UNIFESP
UNIVERSIDADE FEDERAL DE SÃO PAULO
1933

Bacharelado em Ciência da Computação

SC-301 – Programação Orientada a Objetos II

Introdução à UML

Parte 02

Prof. Dr. Fábio Fagundes Silveira

fsilveira@unifesp.br

http://fabiosilveira.net

UNIFESP – Universidade Federal de São Paulo

março/2010

Créditos

- Grade parte dos slides seguintes foram preparados/elaborados pelos Professores:
  - Prof. Dr. Jaelson Freire Brelaz de Castro; e
  - Prof. Dr. Ulrich Schiel

Bibliografia

- *The Unified Modelling Language - User Guide* (G. Booch, J. Rumbaugh, I. Jacobson) - Addison Wesley
- *The Unified Modelling Language - Reference Manual* (J. Rumbaugh, I. jacobson, G. Booch) - Addison Wesley
- *The Unified Software Development Process* (I. Jacobson, G. Booch, R. Rumbaugh) - Addison Wesley
- *UML Distilled* (Martin Fowler) - Addison Wesley

Conteúdo

- Introdução à UML
- Conceitos Gerais
- Apresentação dos diagramas da UML
- Novos diagramas da UML 2.0

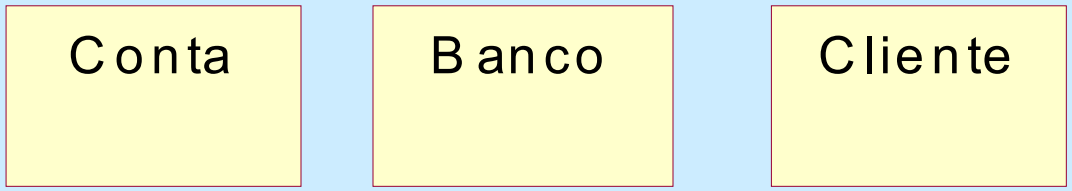
Diagramas de Classes

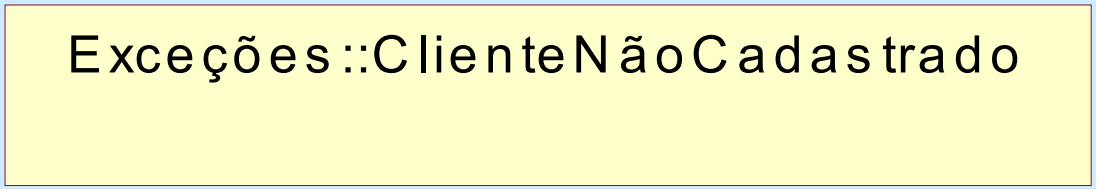
Sobre Classes

- Classes são o elemento mais importante de qualquer sistema orientado a objetos
- Uma classe é uma descrição de um conjunto de objetos com os mesmos **atributos**, **relacionamentos**, **operações** e **semântica**
- Classes são usadas para capturar o **vocabulário** de um sistema
- Classes são abstrações de elementos do domínio do problema, como “Cliente”, “Banco”, “Conta”

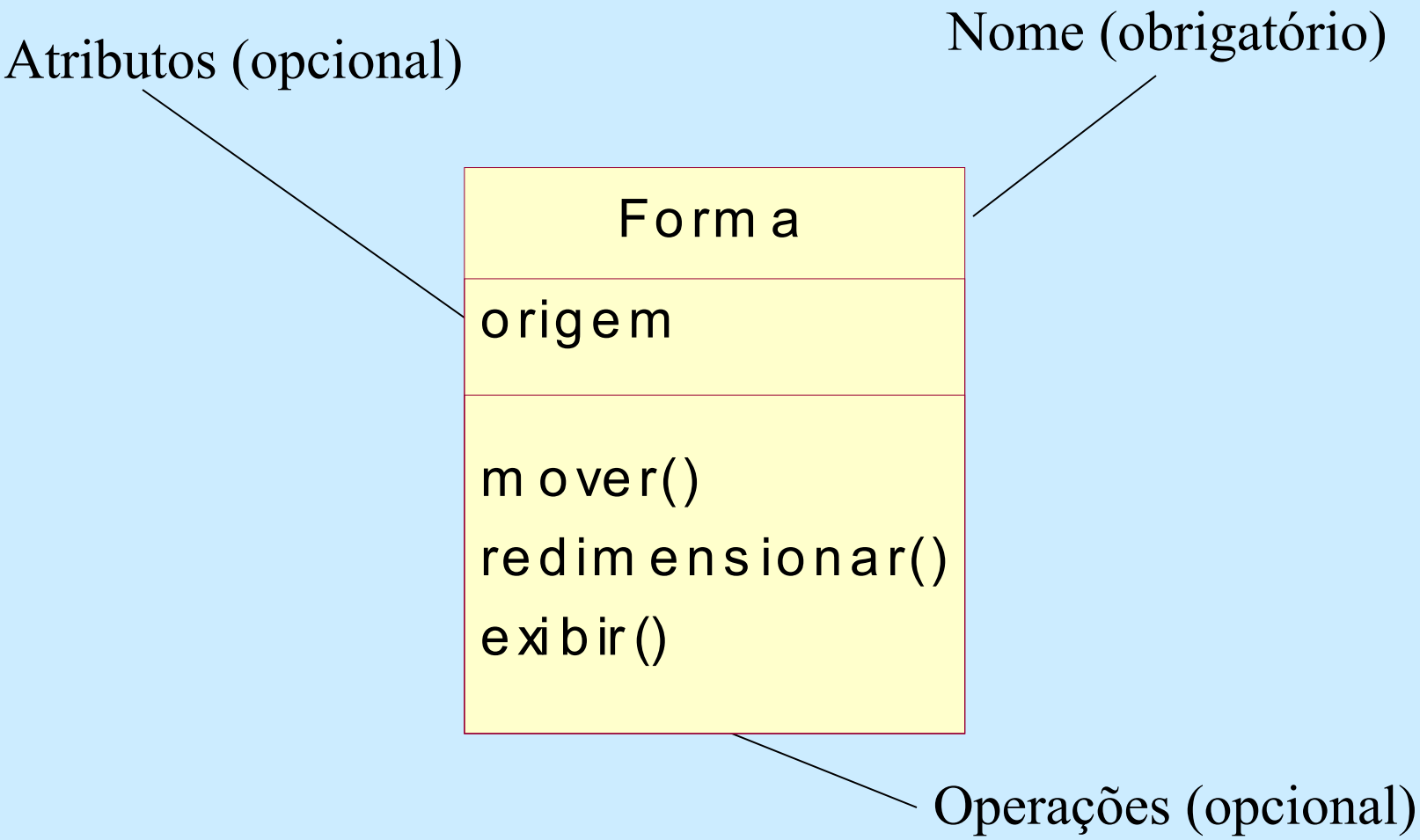
Nomes

- Toda classe deve ter um nome que a distinga das outras classes
- Um nome pode ser simples (apenas o nome), ou pode ser precedido pelo nome do pacote em que a classe está contida





Notação básica

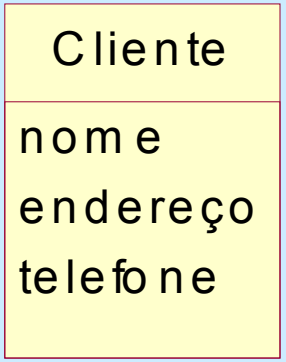


Atributos

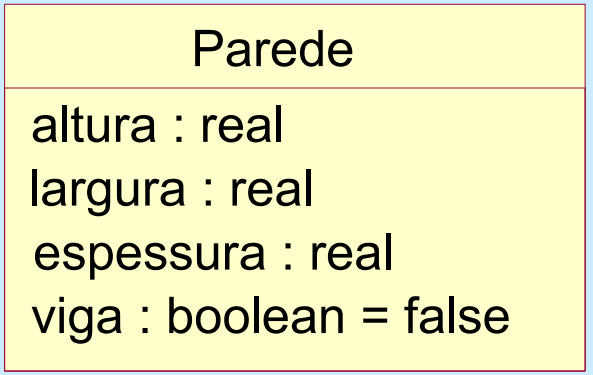
- Um atributo representa alguma propriedade do que está sendo modelado, que é compartilhada por todos os objetos da classe
- Os atributos descrevem os dados contidos nas instâncias de uma classe
- Em um momento dado, um objeto de uma classe conterá valores para todos os atributos descritos na sua classe

Atributos - Notação

- Atributos podem ser identificados apenas com nomes



- Atributos podem ter seus tipos (ou classes) especificados e terem valores padrão definidos

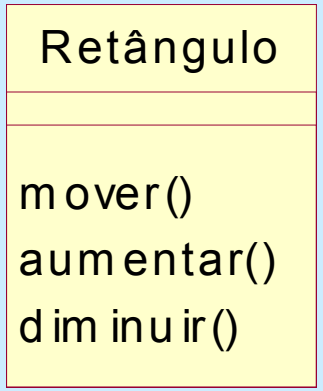


Operações

- Uma operação é uma abstração de alguma coisa que se pode fazer com um objeto e que é compartilhada por todos os objetos da classe
- Um classe pode ter qualquer número de operações, inclusive nenhuma
- Operações são o meio de alterar os valores dos atributos

Operações - Notação

- Como para os atributos, você pode especificar uma operação apenas com seu nome



- Você pode também especificar a **assinatura** da operação: seus parâmetros, o tipo desses parâmetros e o tipo de retorno

Visibilidade

- Você pode usar marcações de acesso para especificar o tipo de acesso permitido aos atributos e operações
- Classificador pode ser classes, interfaces, componentes, nós, use cases, subsistemas

\+ público: todos os classificadores podem usar

\# protegido: qualquer descendente do classificador poderá usar

\- privado: somente o próprio classificador poderá usar

Relacionamentos

- Poucas classes vivem sozinhas
- Tipos de relacionamentos especialmente importantes na modelagem orientada a objetos:
  - Associações
  - Agregação
  - Composição
  - Dependências
  - Generalizações
  - Realização

Relacionamentos

Os relacionamentos ligam as classes/objetos entre si criando relações lógicas entre estas entidades. Os relacionamentos podem ser dos seguintes tipos:

- **Associação** - especifica que objetos de um elemento (classe) estão conectados a objetos de outros elementos

Relacionamentos

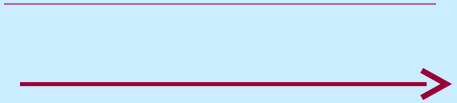
- **Agregação** - relacionamento fraco do tipo “é parte de”. É um tipo especial de associação
- **Composição** - relacionamento forte do tipo “é parte de”. A composição entre um elemento (o “todo”) e outros elementos (“as partes”) indica que as partes só existem em função do “todo”.

