Orientação a Objeto
(Diagrama de Classes UML)

Geraldo Xexéo
DCC/IM & PESC/COPPE
Universidade Federal do Rio de Janeiro
xexeo@cos.ufrj.br
http://www.cos.ufrj.br/~xexeo
http://www.xexeo.net/

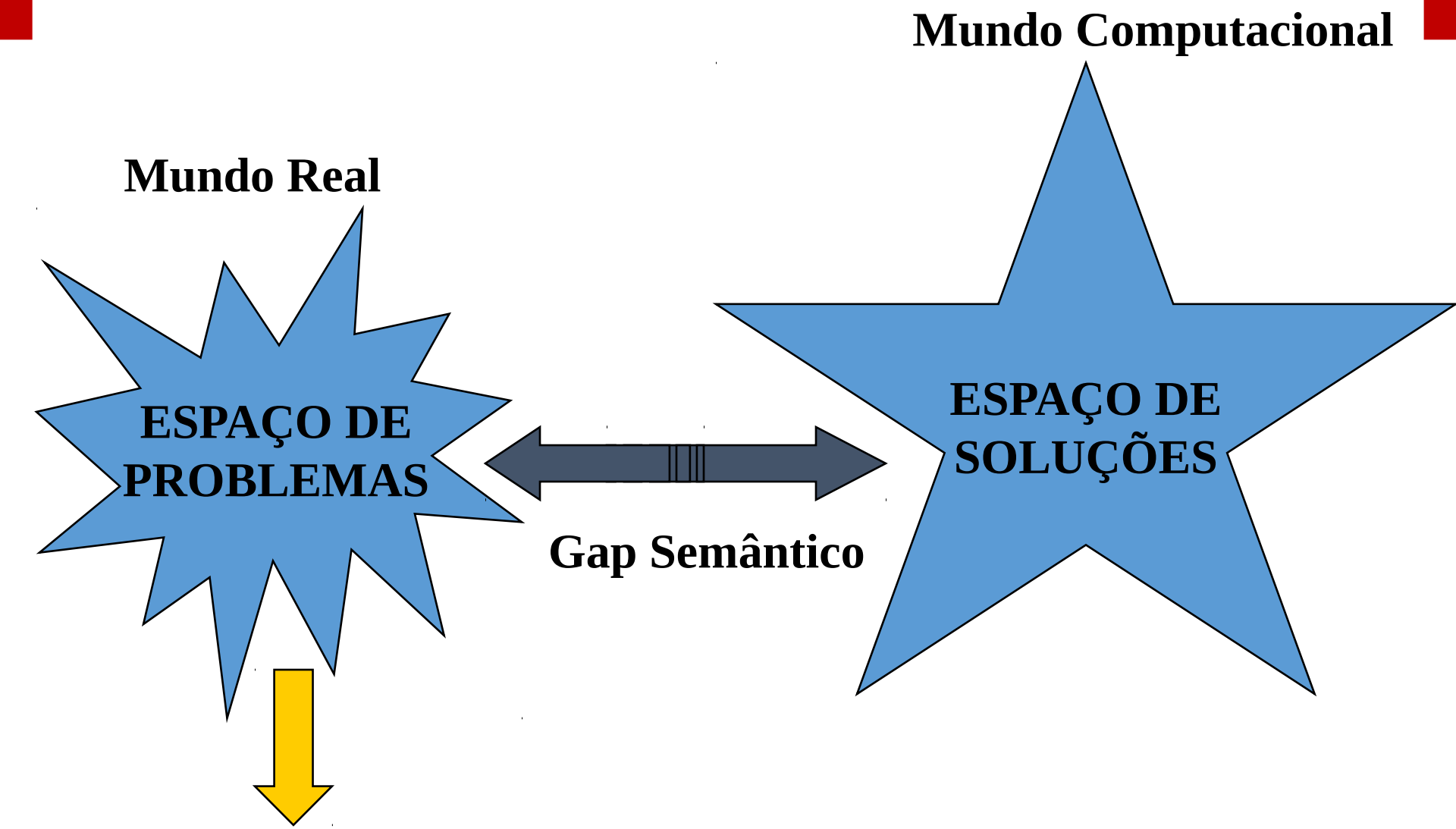
Introdução

Orientação a Objetos

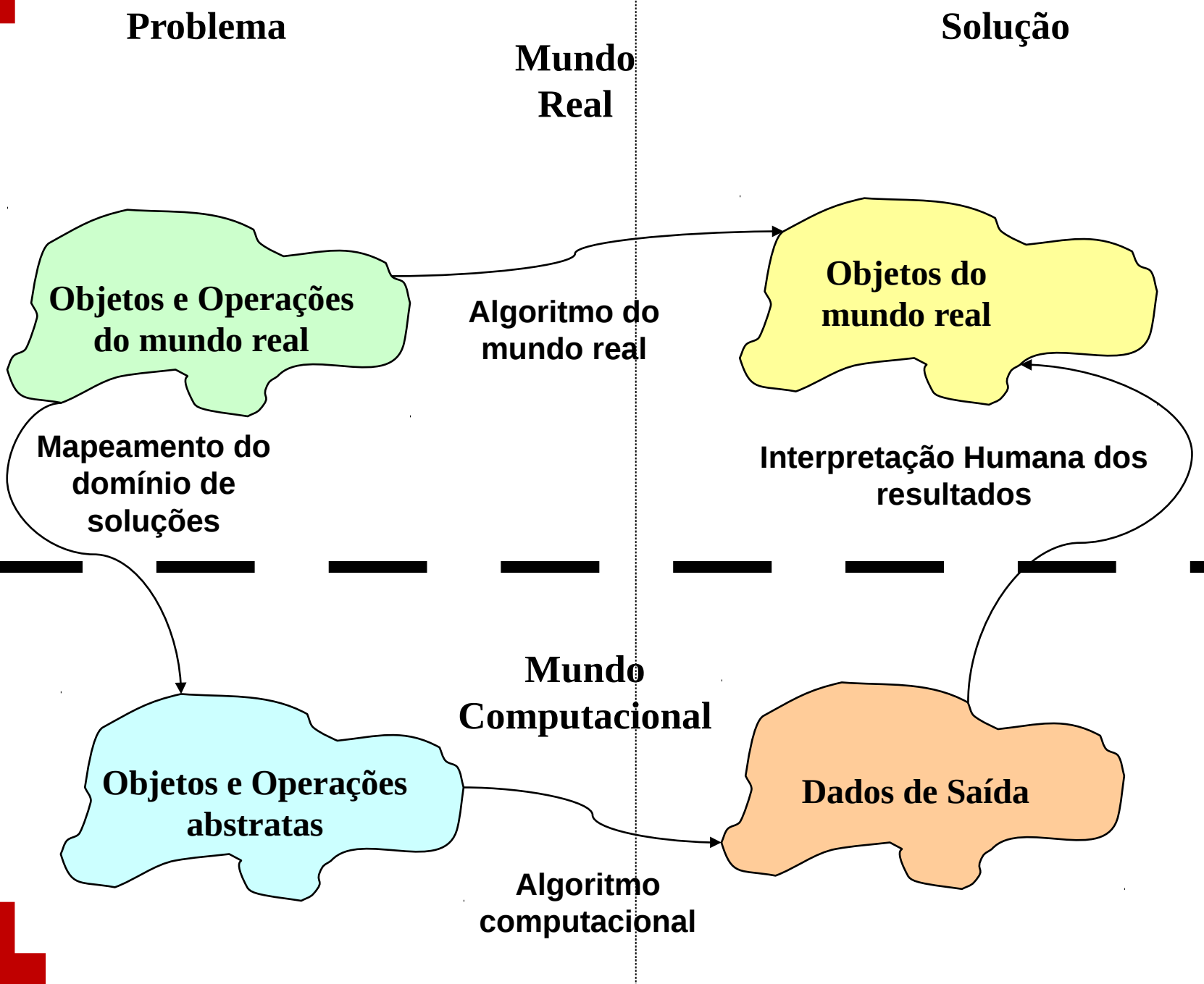
- Surgiu na tentativa de solucionar problemas existentes no desenvolvimento de Softwares Complexos e Confiáveis com baixo custo de desenvolvimento e manutenção

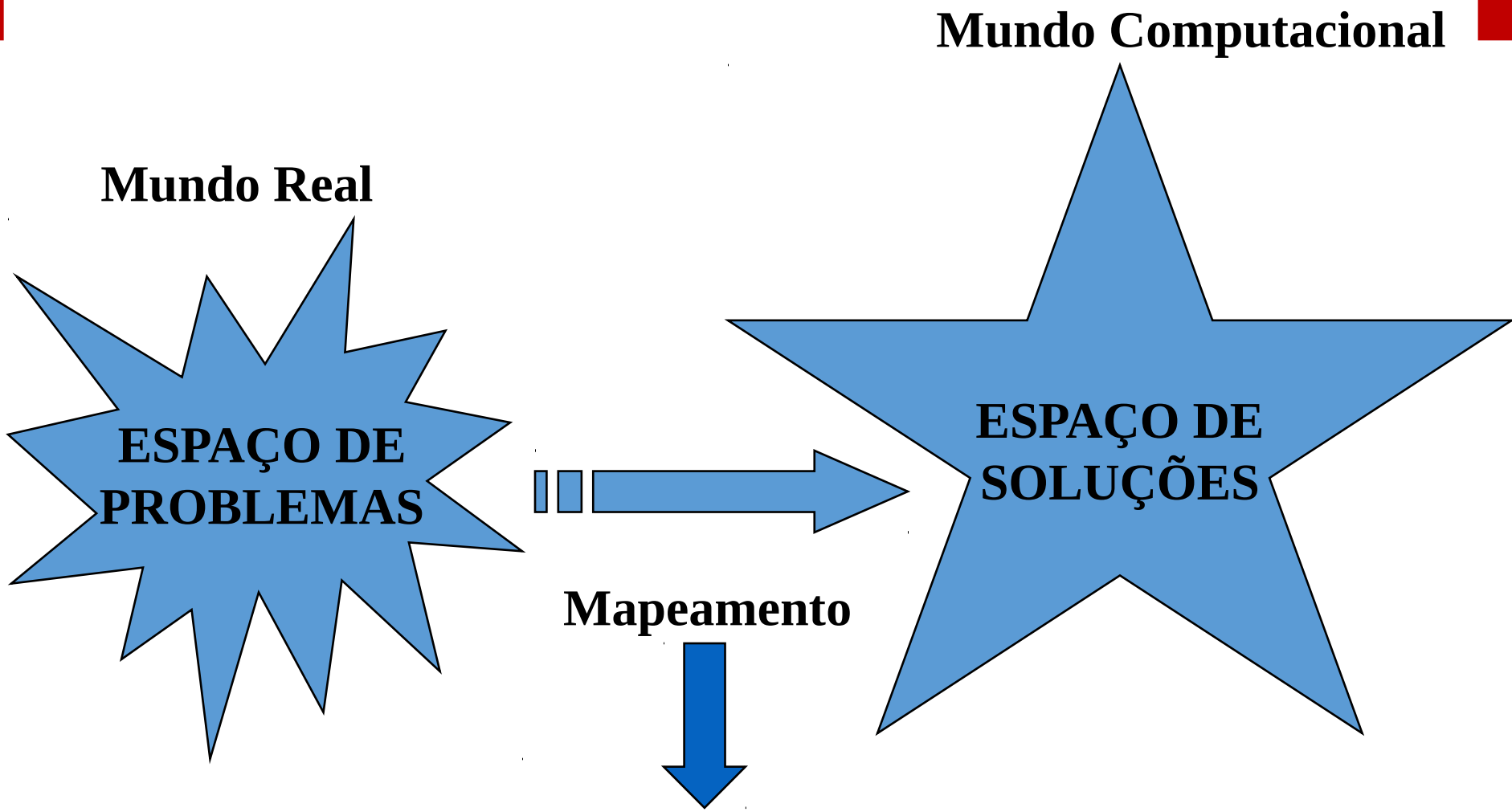
Orientação a Objetos

- Organizar o mundo real como uma coleção de objetos que incorporam:
  - Uma estrutura de dados
  - Um conjunto de operações que manipulam estes dados.
    - Representar esses objetos em um software é mais natural e permanente do que representar a sua funcionalidade (decomposição funcional), pois essa é mutável



Aspectos mais importantes do mundo real para fins de representação no computador

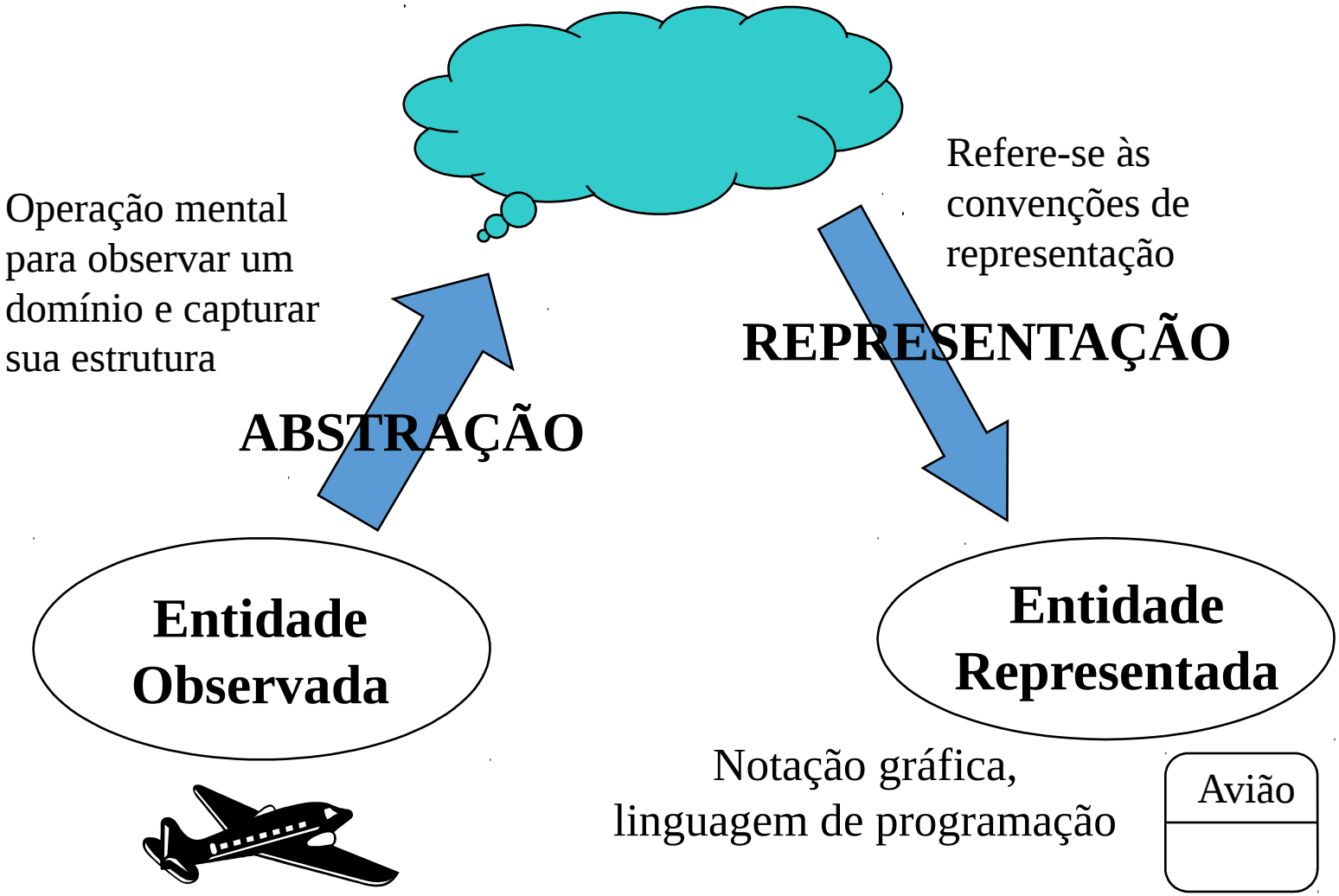

Problema
Solução
Mundo Real
Objetos e Operações do mundo real
Algoritmo do mundo real
Objetos do mundo real
Mapeamento do domínio de soluções
Interpretação Humana dos resultados
Mundo Computacional
Objetos e Operações abstratas
Algoritmo computacional
Dados de Saída



Processo de Identificação
de **Abstrações**

- Se essas abstrações tiverem uma expressão direta (ou próxima) do mundo computacional, a complexidade da solução será diminuida

Abstração e Representação



Modelos e Abstrações

- Todo **modelo** é uma abstração de algo que existe ou se imagina que possa existir no mundo real.
  - Uma representação
- **Abstração** é o processo mental de separar um ou mais elementos de uma totalidade complexa
  - de forma a facilitar a sua compreensão por meio de um modelo.
- Focalizar o essencial
- ignorar propriedades acidentais ou não interessantes ao modelo

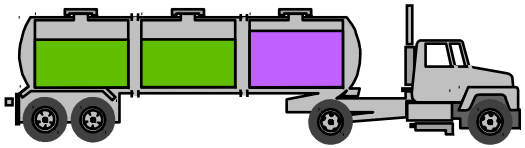
Mapas são Modelos

- Um mapa, por exemplo, é um modelo de uma cidade.
- Dependendo da informação que queremos, colocamos alguns símbolos e tiramos outros do mapa.
- Um mapa também não pode ser “perfeito”, tem que “abstrair” as informações que não são necessárias naquele instante, ou teria que ter o mesmo tamanho da cidade.

Mapas são modelos

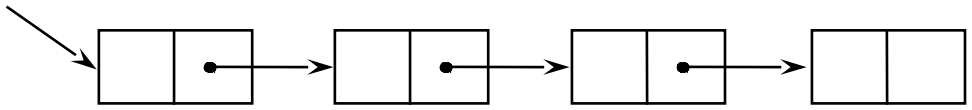
O que é um objeto?

- Um objeto é uma “coisa”
  - Entidade Física
  - Entidade Conceitual
  - Entidade de Software



Caminhão

Processo Químico



Linked List

O que é um objeto?

- Um conceito, abstração ou coisa com significado para a aplicação e que pode ser bem caracterizada separadamente
  - É claro o que é essa coisa e o que não é essa coisa
- Objetos possuem
  - Identidade
  - Comportamento
  - Estado

Objetos

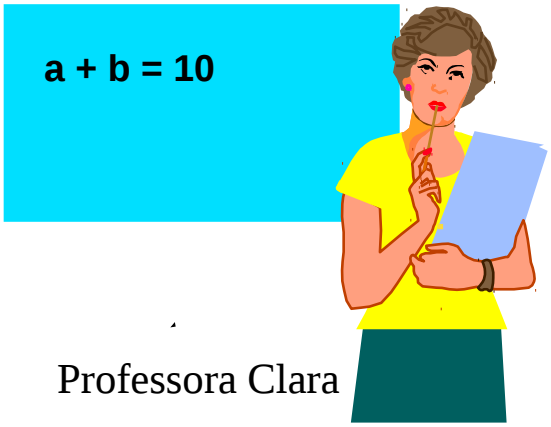
- É uma entidade lógica que contém dados e código para manipular esses dados.
- Exemplos:
  - Coisas tangíveis o livro “A Bíblia”
  - Incidente (evento/ocorrência) a Eleição Presidencial
  - Interação (transação/contrato) o débito de R$100,00 na conta “x” do dia 07/11/2002

Objetos

- Um objeto é uma pessoa, objeto, local, animal, acontecimento, organização ou outra idéia abstrata sobre a qual o sistema deve se lembrar alguma coisa.
- Cada instância de um determinado objeto tem características, um comportamento e uma identidade própria.

O Estado de um Objeto

- É uma das condições em que o objeto pode existir
- É representado pelos valores das suas propriedades em um determinado instante
- Normalmente muda com o tempo
  - Em função do comportamento



Nome: Clara Silva
Registro na UFRJ: 567138
Data de Contratação: 12/10/2000
Cargo: Prof. Adjunto

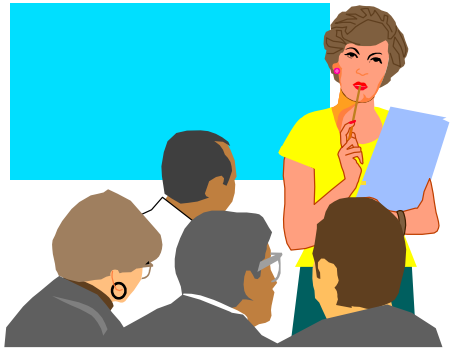
O Comportamento do Objeto

- Determina como o objeto reage
  - Suas mudanças de estado
  - Suas interações com outros objetos
- Define como ele reage a pedidos de outros objetos (e de si mesmo)
- Definido pelo conjunto de operações que ele executa

Registro

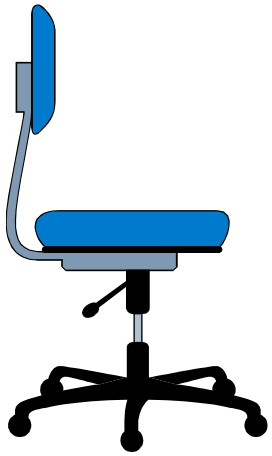
Indica Professor Para
Turma

Retorna Confirmação

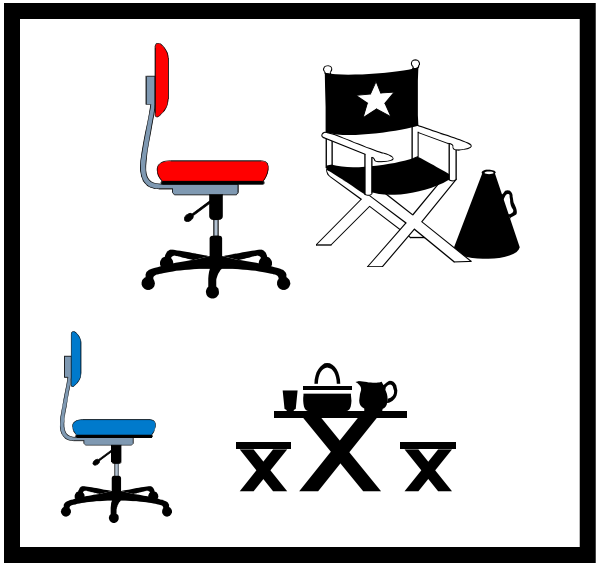


Características

- **Exemplo de objeto do mundo real**
  - **“a coisa em que você está sentado agora”**

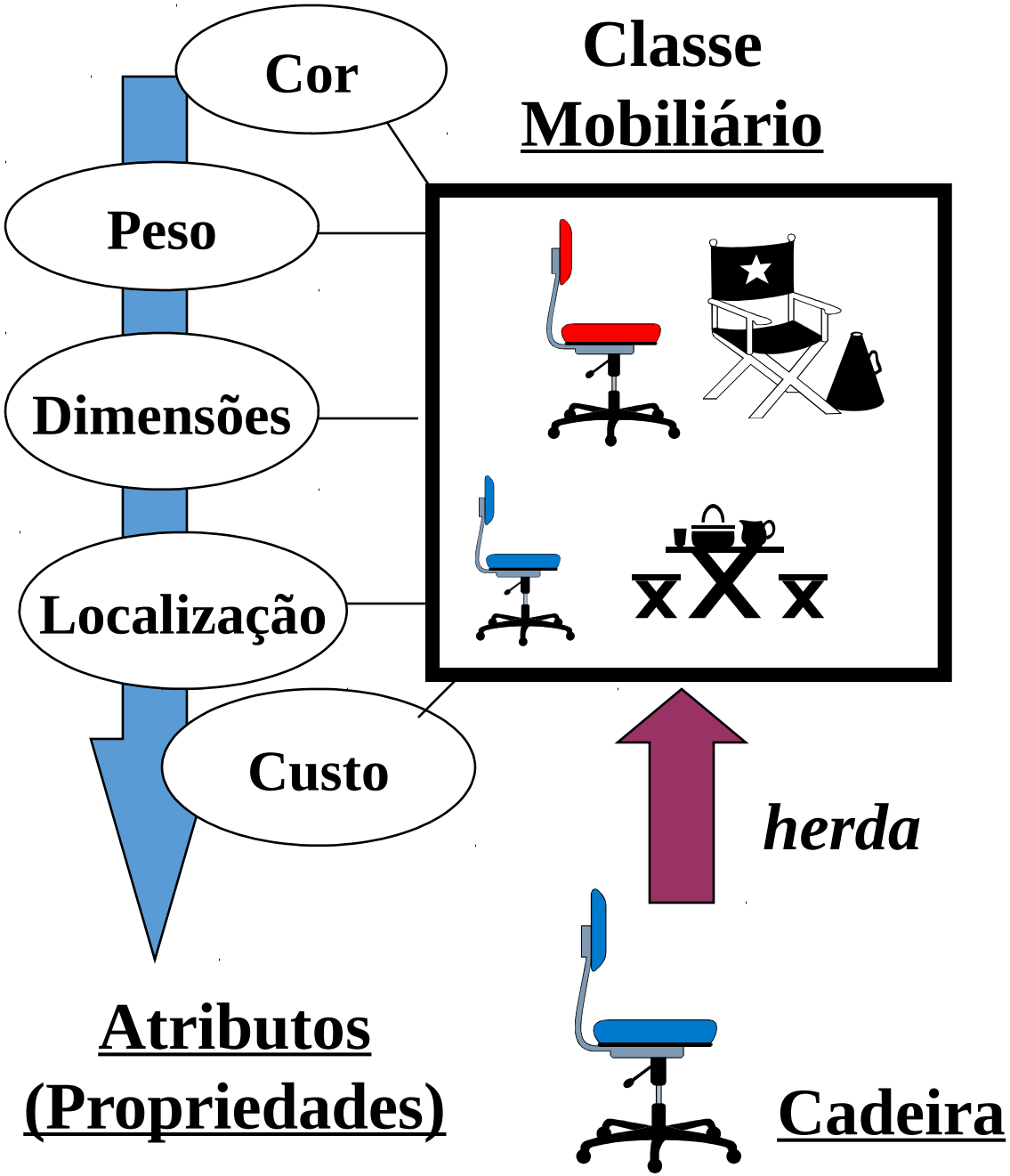


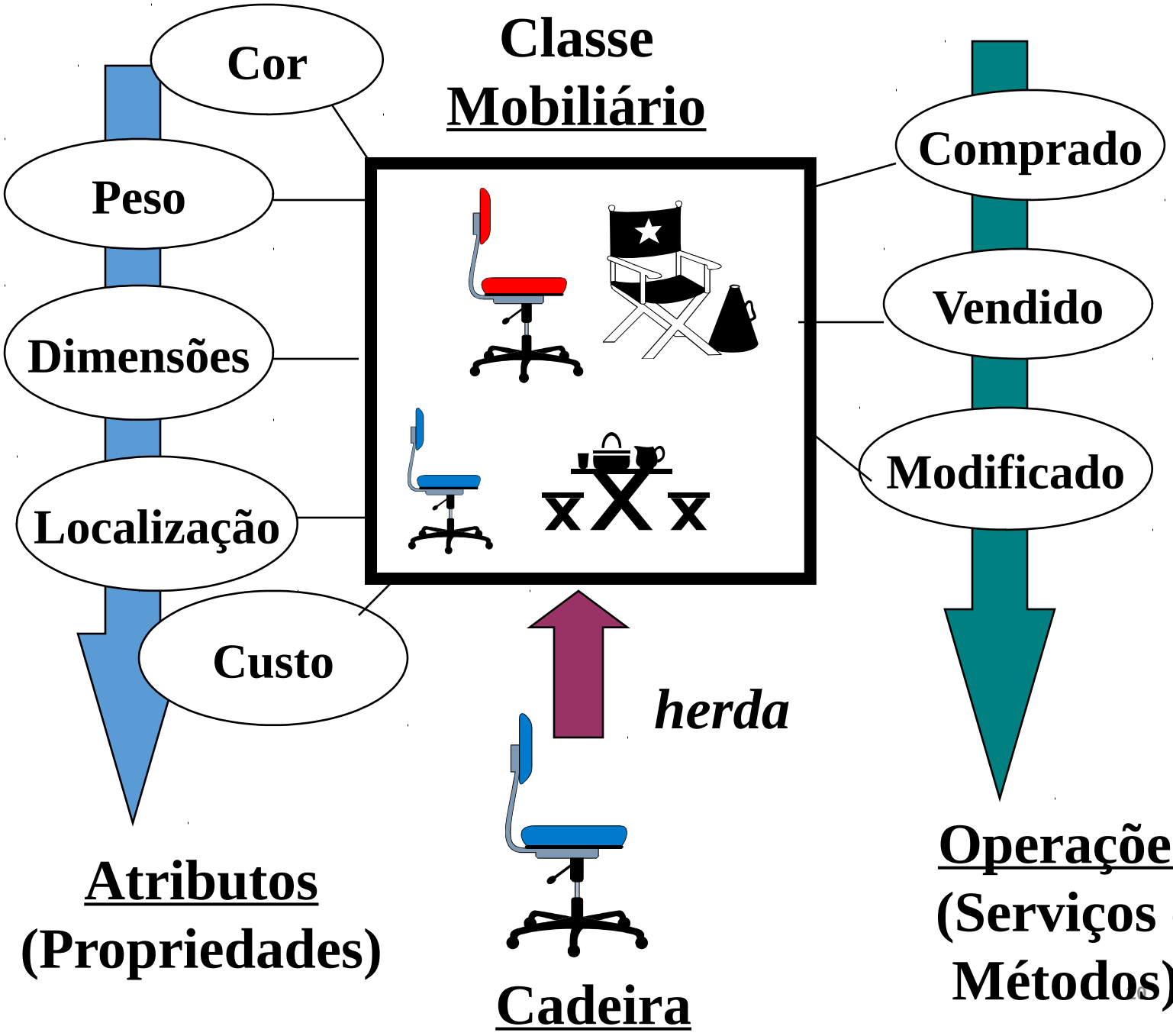
É membro - (instância) de uma classe maior de objetos



Cadeira

Mobiliário


Cor
Peso
Dimensões
Localização
Custo
Classe
Mobiliário
herda
Atributos
(Propriedades)
Cadeira


Cor
Peso
Dimensões
Localização
Custo
Classe
Mobiliário
Comprado
Vendido
Modificado
herda
Atributos
(Propriedades)
Cadeira
Operações
(Serviços e
Métodos)

Abstrações

Abstrações

- Classificação
- Composição
- Generalização
- Identificação

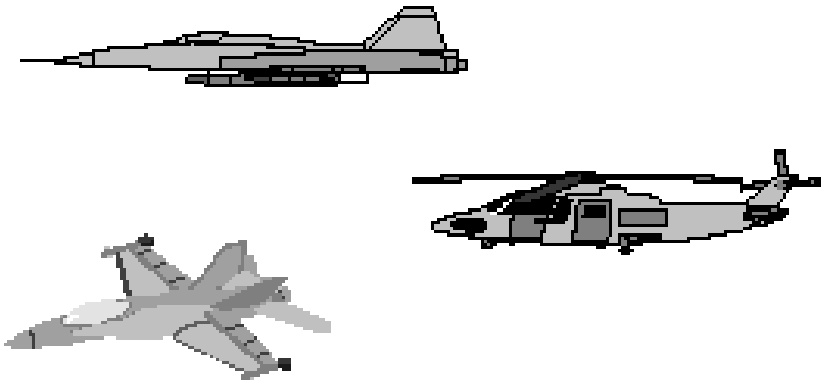
Classificação

- **“é um membro de”**
- No processo de **classificação** eliminamos a parte da individualidade do objeto e o consideramos como um exemplar de uma “classe padrão”.
- Quando fazemos isso, aceitamos que esse objeto, agora uma **instância** da classe, divide com todas as outras instâncias da classe um conjunto de características.

