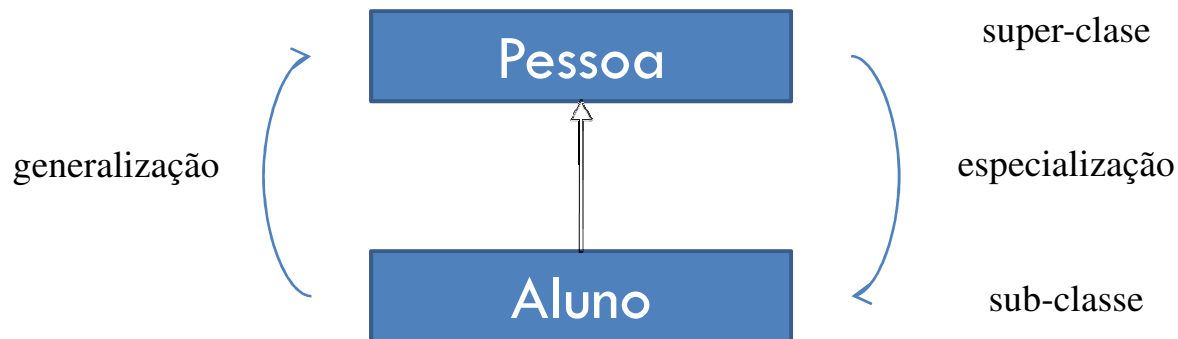
Objetivos



- Generalização
- Associação
- Agregação
- Composição
- Dependência

Generalização

- **Relação semântica “is a” (“é um” / “é uma”)**
 - ▣ um aluno é uma pessoa
- **Relação de herança nas propriedades**
 - ▣ A subclasse herda as propriedades (atributos, operações e relações) da superclasse, podendo acrescentar outras