Relacionamentos

- **Dependência** - relacionamento de uso, no qual uma mudança na especificação de um elemento pode alterar a especificação do elemento dependente
- **Generalização (herança)** - relacionamento entre descrições mais gerais e descrições mais específicas, com mais detalhes sobre alguns dos elementos gerais
- **Realização** - relacionamento entre uma interface e o elemento que a implementa

Relacionamentos - Notação

Associação
Sem/com navegação



Dependência

Agregação

Generalização

Composição

Realização

Dependência

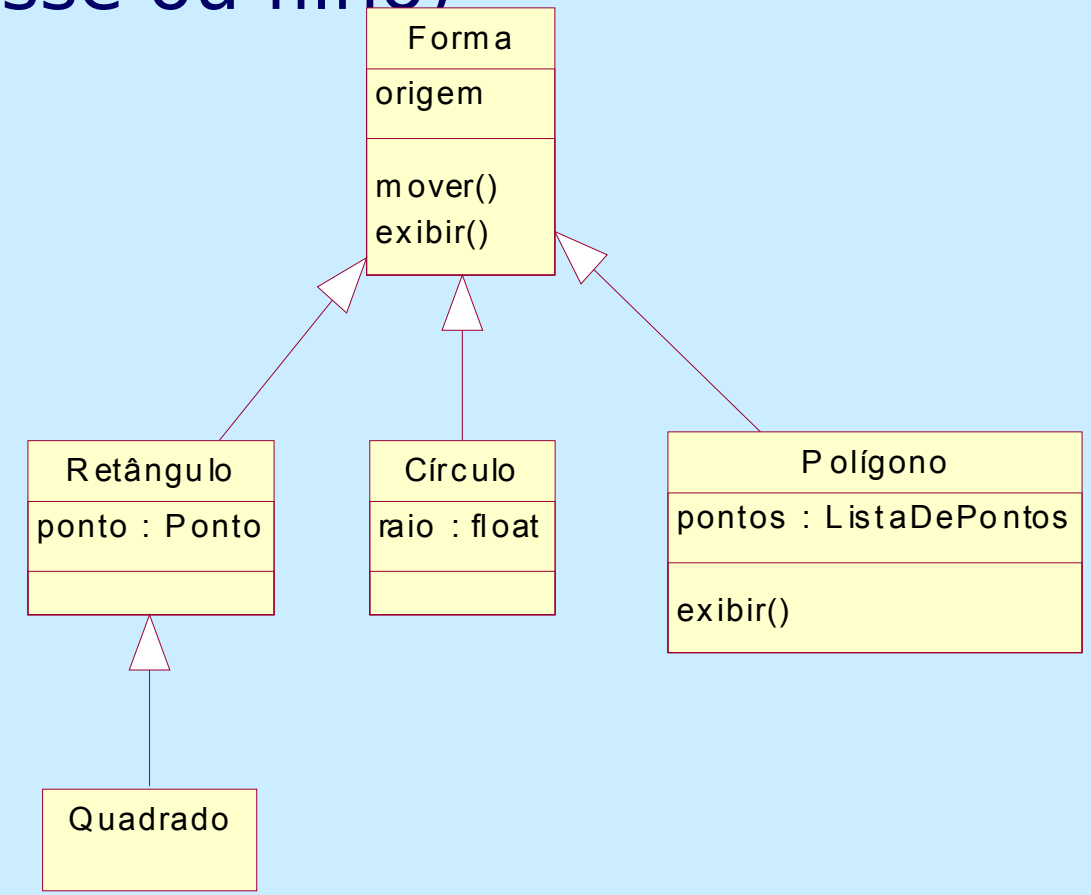
- Dependências são relações de **uso**
- Uma dependência indica que mudanças em um elemento (o “servidor”) podem afetar outro elemento (o “cliente”)
- Uma dependência entre classes indica que os objetos de uma classe **usam** serviços dos objetos de outra classe

Dependência – alguns tipos

- derive – a fonte é computada a partir do destino
- friend – a fonte tem acesso privilegiado ao destino
- instanceOf – o objeto fonte é instância da classe destino
- powertype – o destino é composto por subconjuntos da fonte
- Entre pacotes:
  - access e import
- Entre use-cases:
  - extend e include
- Entre objetos:
  - become, call e copy

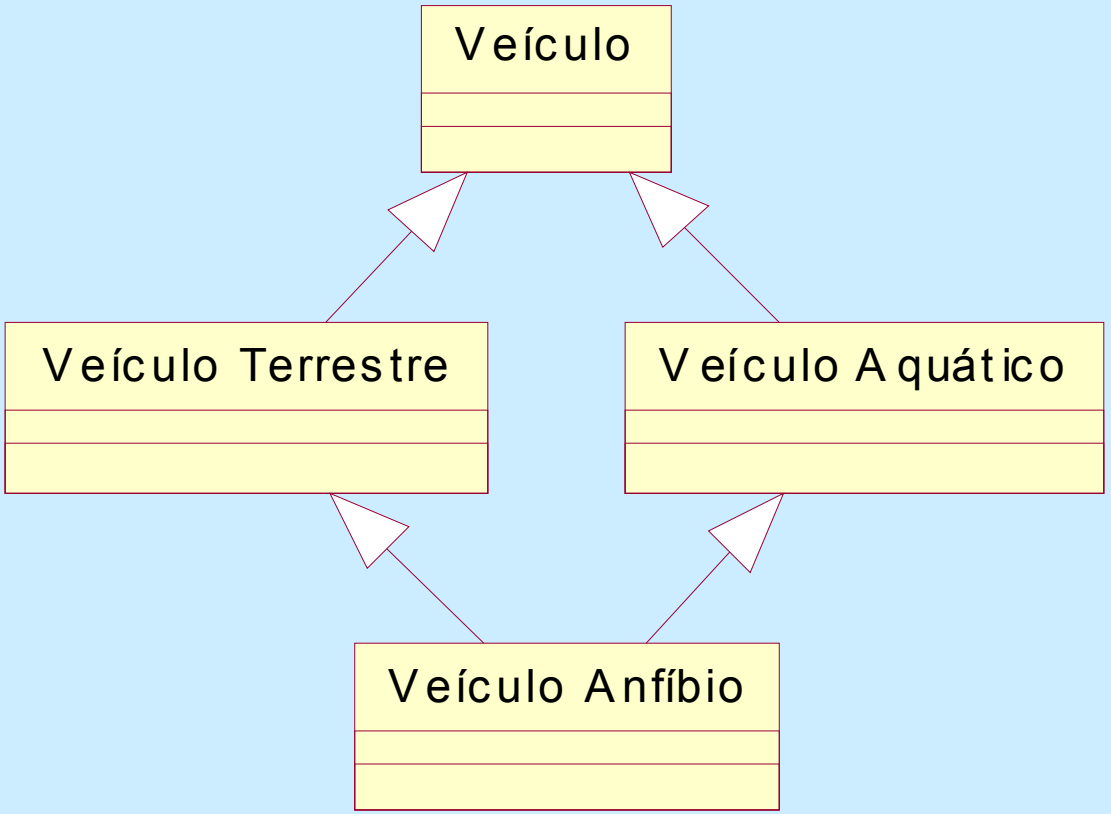
Generalização

- Relacionamento entre um elemento mais geral (chamado de superclasse ou pai) e um mais específico (chamado de subclasse ou filho)



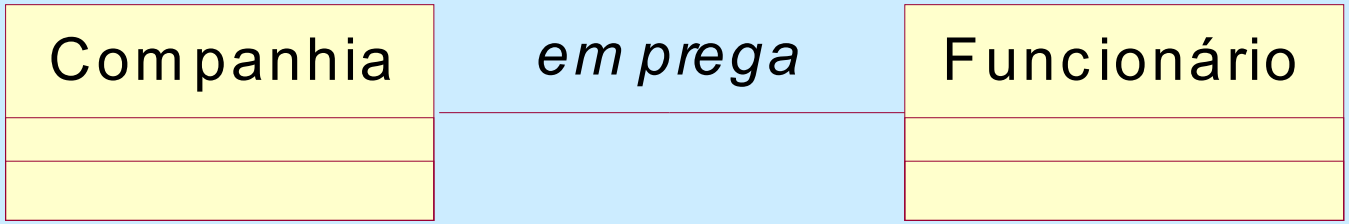
Herança Múltipla

- Ocorrem múltiplas superclasses para uma mesma subclasse



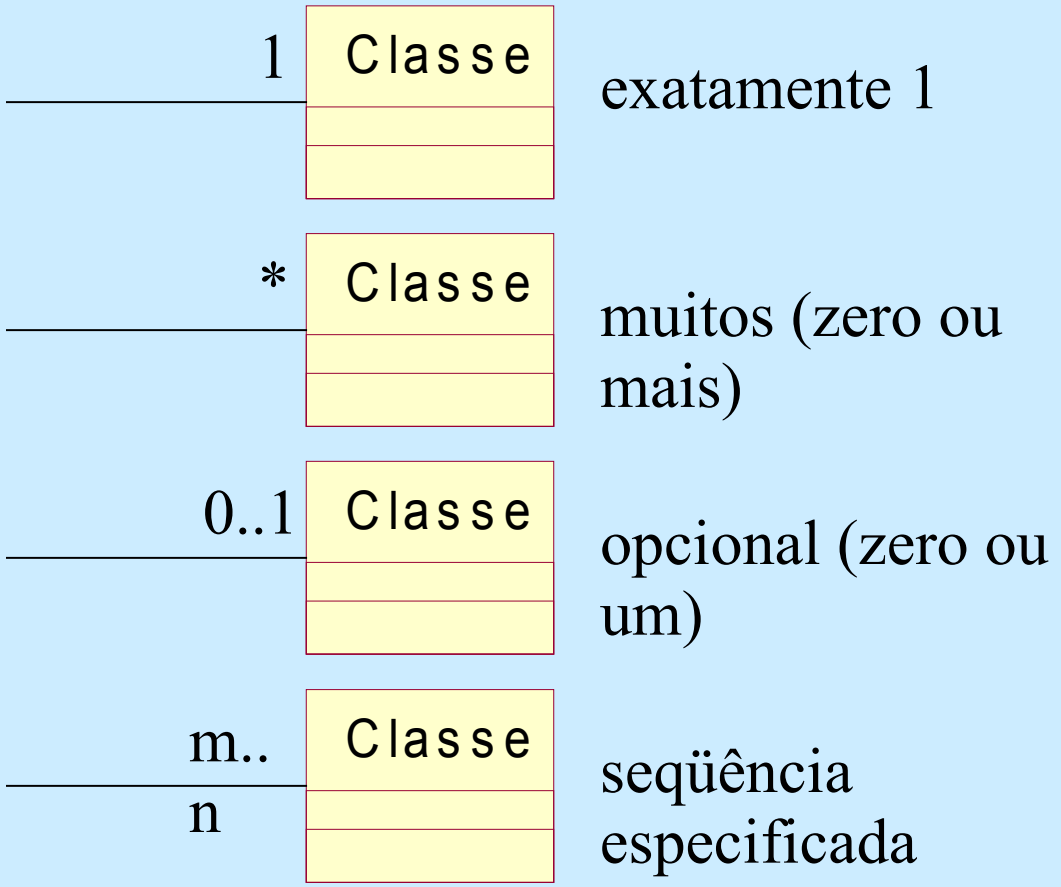
Associação

- A associação é um relacionamento estrutural que especifica que objetos de um elemento estão conectados a objetos de outro elemento