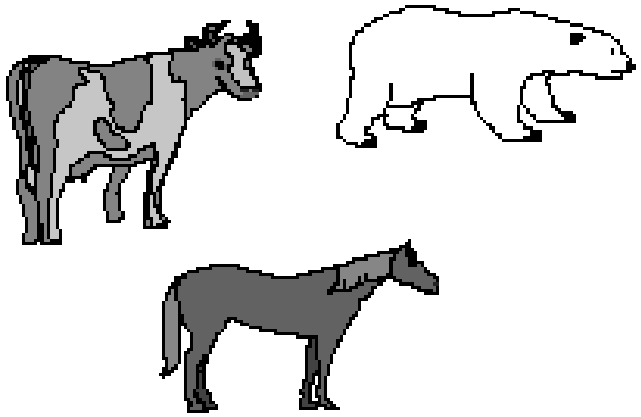
Classificação

- Na classificação o que estamos fazendo é imaginar uma idéia única que descreve, de forma abstrata, todos os objetos de uma classe.
- Abstrai as diferenças, declara as similaridades
- Ao eliminar a necessidade de tratar cada objeto de forma única, simplificamos o problema em questão.
- O processo reverso da classificação é a **instanciação**.
- O conjunto de todas as instâncias de uma classe é a **extensão** dessa classe.

Classificação

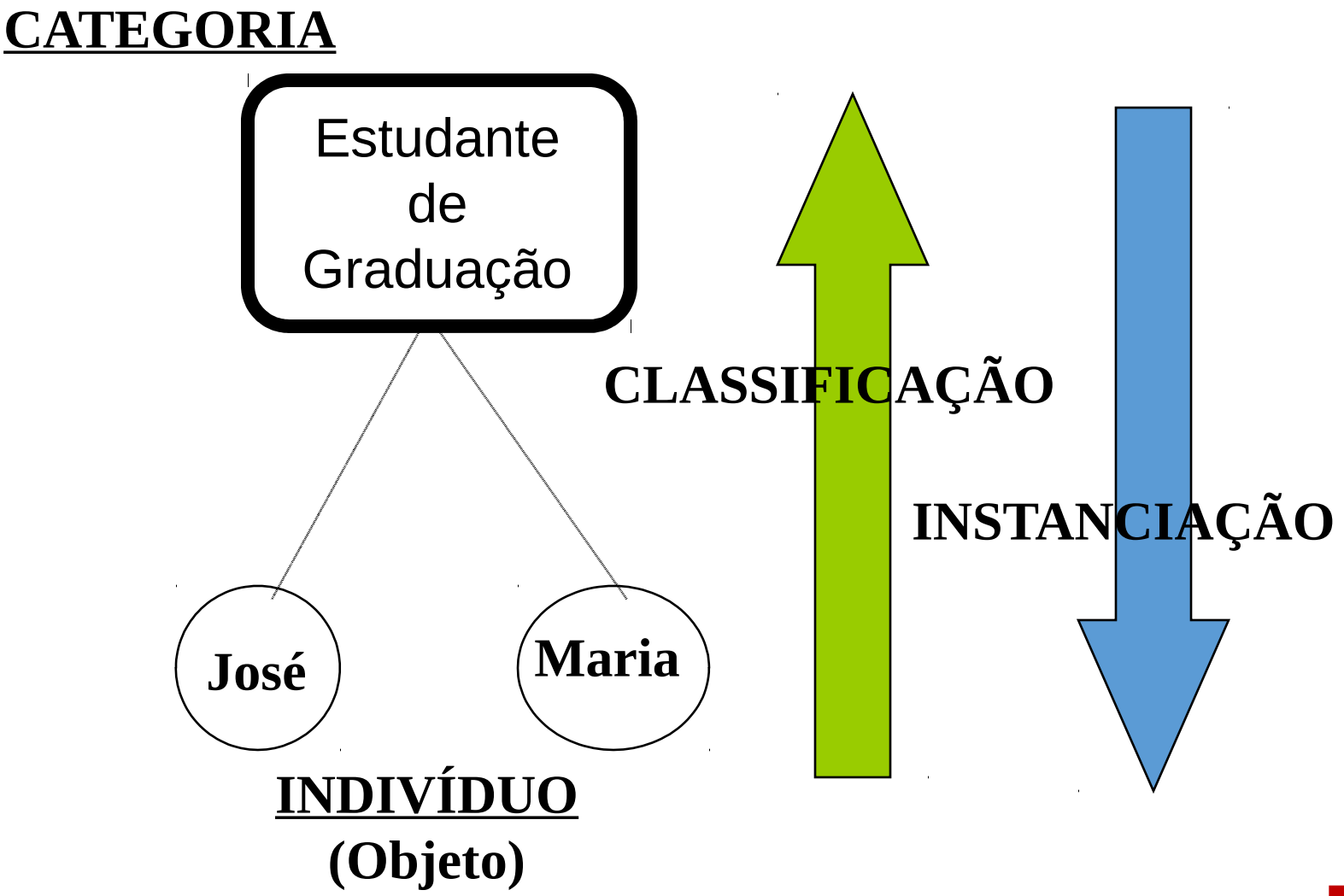


Aeronave

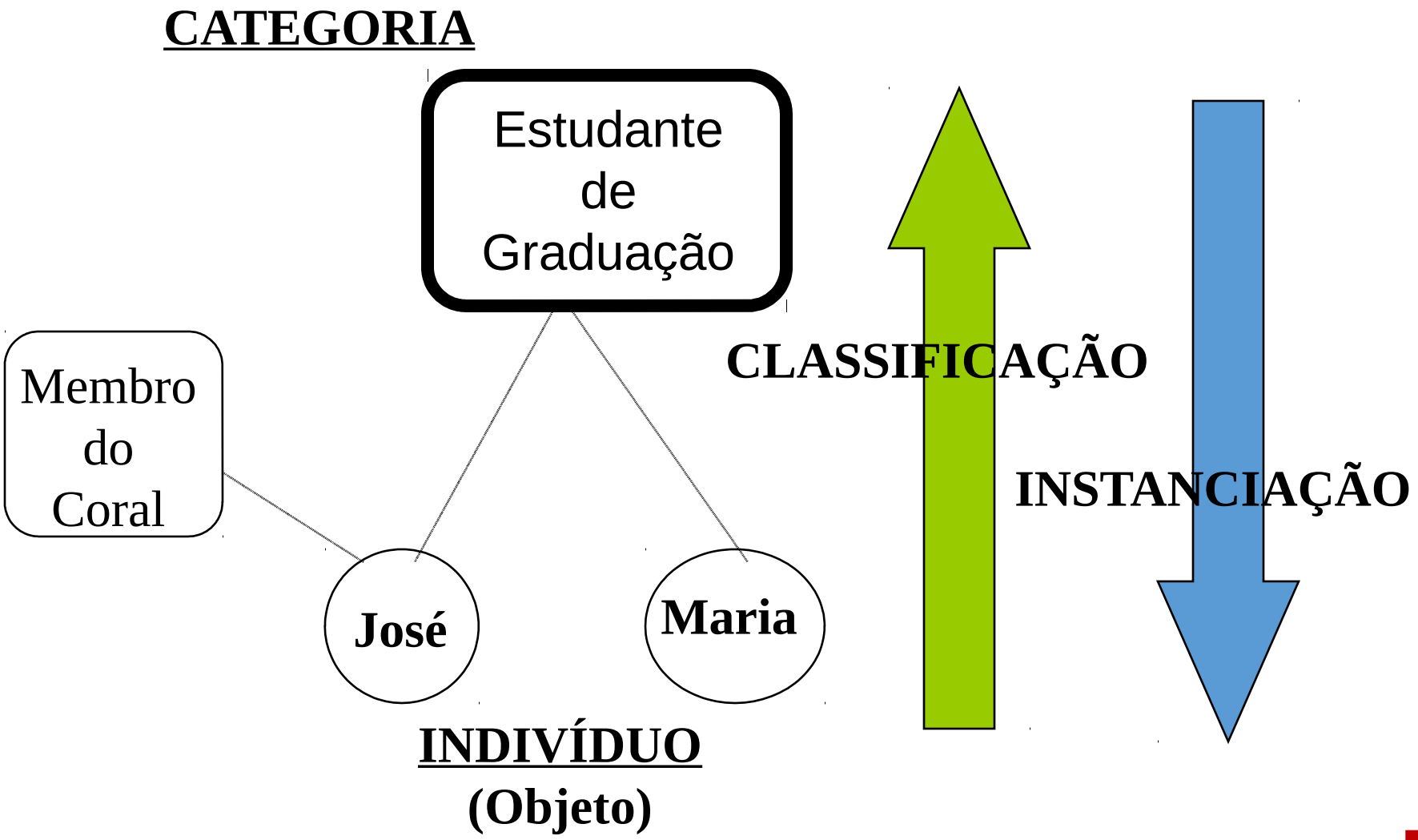


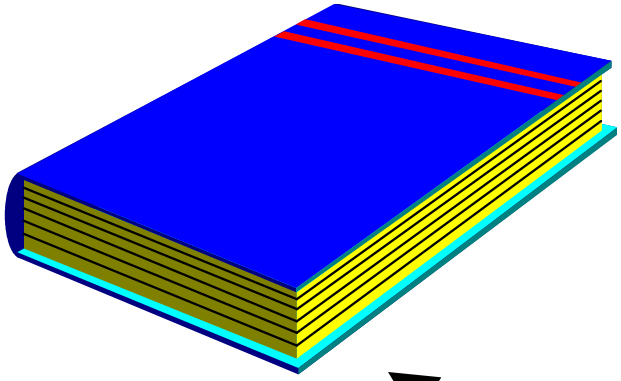
Mamífero

Classificação/Instanciação

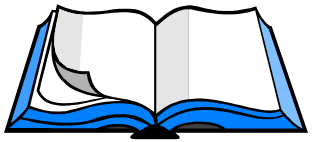


Classificação/Instanciação

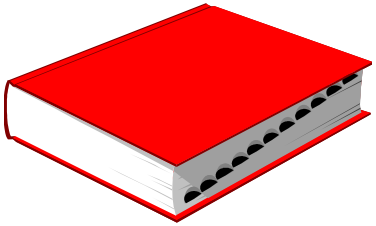




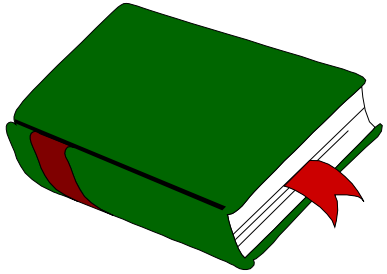
Livro



Bíblia



Relatório

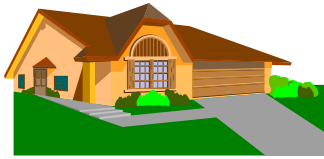


Dicionário

CASA

Portas
Salas
Cozinha
Quartos
Localização
Telhado

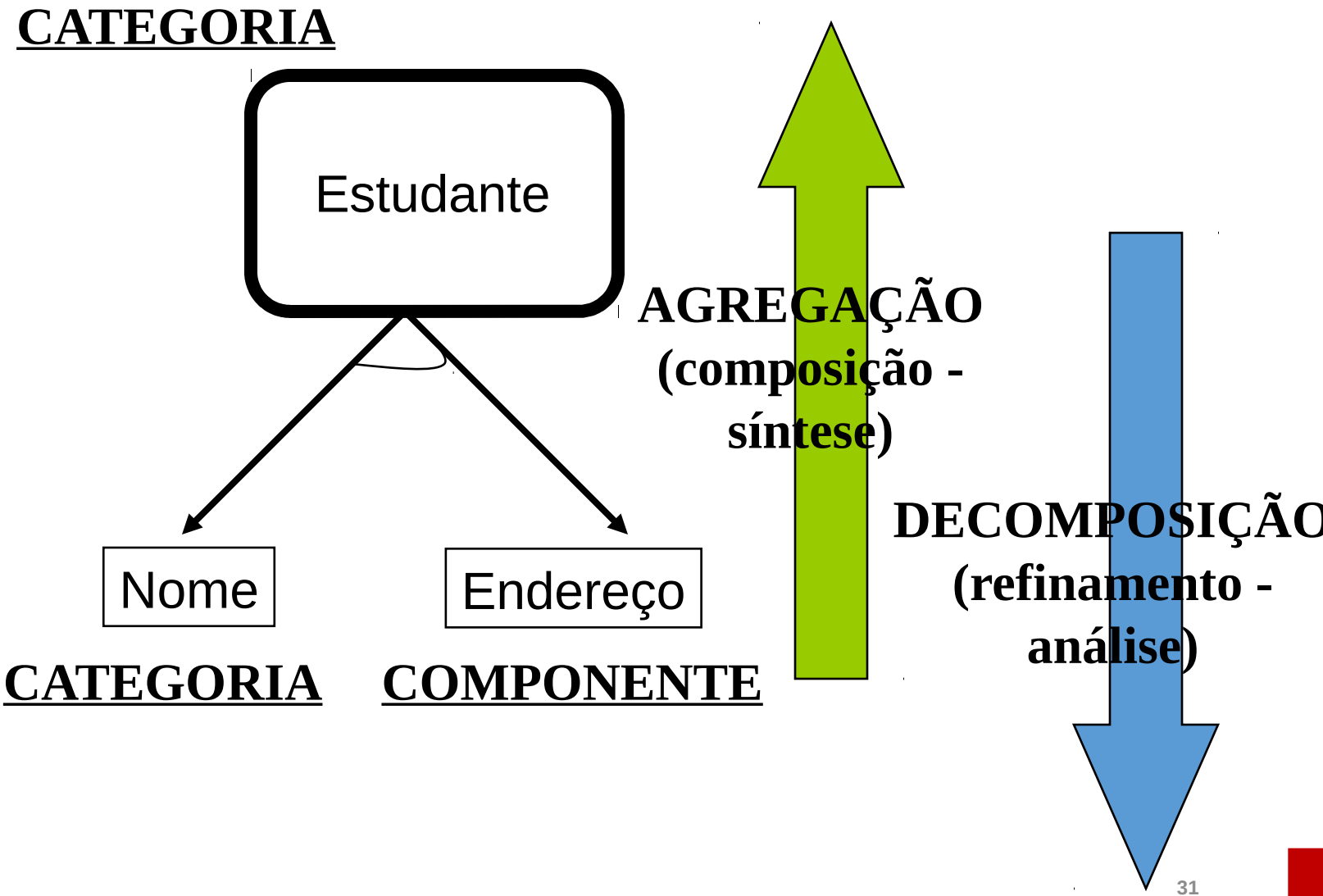
Reformar
Limpar
Pintar
Mobiliar



Composição

- **Composição, Agregação**
- Na **composição** entendemos um objeto complexo formado de um conjunto de outros objetos como um só objeto.
- Um objeto é descrito por suas partes
- O processo reverso da composição é a **decomposição**.

Agregação/Decomposição



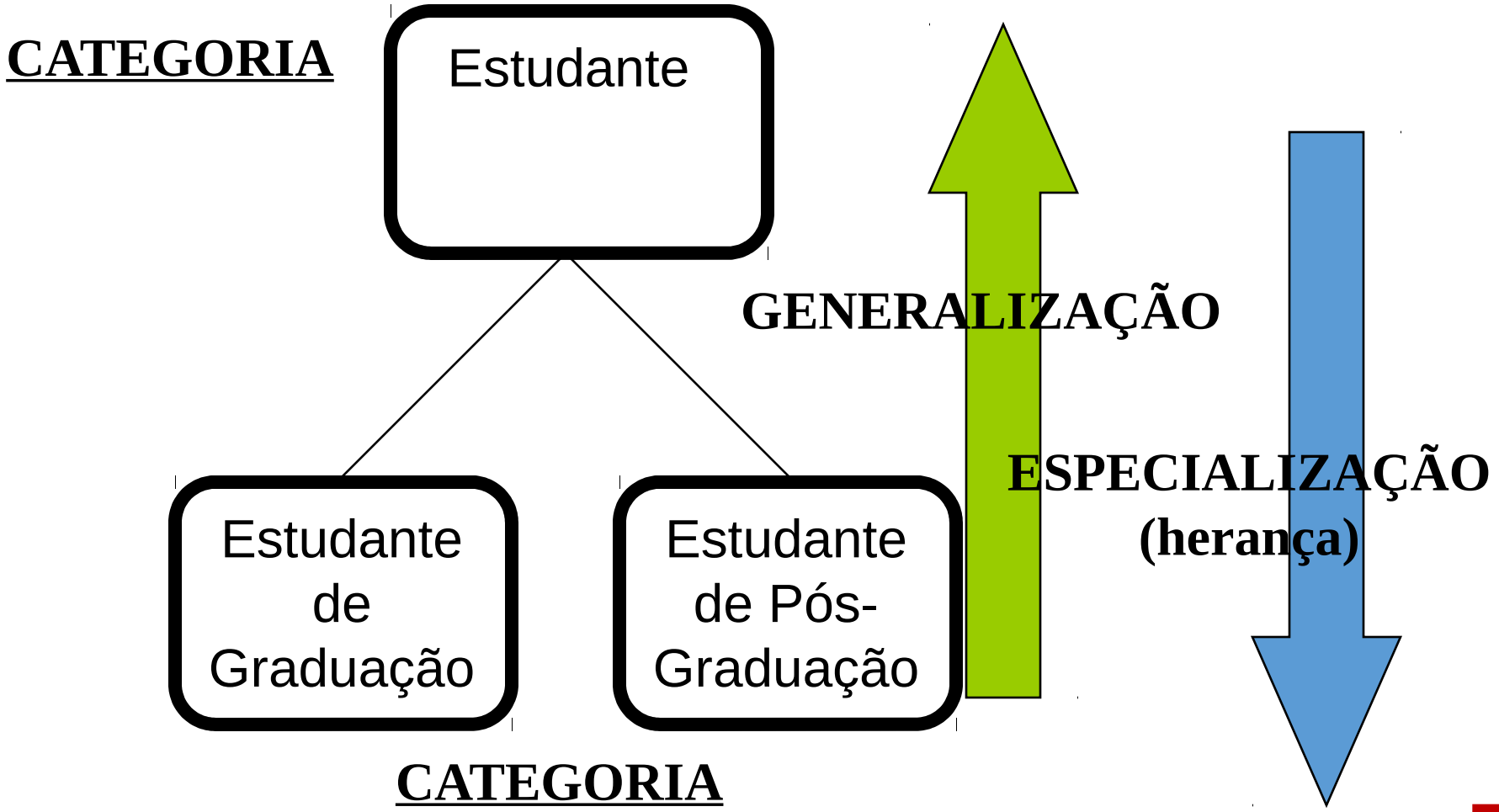
Generalização

- Com a **generalização** nós somos capazes de entender como uma classe pode ser descrita por outra classe, mais geral.
- Com a generalização podemos compreender uma relação muito comum entre classes, que é a que permite que qualquer objeto de uma classe possa ser visto, de uma forma mais geral, como um objeto de outra classe.
- Utilizando judiciosamente a generalização podemos simplificar a forma de tratar objetos de classes similares.
- O processo reverso da generalização é a **especialização**.

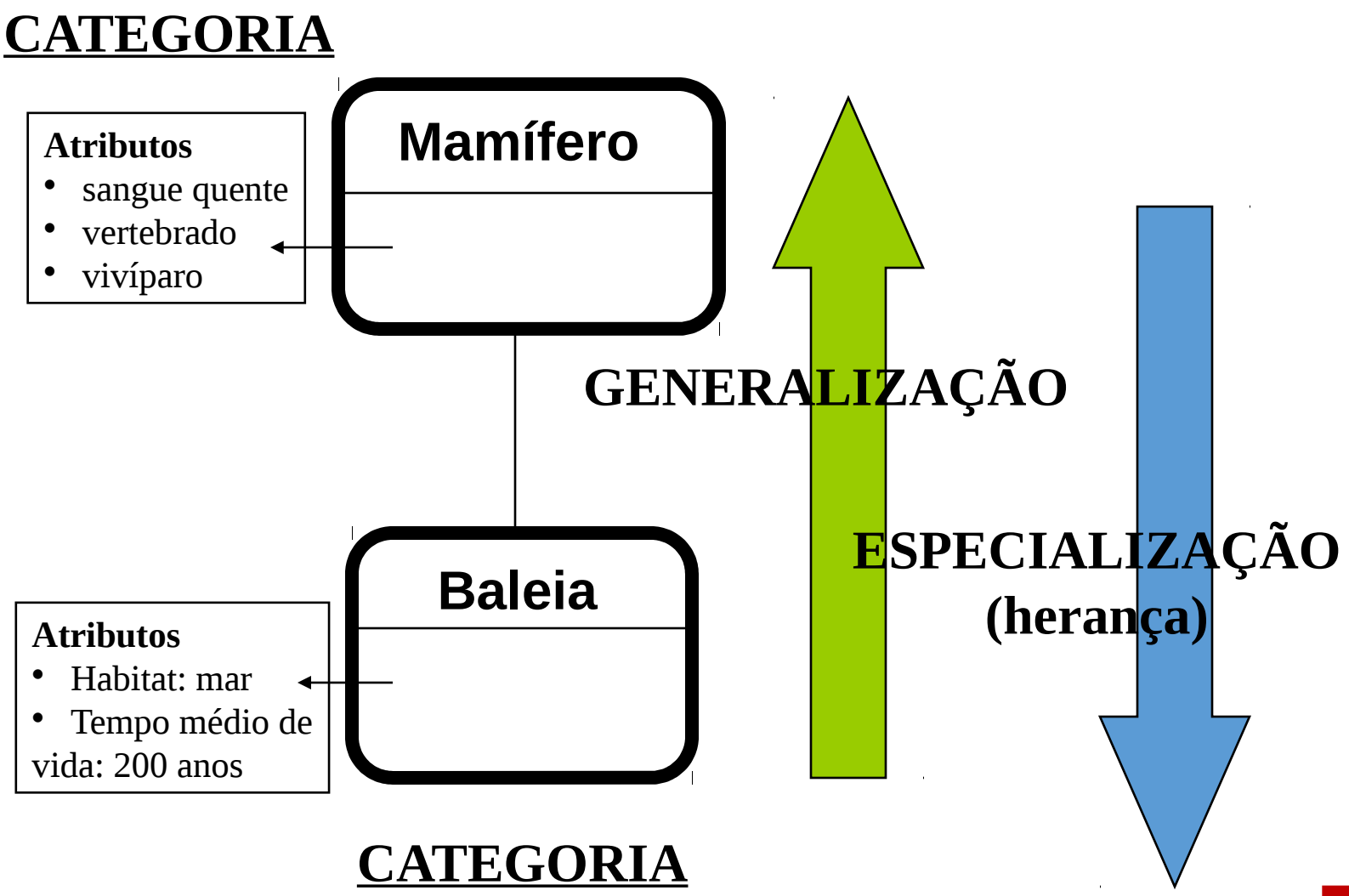
Generalização x Classificação

- É importante ver a diferença entre a classificação e a generalização: a primeira trata da relação entre objeto e classes, enquanto a segunda trata da relação entre classes.

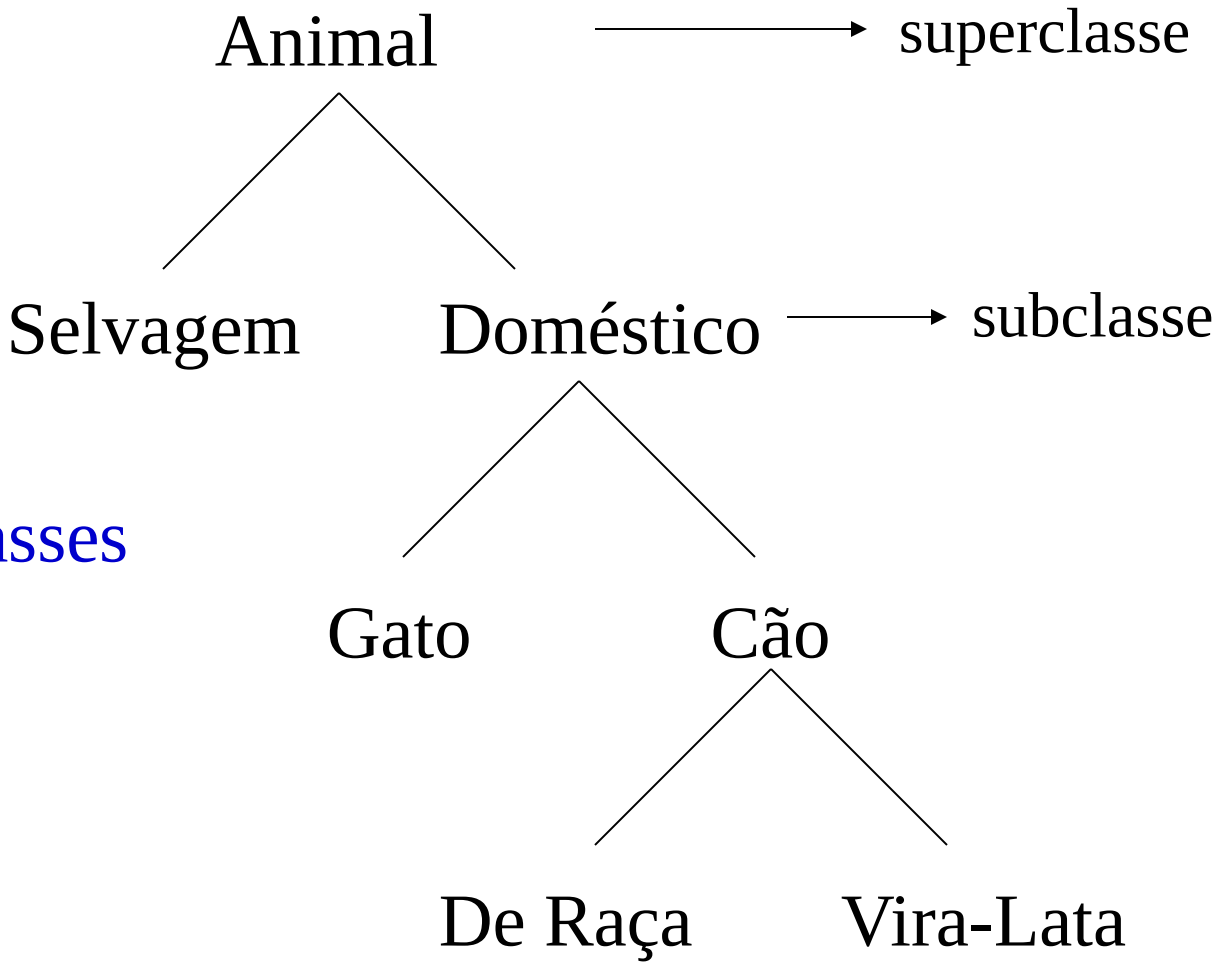
Generalização/ Especialização



Generalização/Especialização

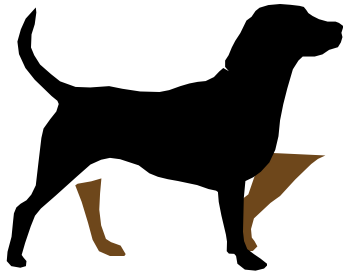


Hierarquia de Classes



Objeto Rex

instância da classe Vira-Lata



Rex

Generalização

Reformar
Limpar
Pintar
Mobiliar

Portas
Salas
Cozinha

Quartos
Localização
Telhado

(Superclasse)

CASA

PRAIA

FAVELA

MANSÃO

(Subclasse)

Limpar Piscina
Contratar Criadagem
Piscina
Quadras

Especialização

Identificação

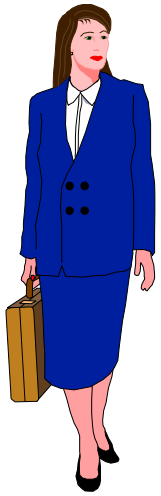
- Com a **identificação** nós somos capazes de entender como caracterizar unicamente um objeto.
  - Um nome identifica uma pessoa
- Ao identificar unicamente um objeto podemos separá-lo de outro objeto semelhante e atribuir a entidades específicas atributos e características que só pertencem a ela, e não pertencem a outros elementos daquela classe.