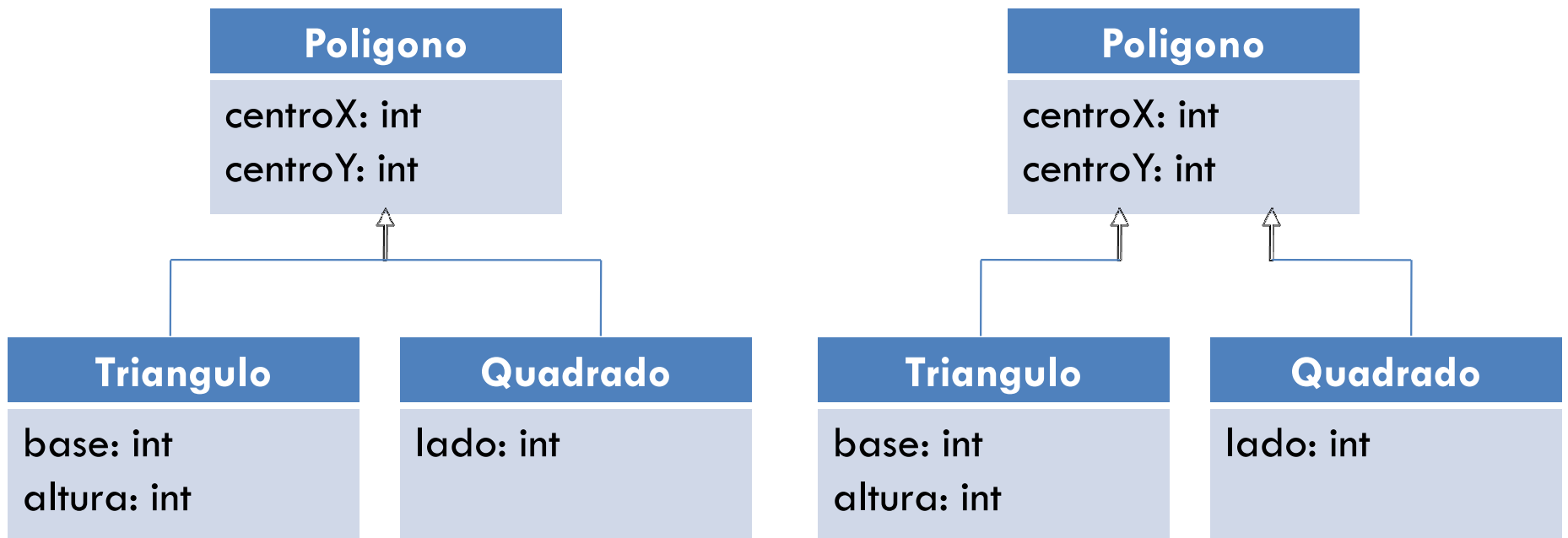


As 3 facetas da generalização

- Substitutabilidade
 - ▣ onde se espera um objeto da superclasse pode-se passar um objeto duma subclasse
- Herança de interface
 - ▣ a subclasse herda as assinaturas (e significados) das operações da superclasse
- Herança ou overriding de implementação
 - ▣ a subclasse pode herdar as implementações das operações da superclasse, mas também pode ter novas implementações de algumas dessas operações
 - ▣ quando em UML se repete numa subclasse a assinatura de uma operação da superclasse, quer dizer que tem uma nova implementação na subclasse

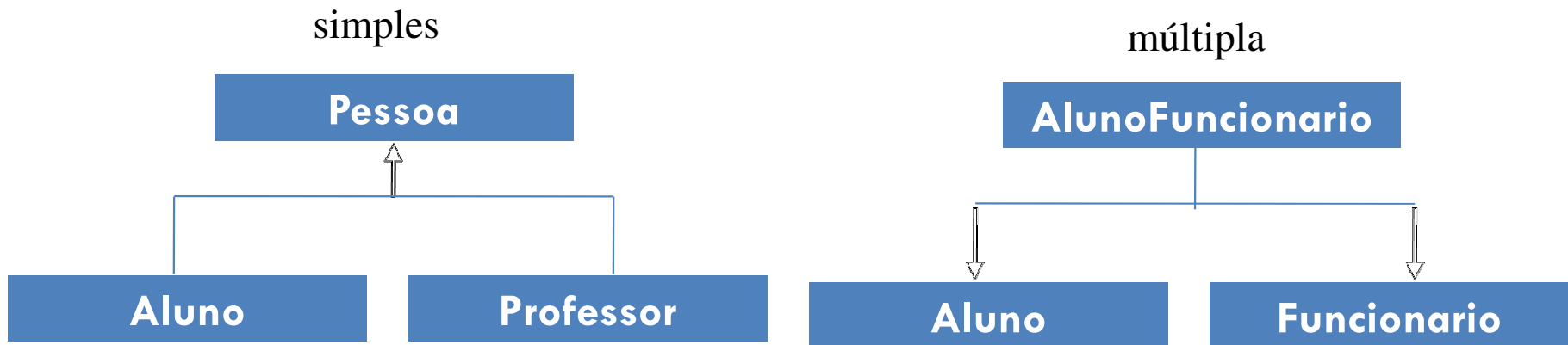
Hierarquia das classes

- Na super classe da hierarquia colocam-se as propriedades que são comuns a todas as suas subclasses
- Evita-se redundância, promove-se reutilização!