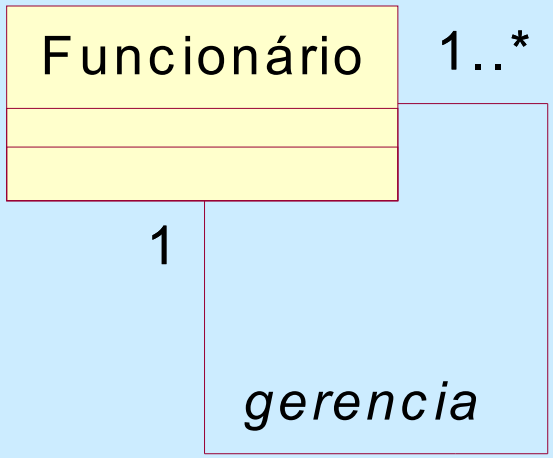
Multiplicidade

- É a cardinalidade de uma associação



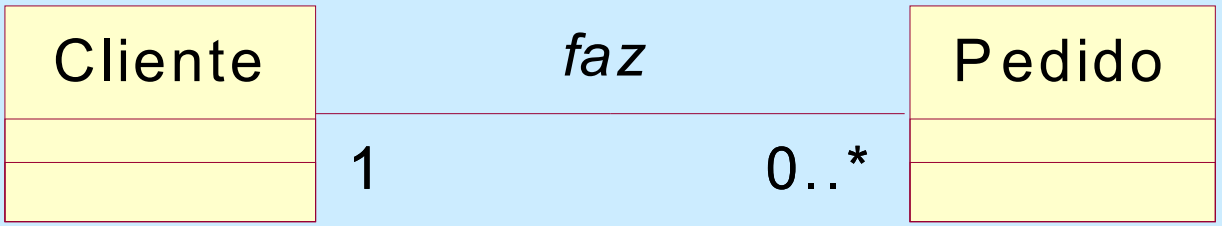
Associação Unária

- Quando há um relacionamento de uma classe para consigo própria



Associação Binária

- Quando há duas classes envolvidas na associação de forma direta de uma para a outra

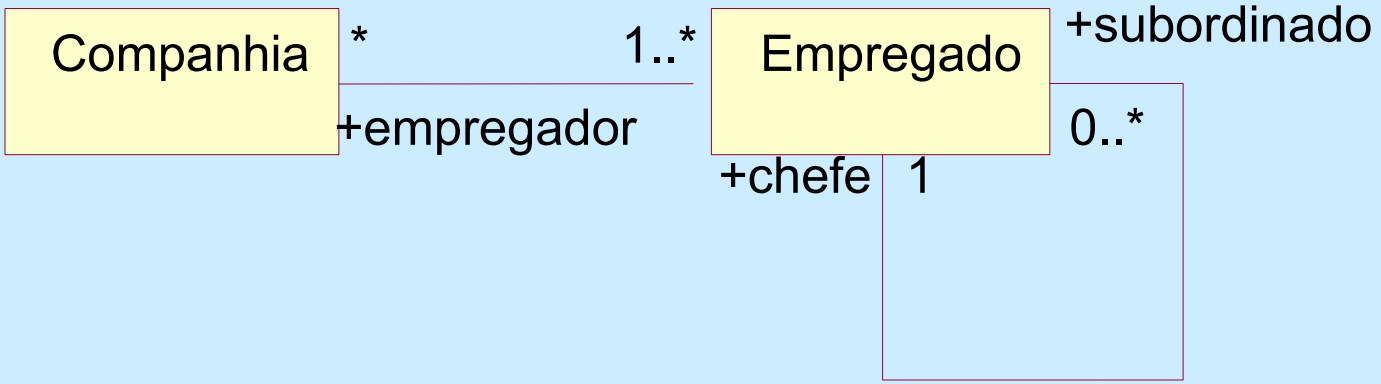


Associação: Navegabilidade

- Em geral a navegação entre as classes de uma associação é bi-direcional
- Porém é possível limitá-la a apenas uma direção

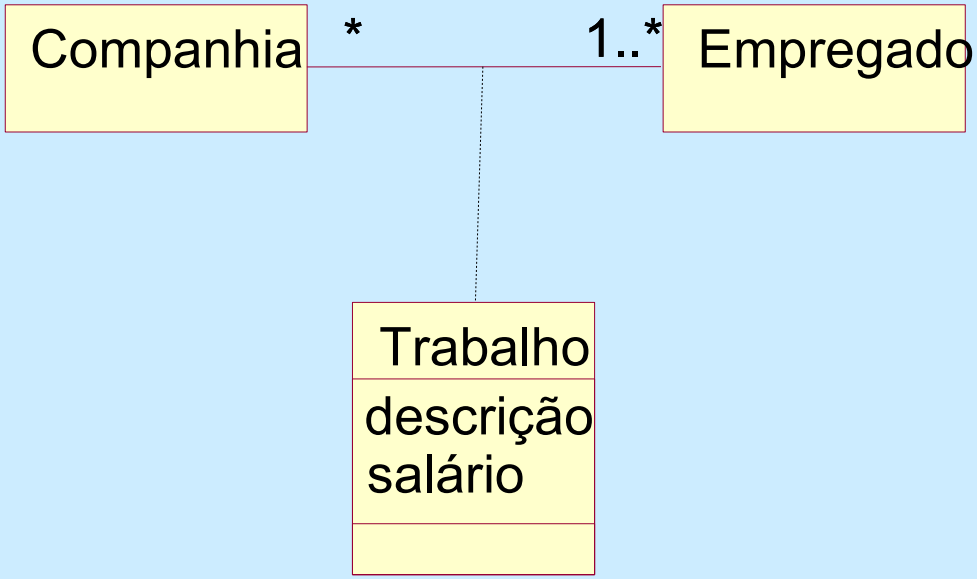
Associação: papéis

- Papéis: um dos lados da associação
- Nomes de papéis são necessários para associação entre dois objetos da mesma classe



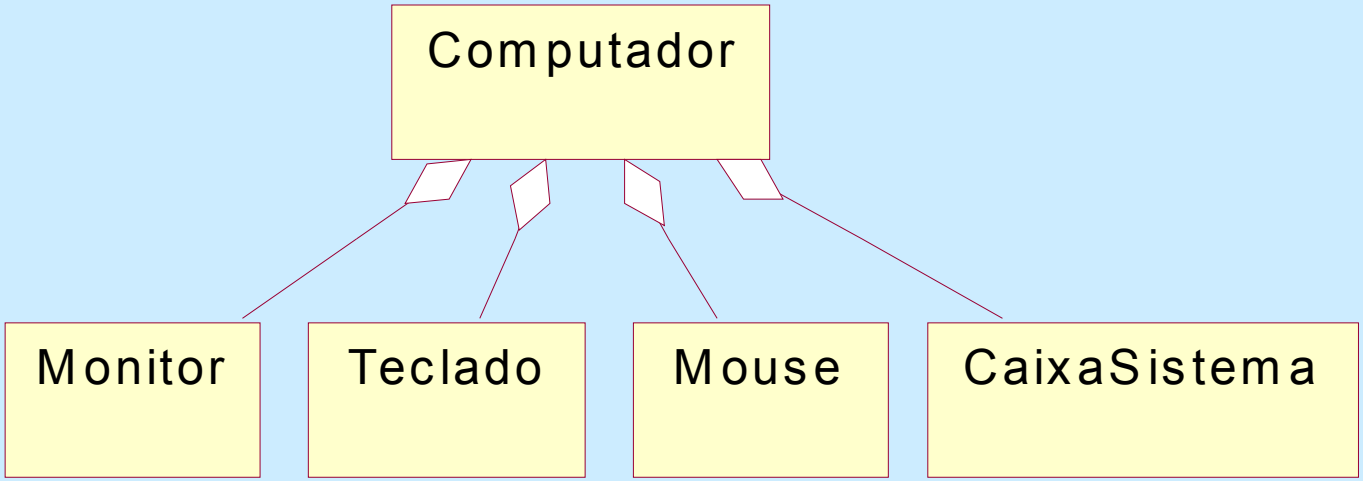
Associação com Atributos

- Modela as propriedades associadas com uma associação
- As propriedades devem ser representadas por uma classe