Identificação

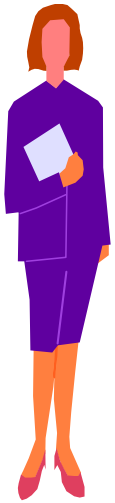
- A identificação permite a que duas instâncias sejam reconhecidas como distintas
  - ou como representações de um mesmo objeto (normalmente devendo ser reunidas em uma).

Identidade do Objeto

- Cada objeto tem uma identidade única, mesmo que seu estado seja idêntico ao de outro objeto



Professora “Clara Silva”



Professora “Clara Silva”

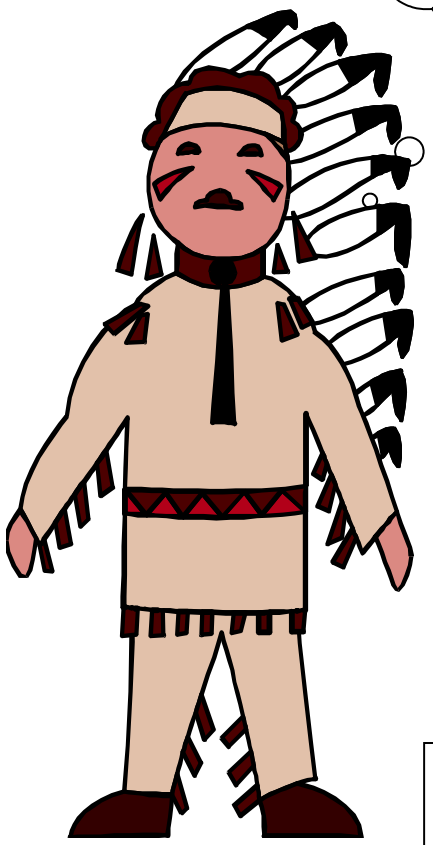
Conceitos Adicionais

Mensagem

- É o mecanismo através do qual os objetos se comunicam, invocando as operações desejadas
- Especificação de uma operação do objeto
- É composta por
  - Seletor:
    - nome simbólico que descreve o tipo da operação
    - descreve O QUE o objeto que envia quer que seja invocado, não como deveria ser invocado
    - o objeto recebedor da mensagem contém a descrição de COMO a operação deveria ser executada
  - Parâmetros:
    - argumentos que uma mensagem pode conter que faz parte da operação e requer uma ordem única

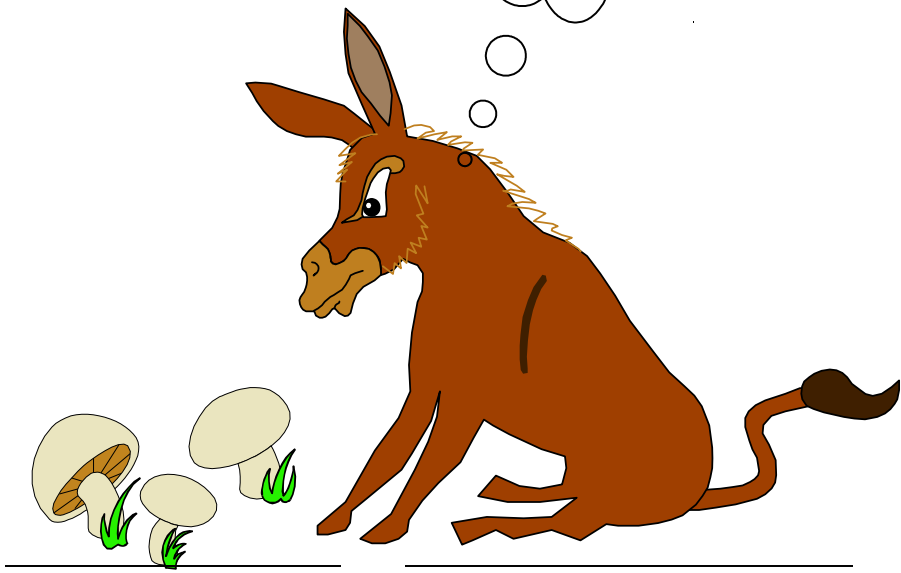
Pessoa

Coma isto logo!



Animal

Comer



Objeto: Animal
Mensagem: COMA ISTO LOGO
Seletor: COMA
Parâmetros: ISTO LOGO

Objeto que envia
EMISSOR

Determina o que deve ser manipulado e aguarda o retorno do objeto que foi ativado

Objeto que recebe
RECEPTOR

Determina como manipular

Mensagem

- Um objeto (Emissor) envia uma mensagem a outro (Receptor) que executará o serviço
- Métodos são invocados por Mensagens
- Exemplo
  - A chamada de um procedimento/função em LP é uma aproximação inicial de uma mensagem, como em

    P(10,20), onde
  - P é o seletor e os valores 10 e 20 são os parâmetros
  - Diferença:
    - a ação da mensagem a ser ativada depende essencialmente do objeto que receber a mensagem

Métodos

- Origem: LP Smalltalk
- Quando um objeto é mapeado dentro do domínio do software, os processos que podem mudar a sua estrutura de dados são denominados Operações ou Métodos
- Métodos são invocados por Mensagens
- Cada objeto possui seu próprio conjunto de métodos
- Definições:
  - São procedimentos definidos e declarados que atuam sobre um objeto ou sobre uma classe de objetos
  - Descrição de uma seqüência de ações a serem executadas por um objeto
  - Através dos métodos que especifica-se a um objeto COMO FAZER alguma coisa
  - São intrínsecos aos objetos e não podem ser separados

Interface (Protocolo)

- Objetos se comunicam através de Mensagens, que invocam os Métodos desejados
- Parte Privada do Objeto (Visão Interna)
  - Métodos
  - Atributos
- Parte Compartilhada do Objeto ou Interface
  - corresponde à parte externa do objeto, isto é, àquela que é vista por outros objetos com a finalidade de invocar os métodos que o objeto realiza
  - agrupa um conjunto de mensagens que o objeto pode responder
  - as mensagens especificam quais operações sobre o objeto podem ser realizadas, mas não como a operação será executada

Ocultamento de Informação (Information Hiding)

- Característica que visa esconder detalhes de implementação
- É alcançado em OO, visto que o objeto, quando implementado, possui uma parte privada (atributos e métodos) e uma parte compartilhada (interface)
- Detalhes de implementação ficam escondidos na parte privada
- Programadores podem introduzir mudanças na implementação de um método sem afetar o comportamento externo desse método

Encapsulamento de Dados

- Objetos encapsulam suas estruturas de dados (atributos)
- Os atributos de um objeto só podem ser manipulados pelos próprios métodos do objeto
- Restringe a visibilidade do objeto mas facilita o reuso
- Os DADOS e os MÉTODOS são empacotados sob um nome e podem ser reusados como uma especificação ou componente de programa

Polimorfismo

- Ao receber uma mensagem para efetuar uma Operação, é o objeto quem determina como a operação deve ser efetuada, pois ele tem o comportamento próprio
- Como a responsabilidade é do Receptor e não do Emissor, pode acontecer que uma mesma mensagem ative métodos diferentes, dependendo da classe de objeto para onde é enviada a mensagem
- Permite a criação de várias classes com interfaces idênticas, porém objetos e implementações diferentes
- Capacidade de uma mensagem ser executada de acordo com as características do objeto que está recebendo o pedido de execução do serviço

Polimorfismo

- Por exemplo, a operação “<“ pode ser definida em várias classes de números: inteiro (método de comparação inteira), string .
- É comum em LPOOs

Detalhando...

Classes

- Todo domínio de aplicação possui muitos objetos
- Uma classe é uma descrição de um grupo de objetos com propriedades similares
  - Atributos
  - Comportamentos
  - Relacionamentos com outros objetos
- Uma classe representa uma semântica comum

Classe

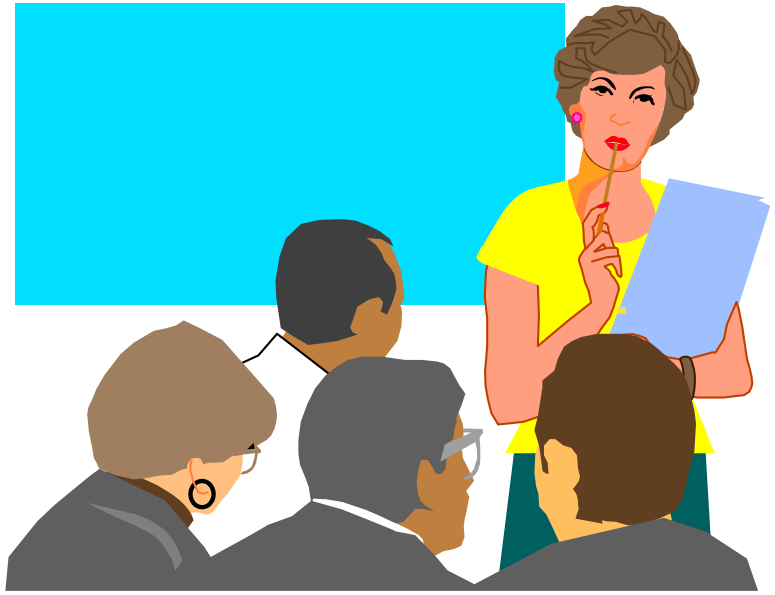
- Enfatiza as características relevantes
  - Ao problema
  - A solução
- Suprime outras características

Exemplo

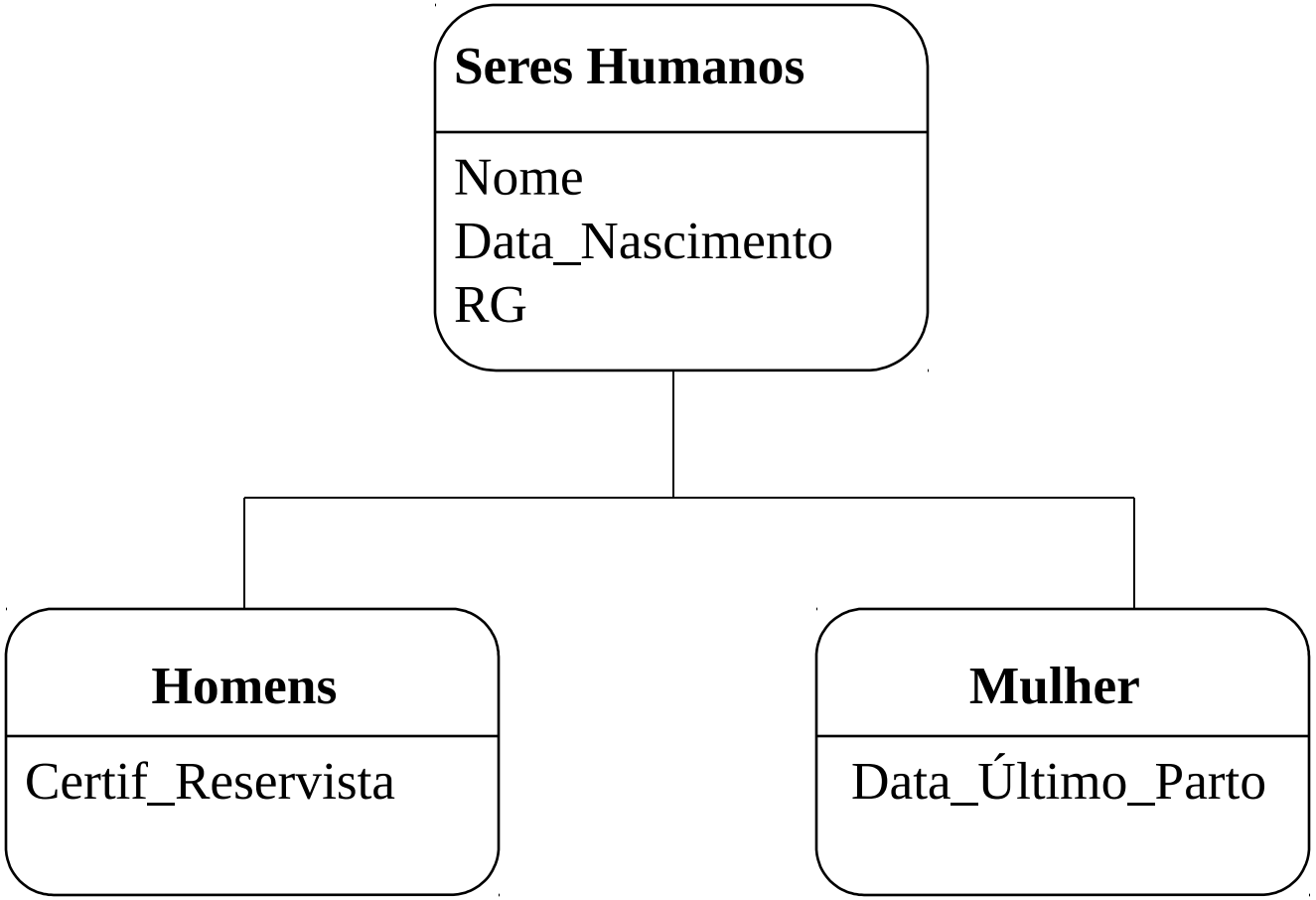
Atributos
Nome
Sala
Tamanho da Turma
Dias
Créditos
Hora de Início
Hora de Fim

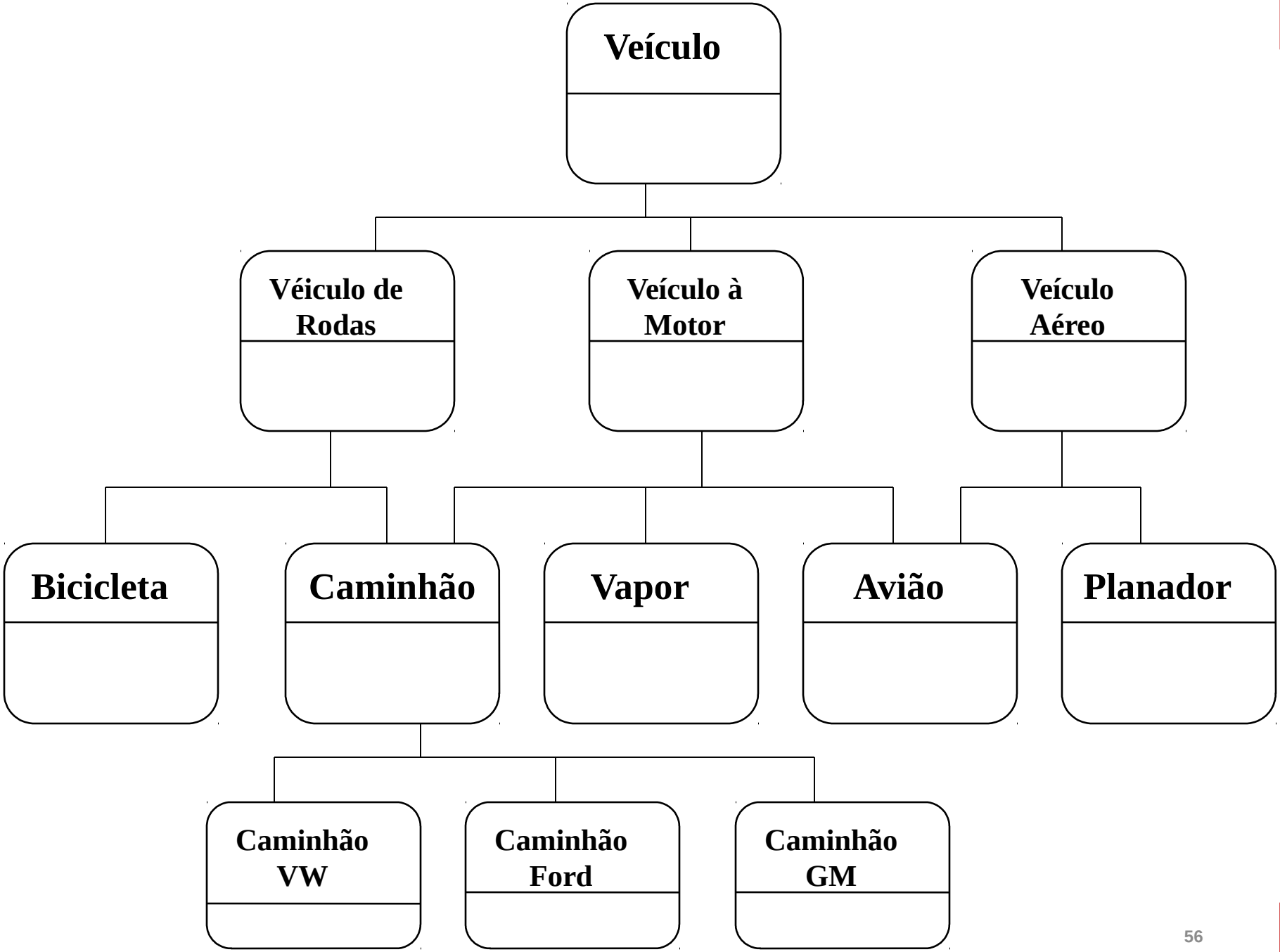
Relacionamentos
Alunos
Pré-Requisitos

Classe
Curso



Comportamento
Matricular aluno
Cancelar Matricula de Aluno
Listar Alunos
Determinar se a Turma está cheia


Seres Humanos
Nome
Data_Nascimento
RG
Homens
Certif_Reservista
Mulher
Data_Último_Parto


Veículo
Véiculo de Rodas
Veículo à Motor
Veículo Aéreo
Bicicleta
Caminhão
Vapor
Avião
Planador
Caminhão VW
Caminhão Ford
Caminhão GM

Nomes das Classes

- Substantivo Singular
- Dificuldade de escolher nome pode significar que a abstração está mal definida
- Nomes devem ser tirados do vocabulário do domínio

Pequenas Regras de Estilo

- Substantivo singular
- Inicia com letra maíuscula
- Não usa “underscore”
  - Usa “composição” de nomes iniciando com letra maiúscula
- Sugestão: usar apenas o alfabeto inglês
- Possibilidade: usar apenas inglês
- Exemplo
  - Estudante
  - Professor
  - AlunoPosGraduacao

Defina a Semântica

- Depois de dar nome a classe
- Descreva concisamente, com foco na abstração e não na implementação
- Use para construir o glossário
- Descreva apenas em relação ao domínio da aplicação

O QUE e não COMO

Exemplo

- Nome: Estudante
  - Definição: Informações básicas sobre estudantes registrados para assistir aula na Universidade
- Nome: Curso
  - Definição: Uma matéria oferecida pela Universidade

Ao longo do tempo as descrições de classe devem ser refinadas para representar a compreensão cada vez melhor que desenvolvemos sobre a classe

UML para Classes

Representação

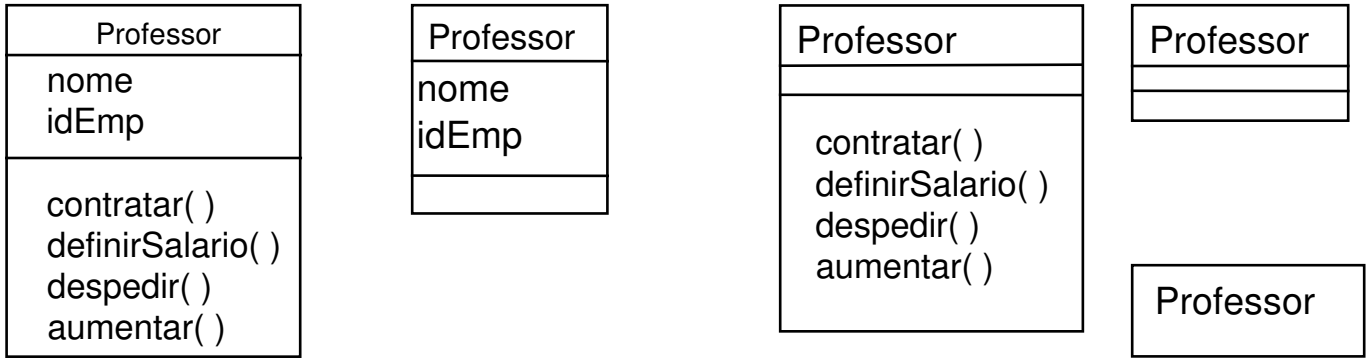
- Um retângulo

Professor

Professora Clara Silva

Representação

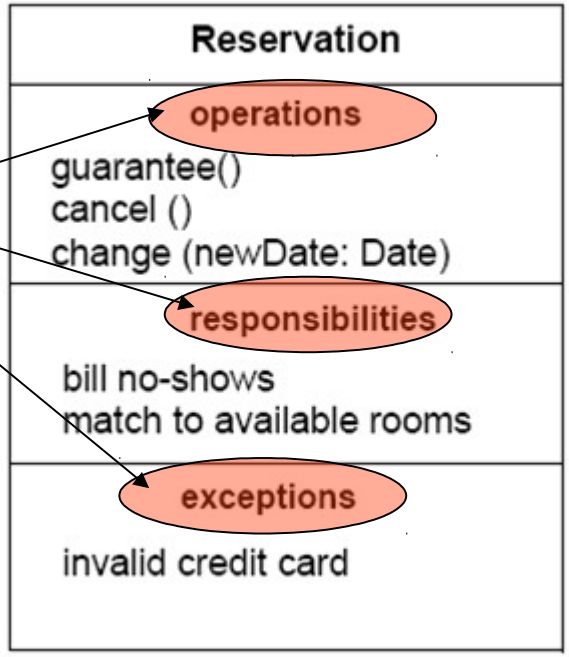
- 3 Partes
  - Nome
  - Estrutura (atributos)
  - Comportamento (operações)
- Apenas o nome é obrigatório



Representação

- Partes Extras
- Criadas pelo modelador

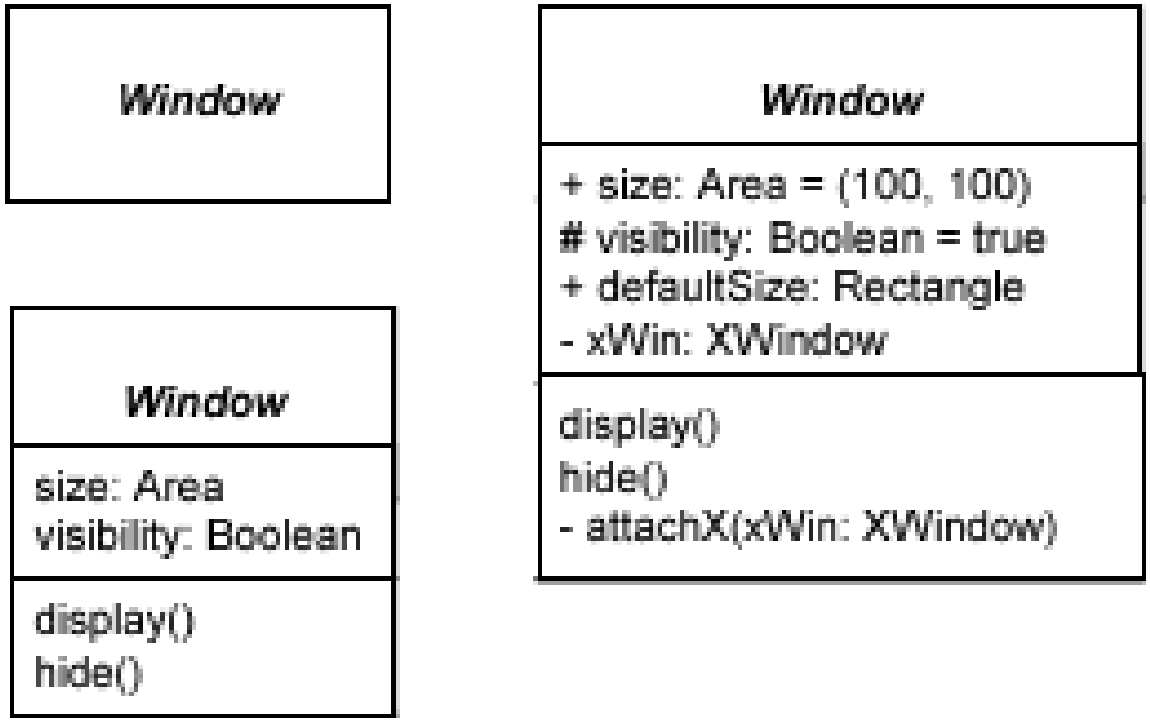
Partes podem ter nomes



Estilo UML

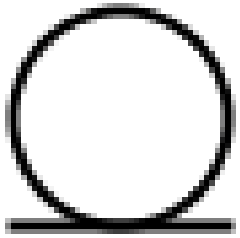
- Nome da classe centralizado em negrito
  - Classes abstratas em itálico
- Primeira letra do nome da classe é maiúscula
- Atributos e operações alinhados a esquerda, caracteres normais
- Atributos e operações começam com minúscula
- Mostre os atributos e operações completos apenas quando o contexto tornar necessário

Do padrão 2.0

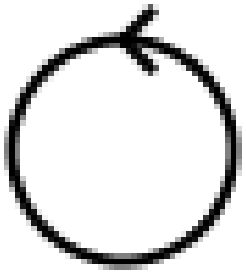

Window
Window
size: Area
visibility: Boolean
display()
hide()
Window
+ size: Area = (100, 100)
# visibility: Boolean = true
+ defaultSize: Rectangle
- xWin: XWindow
display()
hide()
- attachX(xWin: XWindow)

Estereótipos Básicos

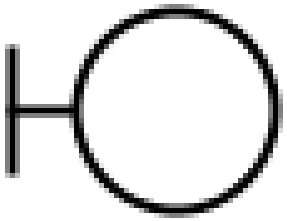
- Entidade



- Controle

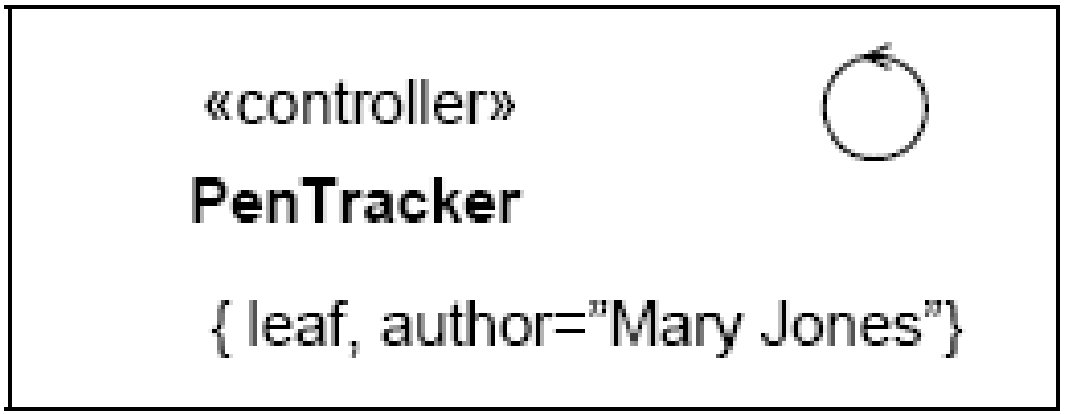


- Fronteira



Ainda as Classes em UML

- Estereótipo
- Lista de propriedades
  - Meta-modelagem
  - Coisas que não podem ser ditas em UML



Encontrando Objetos

Encontrando Objetos

- Objetos são encontrados nos substantivos dos casos de uso e cenários
- Nomes podem ser
  - Objetos
  - Atributos
  - Descrições de estados
  - Atores
  - Nenhuma das opções acima

Filtrando Nomes

- Sublinhe os nomes nos textos
- Atenção
  - Muitos termos podem se referir ao mesmo objeto
    - Sinônimos
  - Um termo pode ser referir a muitos objetos
    - Coleção de objetos
    - Homônimos
  - A linguagem natural é ambígua
- Esta abordagem pode identificar muitos objetos que não são importantes

Cuidado

- De acordo com quem escreveu o texto, um substantivo pode aparecer como verbo ou um verbo como substantivo

Filtragem

- Identifique cada substantivo
  - Nada, Ator, Estados, Candidatos a Objetos
  - Analise os candidatos a objetos
  - Classifique-os
  - Identifique-os

Aplicação Escritório Eletrônico

- Documento de Requisitos
- “Pretende-se automatizar toda a entrada, processamento, saída e arquivamento de documentos em um escritório, onde um executivo e sua secretária desempenham os papéis típicos de suas funções. Por documento, entende-se qualquer tipo de informação que flui para/do escritório e entre o executivo e a secretária: cartas, anotações de telefonemas, documentos técnicos, etc.”

Identificação de Objetos

- O processo de análise para identificação dos objetos e classes DIFÍCIL
- Basicamente a mesma tarefa realizada na análise e no projeto de sistemas (Análise Estruturada)
- Os domínios de aplicação que naturalmente refletem entidades concretas do mundo real se beneficiam da abordagem OO

Identificação de Objetos

- Objetos podem ser:
  - Entidades Externas que produzem ou consomem informações (outros sistemas, pessoas)
  - Coisas que fazem parte do domínio (relatórios, displays, cartas)
  - Eventos que ocorrem no contexto do sistema (transferência de propriedade)
  - Papéis desempenhados por pessoas (gerente, engenheiro, vendedor)
  - Unidades organizacionais (grupo, equipe)
  - Lugares que estabelecem o contexto do problema (piso de fábrica, área de descarga)

Efetuar uma "análise gramatical"

- Isolar nomes (e locuções nominais), verbos (e locuções verbais)
- Os nomes e os verbos que são sinônimos ou que não tem nenhuma relação com o processo de modelagem são omitidos
- Todos os nomes são classes ou atributos (itens de dados)
- Todos os verbos são operações (métodos) - relacionamento entre classes

Operações

O que é uma operação

- Uma classe deve atender a responsabilidades
  - Definem o comportamento de objetos da classe
- Os operadores executam as responsabilidades
- Métodos que executam as mensagens

Operações dependem do Domínio

- Liste as operações relevantes ao problema
  - Um Cliente de banco é diferente de um Cliente de uma Academia de Ginástica
- Atenção ao nome da classe
  - Ele pode ser melhorado, para melhor representar o negócio

Dando Nomes

- Nomes devem indicar o resultado da operações
  - E não os passos
- Exemplos
  - calculateAverage()
    - Ruim, pois indica que a média vai ser calculada e isso é uma decisão de projeto
    - O usuário só está interessado em receber a média
  - getAverage()
    - Bom
    - Indica o resultado que o cliente da operação vai ter

Ainda o Nome

- Indica a visão de quem fornece a operação e não de quem solicita
- Exemplo (Posto de Gasolina)
  - dispenseGas()
  - giveGas()
- Mau exemplo
  - receiveGas()
    - Esse nome é quando o objeto recebe gasolina, não quando fornece gasolina
- O método pertence ao objeto

Operações Primitivas

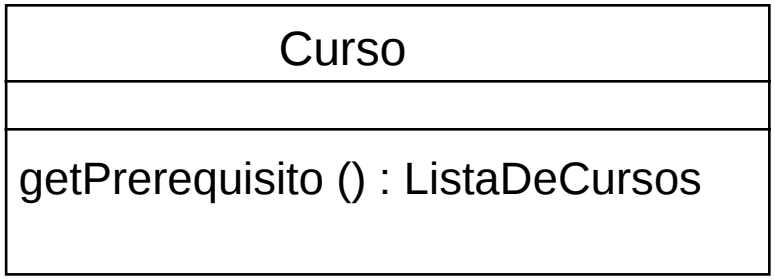
- São as operações que só podem ser implementadas dentro da classe
  - Operações típicas de uma classe são primitiva
- Exemplo
  - Insere item em um conjuto
    - Apenas a classe que implementa o conjunto pode implementar
      - Encapsulamento
  - Insere vários itens em um conjunto
    - Havendo insere um item, então outra classe pode implementar insere vários

Assinatura

- Lista de argumentos
- Classe retornada
- Não precisa ser levantada na análise

Operações em UML

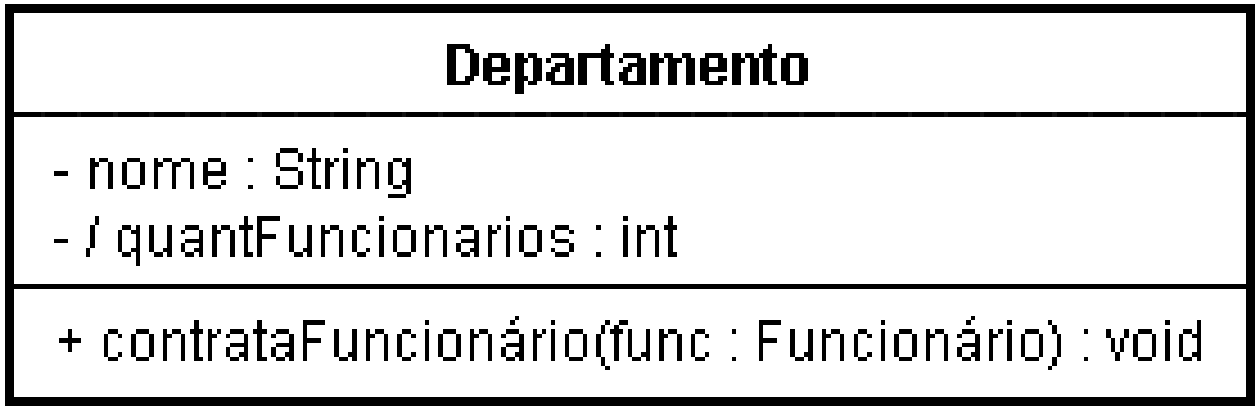
- Terceira parte da Classe



BNF

- [visibility] name ([parameter-list]) ':'

[return-result] [{properties}]



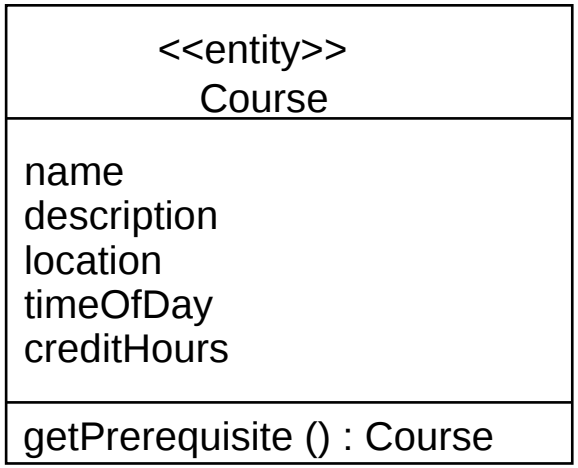
Atributos

O que é um Atributo?

- É uma características dos objetos da classe
- Não possuem identidade
  - Não são objeto por si só
- São descritos normalmente por substantivos
- São definidos de forma clara e concisa

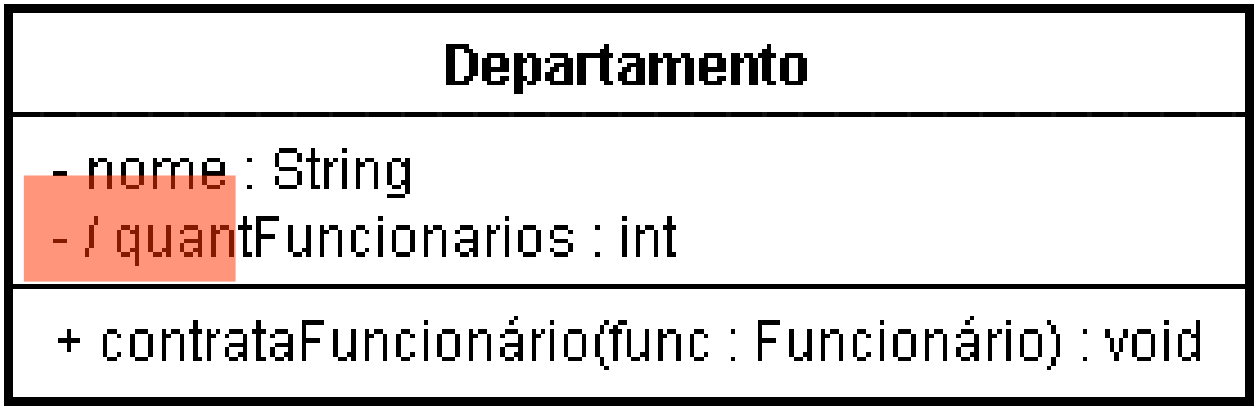
Atributos em UML

- Segunda parte da classe



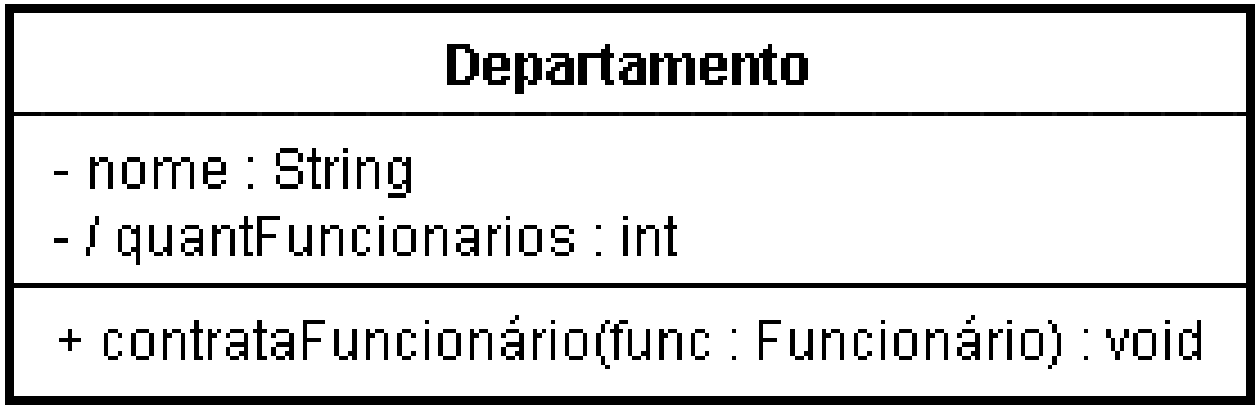
Atributos Derivados

- Podem ter seu valor calculado
  - A partir de outros atributos ou relacionamentos
- Otimização
  - Mais rápido, mais memória
  - Não usar na análise



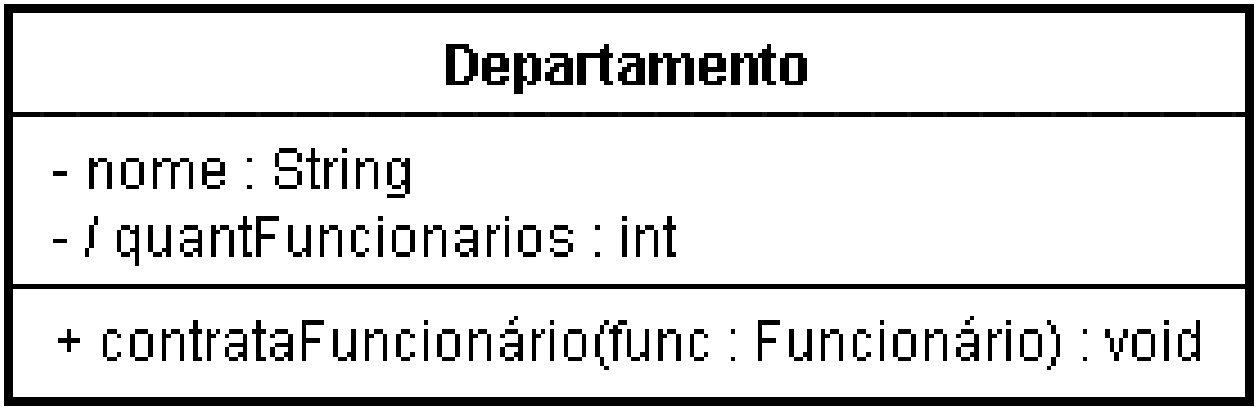
Características do Atributo

- Tipo de Dado
- Valor inicial
- Não são obrigatórios na análise



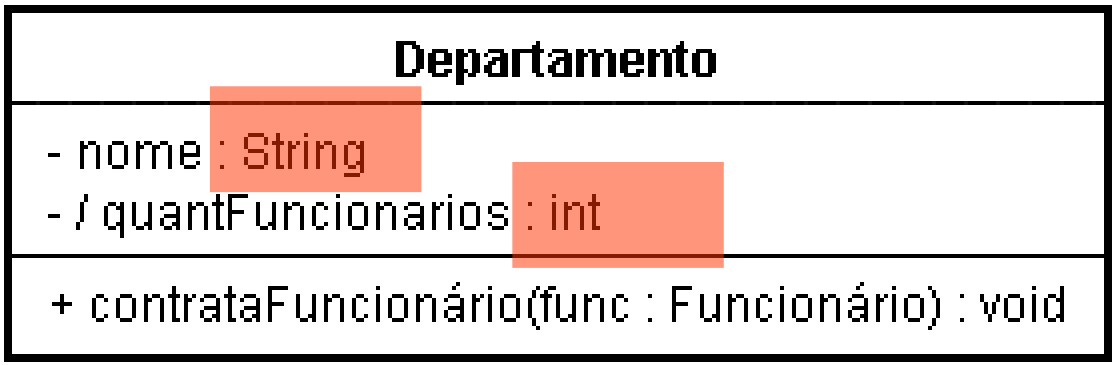
BNF do Atributo

- [visibility] [/] name [: type] [multiplicity] [= default] [{property-string}]
- Visibilidade, property-string veremos mais tarde



Tipo de Dados

- Referência a um tipo de dados primitivo de UML
- Uma enumeração (como Boolean)
- Uma referência ao um tipo de dados de uma linguagem
- Uma referência a outra classe do sistema

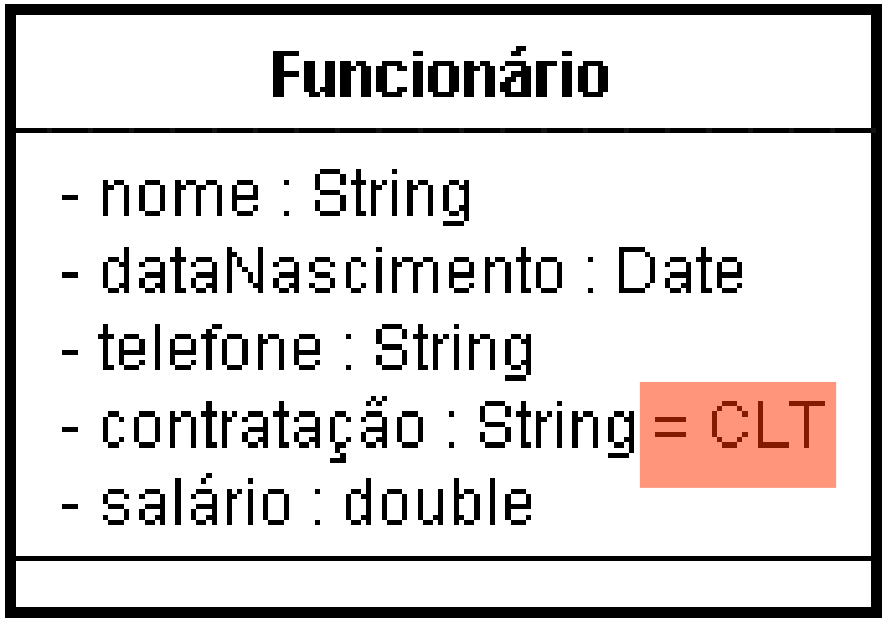


Multiplicidade

- Quantos daquele atributo pode existir
  - Não é um vetor
- Implementado
  - Container...
  - Array...

Valor default

- Valor dado ao atributo quando o objeto é criado



Atributos E Operações Estáticas

Atributos/Operações de Classe

- Algumas Classes apresentam operações ou atributos que não são associadas a instância, mas sim a toda classe
  - Classes de projeto tem maior chance de apresentar essas características
- Não precisa existir um objeto da classe para usar a operação ou atributo
- São “variáveis globais” e “métodos globais” para as instâncias da classe

Atributos e Operações de Classe

- Conhecidos como Estáticos
- São sublinhados nos diagramas

Relacionamentos

Os Relacionamentos

- No mundo real,os objetos estão associados de várias formas
  - Pessoas possuem Objetos
  - Empresas compram de outras empresas (fornecedores) e vendem para empresas e pessoas físicas (clientes)
- Os Sistemas precisam guardar essas associações na sua memória
  - Banco de dados

Relacionamentos e Comportamento

- O comportamento do sistema vêm da colaboração entre os objetos
  - Estruturada por meio dos relacionamentos
  - Executada por meio de mensagens

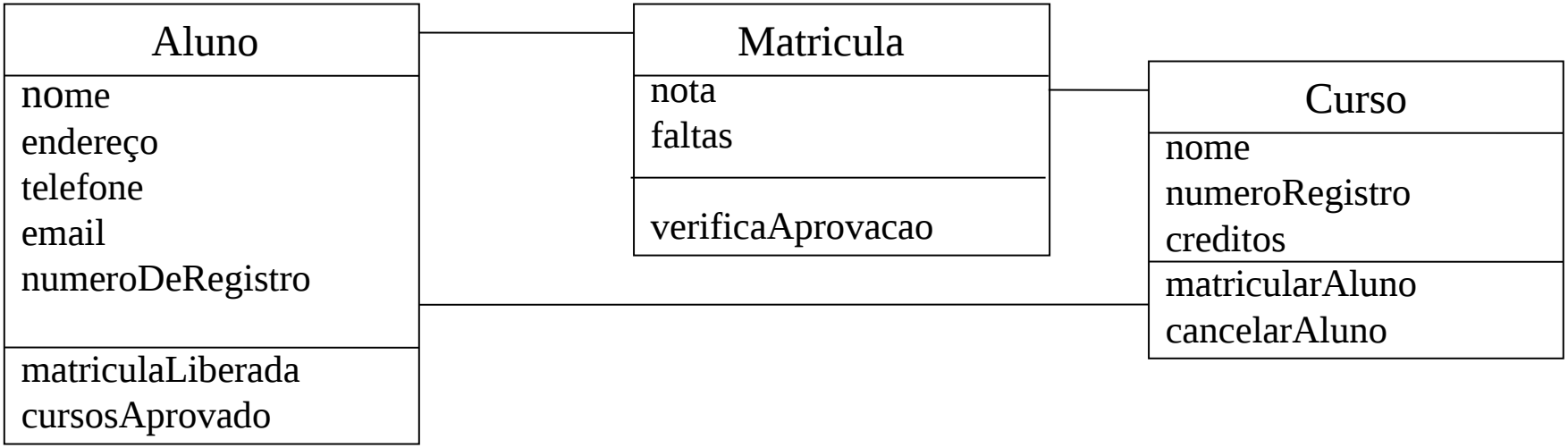
Tipos de Relacionamentos

- Herança
- Associação
- Agregação
- Composição

Associações

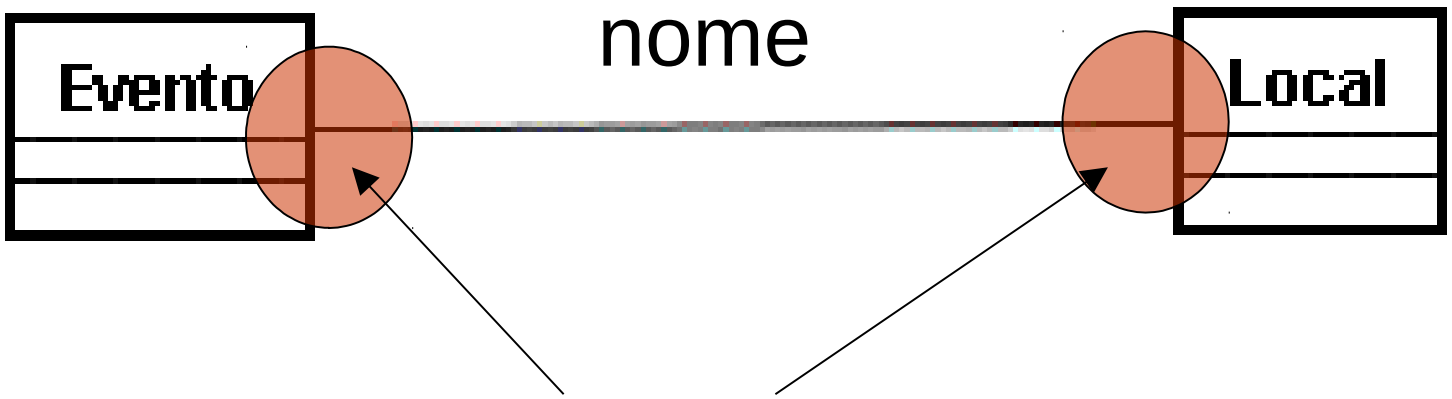
- Uma associação é o relacionamento mais simples
- Representa qualquer forma de conexão / associação / relacionamento entre duas classes
- Objetos de uma classe possuem uma forma de guardar uma referência a objetos de outra classe
  - Ponteiros
- Uma linha
  - Com “decorações”

Associações



Ainda Pouca Informação

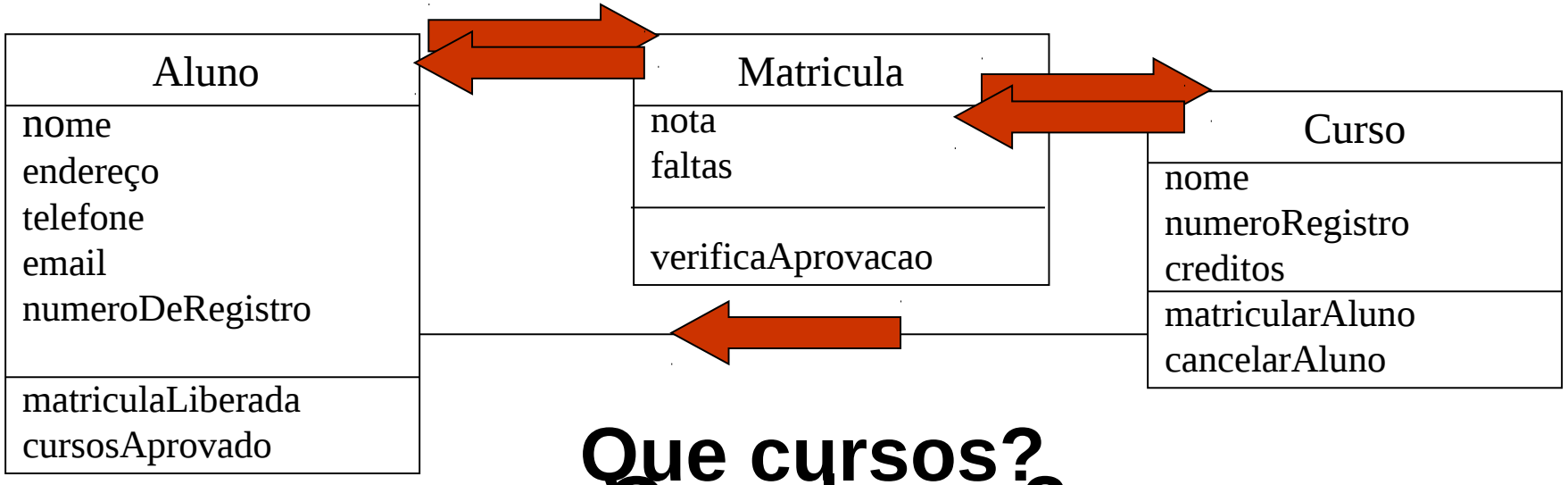
Associações



End point - conexões

Navegando Associações

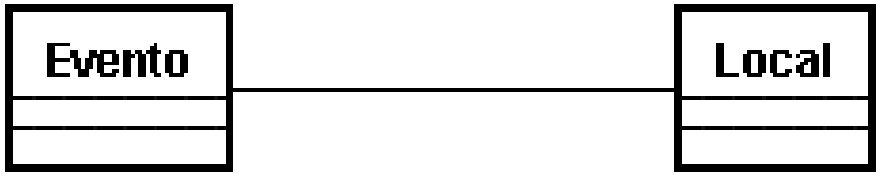
- Bi-direcional
- Ou seja, de um objeto pode se chegar ao outro
- A navegação é entre as instâncias

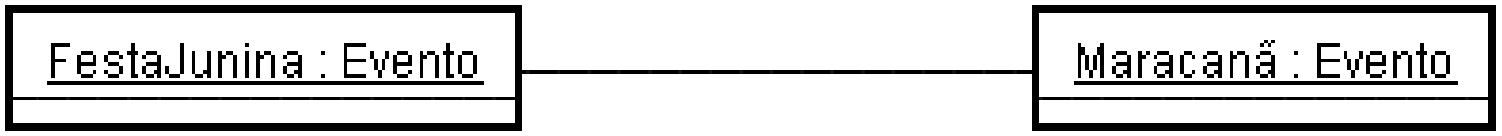


Que cursos?
Que alunos?
Quantos em espera?

Associação e Link

- Uma associação entre classes implica que os objetos possuem links



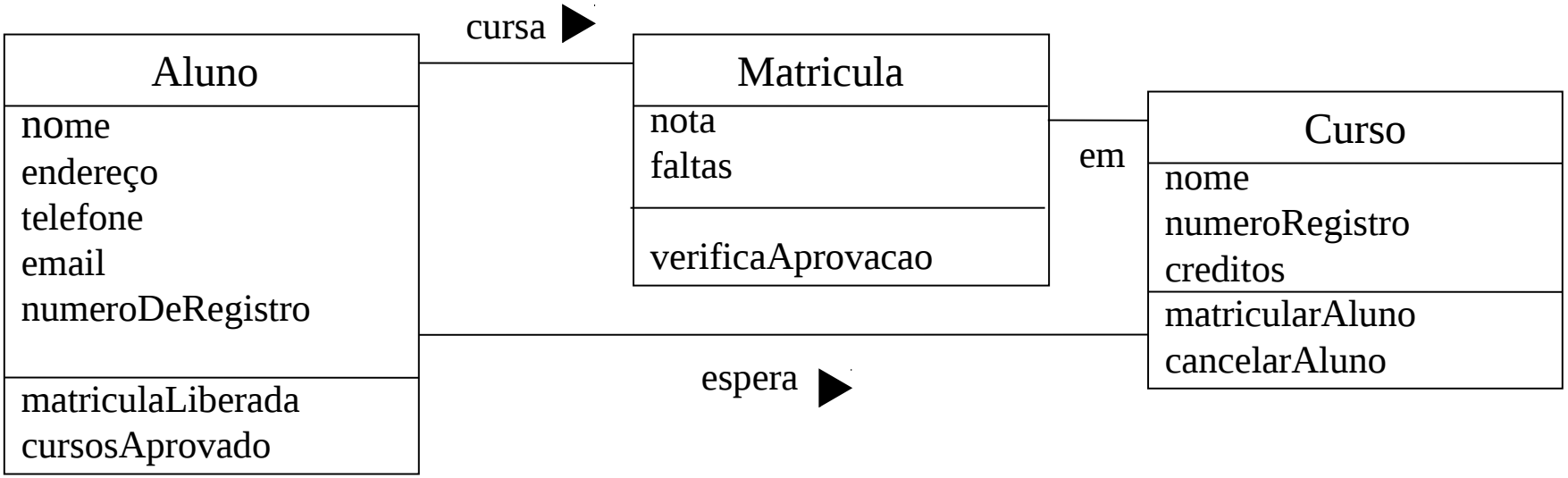


Associações com Nome

- Explica o significado
- Um rótulo no meio da linha
- Verbo ou frase verbal

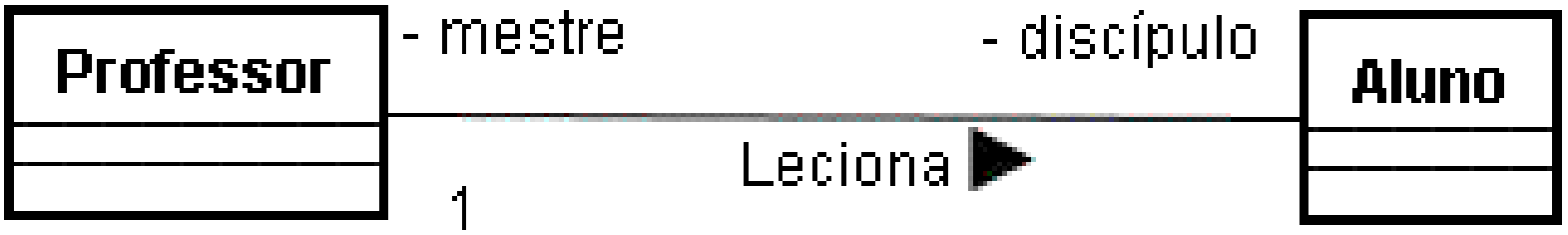
Libere as pontas

Associações



Definindo papéis

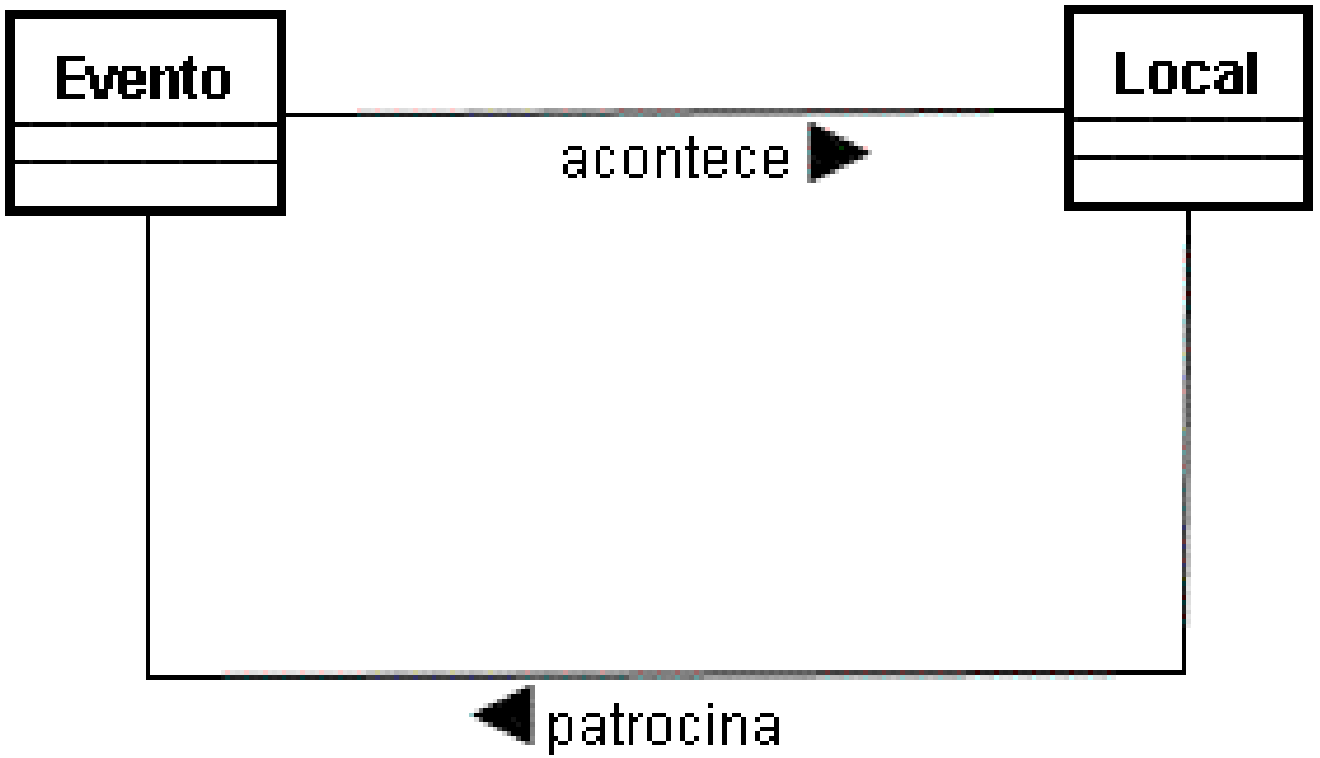
- Um papel explica o objetivo ou a capacidade de uma associação
- Tipicamente substantivos
- Colocado perto da linha da associação
- Pode estar associado a um sentido de leitura



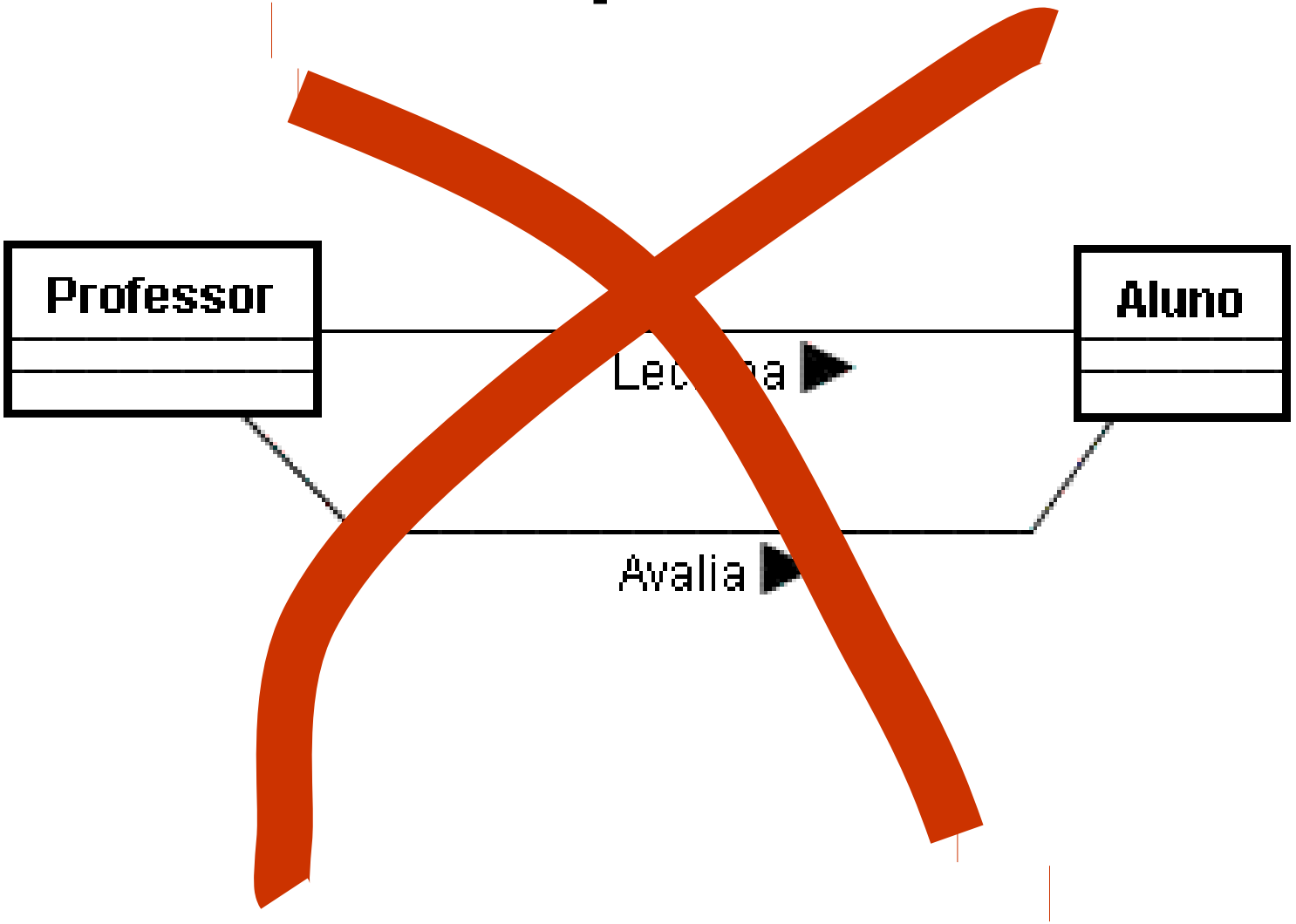
Várias Associações

- Podem existir mais de uma associação entre classes
- Nesse casos os nomes são obrigatórios
- Devemos questionar a necessidade

Várias Associações



Um mau exemplo



Cardinalidade das Associações

- **É o número de instâncias de uma classe relacionadas com UMA instância da outra classe**
- Existe para cada lado da associação

Indicadores de Cardinalidade

- Aparecem em cada ponta da associação
- Indicam o número de objetos daquela ponta que podem aparecer na associação

Cardinalidades

- * para muitos
- 1 para exatemente um
- 0..* para zero ou mais
- 0..1 para zero ou um
- 1..* para um ou mais
- 2..4 para intervalos específicos

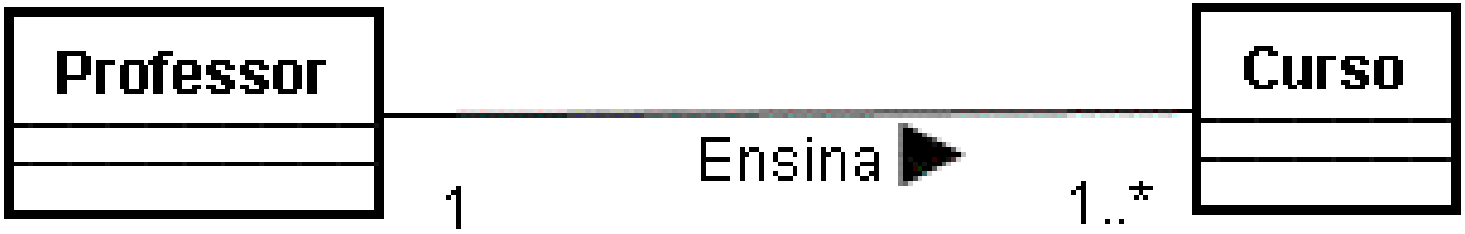
Exemplo

Quantos eventos podem acontecer em um local

Em quantos locais acontece um evento

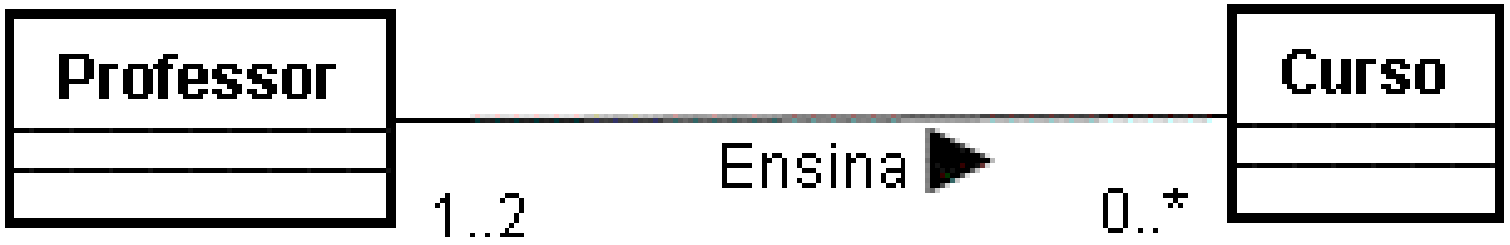
Exemplo

- Nesse exemplo o professor sempre tem que dar um curso
  - Não existe licença
- Um curso sempre tem apenas um professor
  - Cursos não podem ser divididos



Exemplo

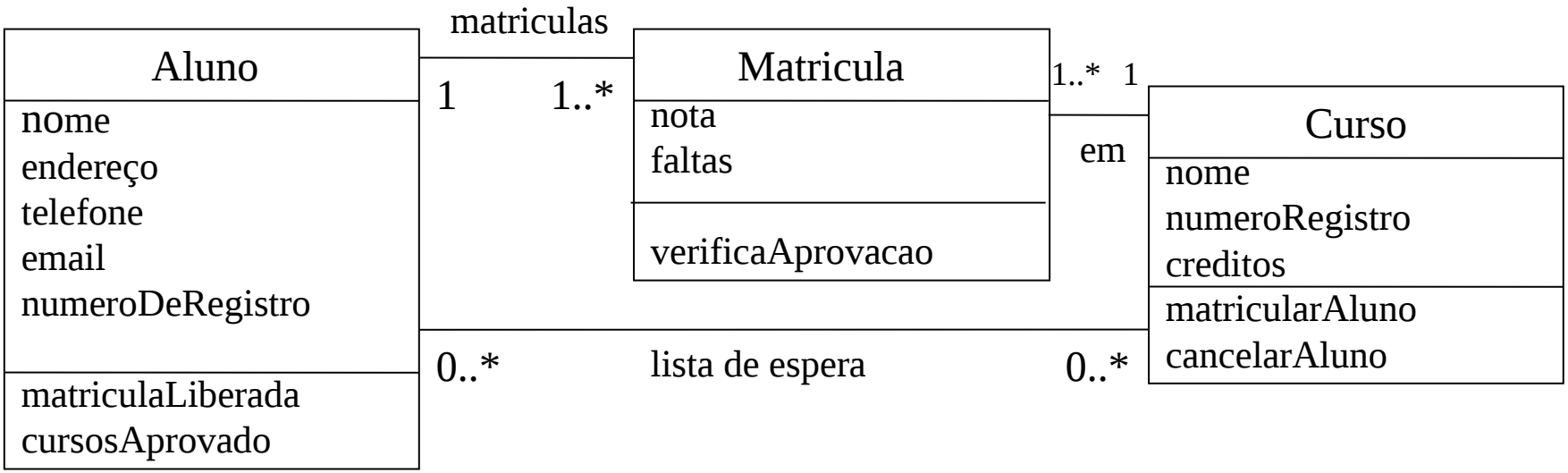
- Nesse caso o professor pode não dar curso nenhum
- Um curso pode ter um ou dois professores, mas não mais que dois



Compreendendo a Cardinalidade

- Relacionamentos
  - Obrigatórios
    - Cardinalidade mínima 1
  - Opcionais
    - Cardinalidade mínima 0
- Cardinalidades mínimas e máxima

Associações



Agregação e Composição

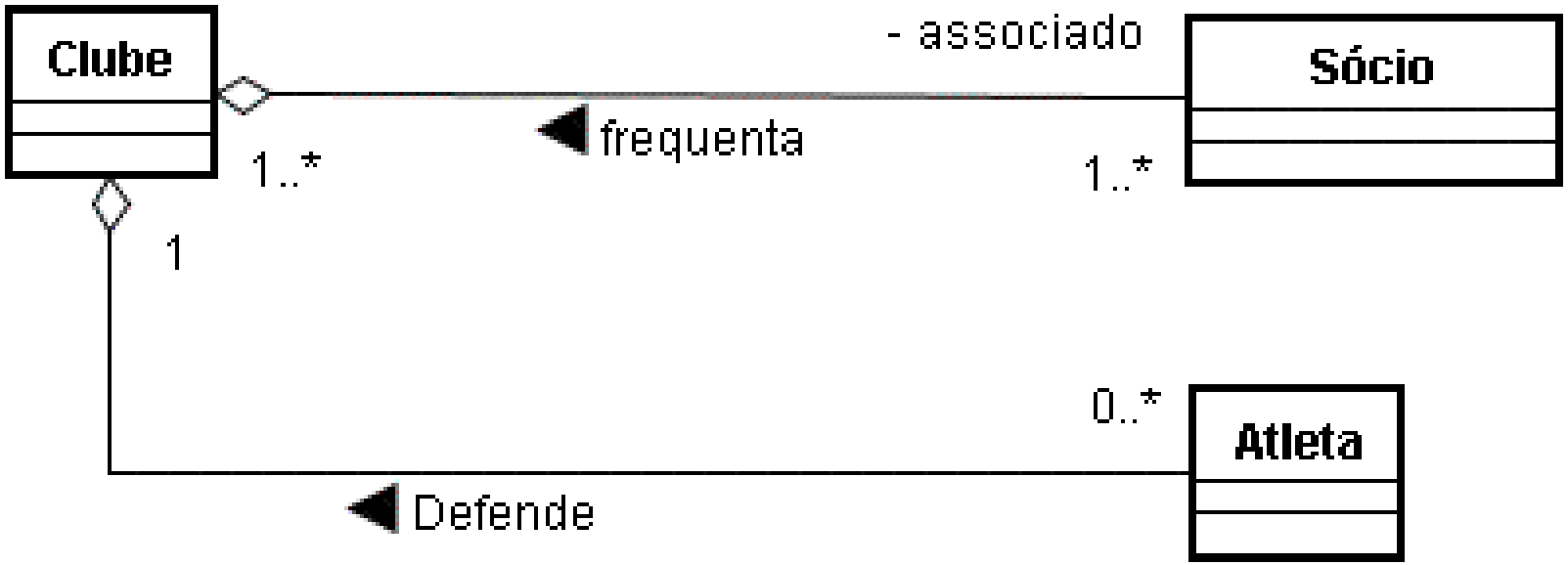
Agregação

- Uma forma especializada da associação
- Define um relacionamento **todo-parte**
- Representada por uma associação com um diamante (losango) junto para classe que denota o todo

Especialização da Agregação

- Na Associação, os objetos conhecem uns aos outros, para colaborar
- Na Agregação
  - A integridade da configuração é protegida
  - Funcionam como uma unidade
  - O controle pode ser feito por um único objeto

Exemplo



Um atleta só defende um clube

Um clube é defendido por 0 ou mais atletas

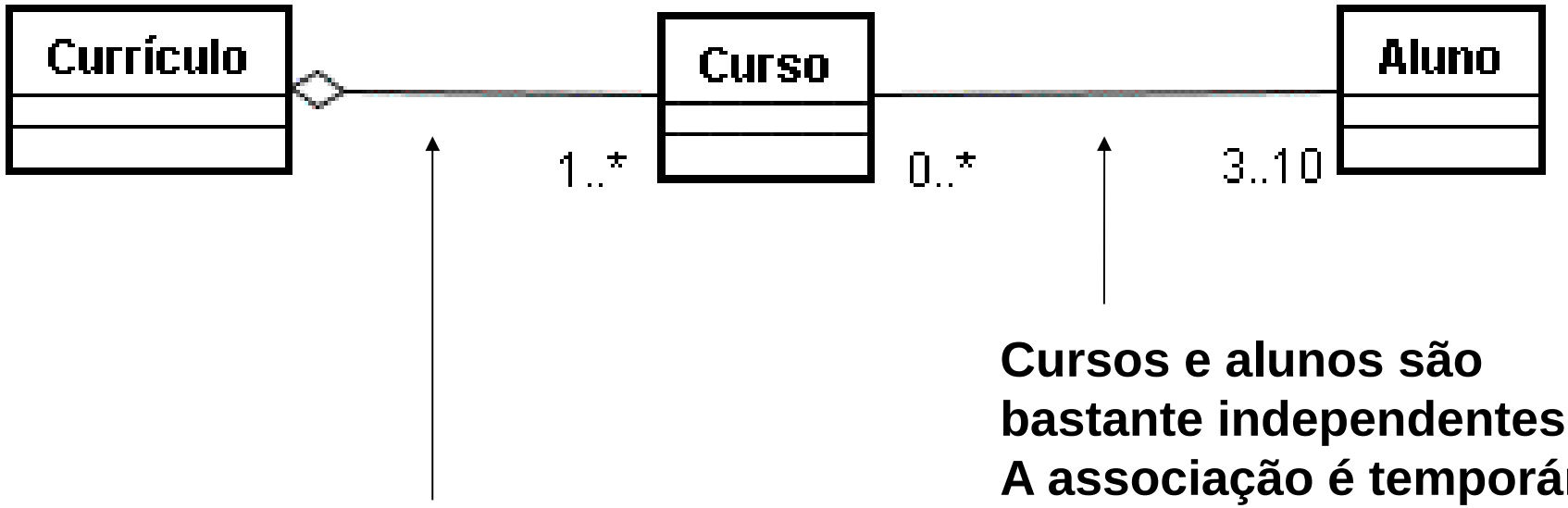
Teste de Agregação

- A frase “é parte de” pode ser usada para descrever o relacionamento
- Algumas operações sobre o todo são aplicadas as partes
- Valores de atributos propagam do todo para as partes?
- Existe uma assimetria intrínseca
  - Um é parte do outro, mas o inverso não é possível

Associação ou Agregação

- Se os objetos estão fortemente ligados pela relação
  - Favorecer agregação
- Se os objetos são independentes, e estão algumas (ou quase sempre) ligados
  - Favorecer a associação

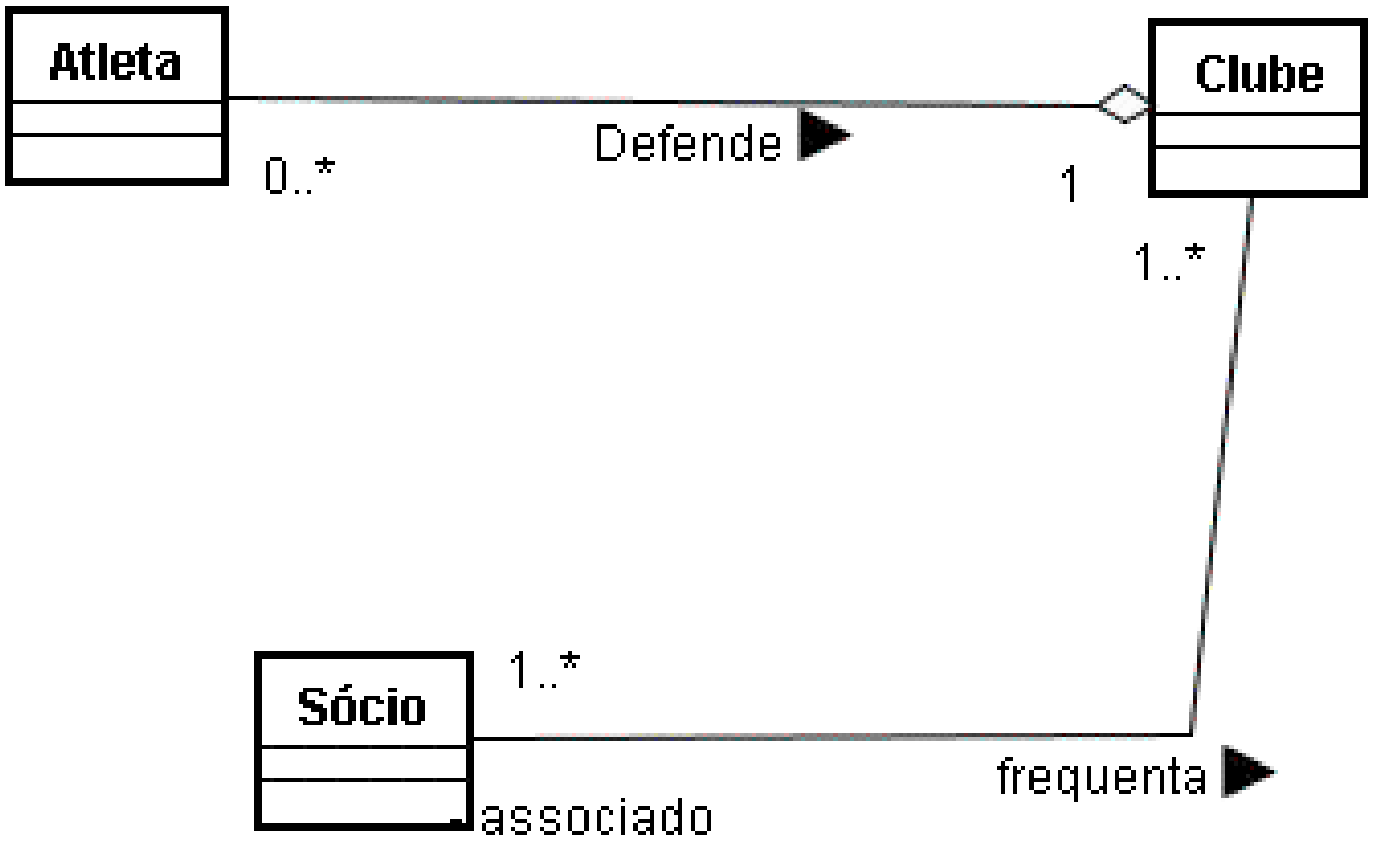
Exemplo



Cursos e alunos são bastante independentes
A associação é temporária

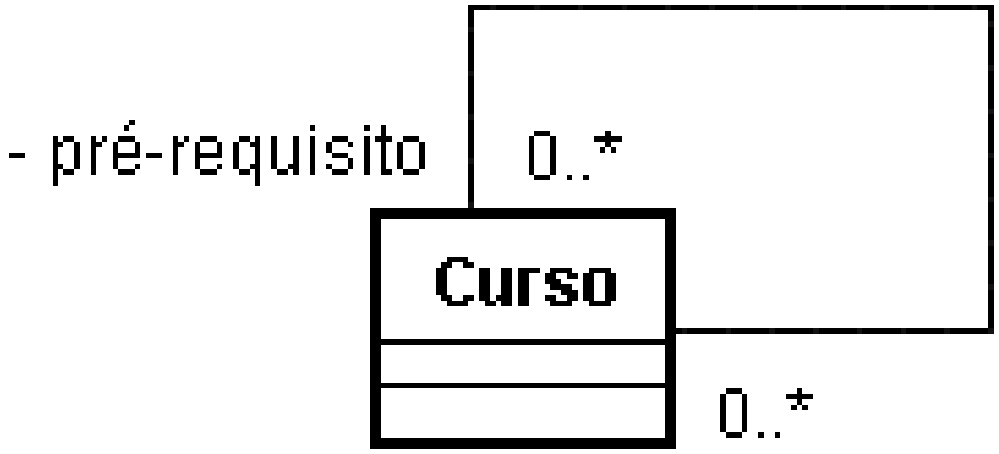
Um currículo precisa ser construído de cursos (1 ou mais)

Exemplo Corrigido 1



Associações Reflexivas

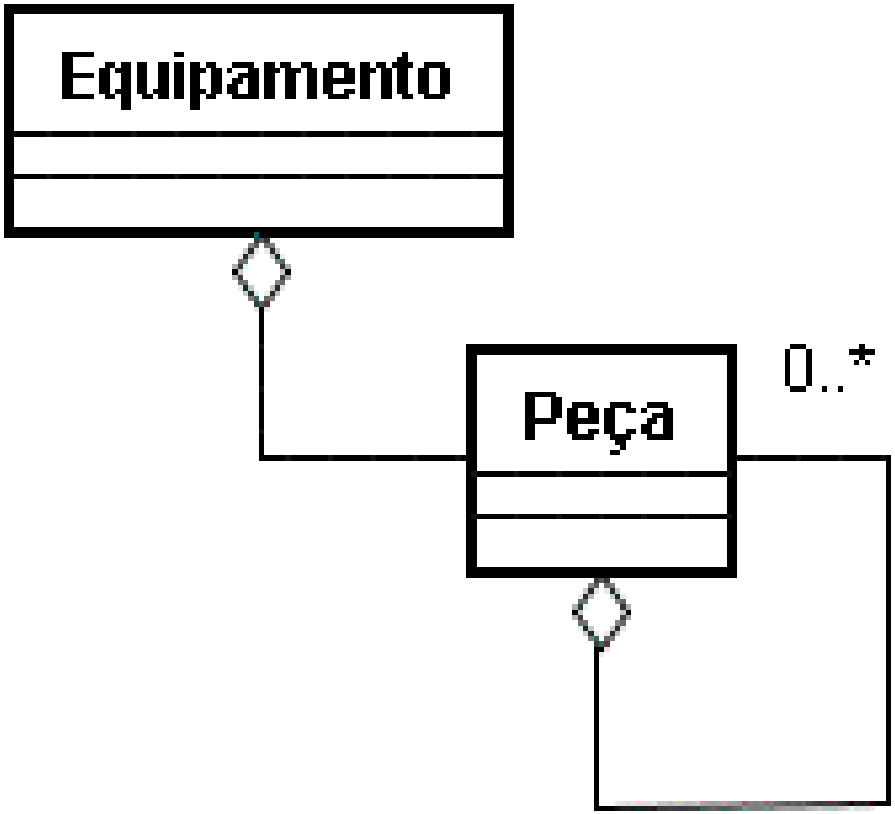
- Cria um relacionamento entre objetos da mesma classe



Agregados Reflexivos

- Agregados também podem ser reflexivos
- Sempre indicam recursividade
- Típico
  - Peças são compostas de outras peças que também podem ser compostas de peças

Agregados Reflexivos



Composição

- Uma especialização da Agregação
- O ciclo de vida da classe membro depende da classe agregada
  - O agregado tem controle sobre a criação e destruição do membro
  - O membro não pode existir sem um agregado

Quando o Agregado é destruído

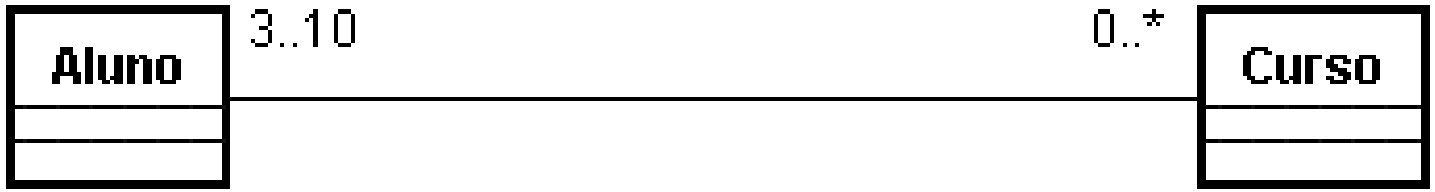
- Duas alternativas
  - O membro é destruído
  - O membro é passado para outro agregado

UML 2.0

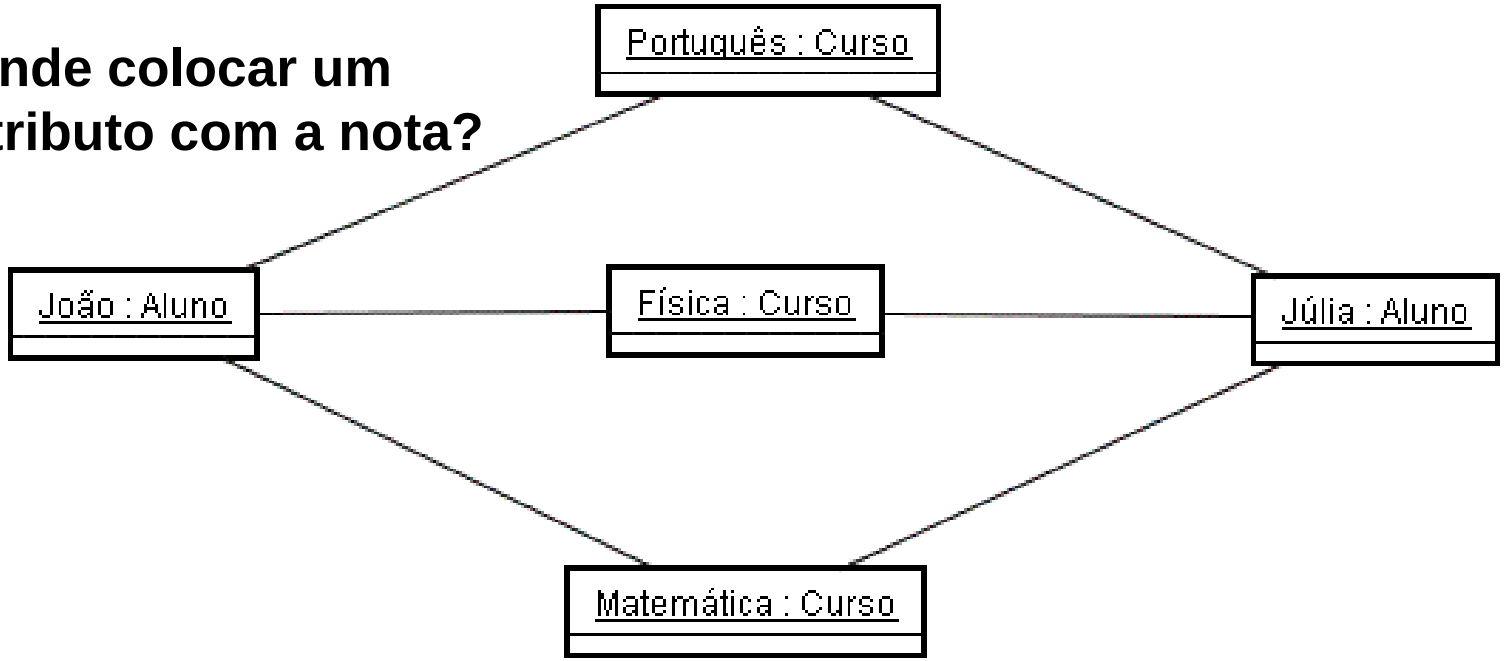
- Agregação
  - None
  - Shared (Agregação)
    - Um membro pode participar de vários agregados
    - Vida independente
    - Se um membro é agregado, o outro tem um tipo de agregação
  - Composite (Composição)
    - Um membro só pode participar de um composto
    - O todo tem responsabilidade sobre a existência e a guarda da parte

Classes de Associação

- Que nota o aluno tirou no curso?



Onde colocar um atributo com a nota?

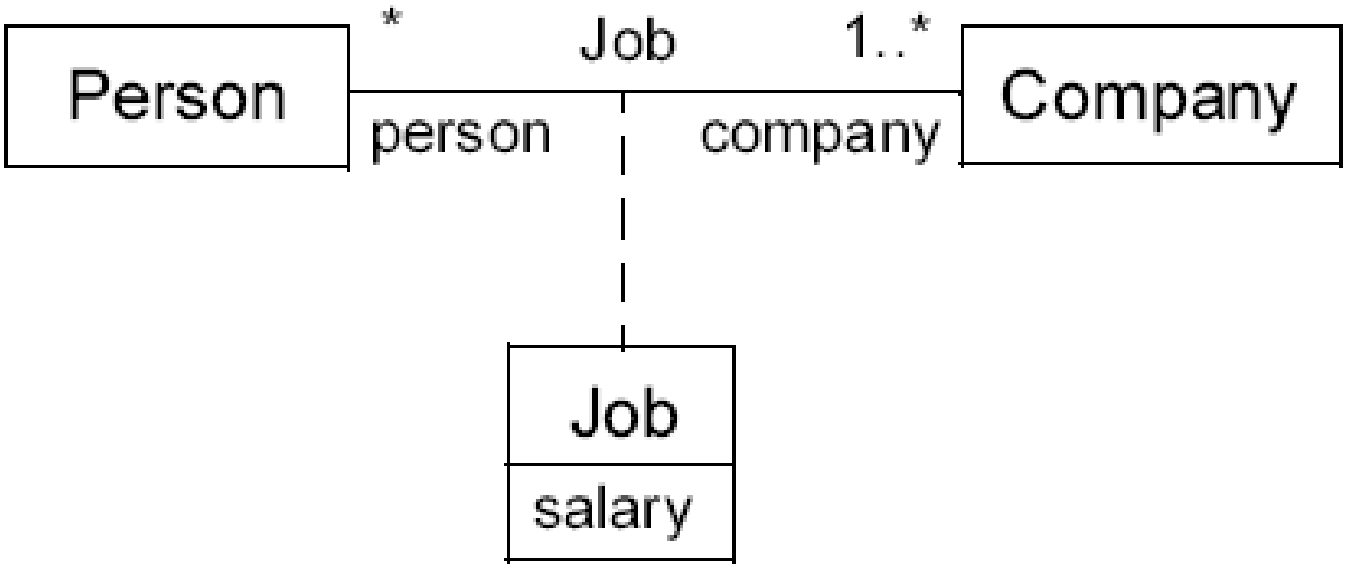


Classes de Associação

- O Atributo não pode ser colocado em Curso, pois existem vários alunos para um curso
  - E não saberíamos de quem é a nota
- O Atributo não pode ser colocado em Aluno, pois ele faz vários cursos
  - E não saberíamos de que curso é a nota
- Concluímos que o atributo pertence ao link
  - E tem que ser representado, de alguma forma, no relacionamento

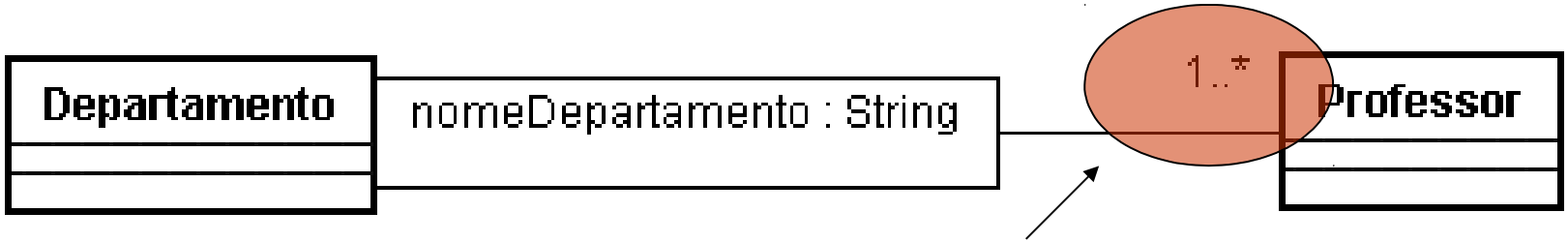
Classes de Associação

- Crie uma classe e ligue ao relacionamento com uma linha pontilhada
- **Apenas uma por relacionamento**



Qualificadores

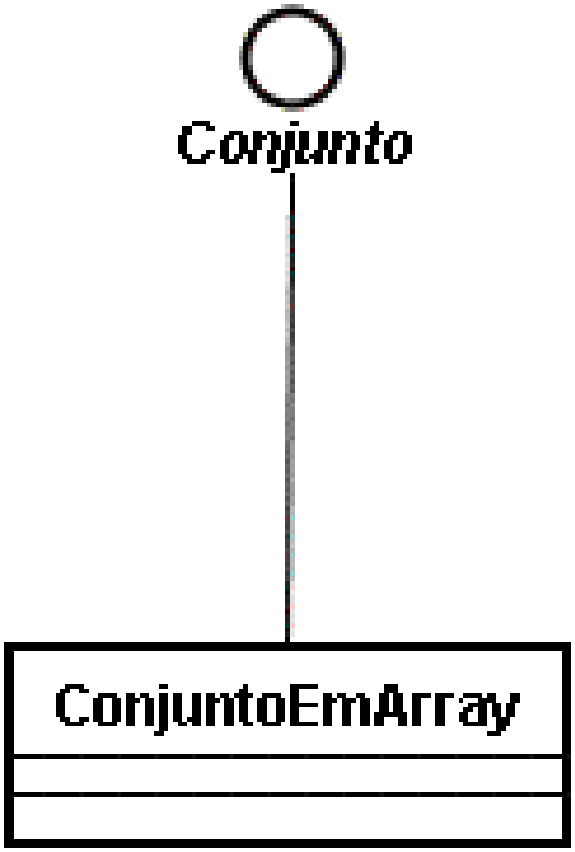
- Um atributo de uma classe que pode ser usado para reduzir a cardinalidade da associação
  - Uma chave

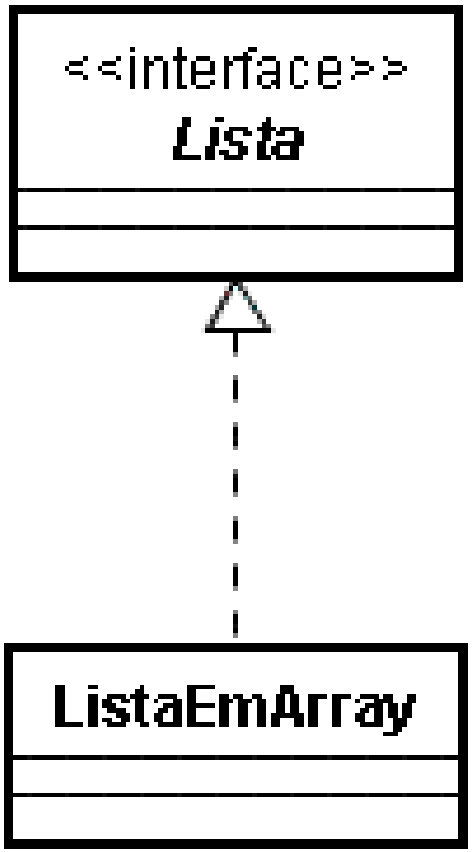


Passa a especificar a Cardinalidade da resposta

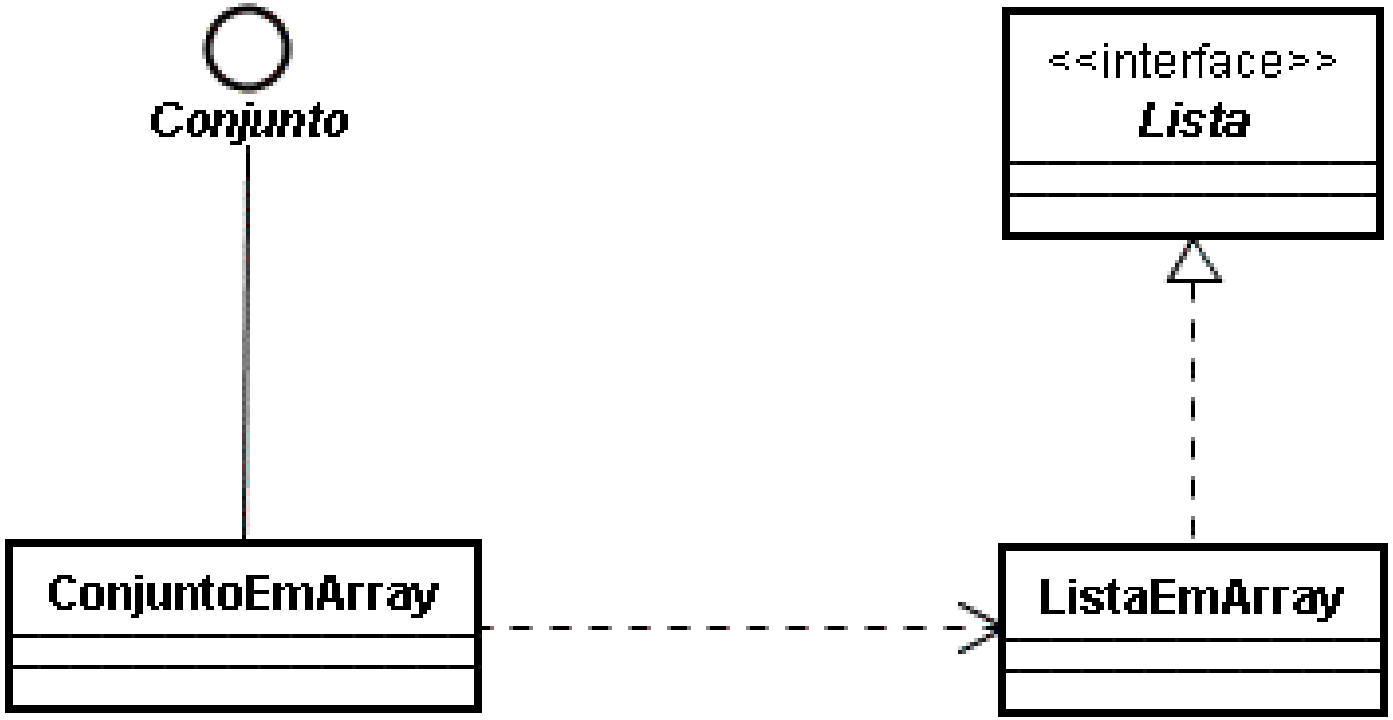
Outros Relacionamentos

Realização





Depende



Herança

Abstração: Generalização

Herança

- Mecanismo que permite definir uma nova classe (subclasse) a partir de uma classe já existente (superclasse)
- Habilidade de um objeto derivar seus atributos (dados) e métodos (funcionalidade) automaticamente de outro objeto

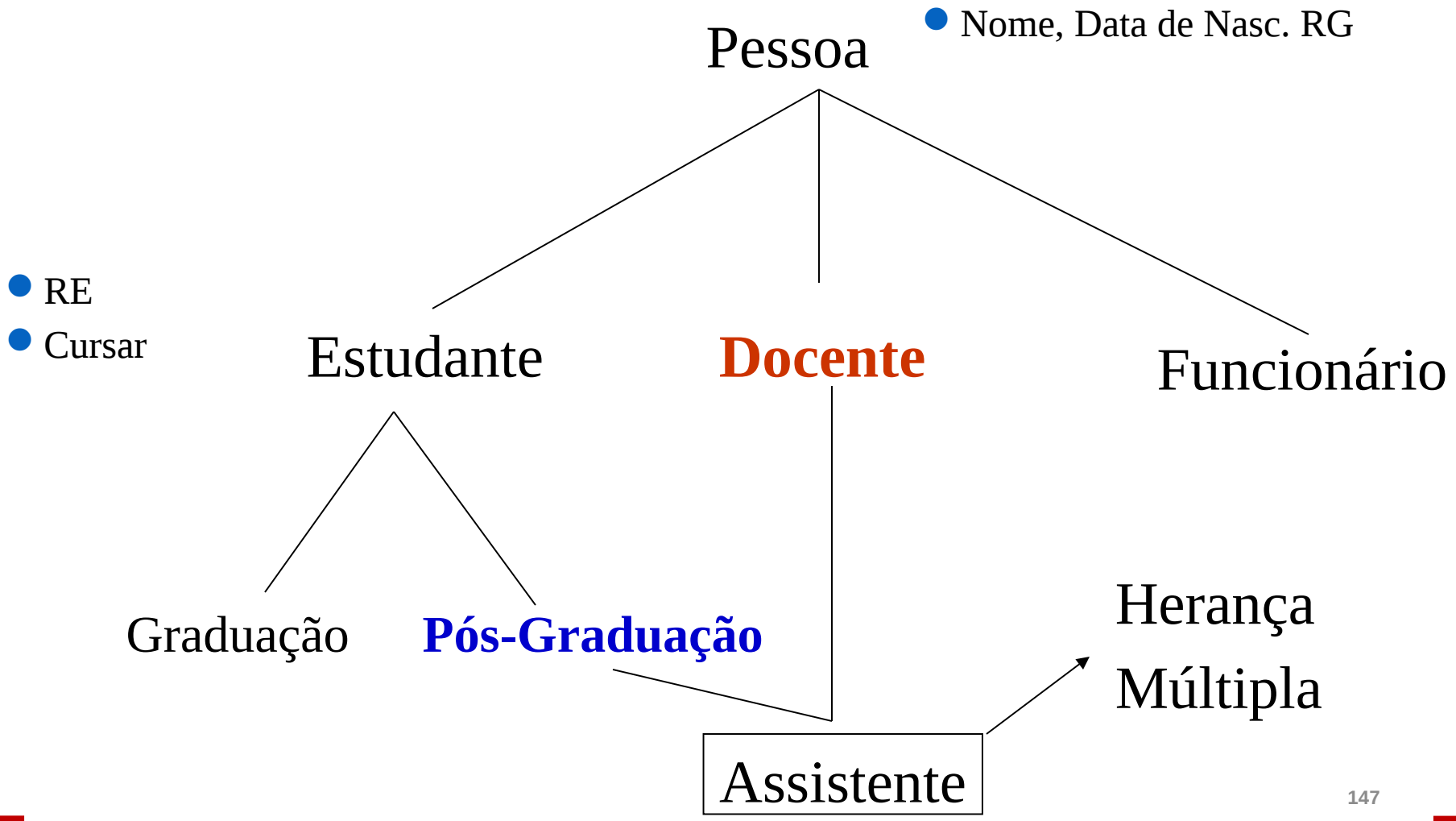
Herança

- Ao se estabelecer uma Especialização (subclasse) de uma classe, a subclasse herda as características comuns da superclasse
  - a especificação dos atributos e dos métodos da superclasse passam a fazer parte da especificação dos atributos e dos métodos da subclasse
  - A subclasse pode adicionar novos atributos, métodos e relacionamentos
  - Reescrever métodos herdados

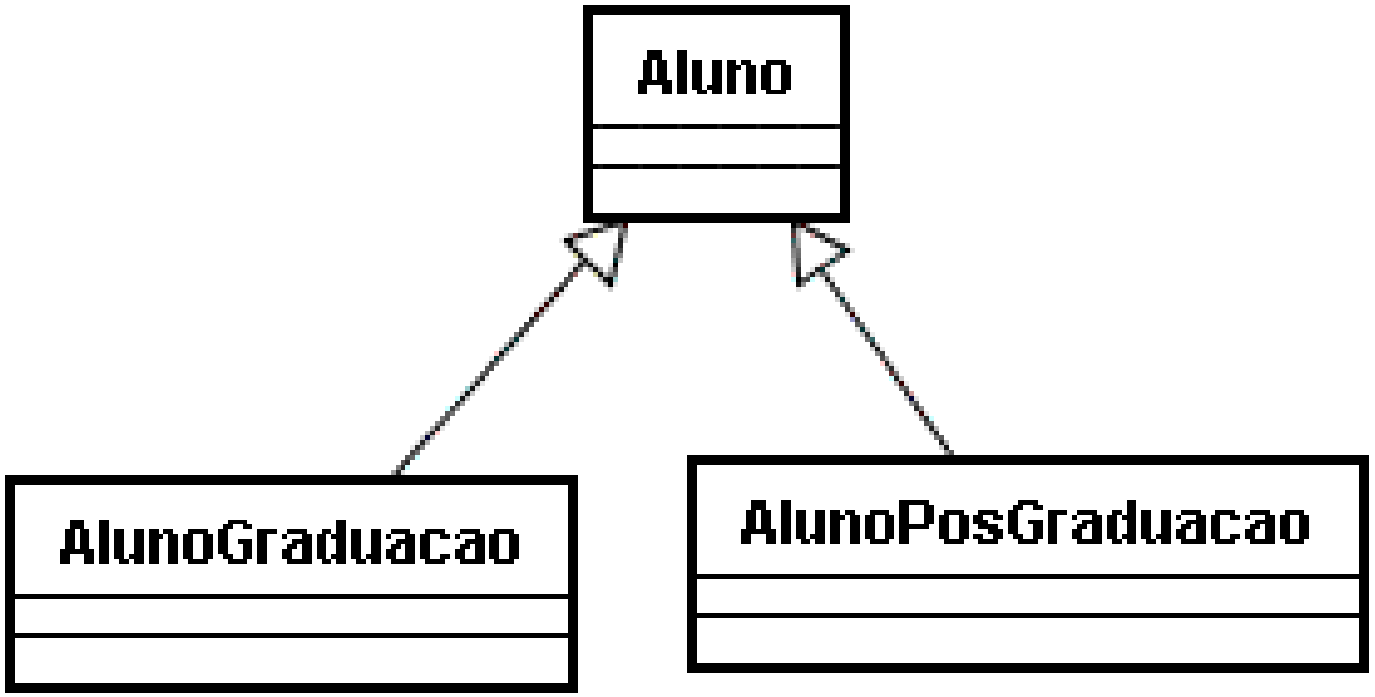
Herança

- Classes podem ser construídas a partir de outras já existentes; facilita extensibilidade
- Quando uma mensagem é enviada para um objeto
  - A procura do método correspondente começa pela classe do objeto
  - Se o método não for encontrado, a procura continua na superclasse
- A Herança pode ser de dois tipos:
  - Herança Simples: quando uma classe é subclasse de somente uma superclasse
  - Herança Múltipla: quando uma classe é subclasse de várias superclasses e conseqüentemente herda as características de cada uma delas

Exemplo: Sistema de Administração Universitária



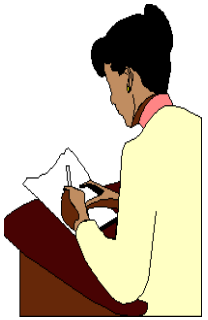
Herança em UML



Encontrando Classes

Encontrando Classes

- Uma classe captura uma abstração chave
  - E apenas uma
- Exemplo
  - Ruim: uma classe Estudante que sabe a informação básica do estudante e possui as matrículas do aluno em um semestre
  - Bom: Uma classe Estudante e uma classe Matrícula do Semestre



Cálculo I
Física I
Lógica

Álgebra I
Cálculo II
Português

Shlaer e Mellor

Encontrando Classes

Shlaer e Mellor

- Objetos tangíveis
- Papéis exercidos
- Eventos
- Interações
- Especificações

Objetos Tangíveis

- Podemos facilmente ver porque objetos tangíveis são bons candidatos a entidades
  - Mapeados diretamente do mundo real
- Normalmente, sistemas de informação falam em algum momento de objetos tangíveis, como produtos e equipamentos.
  - Controlam esses objetos

Papéis Exercidos

- Algumas vezes, porém, um objeto tangível, como uma pessoa, assume uma função ou papel específico, como aluno ou professor.
- Nesse caso, o papel pode ser mais importante que o objeto
- Características notáveis
  - Uma pessoa pode ser aluno e professor
  - Objeto tangível pode ter vários papéis
  - Papéis podem ser exercidos por vários objetos tangíveis

Eventos

- Acontecem em algum momento do tempo e representam classes importantes de entidades.
  - “reunião” em uma agenda .
- Normalmente eventos exigem atributos como data e duração.

Interações

- Um evento que ocorre entre duas entidades
  - “contratação de serviço”
  - “venda de produto”.
- Interações são semelhantes a relacionamentos ou a objetos tangíveis ou eventos, sendo muitas vezes representadas dessa forma.
- Podem ser descritas por um documento do mundo real

Especificação

- Tipos especiais de entidades que classificam outra entidade.
  - Representando a abstração de classificação dentro do modelo, por meio de “meta-classes” no modelo
  - Um bom exemplo é “fábrica”, que é uma especificação para “automóvel”.
- Também podem ser implementadas como um atributo na entidade especificada
  - Decisão de análise.
- Permite maior controle sobre as “classe” válidas para um objeto

UML em Cores

Encontrando Classes

Coad – UML em Cores

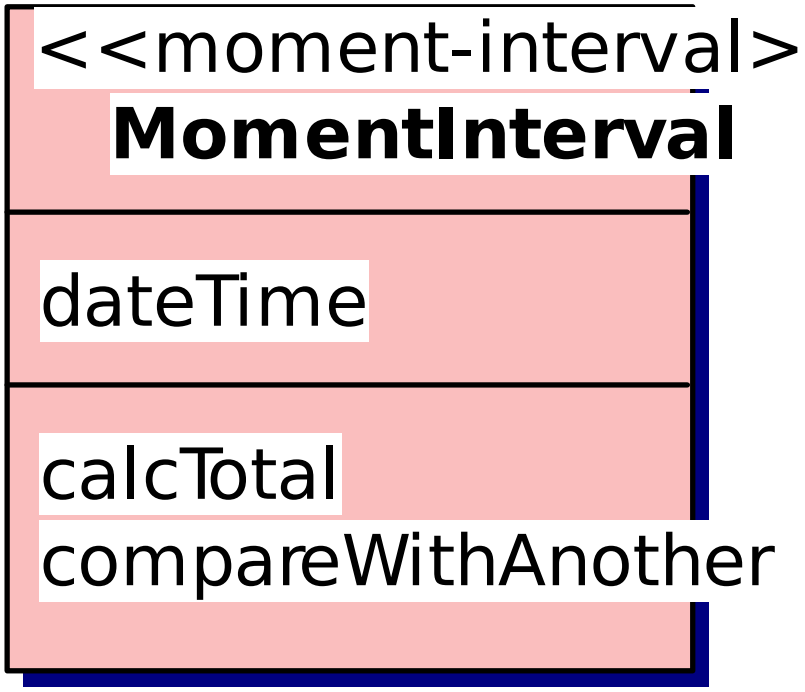
- Objetos do Domínio
  - Momentos ou Intervalos (Rosa)
  - Papéis (Amarelo)
  - Pessoas, Locais ou Coisas (Verde)
  - Descrições (Azul)
- Componentes “Domain-Neutral”

Momentos ou Intervalos

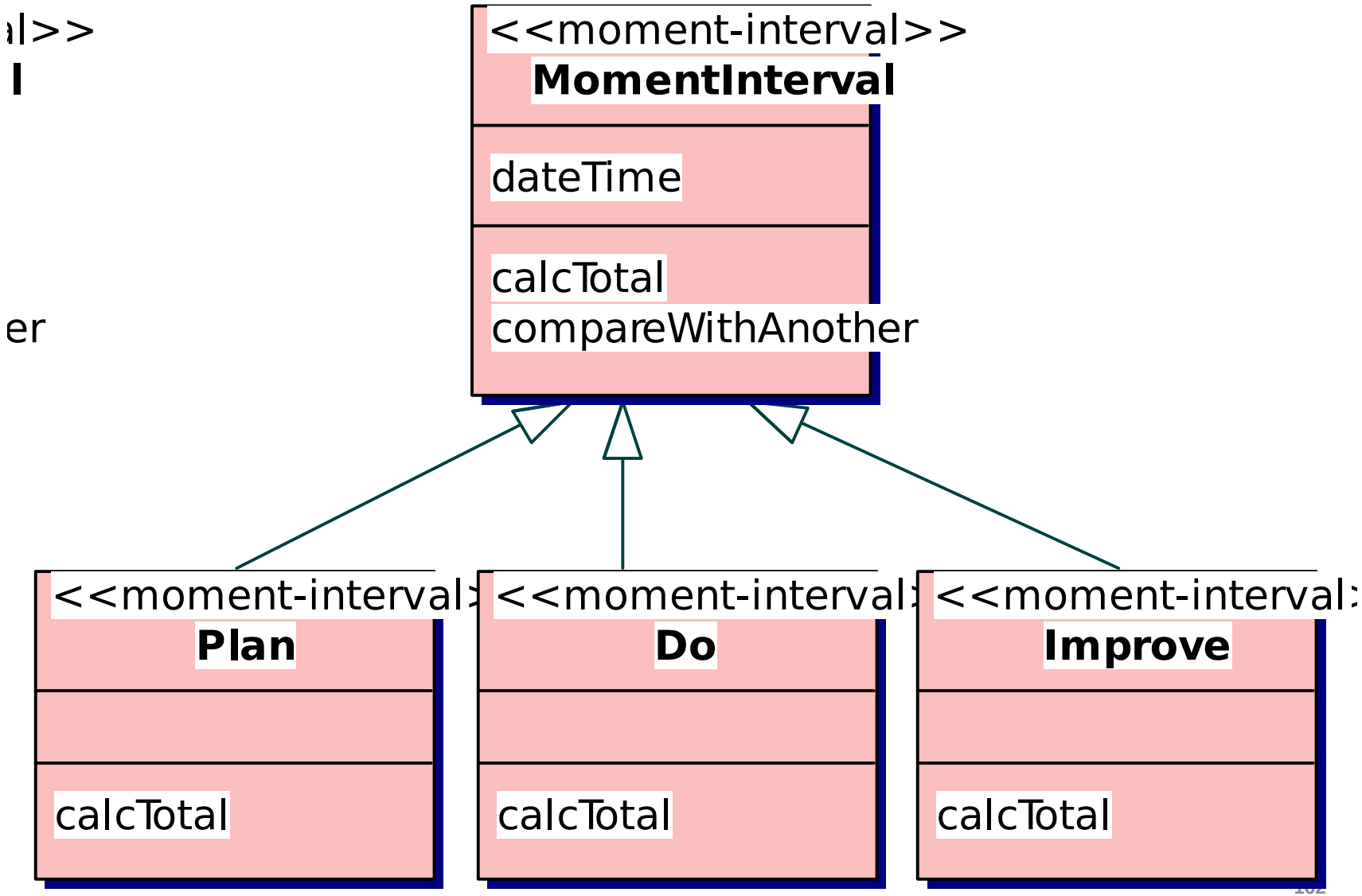
- representa qualquer coisa que precisa ser acompanhada, por motivos de negócio ou legais, e que acontecem em um instante de tempo ou por um período de tempo.
- Muitas vezes pode ser mais fácil começar nossa análise por esse tipo de entidade, pois estamos tratando de atividades de negócio que devem exigir a participação das outras entidades.
- Exemplos são: aulas, consultas, contratação, etc.

- Cor: Rosa / Vermelho

Momento-Intervalo



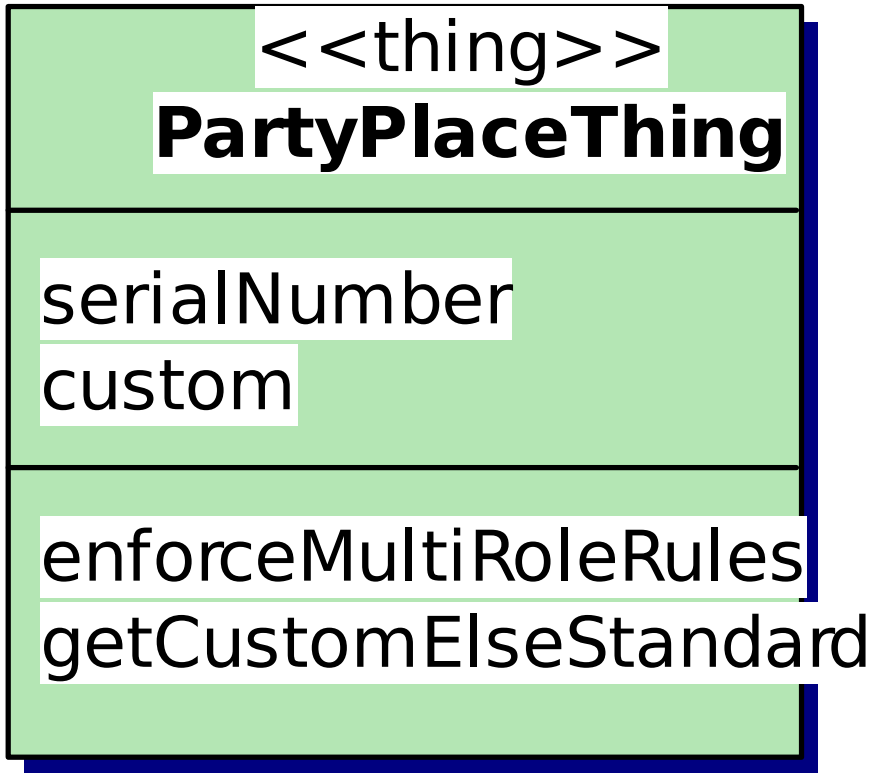
Seqüência de Mom./Int.



Pessoas, Locais ou Coisas

- Pessoas, Locais ou Coisas: representam os objetos tangíveis e localidades.
- Exemplos são: sala, automóvel
- Cor: Verde

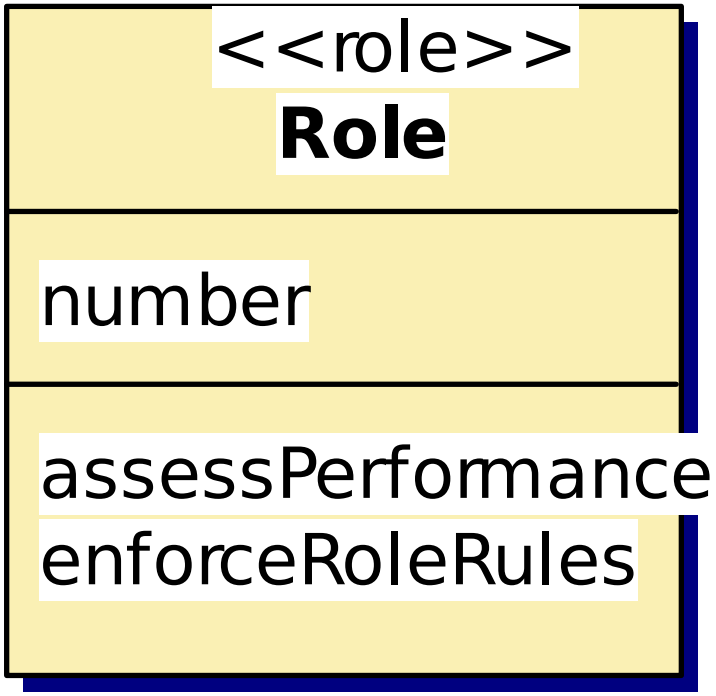
Pessoa-Local-Coisa



Papéis

- representam papéis assumidos pelas pessoas, papéis ou coisas que estão envolvidas com o sistema sendo analisado.
- Cuidado, pois não são apenas os usuários, nem representam os cargos que as pessoas ocupam nas empresas necessariamente.
- Exemplos aluno, professor
- Cor: Amarelo

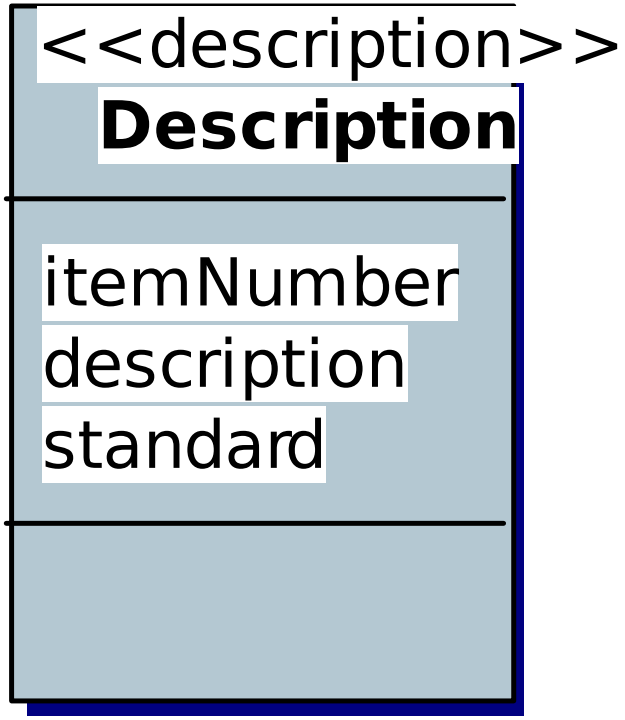
Papel



Descrições

- Descrições: são basicamente as especificações propostas por Shlaer e Mellor.
- Modelos de um produto é um bom exemplo
- Cor: Azul

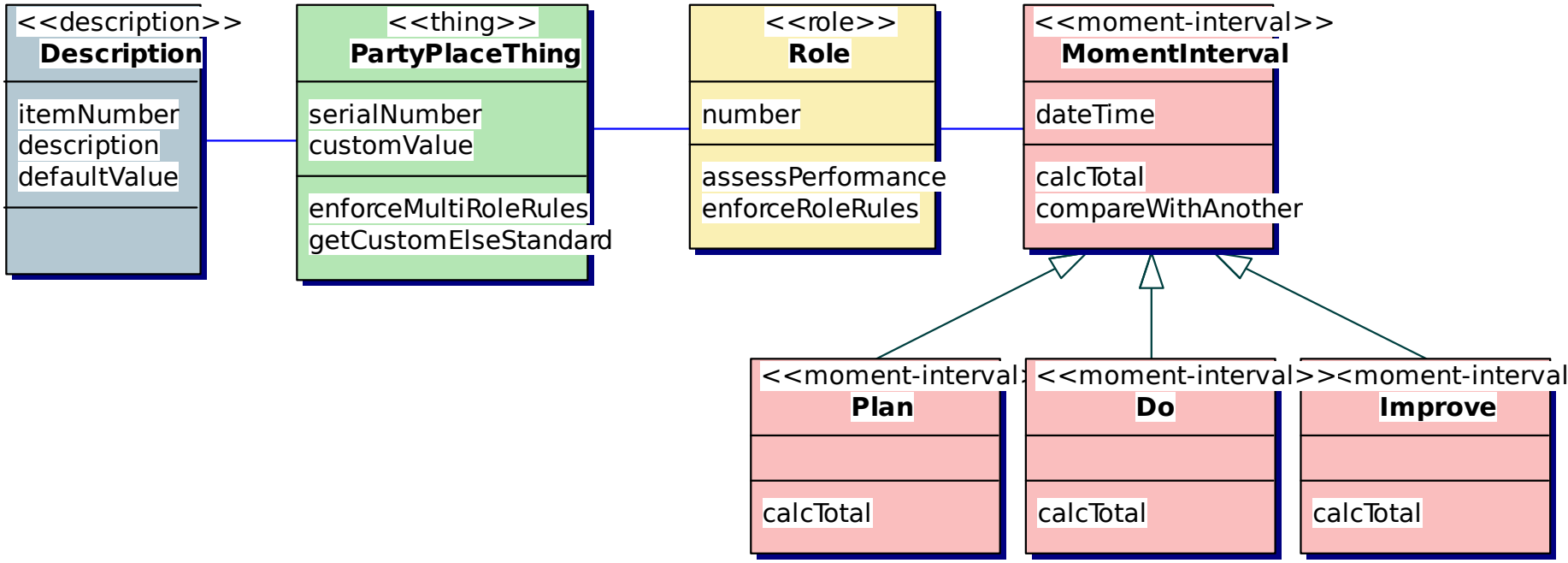
Descrição


<<description>>
Description
itemNumber
description
standard

Componentes “Domain-Neutral”

- Usam os componentes do domínio como partes
- Essas partes aparecem nos componentes de forma repetida e previsível
  - Padrões

Componente


<<description>>
Description
itemNumber
description
defaultValue
<<thing>>
PartyPlaceThing
serialNumber
customValue
enforceMultiRoleRules
getCustomElseStandard
<<role>>
Role
number
assessPerformance
enforceRoleRules
<<moment-interval>>
MomentInterval
dateTime
calcTotal
compareWithAnother
<<moment-interval>>
Plan
calcTotal
<<moment-interval>>
Do
calcTotal
<<moment-interval>>
Improve
calcTotal

Outras Dicas

Encontrando Classes

Ross

- Entidades Núcleo
- Entidades Dependentes
- Entidades de Associação

Entidades Núcleo (Kernel)

- Conceitos mais básicos do domínio do problema
- Não dependem de outras entidades para existir.

Entidades Dependentes:

- Conceitos que são naturalmente dependentes de outra entidade específica (e apenas uma) para sua existência
- Normalmente associados a aspectos de outra entidade que são multi-valorados (como produtos e suas quantidades em um pedido).

Entidades de Associação

- Conceitos que são naturalmente dependentes de mais de uma entidade para sua existência.

James e Suzanne Robertson

- Toda entidade deve ter um papel único e definido no negócio, se você não pode explicá-la, provavelmente não precisa se lembrar dela.
- Entidades devem ter ao menos um atributo que as descrevam, e é preferível que tenham vários.
  - Objetos podem ter só comportamento
- Entidades devem ter mais de uma instância.
  - Se a instância é única, então não deve ser uma entidade, mas uma informação constante, que é parte do negócio da empresa (uma regra de negócio?).
  - Existem objetos com instância única
  - Singleton
- Entidades devem possuir instâncias unicamente identificáveis.

Dicas JR+SR

- Entidades não possuem valores, apenas atributos possuem valores.
- Pessoas e organizações que interagem com o sistema são candidatos a entidade quando precisamos nos lembrar alguma coisa específica sobre elas, para gerar relatórios ou processar dados entrados.
  - Isso não se aplica a “logons” ou “passwords” utilizados para a segurança do sistema, pois segurança é um problema tratado no projeto físico.
  - Devemos aplicar essa regra em relação à necessidade de identificação e endereçamento, por exemplo.

Dicas JR+SR

- Relatórios raramente são entidades. Normalmente eles são apenas os resultados de um processo que acessa várias entidades.
  - Porém, podem ser objetos
  - Diferença entre objetos e entidades
    - O comportamento define o objeto
- Linhas de relatório geralmente são entidades. Nomes de colunas indicam entidades ou seus atributos. Porém, nenhum valor calculado ou derivado é atributo ou entidade.

Dicas JR+SR

- Substantivos em regras de negócio são normalmente entidades
- Produtos, quando não são únicos, são normalmente entidades.
- Papéis, como funcionário, atendente, apostador, etc., são bons candidatos para entidades.
- Um grupo de dados que se repete em uma entrada ou saída de dados é normalmente uma entidade (ou mais).

Onde Encontrar Objetos

- Relatórios
- Formulários de entrada de dados
- Arquivos, tanto de papel quanto no computador.
- Fichas, como fichas de cadastro, de empréstimo, etc.
- Pedidos, requisições e documentos do gênero.
- Documentos contábeis e fiscais, como nota fiscal.
- Planilhas de dados, em papel ou eletrônicas.
- Listagens, registros, agendas, protocolos e outros documentos de trabalho.
- Sistemas já existentes
- Bancos de dados já existentes
- sistemas semelhantes já resolvidos
- padrões de projeto[1] ou padrões internacionais sobre o assunto sendo tratado.

Pacotes

Organizando Classes

- Modelos OO possuem muitas classes
  - Necessidade de organizá-las
- Classes podem ser agrupadas em Pacotes

Pacotes

- São um mecanismo de agrupamento onde podemos colocar qualquer “classificador” de UML
  - Definições de Classe
  - Casos de Uso
  - Estados
  - ...

Tipos de Elementos

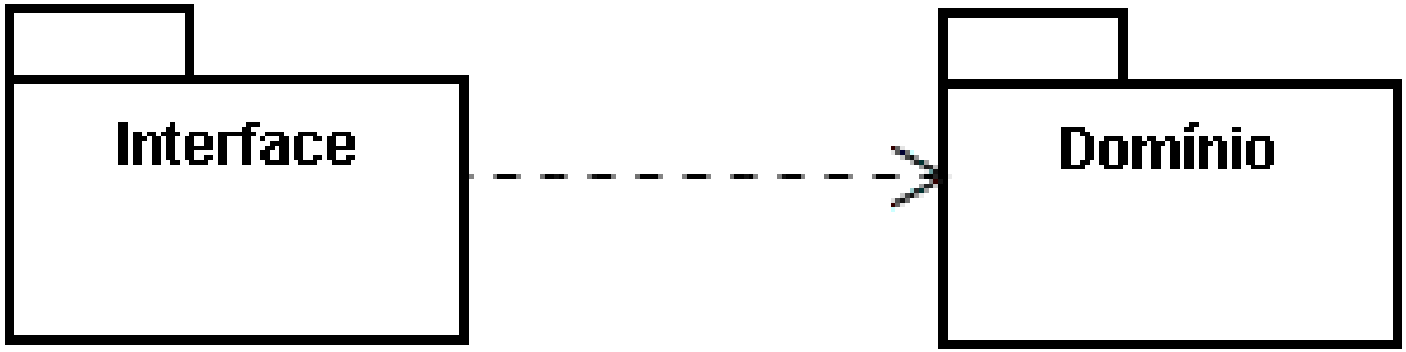
- Elementos possuídos pelo pacote
- Elementos importados pelo pacote
  - “Visitados”
- Elementos acessados que acessam o pacote
  - “Visitantes”

Espaço de Nomes

- Cada pacote fornece um espaço único para os nomes
- Namespace
- Os espaços de nomes são definidos de forma hierárquica

Dependência (Pacotes)

- Explica a relação entre pacotes



Diagramas de Classe

O que são Diagramas de Classe

- A visão lógica do sistema é formada de muitos pacotes e classes
- Um Diagrama de Classes é uma visão de algumas (ou todas) as classes, e pacotes, na visão lógica
  - Normalmente temos muitos diagramas de classe

Diagramas de Classe

- O **diagrama de classe principal** normalmente mostra os pacotes principais do sistema
- Cada pacote de ter seu diagrama de classes próprio
- Classes de um cenário
- Classes privadas de um pacote
- Uma hierarquia de herança

Visibilidade

Visibilidade

- Privada
  - Apenas objetos da própria classe vêem o atributo, método ou relacionamento
- Protegida
  - Objetos da própria classe e suas sub-classes vêem o atributo, método ou relacionamento
- Pública
  - Qualquer objeto vê o atributo, método ou relacionamento
- Pacote
  - Objetos que estão no mesmo Pacote (veremos mais tarde) vêem o relacionamento

Símbolos para Visibilidade

- - Privada
- # Protegida
- + Pública
- ~ Pacote

Restrições

O que são restrições

- Uma restrição é uma condição que pode ser expressa em:
  - Linguagem natural
  - Outra linguagem compreensível por uma máquina
- Declara alguma semântica para algum elemento de UML

Restrições

- É uma expressão booleana
- Não possuem efeitos colaterais
- Não podem ser aplicadas a si mesma
- Restringe a extensão de um elemento além das imposições das construções de UML
- Mostrada como uma string entre chaves

Exemplo

- { deve ser menor que 11 }
- tamanho : int { tamanho < 11 }
- tamanho : int { atéDez : tamanho < 11}

Comentários

- Existem restrições pré-definidas
  - {xor} para associações
- Podem ser escritas em OCL
  - Não é obrigatório
  - Java? C#
    - MDA
- Nome opcional

Do padrão UML

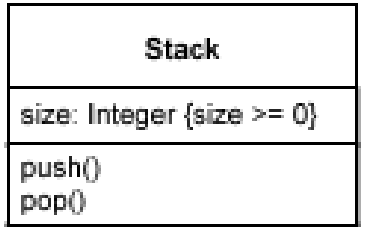


Figure 31 - Constraint attached to an attribute

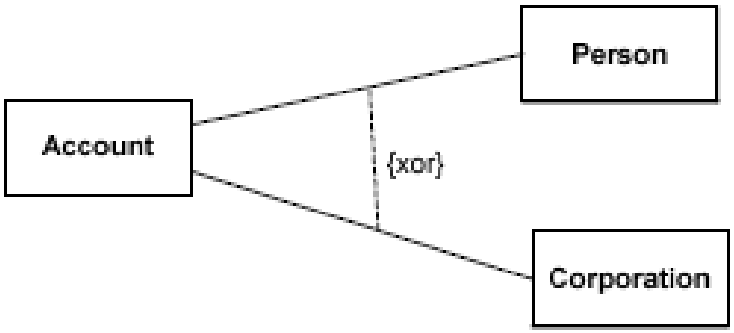


Figure 32 - {xor} constraint

Exemplos do Padrão

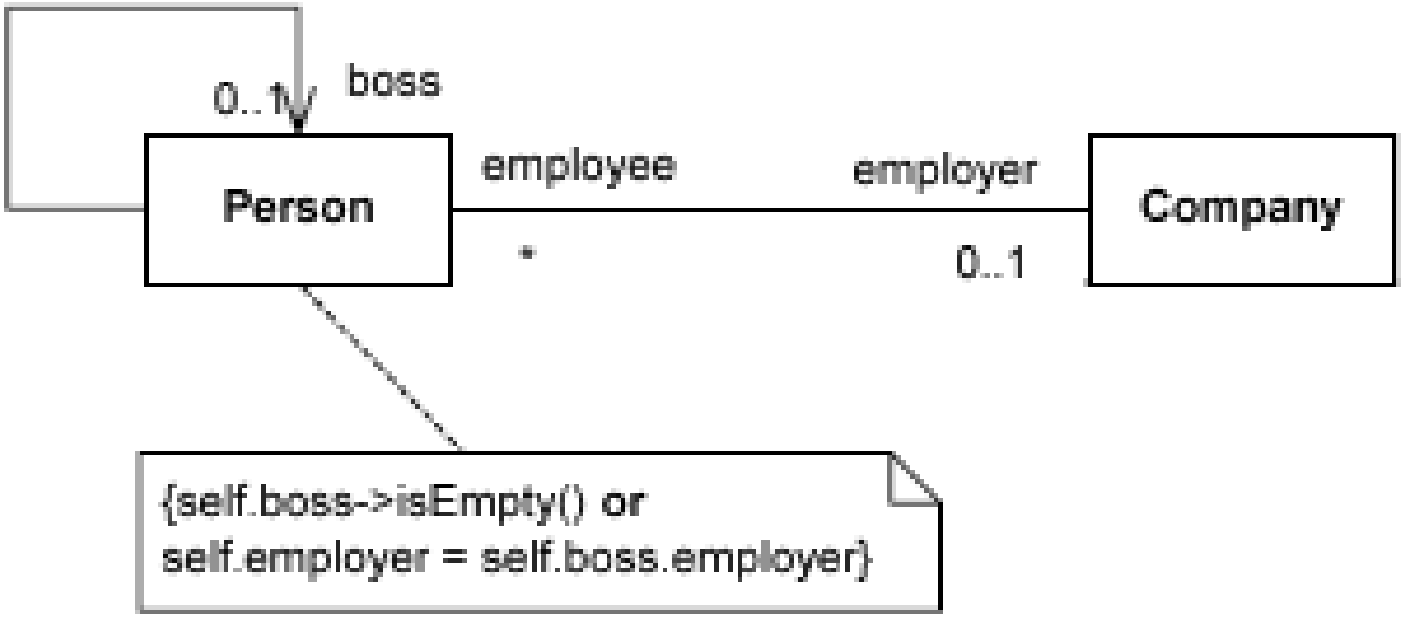


Figure 33 - Constraint in a note symbol

Restrições

- Existem algumas restrições que já fazem parte do padrão
  - {xor} para relacionamentos
  - ...

Estereótipos

Estereótipos

- Representam uma classificação para a classe
  - Um “tipo de classe”
- A maioria das classes se encaixa em um estereótipo
- São definidos com <<nome do estereótipo>>

Estereótipos

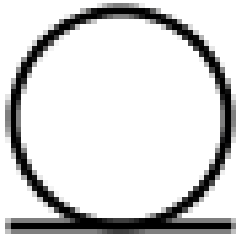
- Aplicados em toda UML!

Estereótipos comuns

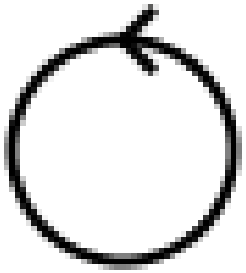
- Classe de fronteira
- Classe Entidade
- Classe de Controle
- Classe de Exceção
- Metaclasse
- Classe parametrizada
- Classe de utilidade

Estereótipos Básicos

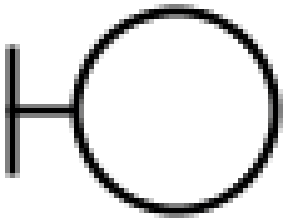
- Entidade



- Controle

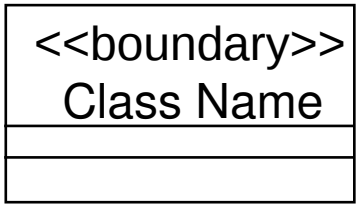


- Fronteira



Classes de Fronteira

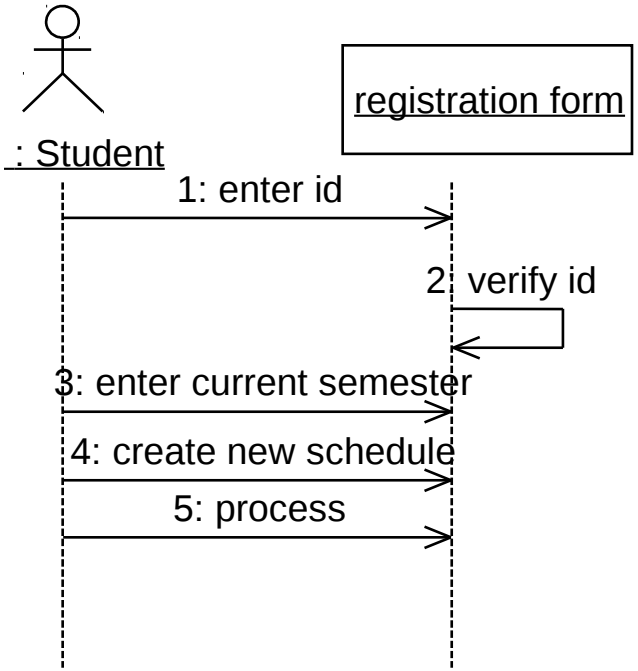
- Modela a comunicação entre o ambiente e o interior do sistema
- Típicas
  - Janelas
  - Protocolos
  - Interfaces
  - Sensores



Encontrando

- A informação trocada entre um ator e o sistema está contida em uma classe de fronteira
  - Analisar os Cenários

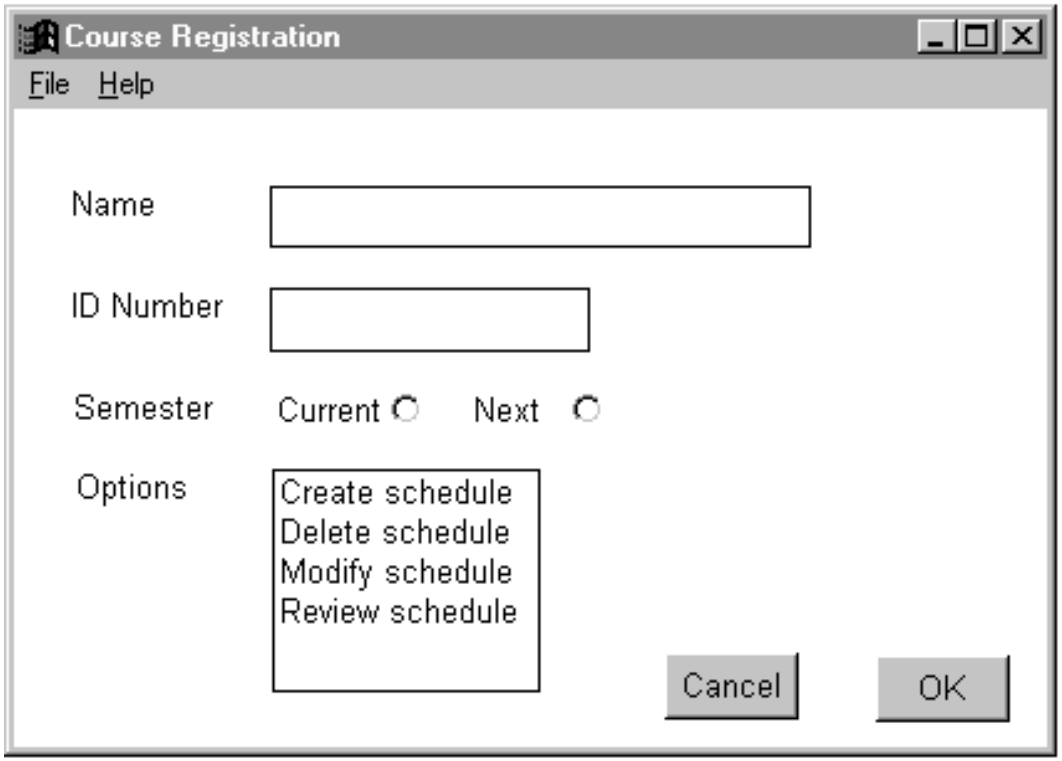
Fronteiras


: Student
registration form
1: enter id
2: verify id
3: enter current semester
4: create new schedule
5: process


<<boundary>>
RegistrationForm

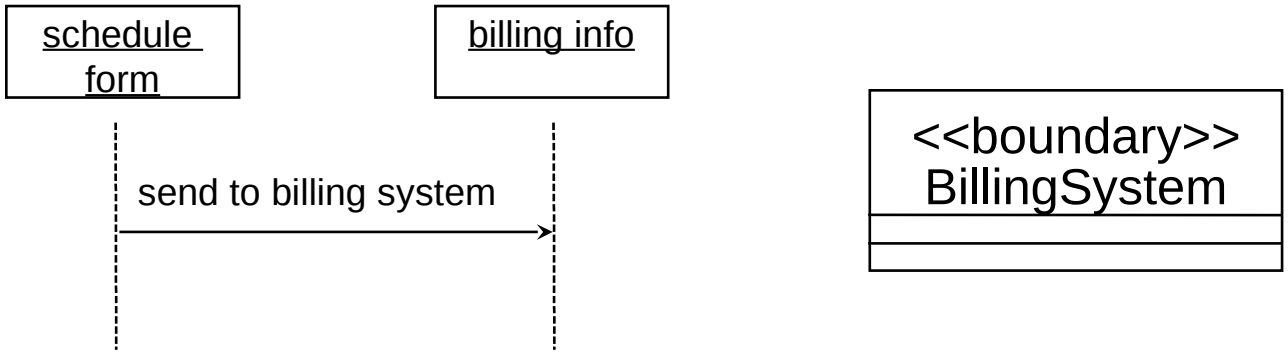
Janelas

- Classes de fronteira podem ser discutidas com o usuário por meio de desenhos ou protótipos



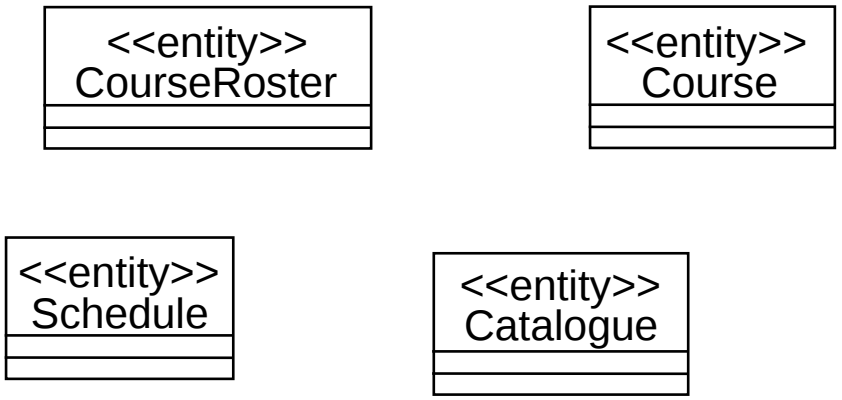
Interfaces com outros sistemas

- Informação é trocada com outro sistema
- Um protocolo é usado para a conversação



Classe Entidade

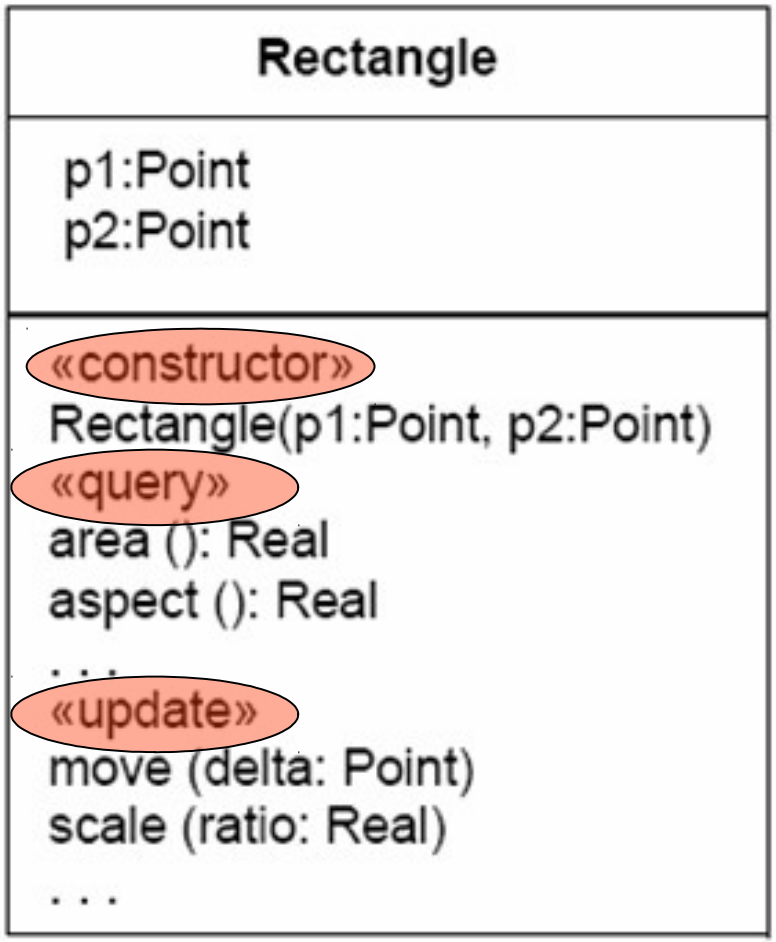
- Modela informação e comportamento associado que é persistente
  - Pode refletir um fenômeno da vida real
  - Pode ser necessária para o interior do sistema
  - Atores fornecem normalmente os valores do atributo



Classe de Controle

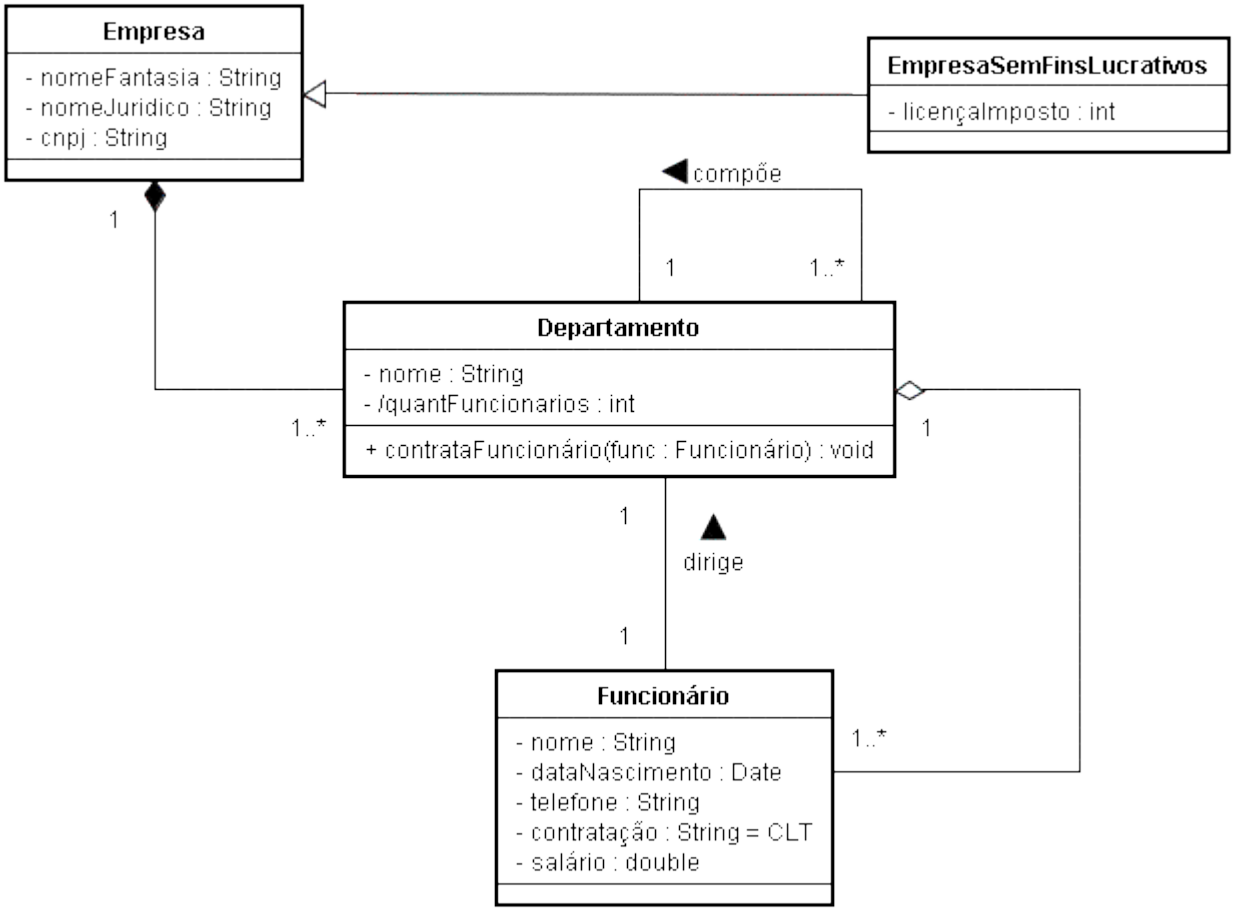
- Controla o comportamento de uma ou mais classes
  - Cria, inicia e apaga objetos
  - Dá seqüência e coordena a execução de objetos
  - Controla concorrência
  - É normalmente um objeto intangível

Estereótipos

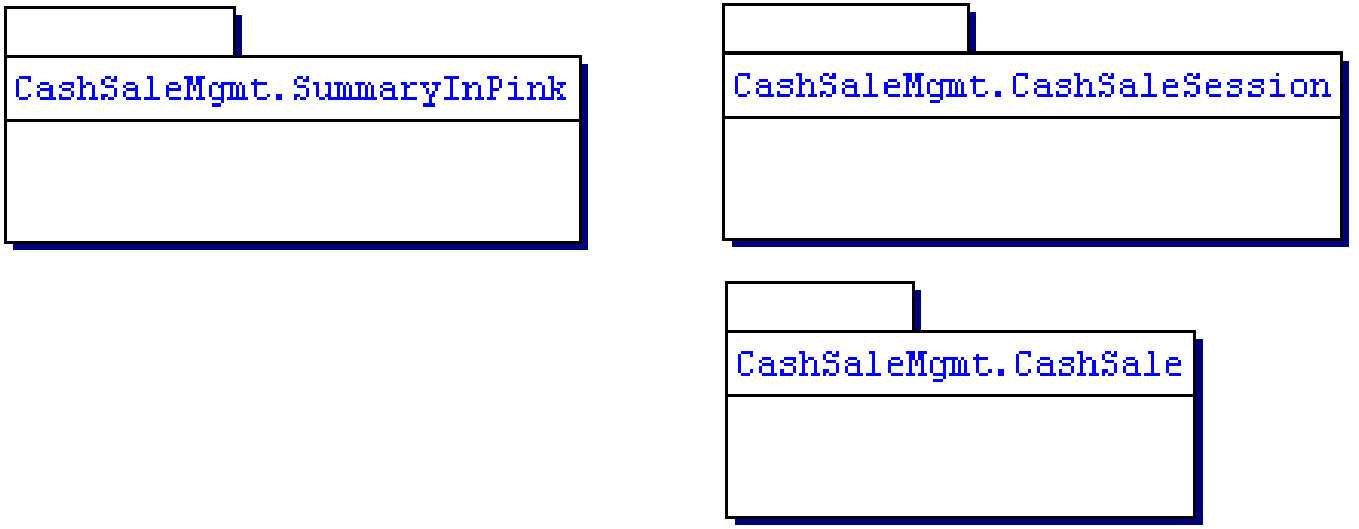


Exemplos

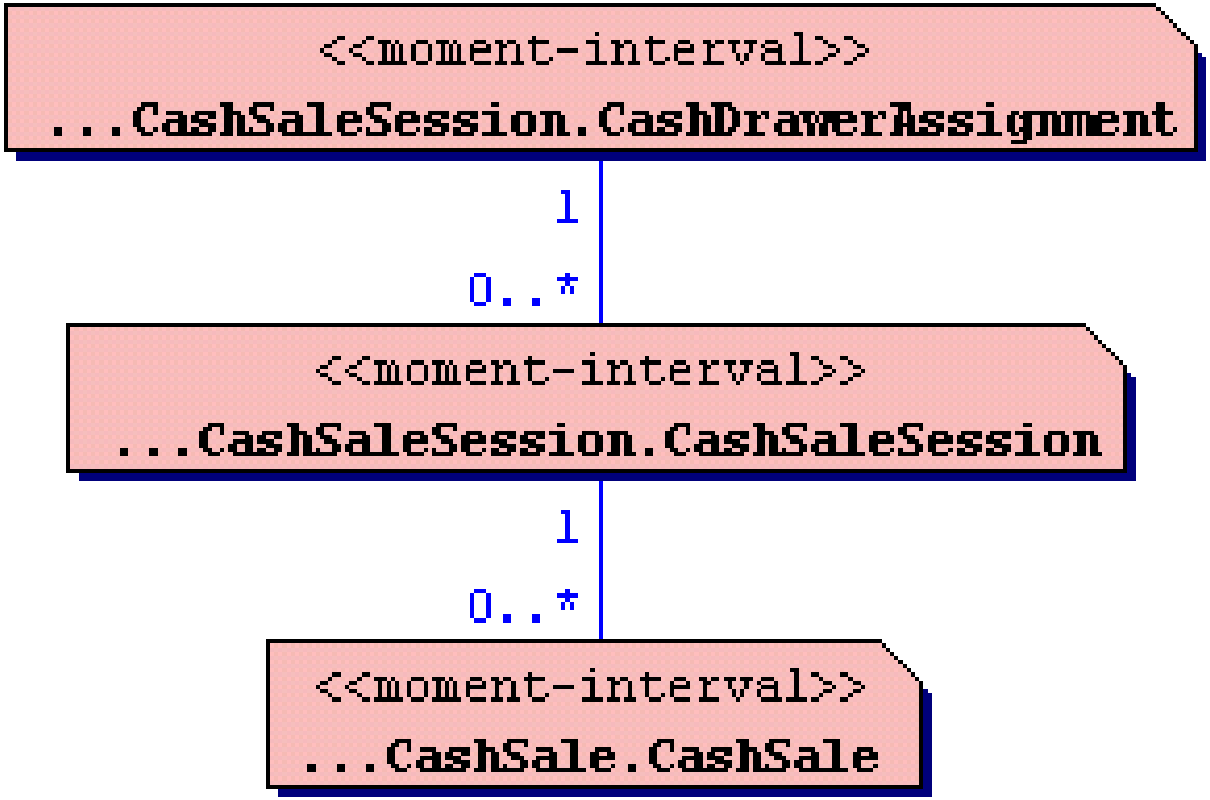
Exemplo

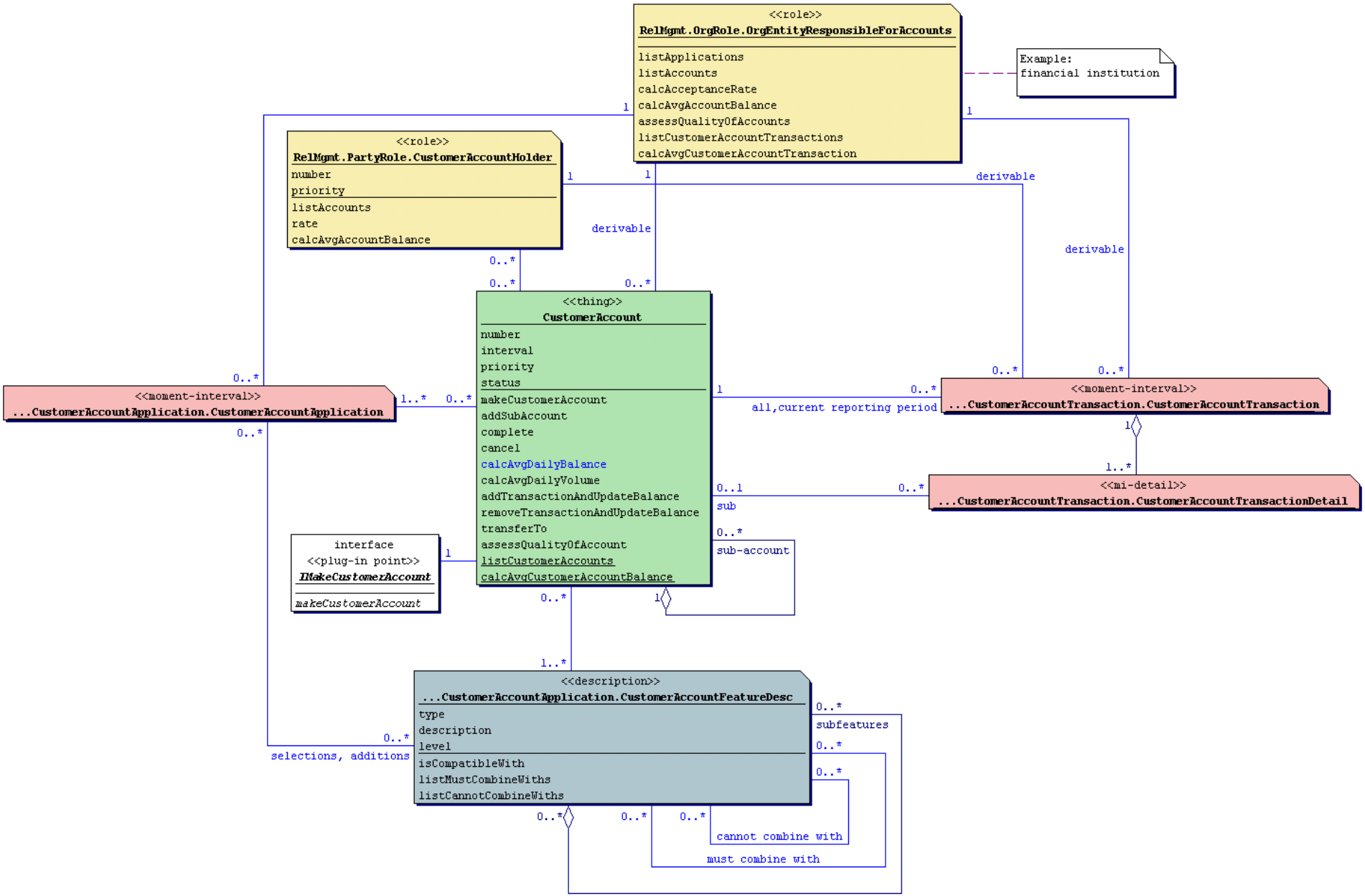

Empresa
- nomeFantasia : String
- nomeJuridico : String
- cnpj : String
EmpresaSemFinsLucrativos
- licençaImposto : int
compõe
1
1
1..*
Departamento
- nome : String
- /quantFuncionarios : int
+ contrataFuncionário(func : Funcionário) : void
1..*
1
1
dirige
1
Funcionário
- nome : String
- dataNascimento : Date
- telefone : String
- contratação : String = CLT
- salário : double
1..*

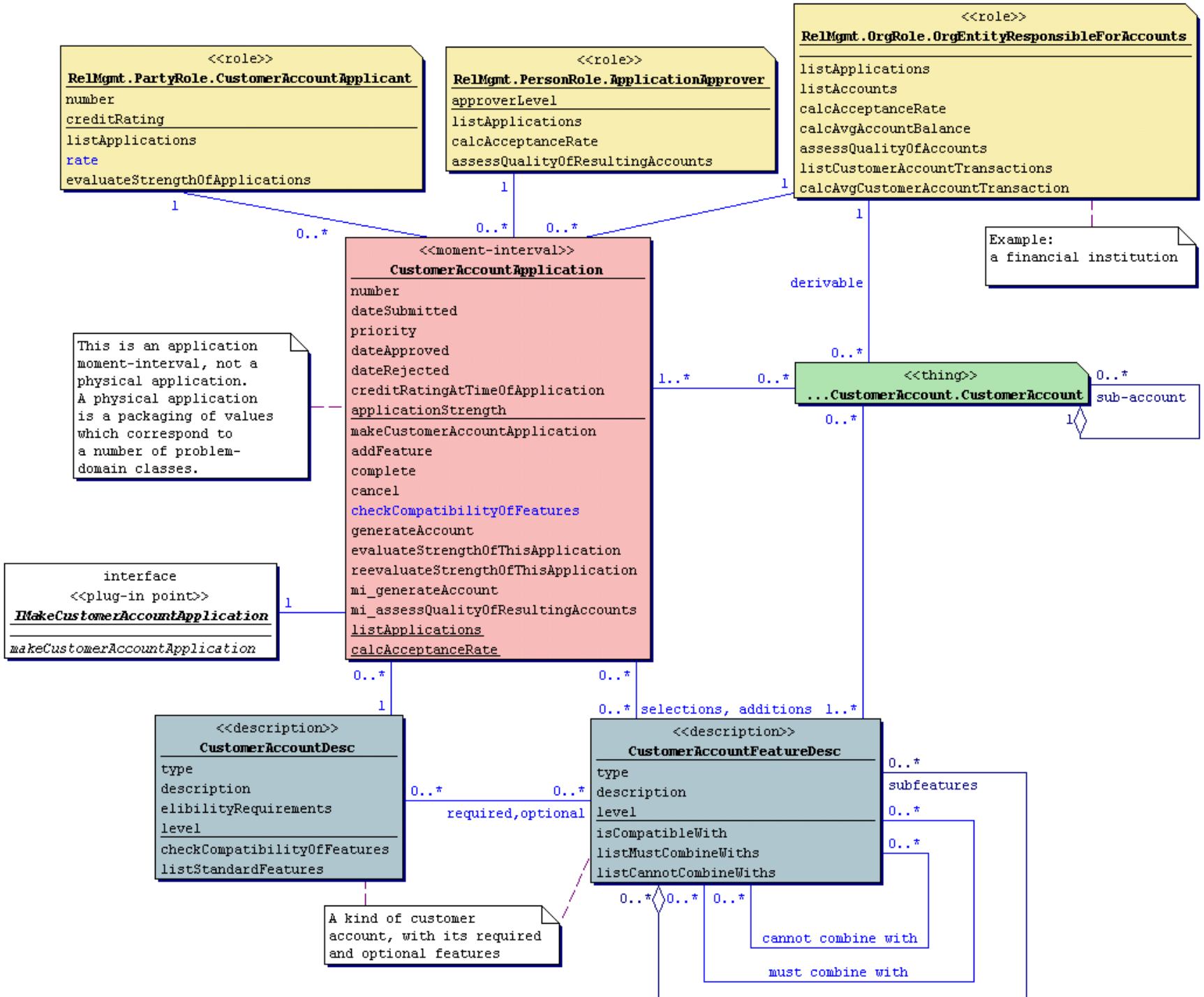
Exemplo: CashSaleMgmt Package

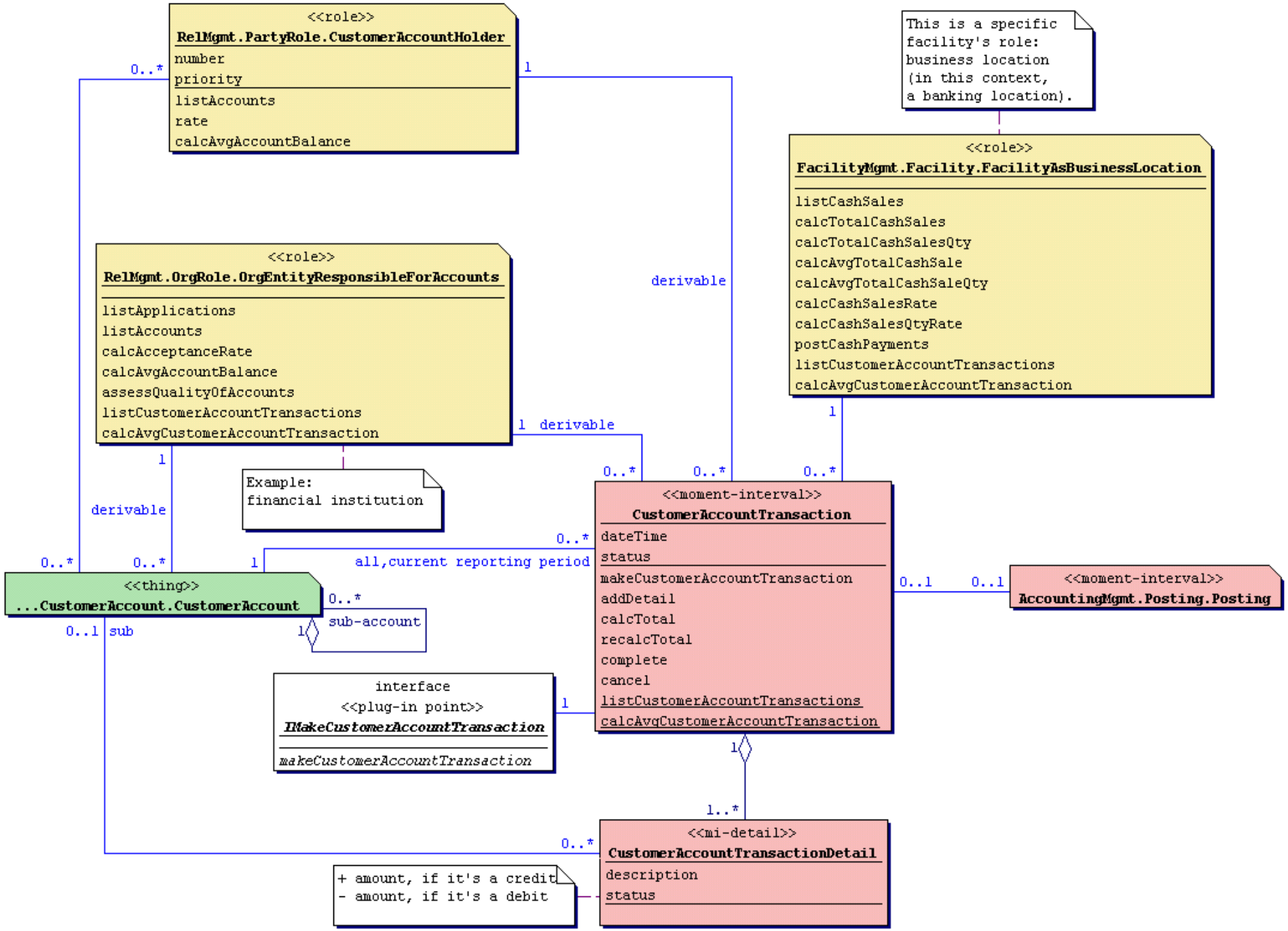


Sumário em Rosa




<<role>>
RelMgmt.OrgRole.OrgEntityResponsibleForAccounts
listApplications
listAccounts
calcAcceptanceRate
calcAvgAccountBalance
assessQualityOfAccounts
listCustomerAccountTransactions
calcAvgCustomerAccountTransaction
Example:
financial institution
<<role>>
RelMgmt.PartyRole.CustomerAccountHolder
number
priority
listAccounts
rate
calcAvgAccountBalance
derivable
derivable
derivable
<<thing>>
CustomerAccount
number
interval
priority
status
makeCustomerAccount
addSubAccount
complete
cancel
calcAvgDailyBalance
calcAvgDailyVolume
addTransactionAndUpdateBalance
removeTransactionAndUpdateBalance
transferTo
assessQualityOfAccount
listCustomerAccounts
calcAvgCustomerAccountBalance
<<moment-interval>>
...CustomerAccountApplication.CustomerAccountApplication
<<moment-interval>>
...CustomerAccountTransaction.CustomerAccountTransaction
all,current reporting period
<<mi-detail>>
...CustomerAccountTransaction.CustomerAccountTransactionDetail
sub
sub-account
interface
<<plug-in point>>
IMakeCustomerAccount
makeCustomerAccount
<<description>>
...CustomerAccountApplication.CustomerAccountFeatureDesc
type
description
level
isCompatibleWith
listMustCombineWiths
listCannotCombineWiths
subfeatures
selections, additions
cannot combine with
must combine with


<<role>>
RelMgmt.PartyRole.CustomerAccountApplicant
number
creditRating
listApplications
rate
evaluateStrengthOfApplications
<<role>>
RelMgmt.PersonRole.ApplicationApprover
approverLevel
listApplications
calcAcceptanceRate
assessQualityOfResultingAccounts
<<role>>
RelMgmt.OrgRole.OrgEntityResponsibleForAccounts
listApplications
listAccounts
calcAcceptanceRate
calcAvgAccountBalance
assessQualityOfAccounts
listCustomerAccountTransactions
calcAvgCustomerAccountTransaction
Example:
a financial institution
<<moment-interval>>
CustomerAccountApplication
number
dateSubmitted
priority
dateApproved
dateRejected
creditRatingAtTimeOfApplication
applicationStrength
makeCustomerAccountApplication
addFeature
complete
cancel
checkCompatibilityOfFeatures
generateAccount
evaluateStrengthOfThisApplication
reevaluateStrengthOfThisApplication
mi_generateAccount
mi_assessQualityOfResultingAccounts
listApplications
calcAcceptanceRate
This is an application moment-interval, not a physical application. A physical application is a packaging of values which correspond to a number of problem-domain classes.
interface
<<plug-in point>>
IMakeCustomerAccountApplication
makeCustomerAccountApplication
<<thing>>
...CustomerAccount.CustomerAccount
sub-account
derivable
selections, additions
<<description>>
CustomerAccountDesc
type
description
elibilityRequirements
level
checkCompatibilityOfFeatures
listStandardFeatures
required,optional
<<description>>
CustomerAccountFeatureDesc
type
description
level
isCompatibleWith
listMustCombineWiths
listCannotCombineWiths
subfeatures
cannot combine with
must combine with
A kind of customer account, with its required and optional features


<<role>>
RelMgmt.PartyRole.CustomerAccountHolder
number
priority
listAccounts
rate
calcAvgAccountBalance
This is a specific facility's role: business location (in this context, a banking location).
<<role>>
FacilityMgmt.Facility.FacilityAsBusinessLocation
listCashSales
calcTotalCashSales
calcTotalCashSalesQty
calcAvgTotalCashSale
calcAvgTotalCashSaleQty
calcCashSalesRate
calcCashSalesQtyRate
postCashPayments
listCustomerAccountTransactions
calcAvgCustomerAccountTransaction
<<role>>
RelMgmt.OrgRole.OrgEntityResponsibleForAccounts
listApplications
listAccounts
calcAcceptanceRate
calcAvgAccountBalance
assessQualityOfAccounts
listCustomerAccountTransactions
calcAvgCustomerAccountTransaction
Example: financial institution
<<moment-interval>>
CustomerAccountTransaction
dateTime
status
makeCustomerAccountTransaction
addDetail
calcTotal
recalcTotal
complete
cancel
listCustomerAccountTransactions
calcAvgCustomerAccountTransaction
<<thing>>
...CustomerAccount.CustomerAccount
<<moment-interval>>
AccountingMgmt.Posting.Posting
interface
<<plug-in point>>
IMakeCustomerAccountTransaction
makeCustomerAccountTransaction
<<mi-detail>>
CustomerAccountTransactionDetail
description
status
+ amount, if it's a credit
- amount, if it's a debit
0..*
1
derivable
all,current reporting period
sub
sub-account
0..1
1..*