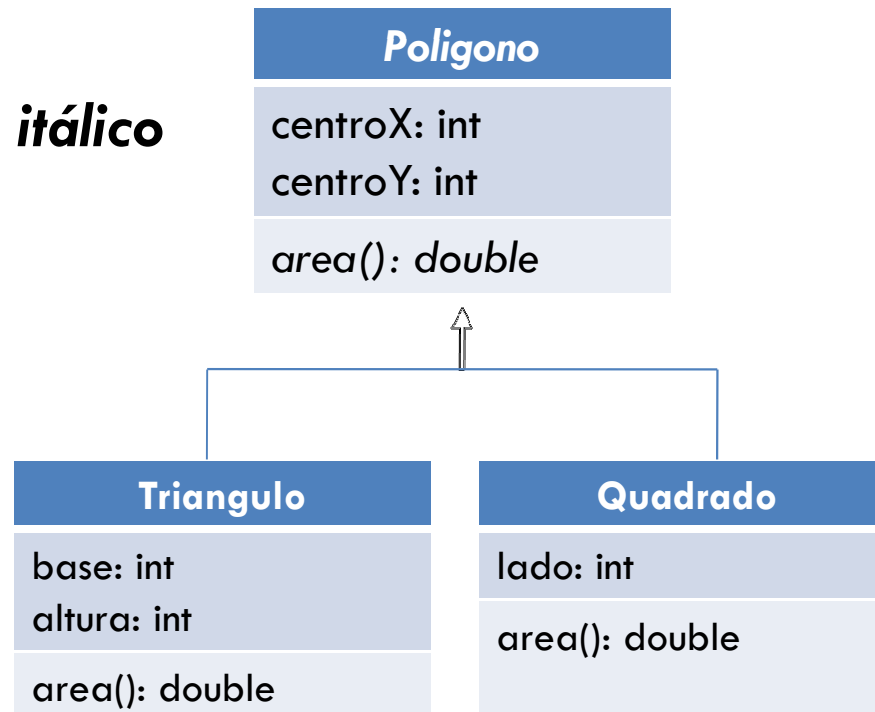
Herança simples / múltipla

- Herança Simples → quando uma subclasse possui apenas uma superclasse
- Herança Múltipla → quando uma subclasse possui mais de uma superclasse



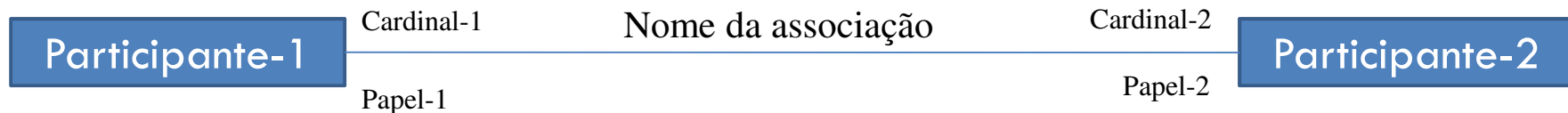
Classes e operações abstratas

- ❑ Classe abstrata: classe que não pode ter instâncias diretas. As instâncias somente pelas subclasses concretas
- ❑ Operação abstrata: operação com implementação a definir nas subclasses. Uma classe com operações abstratas tem de ser abstrata
- ❑ Notação : ***nome em itálico***



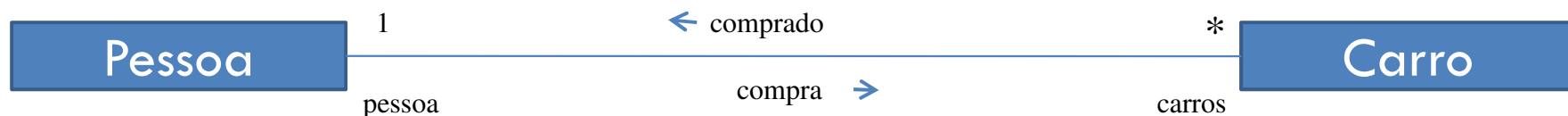
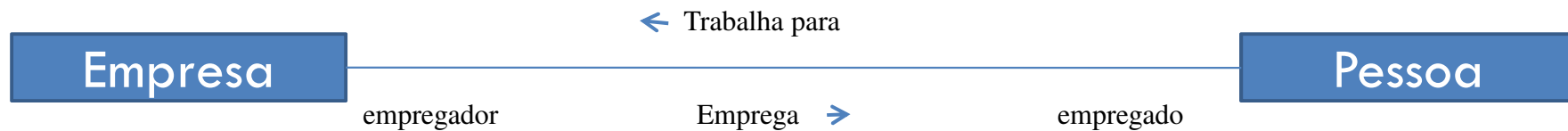
Associação

- ❑ Uma associação é uma relação entre objetos das classes participantes (um objeto de cada classe em cada ligação)
- ❑ Essa ligação é uma instância de uma associação
- ❑ Implementado através de uma referência entre os objetos relacionados
- ❑ Pode haver mais do que uma associação (com nomes diferentes) entre o mesmo par de classes
- ❑ Papéis nos extremos da associação podem ter indicação de visibilidade (pública, privada, etc.)