Agregação

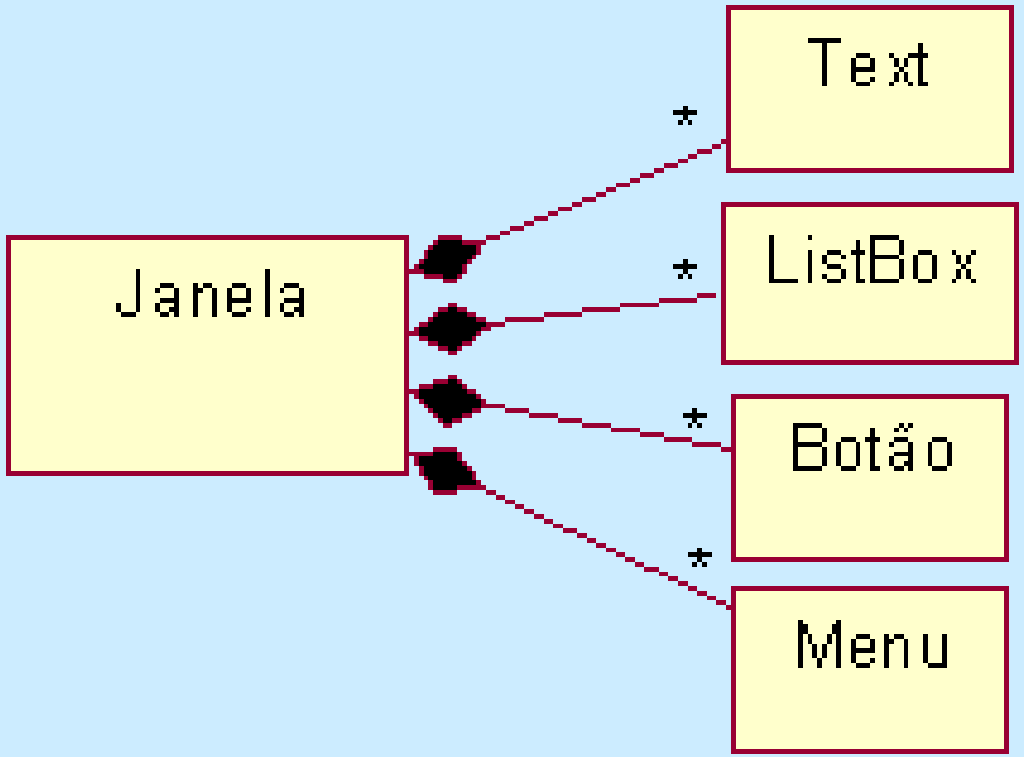
- Uma forma especial de associação entre o todo e suas partes, no qual o todo é composto de partes
- Não impõe que a vida das “Partes”’ esteja relacionado com a vida do “Todo”



Composição

- Uma forma mais forte de agregação
- Há uma coincidência da vida das partes
- Uma vez criada a parte ela irá viver e morrer com ele
- O “Todo” é responsável pelo gerenciamento da criação e destruição das partes

Composição



Interfaces

- Uma interface é um conjunto de operações usado para especificar um serviço de uma classe ou componente
- Diferentemente das classes, as interfaces não especificam nenhuma estrutura
- Interfaces **não podem conter atributos**

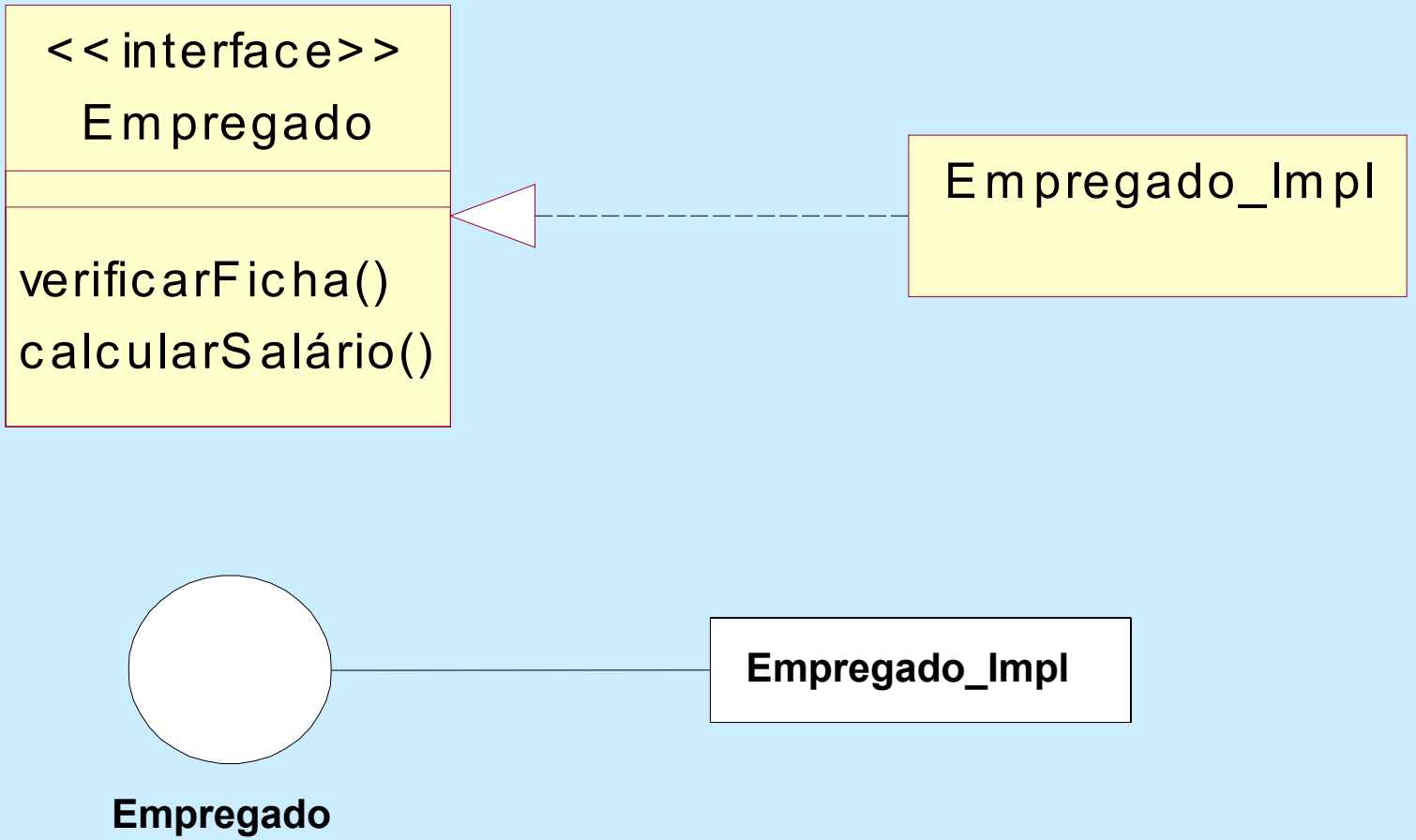
Interfaces

- Com as interfaces, é possível se concentrar apenas nos **serviços** oferecidos por classes ou componentes
- O uso de interfaces é uma maneira elegante e poderosa de **isolar** a especificação da implementação
- Uma interface especifica o **contrato** para uma classe ou componente, sem definir como ele será implementado

Interfaces e Realização

- **Realização** é uma relação pela qual um elemento especifica o contrato que outro elemento deve implementar
- A realização é um relacionamento entre uma especificação e sua implementação
- É um relacionamento semântico entre classificadores no qual um classificador especifica um contrato que outro classificador garante cumprir

Realização - Notação



Diagramas de Objetos

Diagramas de objetos

- Os diagramas de objetos mostram uma “fotografia” de um sistema OO em execução
- São mostrados os objetos, com os **valores** de seus atributos e as ligações (**links**) entre eles
- Os diagramas de objetos são úteis para a modelagem de estruturas de dados complexas, por exemplo

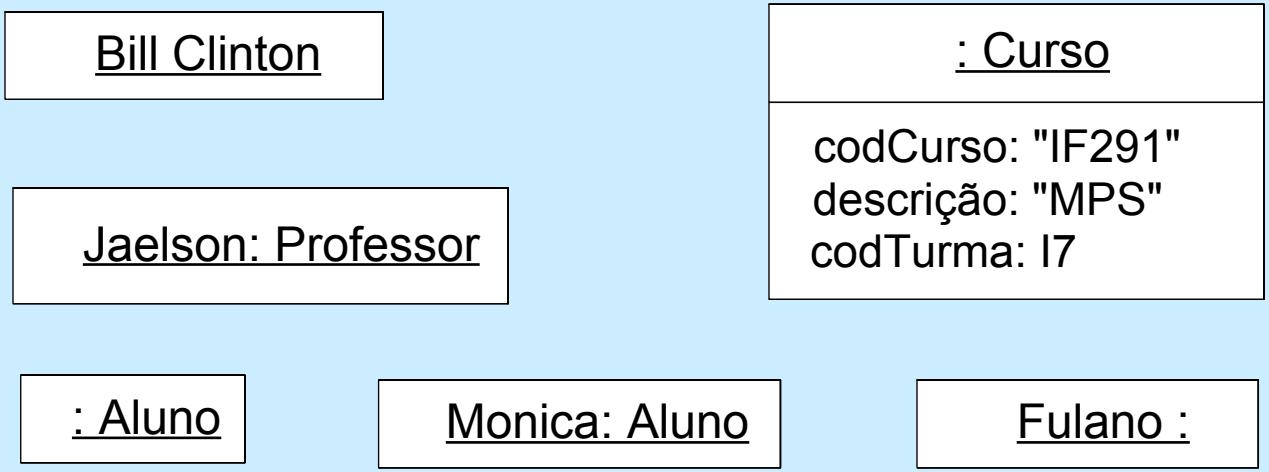
Diagramas de objetos

- É comum haver centenas ou milhares de objetos em um sistema em execução, a maioria deles anônimos
- Um diagrama de objetos mostra apenas uma **parte** dos objetos no sistema
- Um diagrama de objetos não mostra a evolução do sistema com o tempo

Diagramas de objetos

- Diagramas de objetos são **estáticos**
- Para mostrar o **comportamento** de um objeto, use **diagramas de comunicação**, **diagramas de seqüência**, ou **diagramas de máquina de estados**
- É comum colocar um **diagrama de classes** junto com um diagrama de objetos, para facilitar a identificação dos objetos

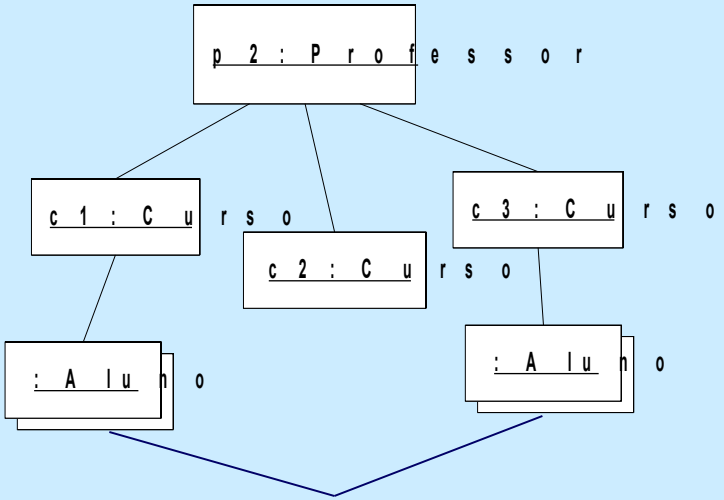
Objetos Simples



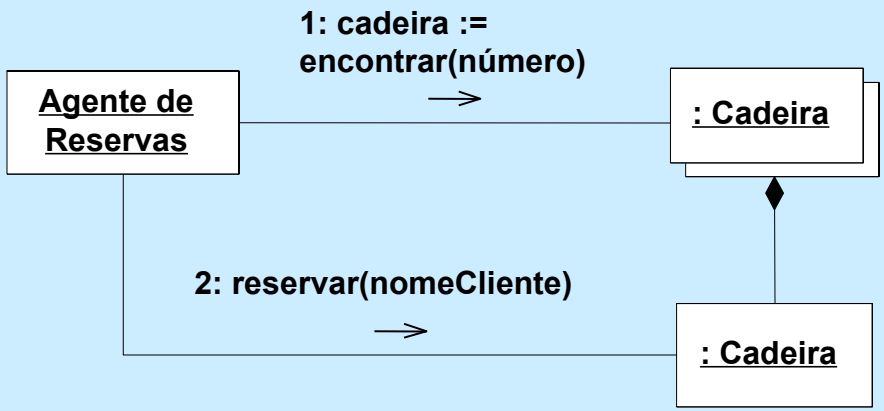
MultiObjects

- **Multiobjects** são conjuntos de objetos, com um número indeterminado de elementos
- São usados, por exemplo, em diagramas de comunicação para modelar uma mensagem enviada para vários objetos ao mesmo tempo

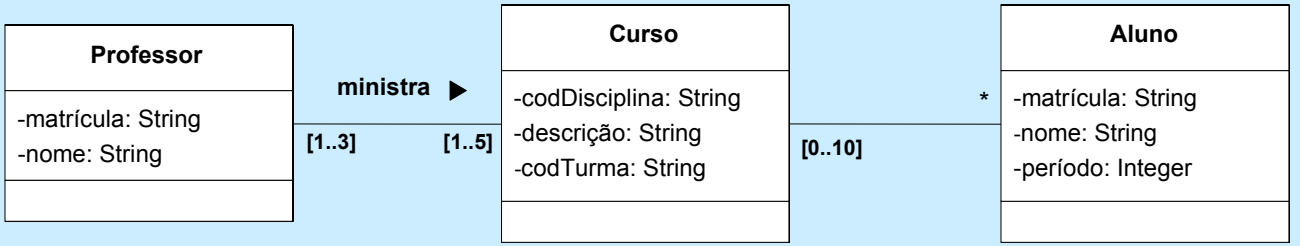
MultiObjects - Exemplos

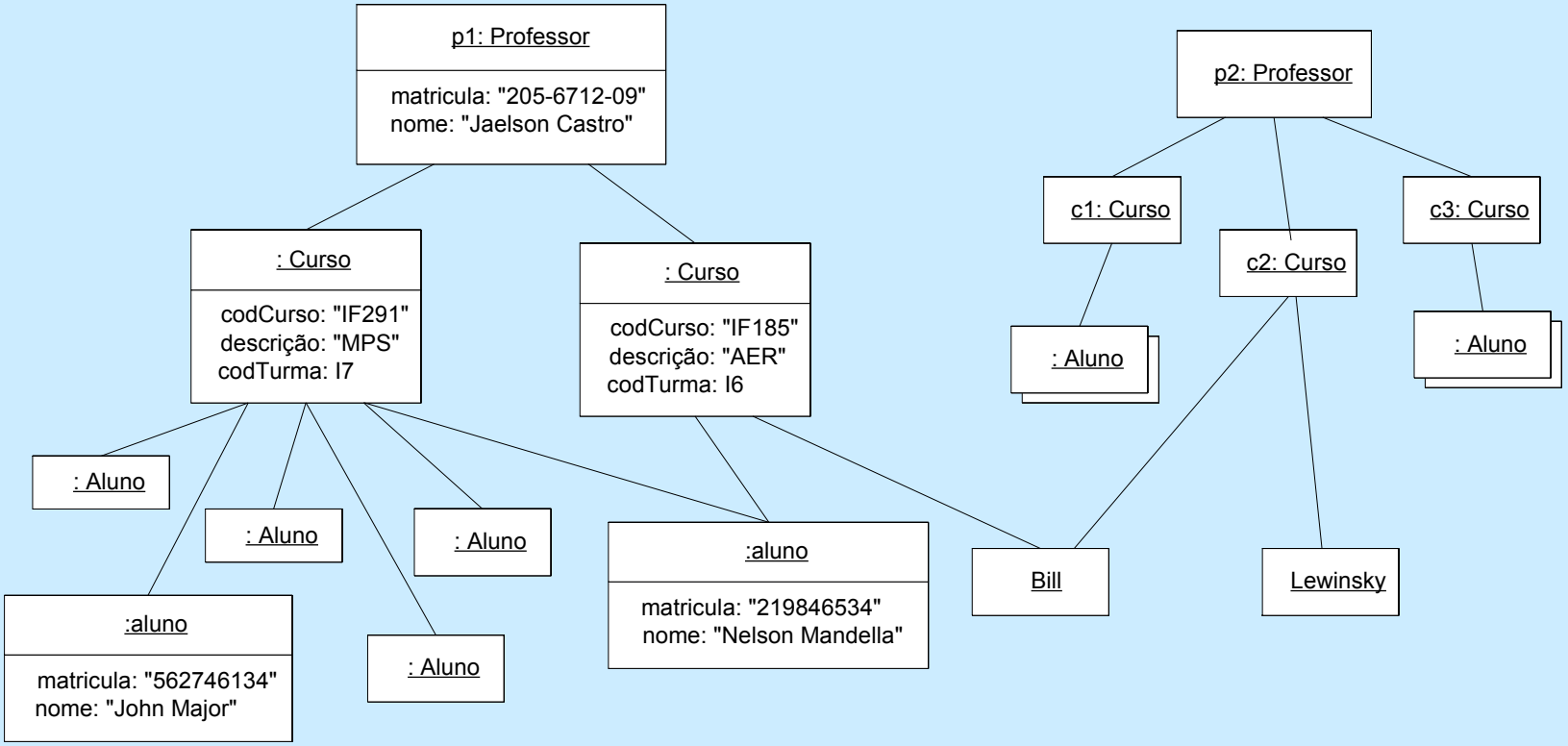


MultiObjects



Diagramas de Objetos





Diagramas de Seqüência e Comunicação

Seqüenciamento

- Quando um objeto envia uma mensagem para outro objeto, o objeto que recebe a mensagem pode enviar outras mensagens e assim por diante, formando uma **sequência** de mensagens
- O sequenciamento pode ser
  - procedural, com aninhamento (mensagens **síncronas**)
  - ou plano, sem aninhamento (mensagens **assíncronas**)

Diagramas de Seqüência

- Diagramas de Seqüência enfatizam a ordenação das mensagens trocadas entre os objetos
- Um **cenário** é uma seqüência específica de ações que ilustra um comportamento
- Diagramas de Seqüência podem modelar **apenas um cenário** ou um **conjunto de cenários**
- Diagramas de Seqüência podem mostrar decisões simples e iterações

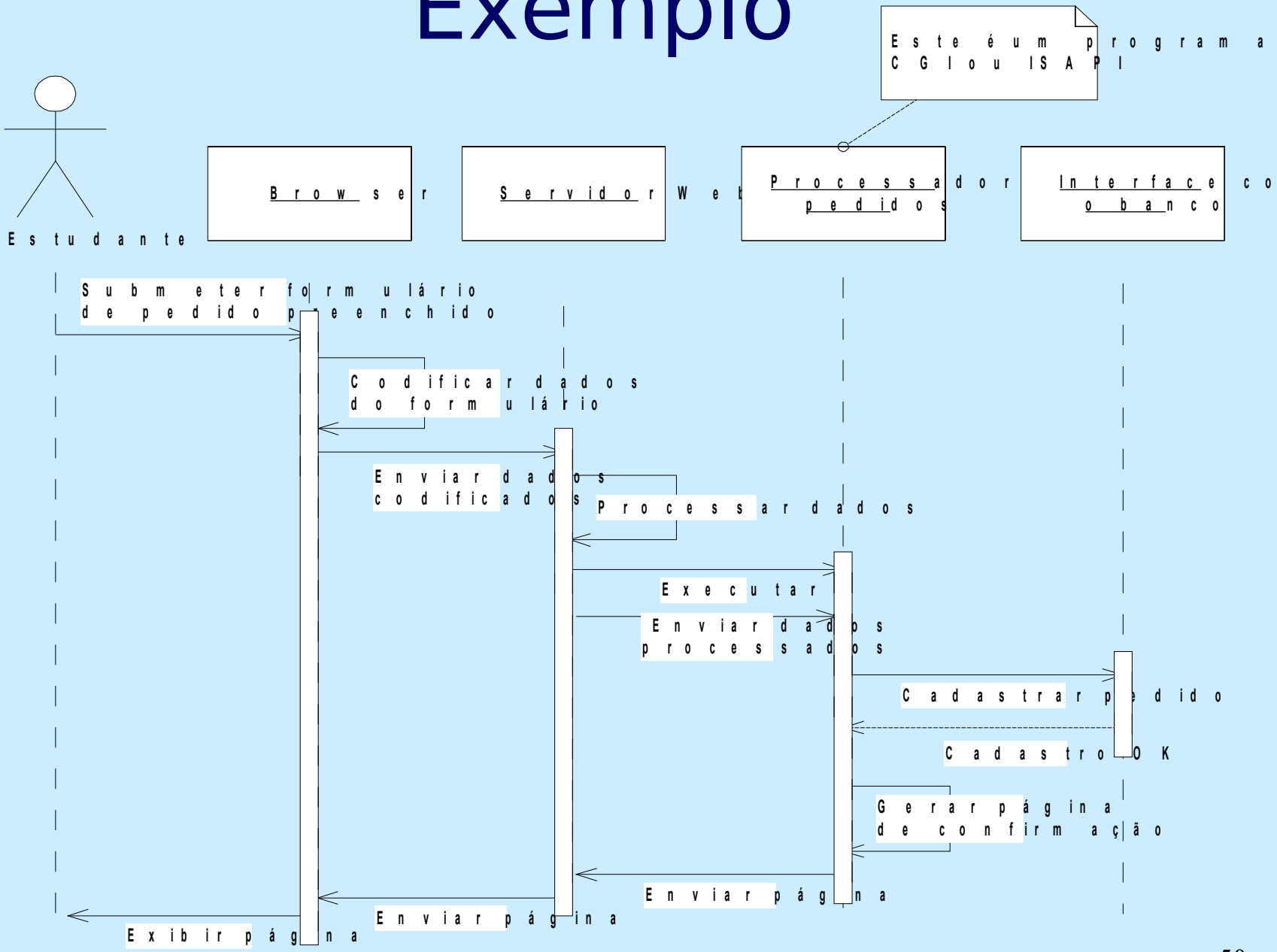
Mensagens

- Definição formal: uma mensagem é a especificação de uma comunicação entre objetos, onde são passadas informações, com a esperança de que ocorra alguma atividade
- Na maioria das vezes, uma mensagem resulta na execução de uma operação