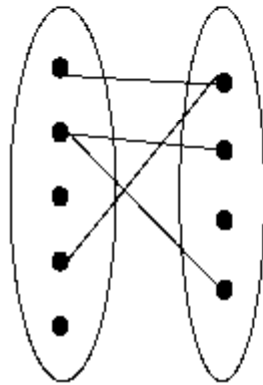
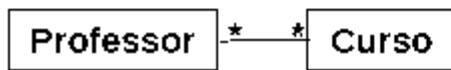
Nome da Associação

- ❑ A indicação do nome é opcional
- ❑ O nome é indicado no meio da linha que une as classes participantes
- ❑ Pode-se indicar o sentido em que se lê o nome da associação

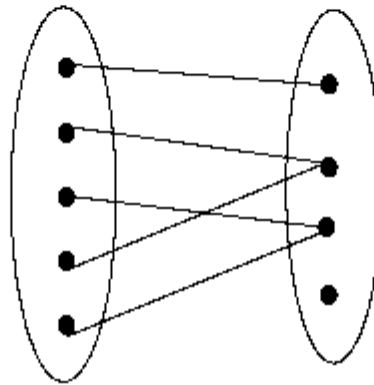
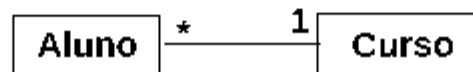


Multiplicidade de Associações

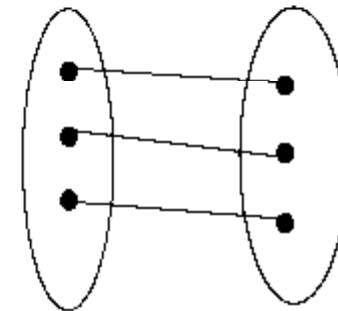
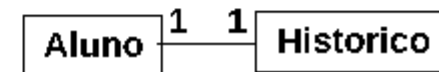
Muitos-para-Muitos