Mensagens

- Tipos principais de mensagens:
  - **Chamada** (Call)
  - **Retorno** (Return)
  - **Envio** (Send)
  - **Criação** (Create)
  - **Destruição** (Destroy)

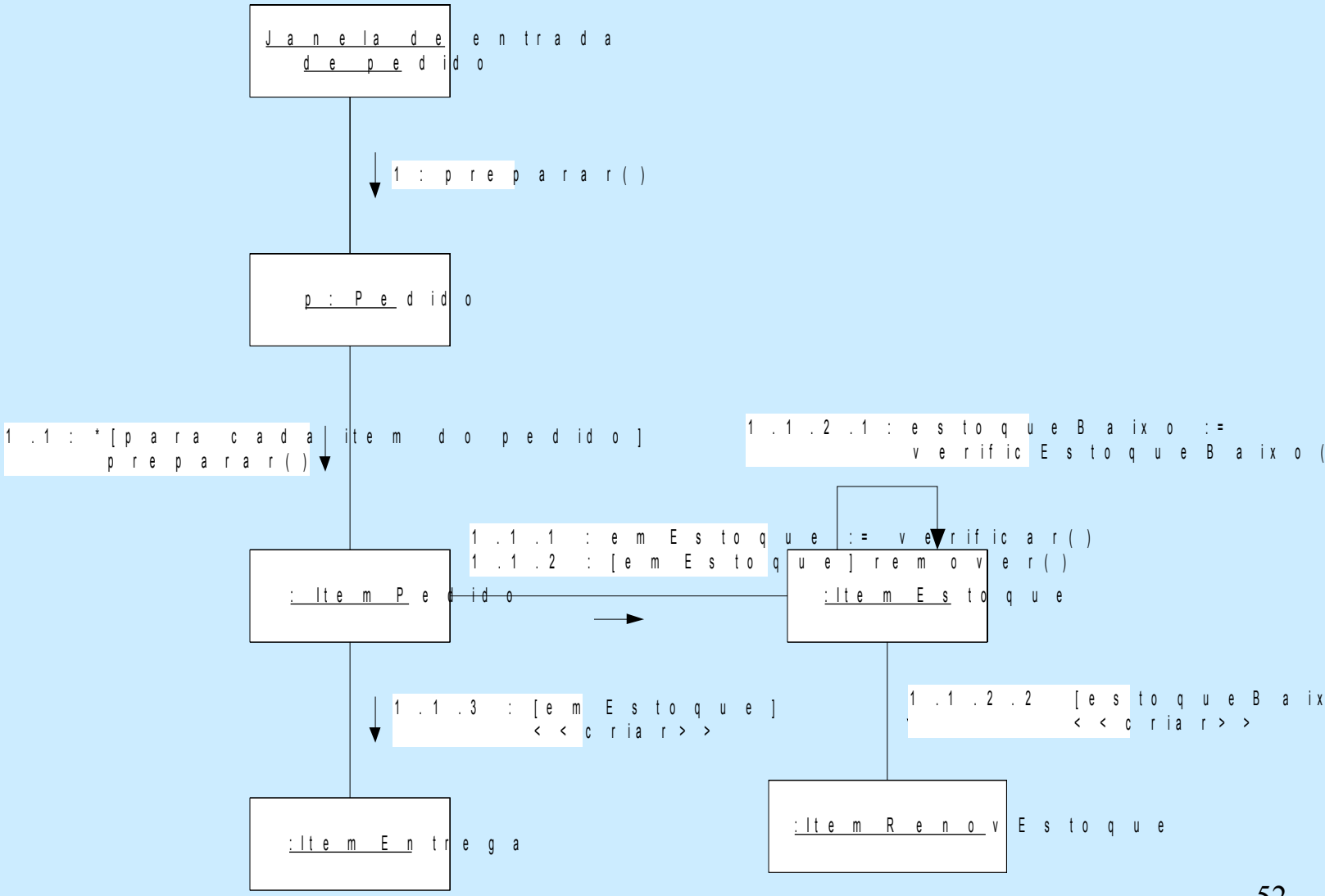
Exemplo


Este é um programa CGI ou ISAPI
Estudante
Browser
Servidor Web
Processador de pedidos
Interface com o banco
Submeter formulário de pedido preenchido
Codificar dados do formulário
Enviar dados codificados
Processar dados
Executar
Enviar dados processados
Cadastrar pedido
Cadastro OK
Gerar página de confirmação
Enviar página
Enviar página
Exibir página

Diagramas de Comunicação

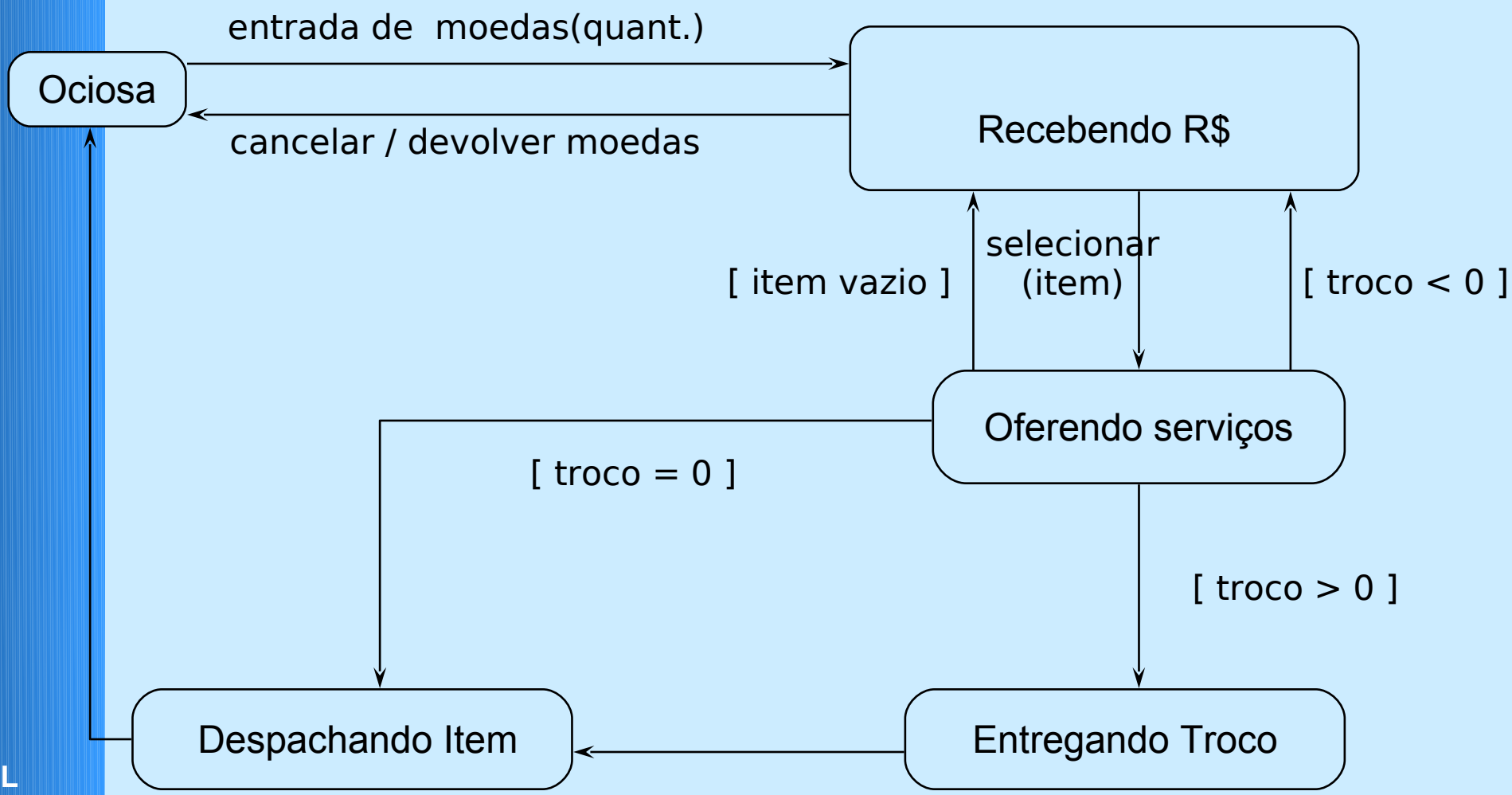
- Diagramas de Comunicação enfatizam a **organização** dos objetos em uma interação
- Praticamente tudo que pode ser mostrado em um diagrama de seqüência pode também ser mostrado em um diagrama de comunicação
- Diagramas de Comunicação podem ser transformados em diagramas de seqüência e vice-versa