Muitos-para-1

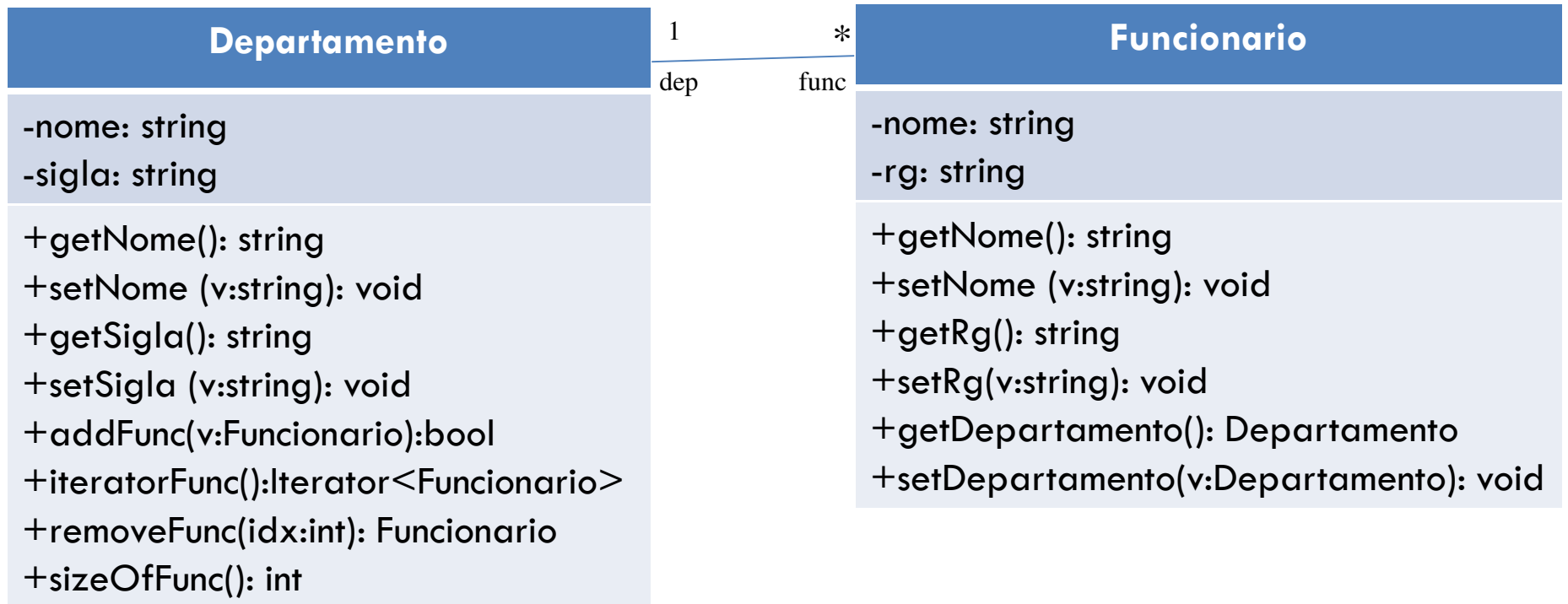


1-para-1



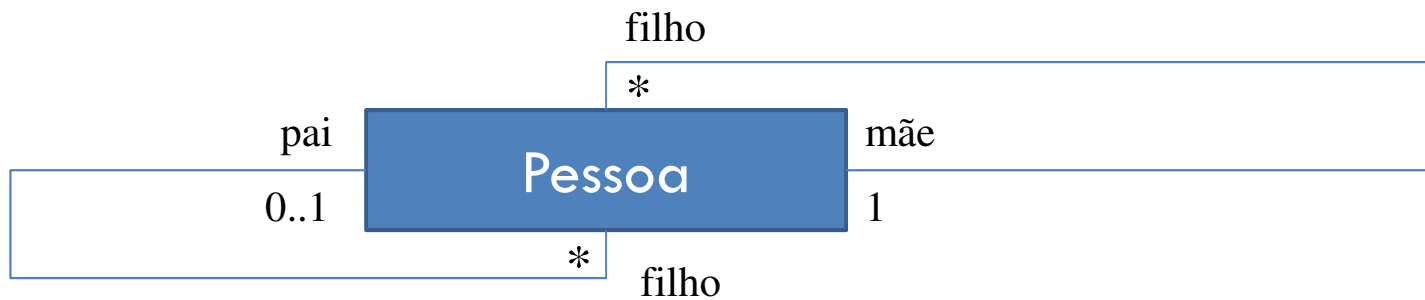
- 1 - exatamente um
- 0..1 - zero ou um (zero a 1)
- *
- 0..* - zero ou mais
- 1..* - um ou mais
- 1, 3..5 - um ou três a 5

Acesso a Multiplicidade



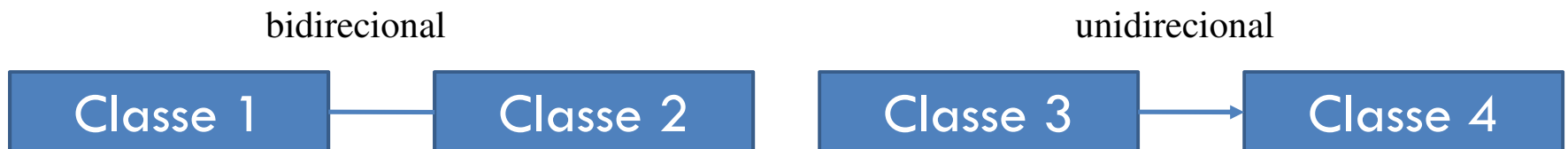
Associação reflexiva

- Pode- se associar uma classe com ela própria (em papéis diferentes)

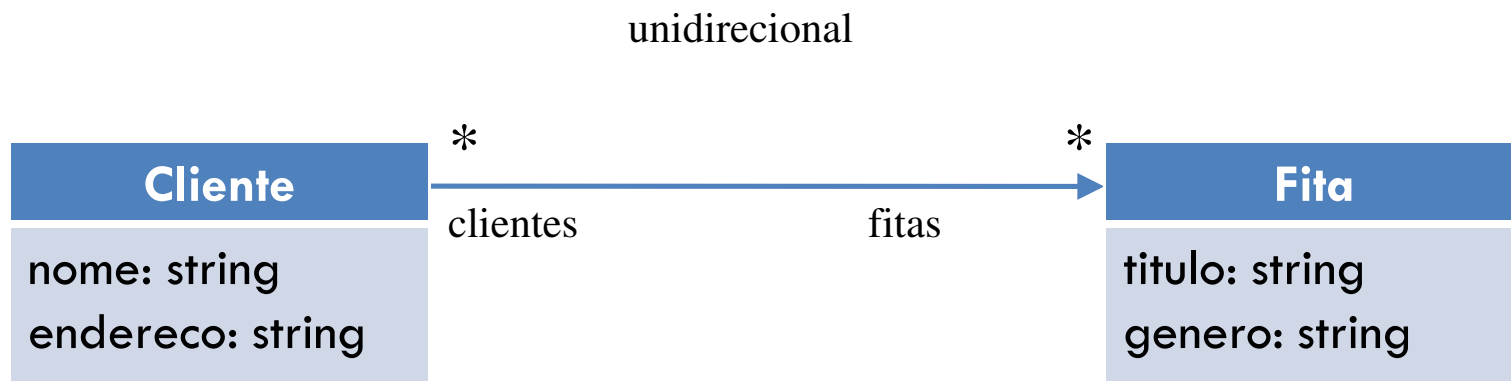
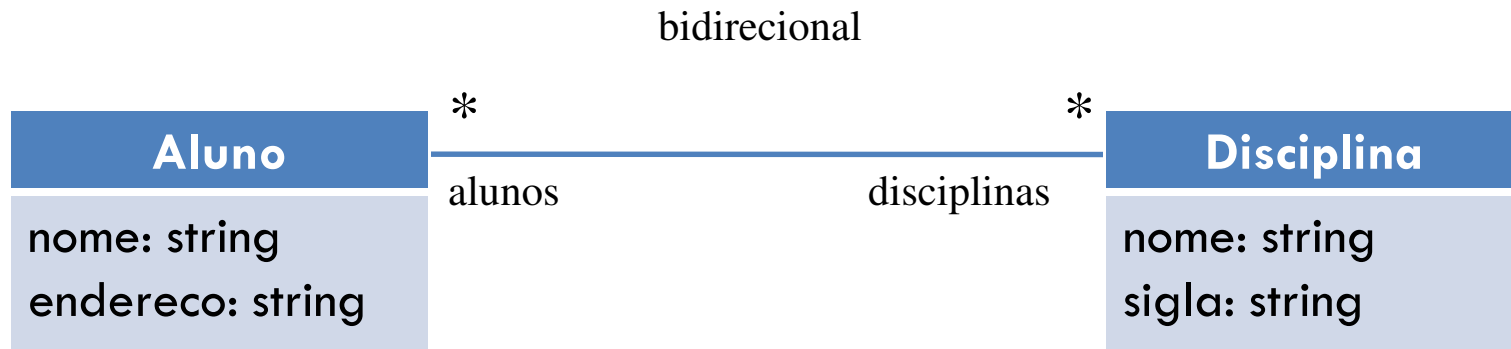


Associações bidirecionais / unidirecional

- As associações são classificadas quanto à navegabilidade em:
- Bidirecional → ambos objetos possuem referência.
- Unidirecional → deve ser armazenada em uma variável de instância na classe de origem da associação e seu tipo deve ser a classe de destino.