Exemplo

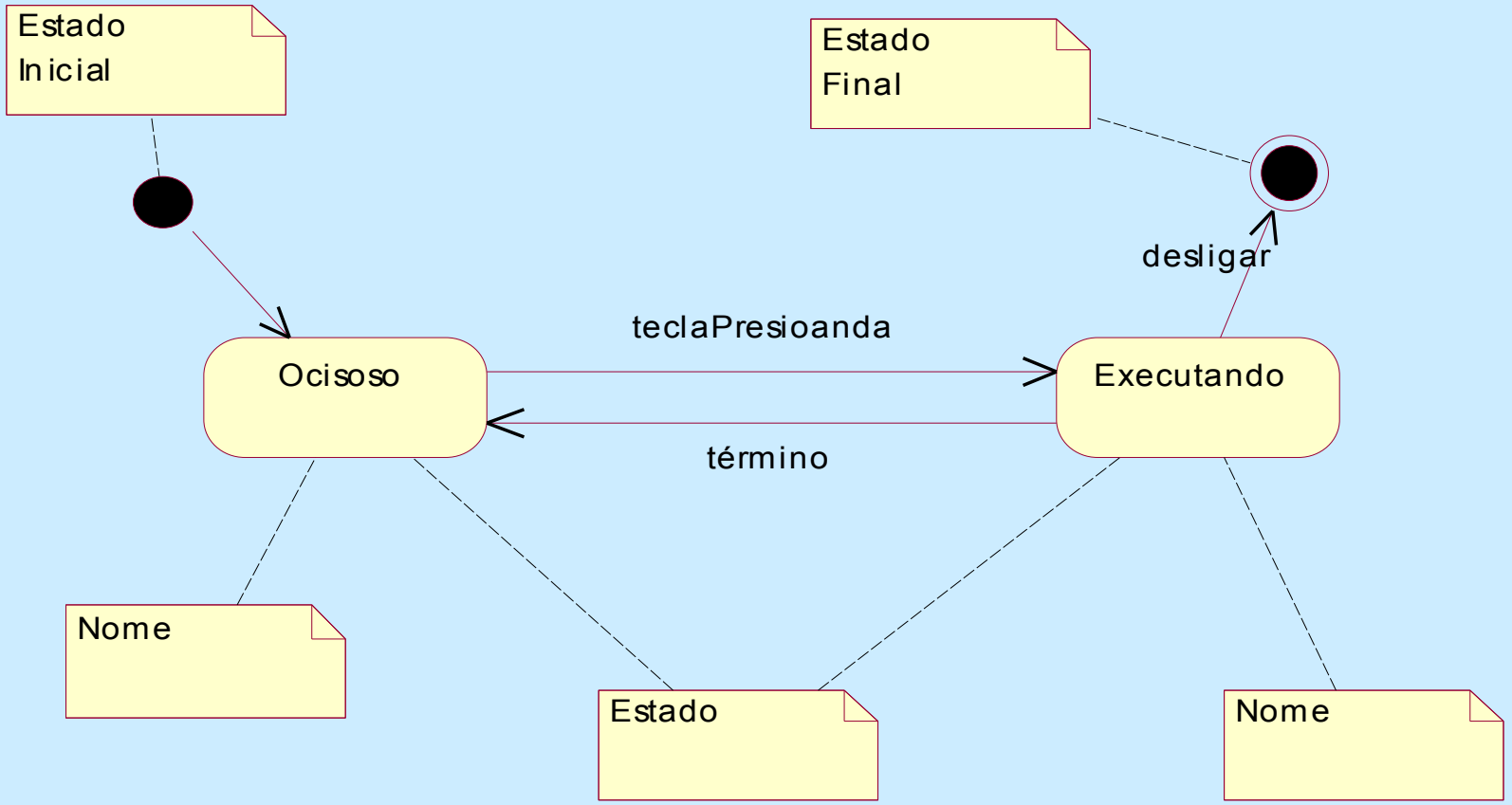

Janela de entrada de pedido
1 : preparar()
p : Pedido
1.1 : *[para cada item do pedido] preparar()
1.1.2.1 : estoqueBaixo := verificEstoqueBaixo(
1.1.1 : emEstoque := verificar()
1.1.2 : [emEstoque] remover()
:ItemPedido
:ItemEstoque
1.1.3 : [emEstoque] <<criar>>
1.1.2.2 [estoqueBaix
<<criar>>
:ItemEntrega
:ItemRenovEstoque

Diagramas de Máquina de Estados

Máquina de Vendas



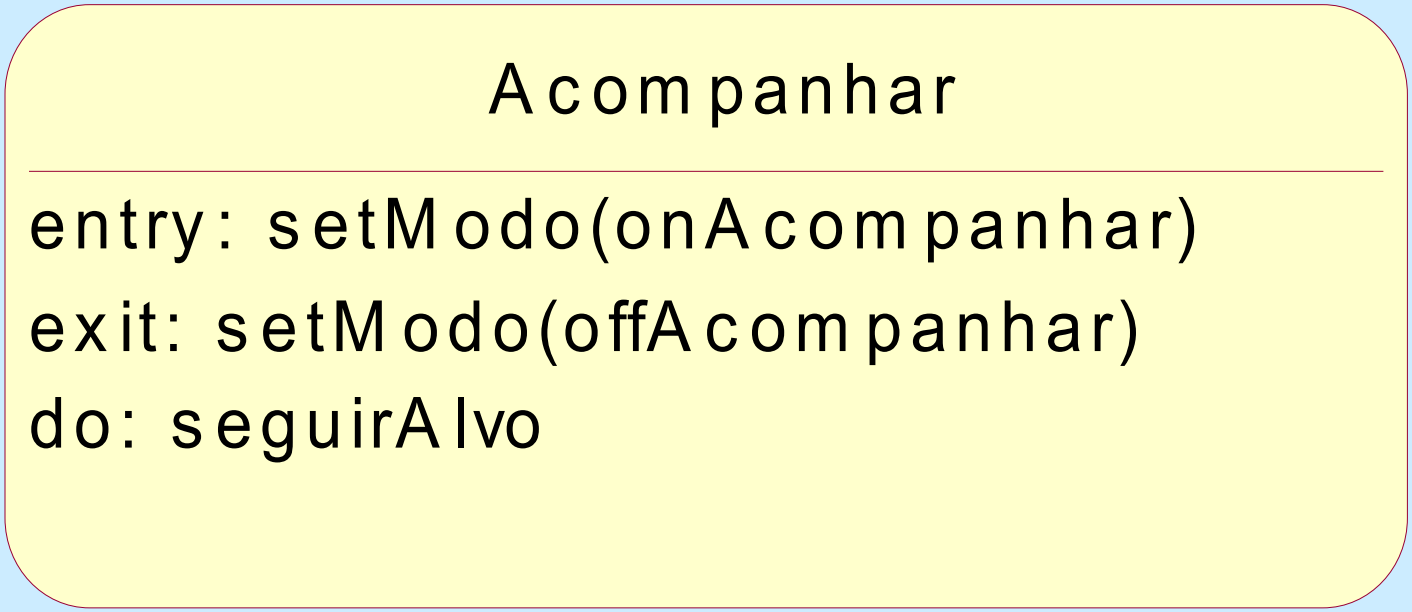
Estados


Estado
Inicial
Estado
Final
desligar
teclaPresioanda
Ocisoso
Executando
término
Nome
Estado
Nome

Partes de um Estado

- Nome
- Ações de entrada (Entry)
- Ações da saída (Exit)
- Atividades (do:)

Partes de um estado



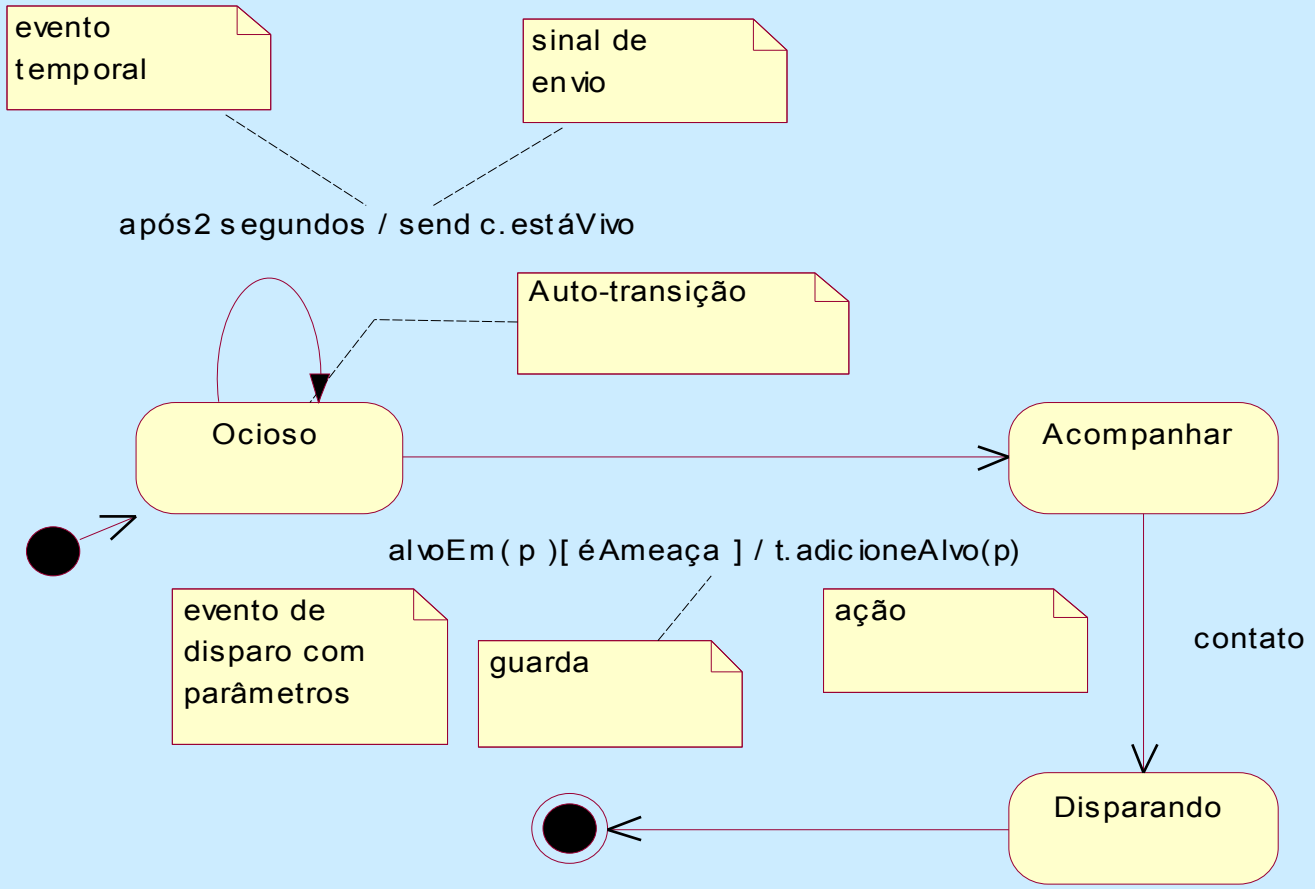
Transição

- É um relacionamento entre dois estados indicando que o objeto no primeiro estado irá executar certas ações e entrar no segundo estado quando o evento especificado ocorrer e as condições especificadas forem satisfeitas
- Uma transição de estado é uma mudança de estado causada por um evento