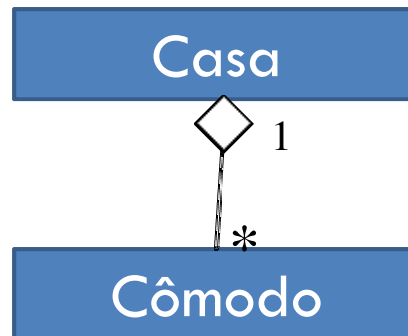
Exemplos



A classe Cliente recebe a Lista de Fitas e a classe Fita não recebe nada

Agregação

- Associação com o significado contém (é constituído por) / faz parte de (part of)
- Relação de inclusão nas instâncias das classes
- Hierarquias de objetos.
- Exemplo:

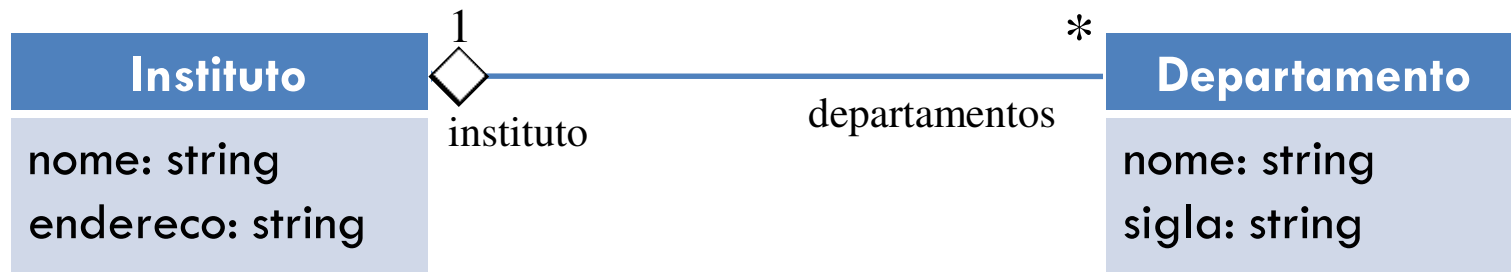


Uma casa **contém** vários cômodos
Um cômodo **faz parte** de uma casa

Quando a Casa for eliminada, os cômodos também devem ser eliminados através da classe Casa

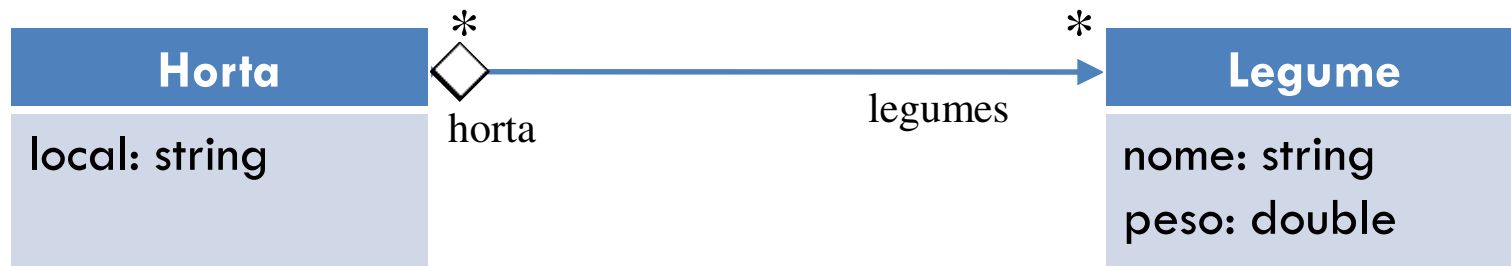
Exemplos

bidirecional



Quando Instituto for eliminado, os Departamentos também devem ser eliminados através da classe Instituto. O mesmo vale para a relação Horta e Legume abaixo

unidirecional



Composição



- Forma mais forte de agregação aplicável quando:
 - ▣ existe um forte grau de pertence das partes ao todo
 - ▣ cada parte só pode fazer parte de um todo
 - ▣ o topo e as partes têm tempo de vida coincidente, ou, pelo menos, as partes nascem e morrem dentro de um todo
 - ▣ a eliminação do todo propaga-se para as partes, em cascata
- Notação: losango cheio
- A Classe com a agregação cria, acessa e destrói a instância da outra classe.

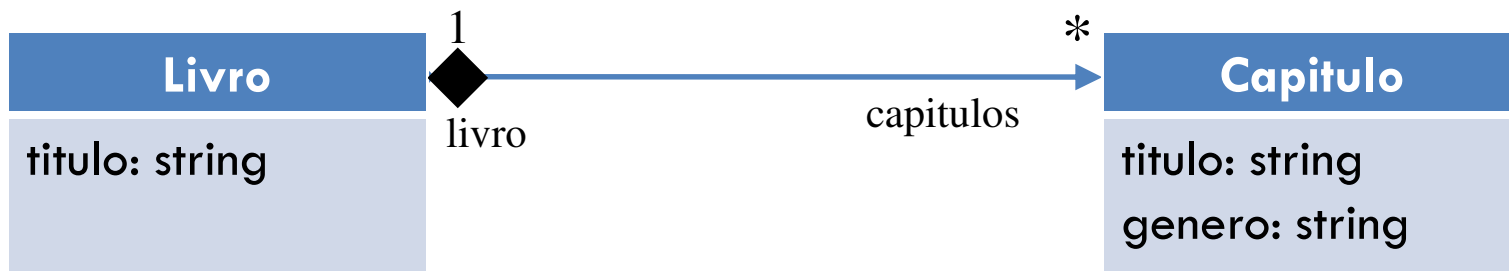
Exemplos

bidirecional



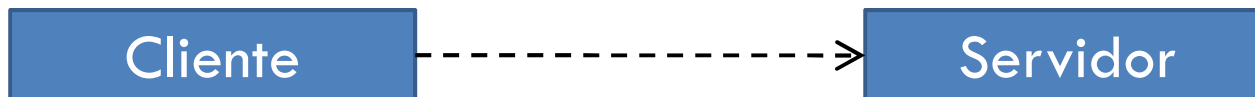
Classe Carro cria Motor no construtor, destrói Motor no destrutor e acessa Motor no programa principal via um Método na classe Carro

unidirecional

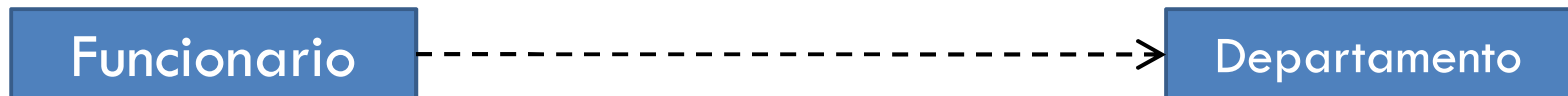


Dependência

- Relação de uso entre dois elementos (classes, componentes, etc.), em que uma mudança na especificação do elemento usado pode afetar o elemento utilizador
- Exemplo típico: classe1 que depende de outra classe2 porque usa operações ou definições da classe2
- Úteis para gestão de dependências
- Construtor alterado. Ex.: Cliente só pode existir se Servidor existir. Assim, Cliente recebe Servidor no construtor alterado



Exemplos



Para que um Estado exista é necessário que um País exista antes, desta forma, Estado deve receber no **construtor** a instância de País.