Partes da transição

- Estado fonte
- Evento de disparo
- Condição de guarda
- Ação
- Estado alvo

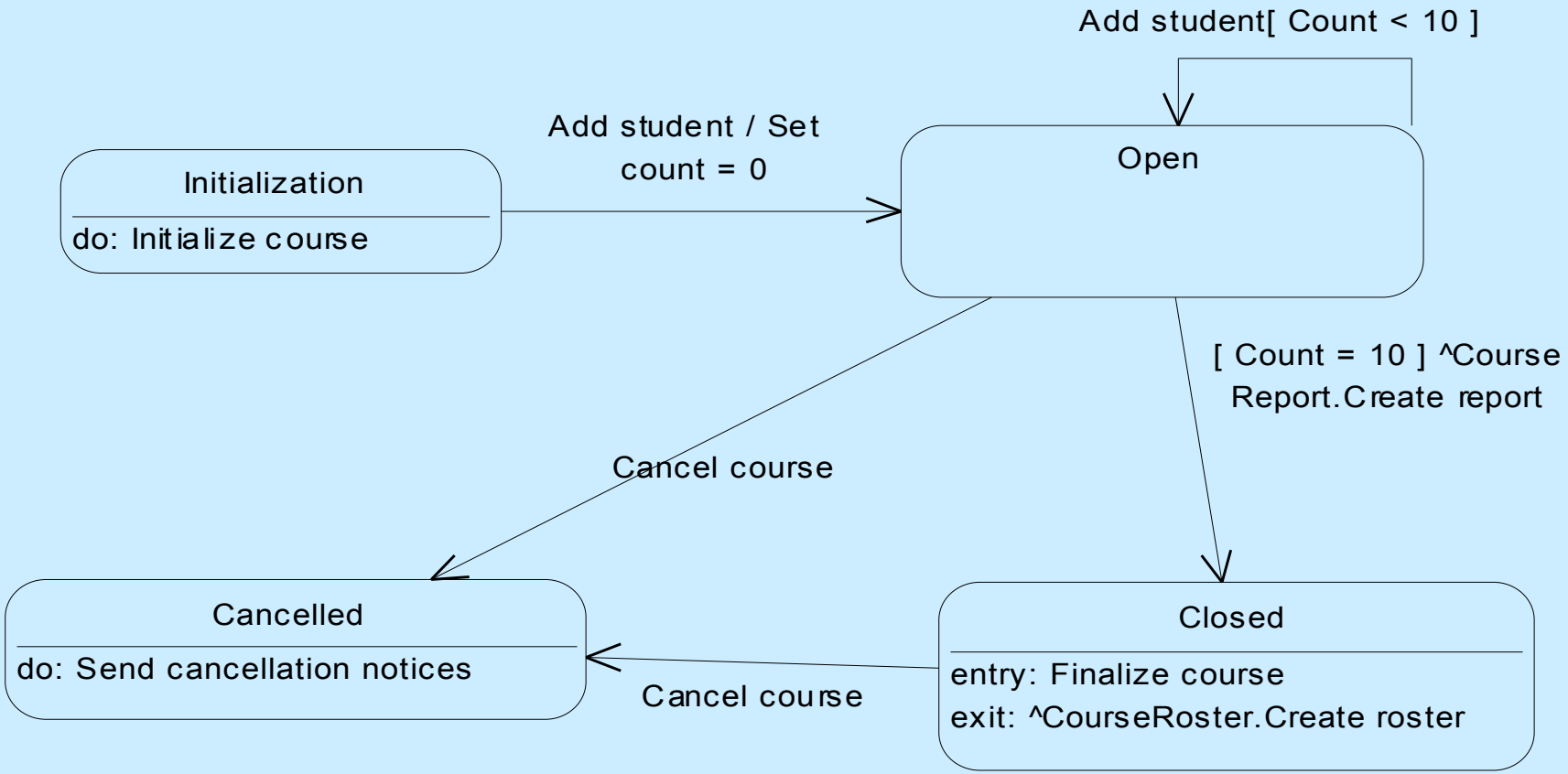
Sistema de disparo de míssil



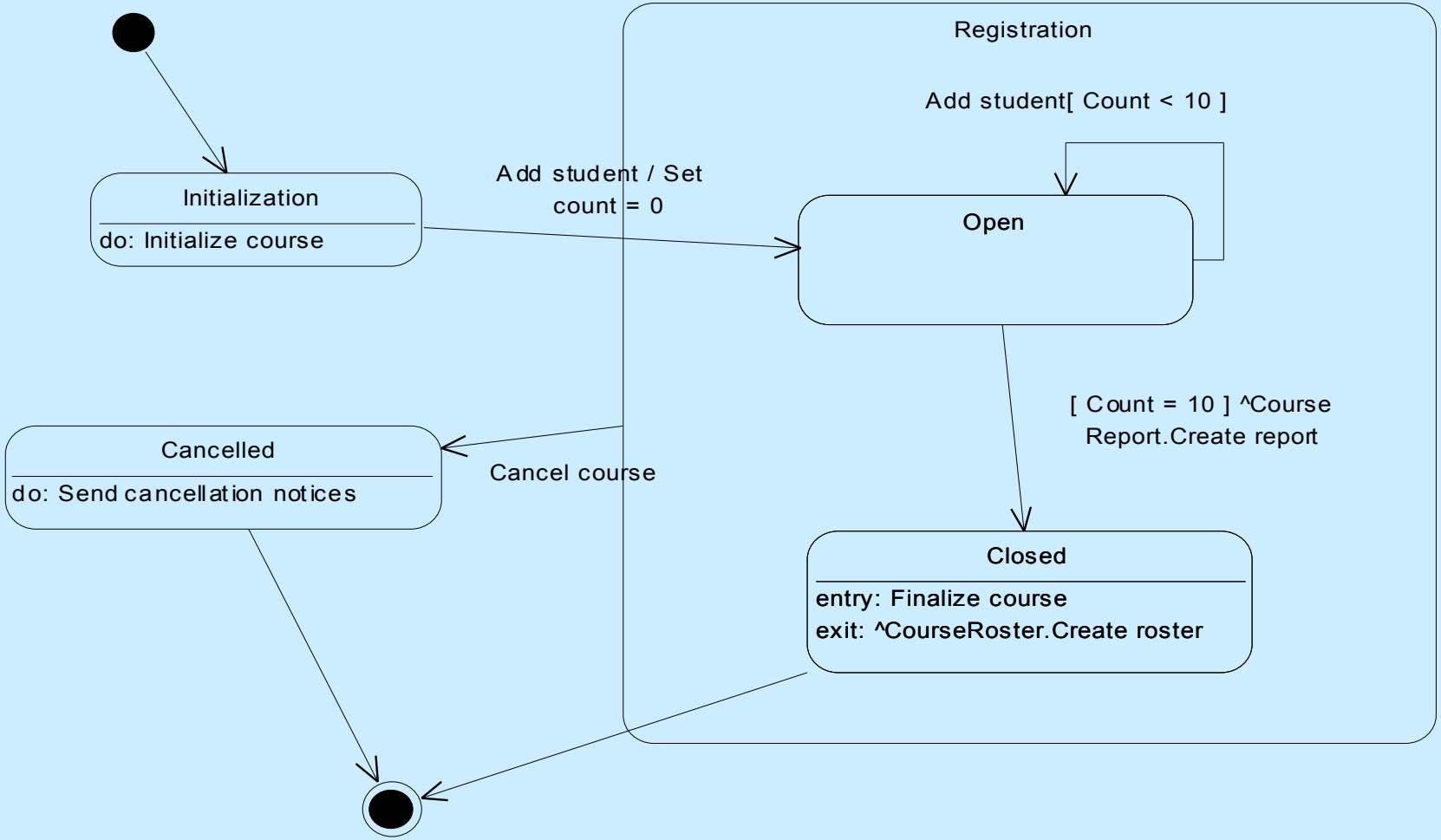
Problemas com Máquinas de Estados

- Máquinas de estados não estruturados não possuem bom poder de expressão e tornam-se impraticáveis para problemas grandes
- Existe uma explosão combinatorial em diagramas planos
- As formas de estruturação:
  - Refinamento
  - Concorrência

Refinamento

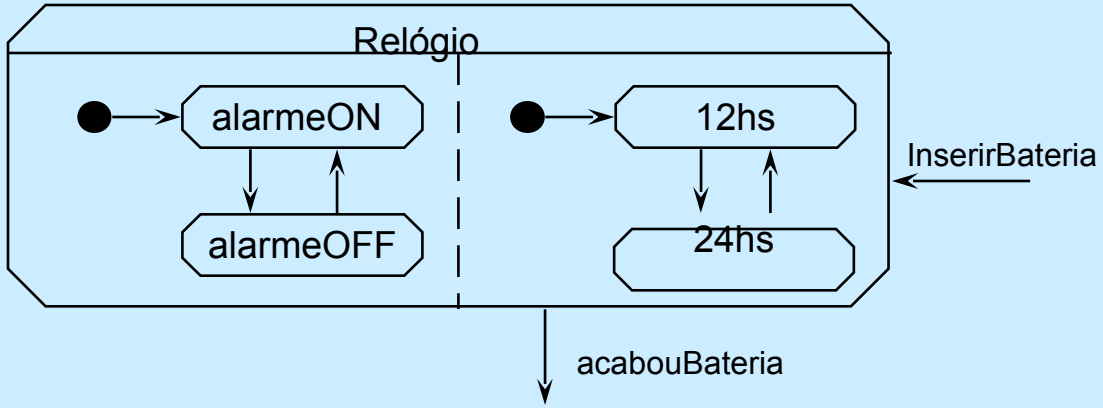


Refinamento: SubEstados



Concorrência dentro de um objeto

- Pode ser mostrada com partições pontilhadas
- Normalmente surge de dois ou mais atributos ortogonais



Concorrência dentro de um objeto

Divisão do Controle: