

Unified Modeling Language – UML (Linguagem de Modelagem Unificada)

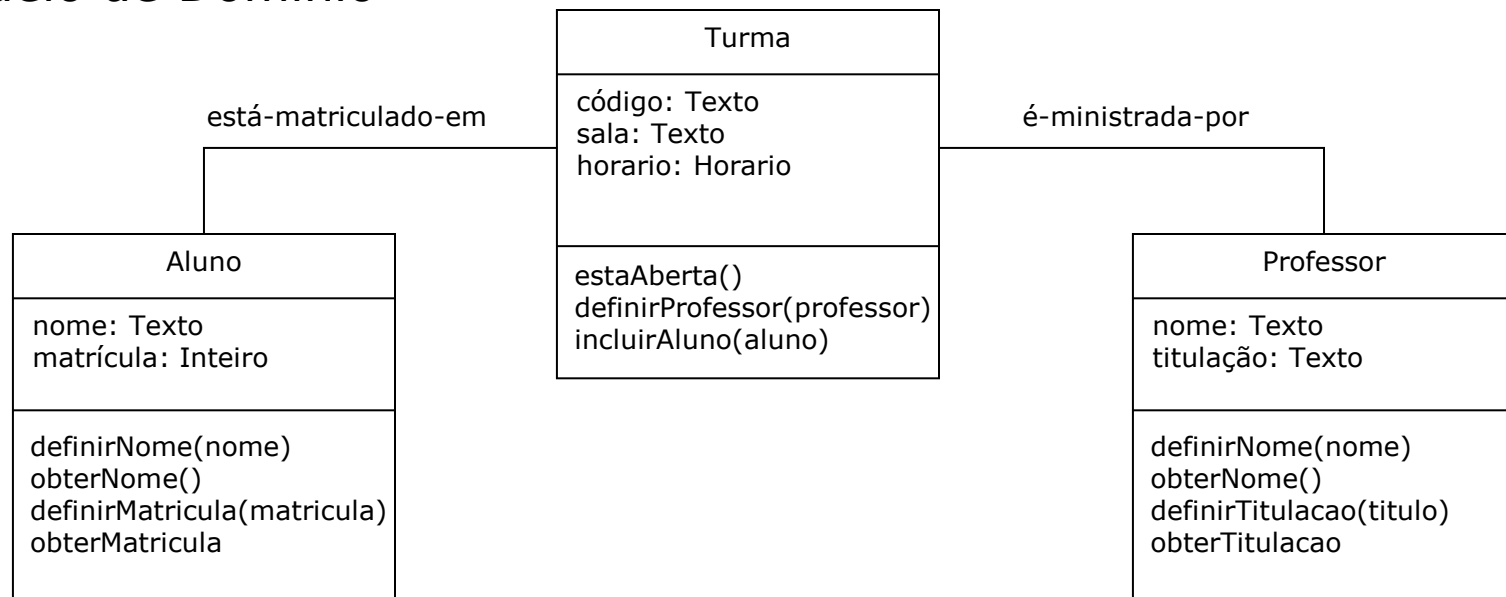
Profa. Me. Viviane de Fatima Bartholo

UML: Diagrama de Classes

Projeto de Sistemas de Software

- Introdução – Diagrama de classes
- Elementos do diagrama de classes
- Exemplo: Sistema de matrícula

- É o diagrama central da modelagem orientada a objetos.
- Mostra um conjunto de classes e seus relacionamentos.
- Representa
 - Modelo Conceitual
 - Modelo de Domínio



Perspectivas de um Diagrama de Classes

O diagrama de classes evolui com o sistema e pode ter diferentes perspectivas

- Na **análise** – identificamos objetos (classes) no domínio do problema
- No **projeto** – pensamos em objetos (classes) para a solução

Perspectivas de um Diagrama de Classes

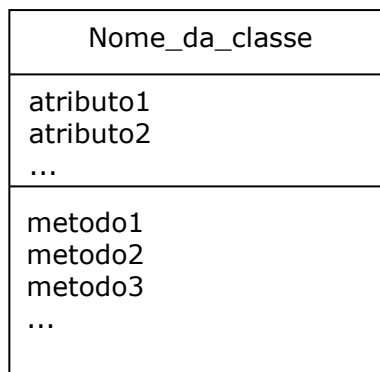
- **O modelo conceitual (análise)** representa as classes no domínio do negócio em questão. Não leva em consideração restrições inerentes à tecnologia a ser utilizada na solução de um problema.
- **O modelo de classes de especificação (projeto)** é obtido através da adição de detalhes ao modelo anterior conforme a solução de software escolhida.
- **O modelo de classes de implementação** corresponde à implementação das classes em alguma linguagem de programação.

- Elementos de um diagrama de classes
 - Classes
 - Relacionamentos
 - Associação
 - Agregação
 - Composição
 - Generalização
 - Dependência

- Elementos de um diagrama de classes
 - **Classes**
 - Relacionamentos
 - Associação
 - Agregação
 - Composição
 - Generalização
 - Dependência

Classes

- Graficamente, as classes são representadas por retângulos incluindo nome, atributos e métodos.



- Devem receber nomes de acordo com o vocabulário do domínio do problema.
- É comum adotar um padrão para nomeá-las
Ex: todos os nomes de classes serão substantivos singulares com a primeira letra maiúscula

Classes

- Atributos
 - Representam o conjunto de características (estado) dos objetos daquela classe
 - Visibilidade:
 - + público: visível em qualquer classe de qualquer pacote
 - # protegido: visível para classes do mesmo pacote
 - privado: visível somente para classe

Exemplo:

+ nome : String

Classes

- Métodos
 - Representam o conjunto de operações (comportamento) que a classe fornece
 - Visibilidade:
 - + público: visível em qualquer classe de qualquer pacote
 - # protegido: visível para classes do mesmo pacote
 - privado: visível somente para classe

Exemplo:

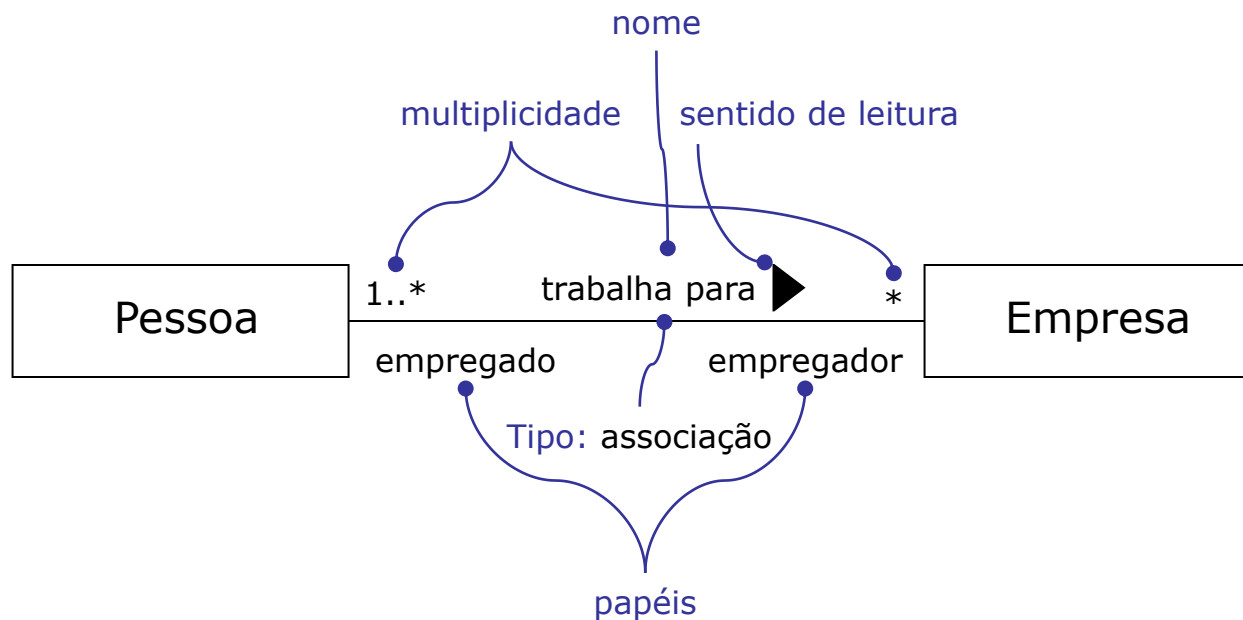
- getNome() : String

- Elementos de um diagrama de classes
 - Classes
 - **Relacionamentos**
 - Associação
 - Agregação
 - Composição
 - Generalização
 - Dependência

Relacionamentos

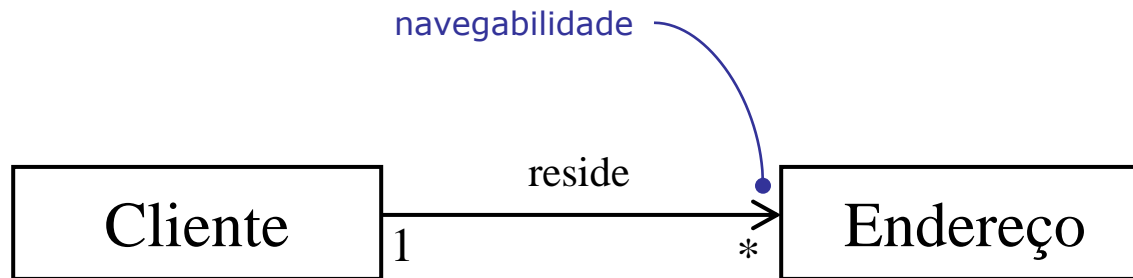
- Os relacionamentos possuem:
 - **Nome:** descrição dada ao relacionamento (faz, tem, possui,...)
 - **Sentido de leitura**
 - **Navegabilidade:** indicada por uma seta no fim do relacionamento
 - **Multiplicidade:** 0..1, 0..*, 1, 1..*, 2, 3..7
 - **Tipo:** associação (agregação, composição), generalização e dependência
 - **Papéis:** desempenhados por classes em um relacionamento

- Relacionamentos



E a navegabilidade?

- Relacionamentos

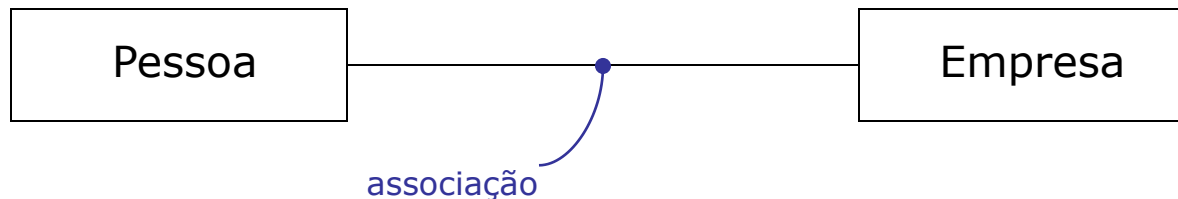


- O cliente sabe quais s o seus endere os, mas o endere o n o sabe a quais clientes pertence

- Elementos de um diagrama de classes
 - Classes
 - **Relacionamentos**
 - Associação
 - Agregação
 - Composição
 - Generalização
 - Dependência

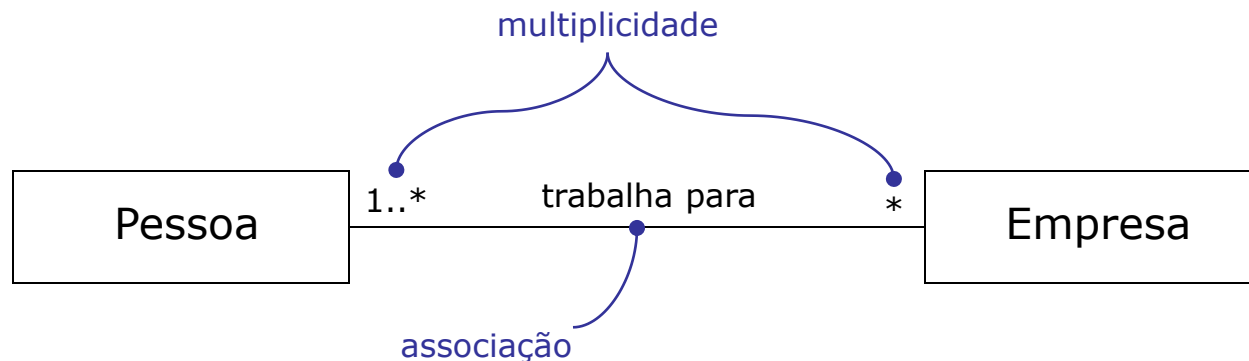
Relacionamentos: Associação

- Uma **associação** é um relacionamento estrutural que indica que os objetos de uma classe estão vinculados a objetos de outra classe.
- Uma associação é representada por uma linha sólida conectando duas classes.



Relacionamentos: Associação

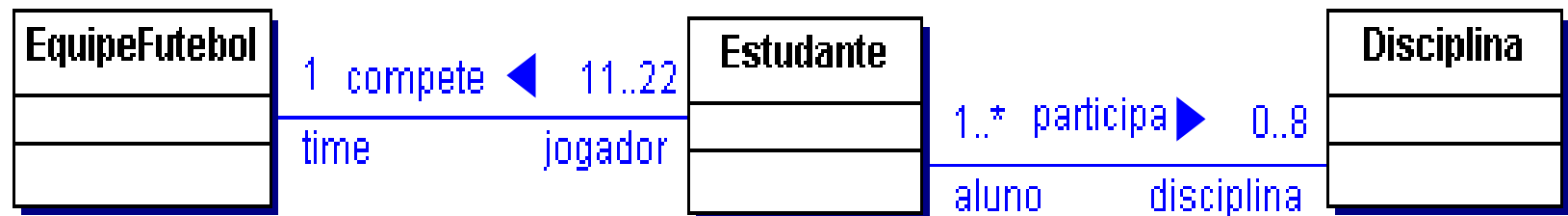
- Indicadores de multiplicidade:
 - 1 Exatamente um
 - 1..* Um ou mais
 - 0..* Zero ou mais (muitos)
 - * Zero ou mais (muitos)
 - 0..1 Zero ou um
 - m..n Faixa de valores (por exemplo: 4..7)



Relacionamentos: Associação

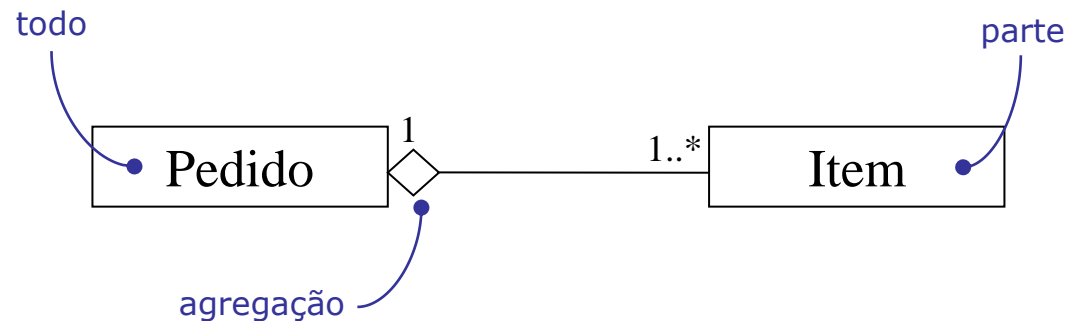
Exemplo:

- Um **Estudante** pode ser um **aluno** de uma Disciplina e um **jogador** da Equipe de Futebol
- Cada Disciplina deve ser cursada por no mínimo 1 aluno
- Um aluno pode cursar de 0 até 8 disciplinas



- Elementos de um diagrama de classes
 - Classes
 - **Relacionamentos**
 - Associação
 - **Agregação**
 - Composição
 - Generalização
 - Dependência

- Relacionamento: Agregação
 - É um tipo especial de associação
 - Utilizada para indicar “todo-parte”

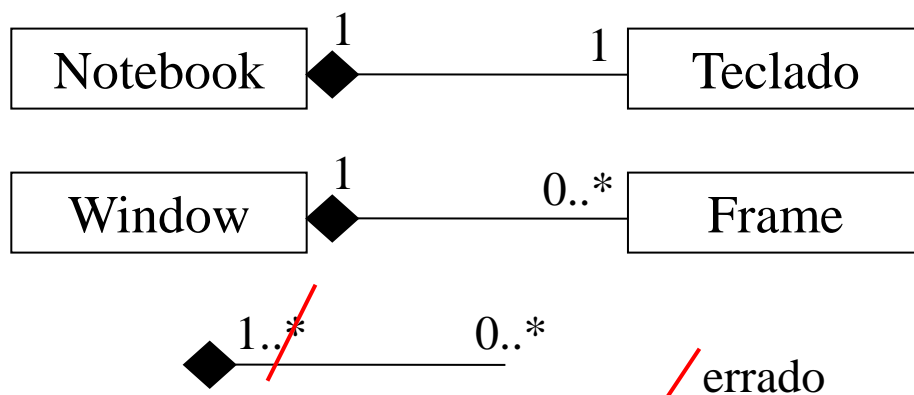


- um objeto “parte” pode fazer parte de vários objetos “todo”

- Elementos de um diagrama de classes
 - Classes
 - **Relacionamentos**
 - Associação
 - Agregação
 - **Composição**
 - Generalização
 - Dependência

- Relacionamento: Composição

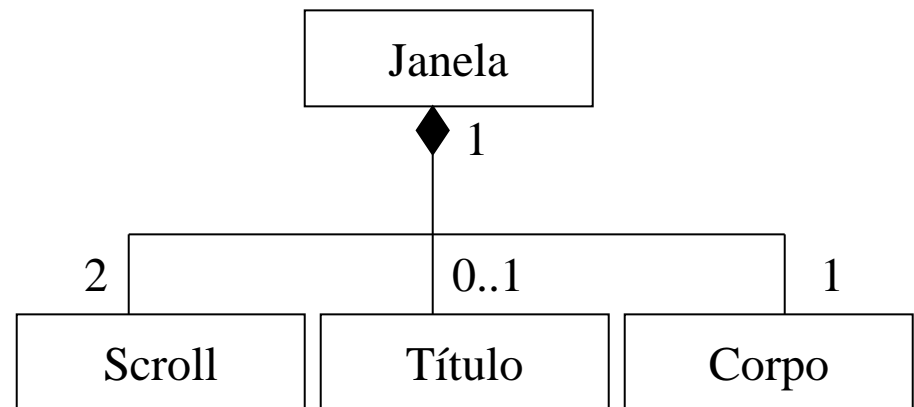
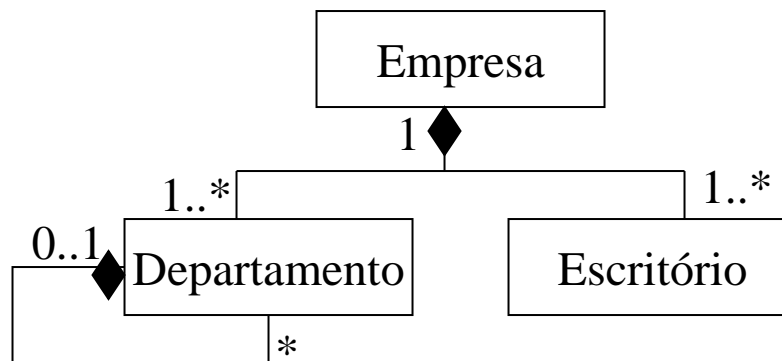
- É uma variante semanticamente mais “forte” da agregação
- Os objetos “parte” só podem pertencer a um único objeto “todo” e têm o seu tempo de vida coincidente com o dele



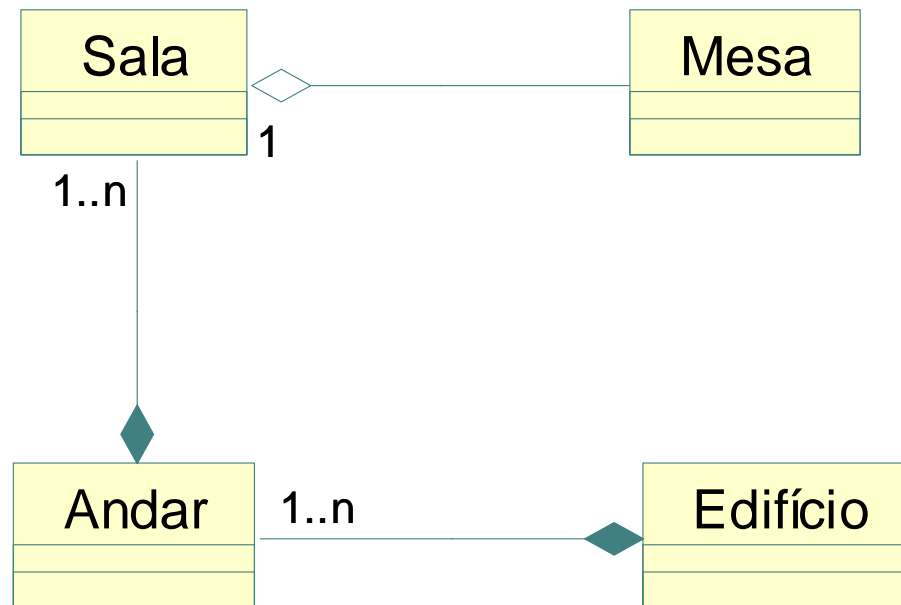
- Quando o “todo” *morre* todas as suas “partes” também *morrem*

- Relacionamento: Composição

Ex:

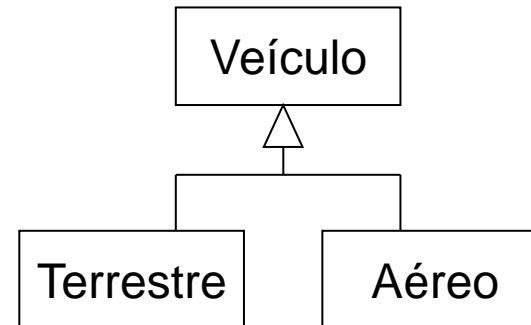
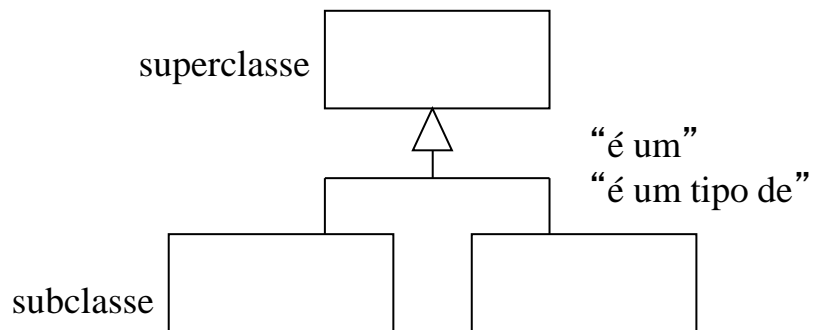


- Agregação X Composição



- Elementos de um diagrama de classes
 - Classes
 - **Relacionamentos**
 - Associação
 - Agregação
 - Composição
 - Generalização
 - Dependência

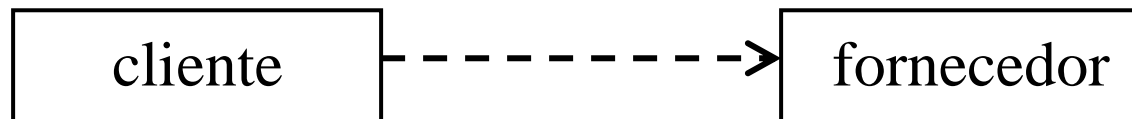
- Relacionamento: Generalização
 - É um relacionamento entre itens gerais (superclasses) e itens mais específicos (subclasses)



- Elementos de um diagrama de classes
 - Classes
 - **Relacionamentos**
 - Associação
 - Agregação
 - Composição
 - Generalização
 - Dependência

- Relacionamento: Dependência
 - Representa que a alteração de um objeto (o objeto independente) pode afetar outro objeto (o objeto dependente)

Ex:



Obs:

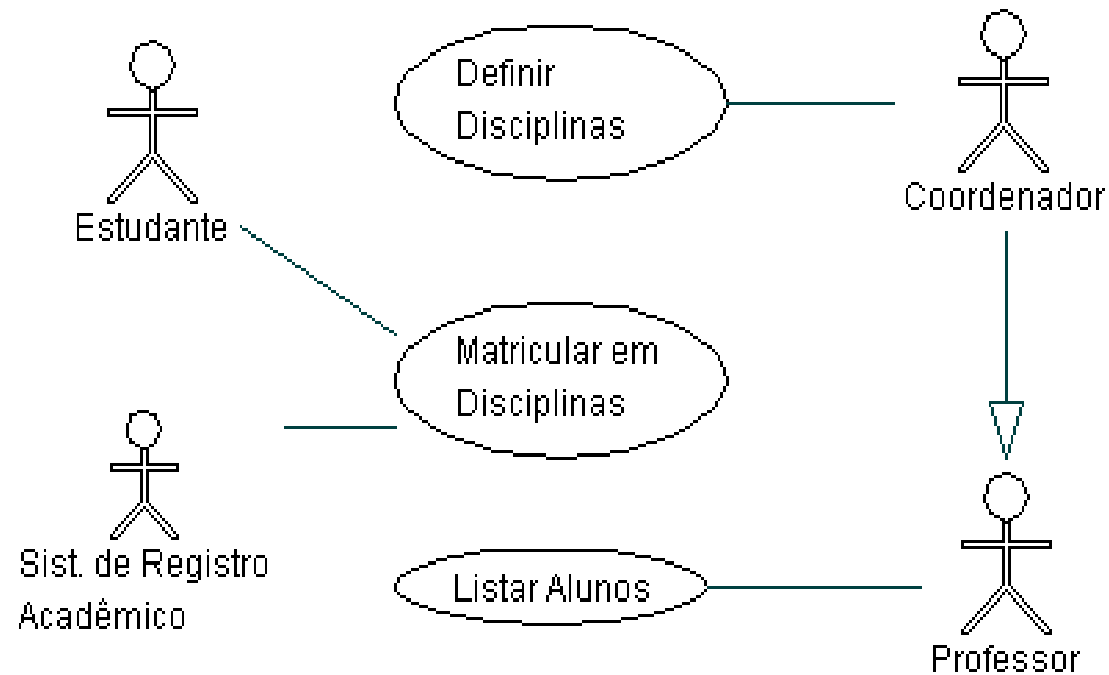
- A classe cliente depende de algum serviço da classe fornecedor
- A mudança de estado do fornecedor afeta o objeto cliente
- A classe cliente não declara nos seus atributos um objeto do tipo fornecedor
- Fornecedor é recebido por parâmetro de método

Descrição

A Universidade XYZ deseja informatizar seu sistema de matrículas:

- A universidade oferece vários cursos.
- O **Coordenador** de um curso define as disciplinas que serão oferecidas pelo seu curso num dado semestre.
- Várias disciplinas são oferecidas em um curso.
- Várias turmas podem ser abertas para uma mesma disciplina, porém o número de estudantes inscritos deve ser entre 3 e 10.
- **Estudantes** selecionam 4 disciplinas.
- Quando um estudante matricula-se para um semestre, o **Sistema de Registro Acadêmico (SRA)** é notificado.
- Após a matrícula, os estudantes podem, por um certo prazo, utilizar o sistema para adicionar ou remover disciplinas.
- **Professores** usam o sistema para obter a lista de alunos matriculados em suas disciplinas.
- Todos os usuários do sistema devem ser validados.

Diagrama de Casos de Uso



Descrição do Caso de Uso “Matricular em Disciplina”

- Esse caso de uso se inicia quando o Estudante de Curso inicia uma sessão no sistema e apresenta suas credenciais.
- O sistema verifica se a credencial é válida.
- O sistema solicita que o estudante realize sua matrícula, selecionando 4 disciplinas.
- O estudante preenche um formulário eletrônico de matrícula e o submete para uma análise de consistência.
- O sistema analisa as informações contidas no formulário.
 - Se as informações são consistentes, o estudante é incluído em turmas abertas de 4 disciplinas, iniciando pelas preferenciais.
 - Se as informações não são consistentes, o sistema informa o motivo da inconsistência e solicita que o formulário seja alterado.

Diagrama de Classes: identificando as classes

Professor

Coordenador

Estudante

Universidade

Disciplina

Turma

Curso

FormularioMatricula

AnalizadorMatricula

SistemaRegistroAcademico

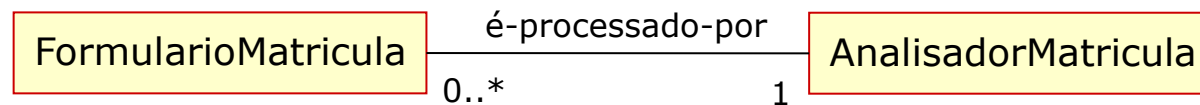
ListaAlunos

Diagrama de Classes: identificando os relacionamentos

- Exemplos de candidatos a relacionamentos:
 - **A** é parte física ou lógica de **B**.
 - **A** está contido fisicamente ou logicamente em **B**.
 - **A** é uma descrição de **B**.
 - **A** é membro de **B**.
 - **A** é subunidade organizacional de **B**.
 - **A** usa ou gerencia **B**.
 - **A** se comunica/interage com **B**.
 - **A** está relacionado com uma transação **B**.
 - **A** é possuído por **B**.
 - **A** é um tipo de **B**.

Diagrama de Classes: identificando os relacionamentos

- O formulário de matrícula é processado por um analisador de matrícula



- O analisador de matrícula gerencia a disciplina

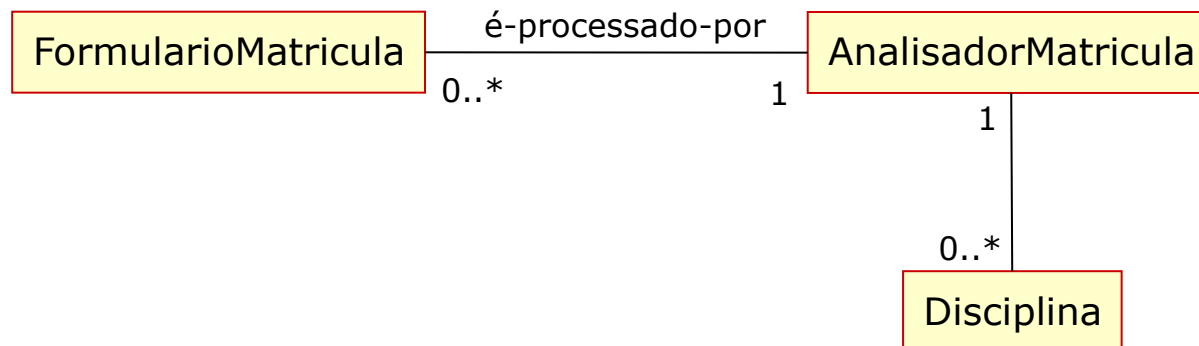


Diagrama de Classes

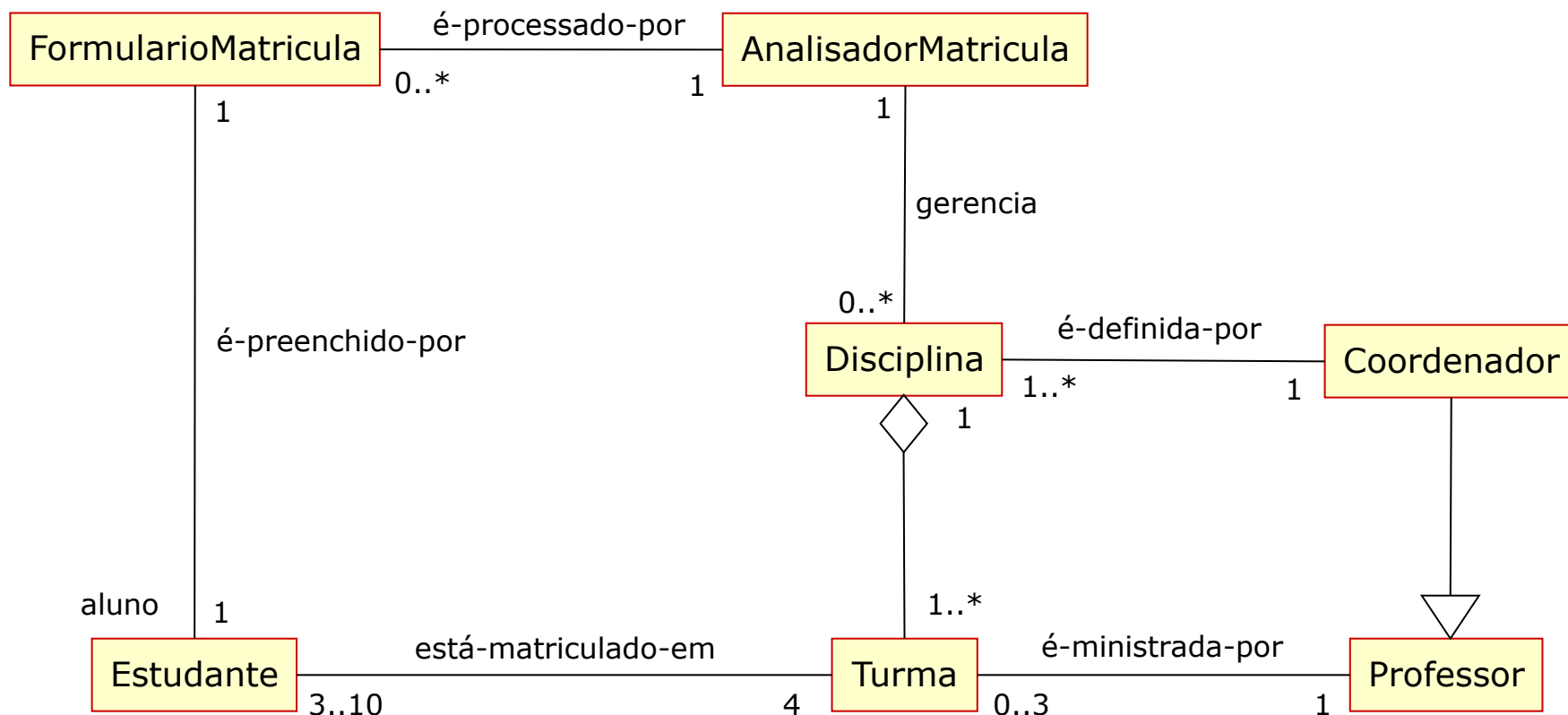


Diagrama de Classes: identificando os atributos

- Os atributos podem ser encontrados examinando-se as descrições dos casos de uso e também pelo conhecimento do domínio do problema.

- Cada turma oferecida possui um código, uma sala e um horário.

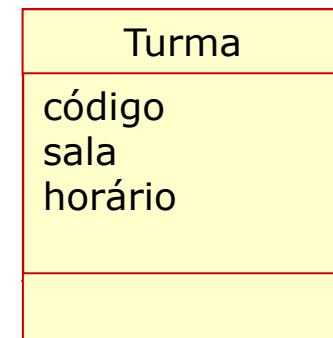
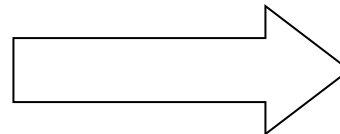


Diagrama de Classes

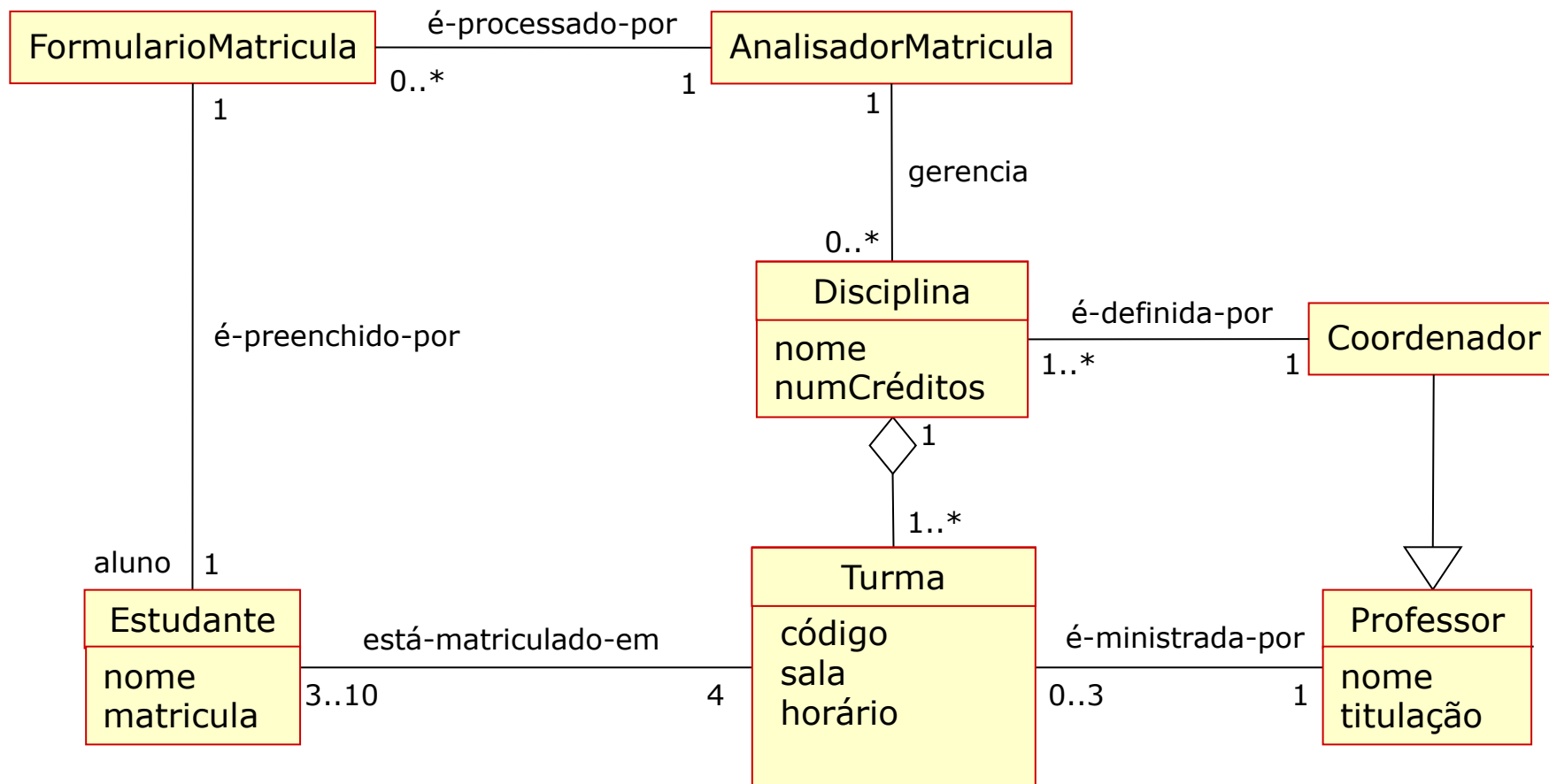


Diagrama de Classes: identificando os métodos

- Somente depois de modelar os diagramas de sequência

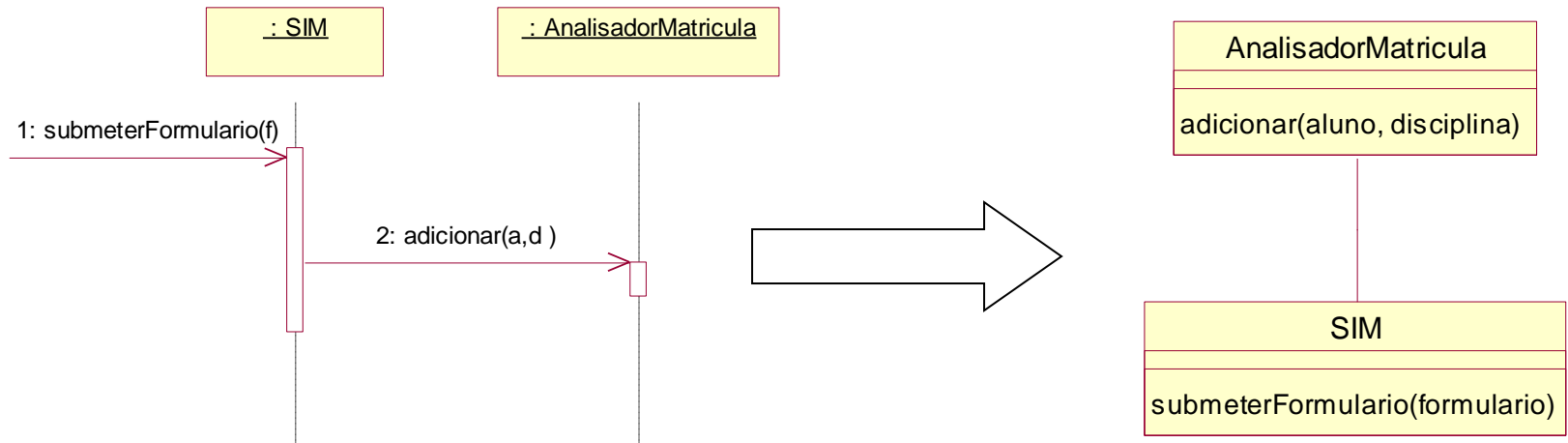
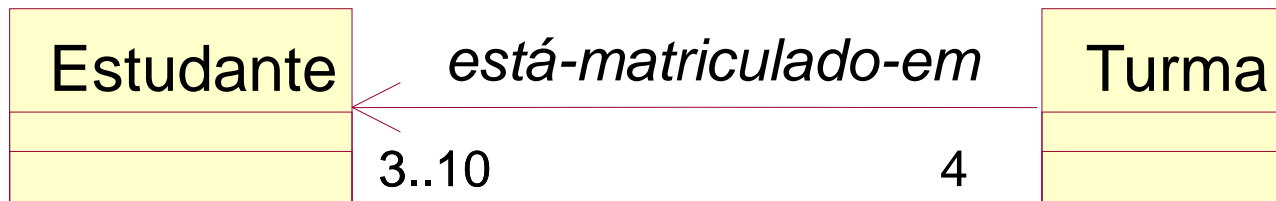


Diagrama de Classes:

- E a navegabilidade?



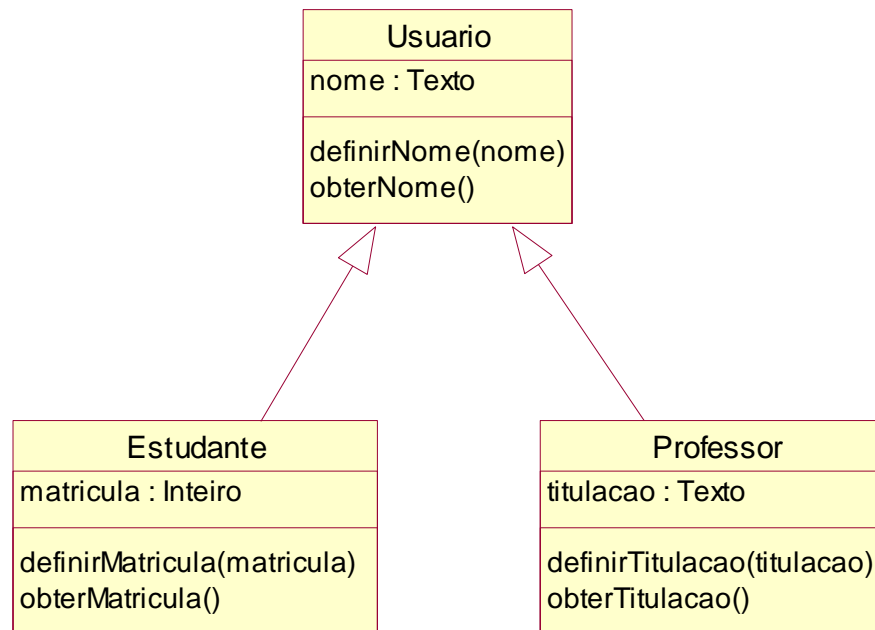
```
public class Estudante {
    private String nome;
    private String matricula;
    ...
}
```

```
public class Turma {
    private String codigo;
    private String sala;
    private Estudante alunos[];
    ...
}
```

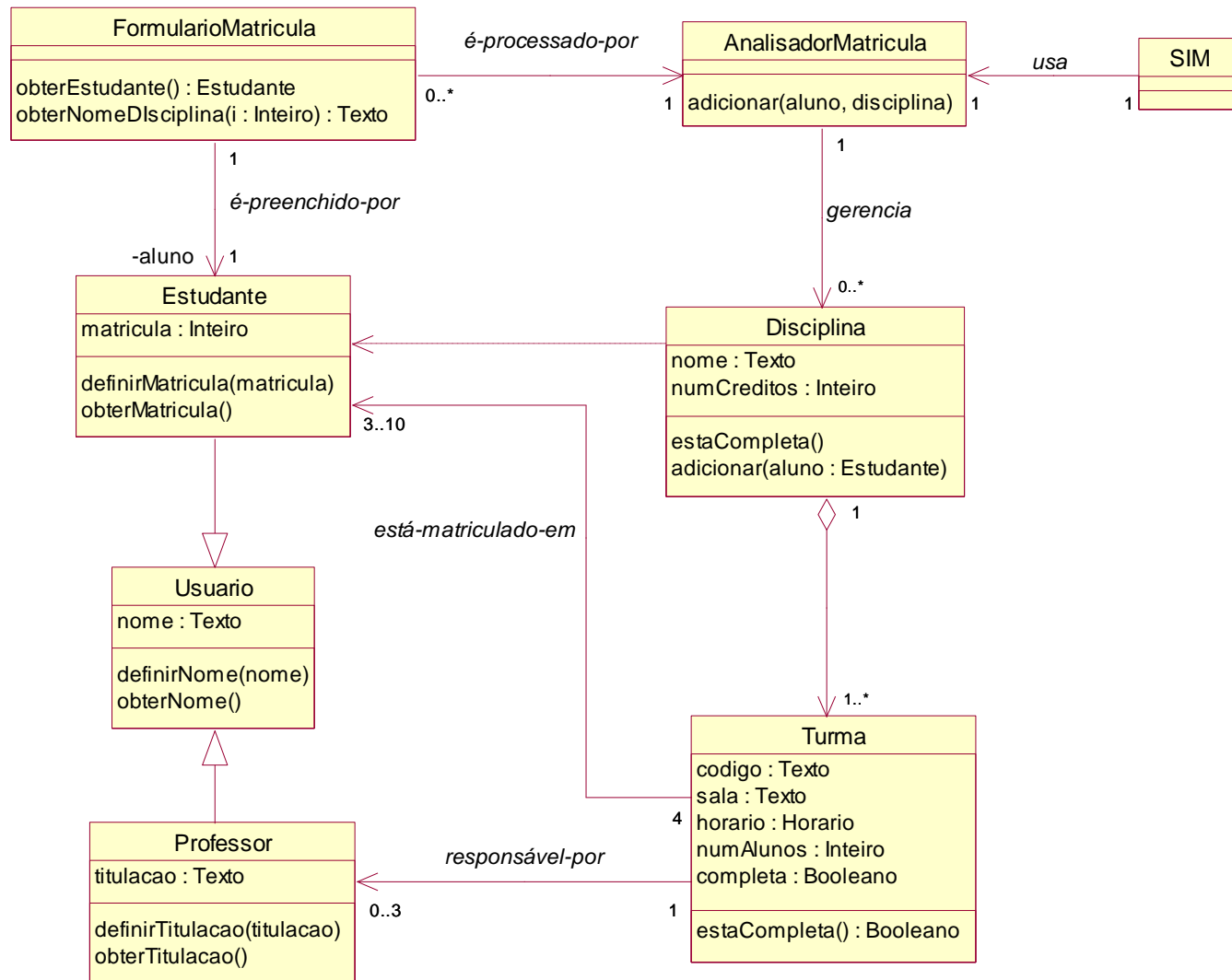
OBS: Turma não aparece como atributo de Estudante!

Diagrama de Classes:

- Acrescentando generalizações:
 - Atributos, operações e/ou relacionamentos comuns podem ser movidos para uma classe mais geral.



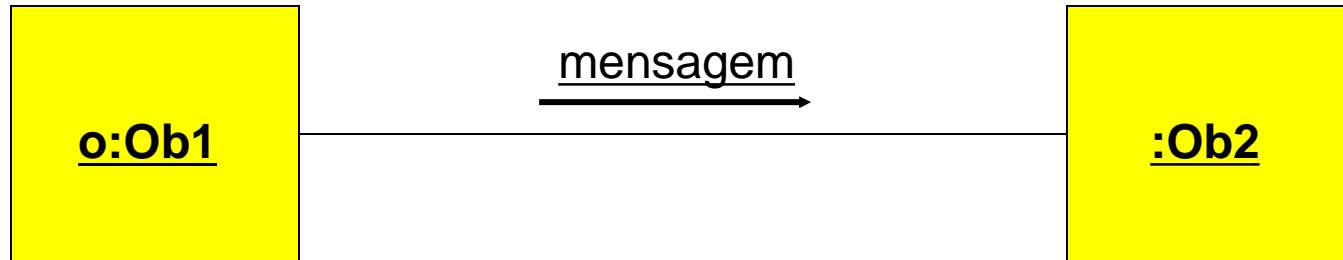
Exemplo: Sistema de Matrícula



Diagramas de Seqüência

Projeto de Sistemas de Software

- Comportamento que
 - Envolve conjunto de mensagens trocadas entre objetos dentro de um determinado contexto
 - Objetiva atingir resultado específico
- Acontecem em função da troca de mensagens entre objetos
- Usadas para a modelagem dos aspectos dinâmicos de um sistema



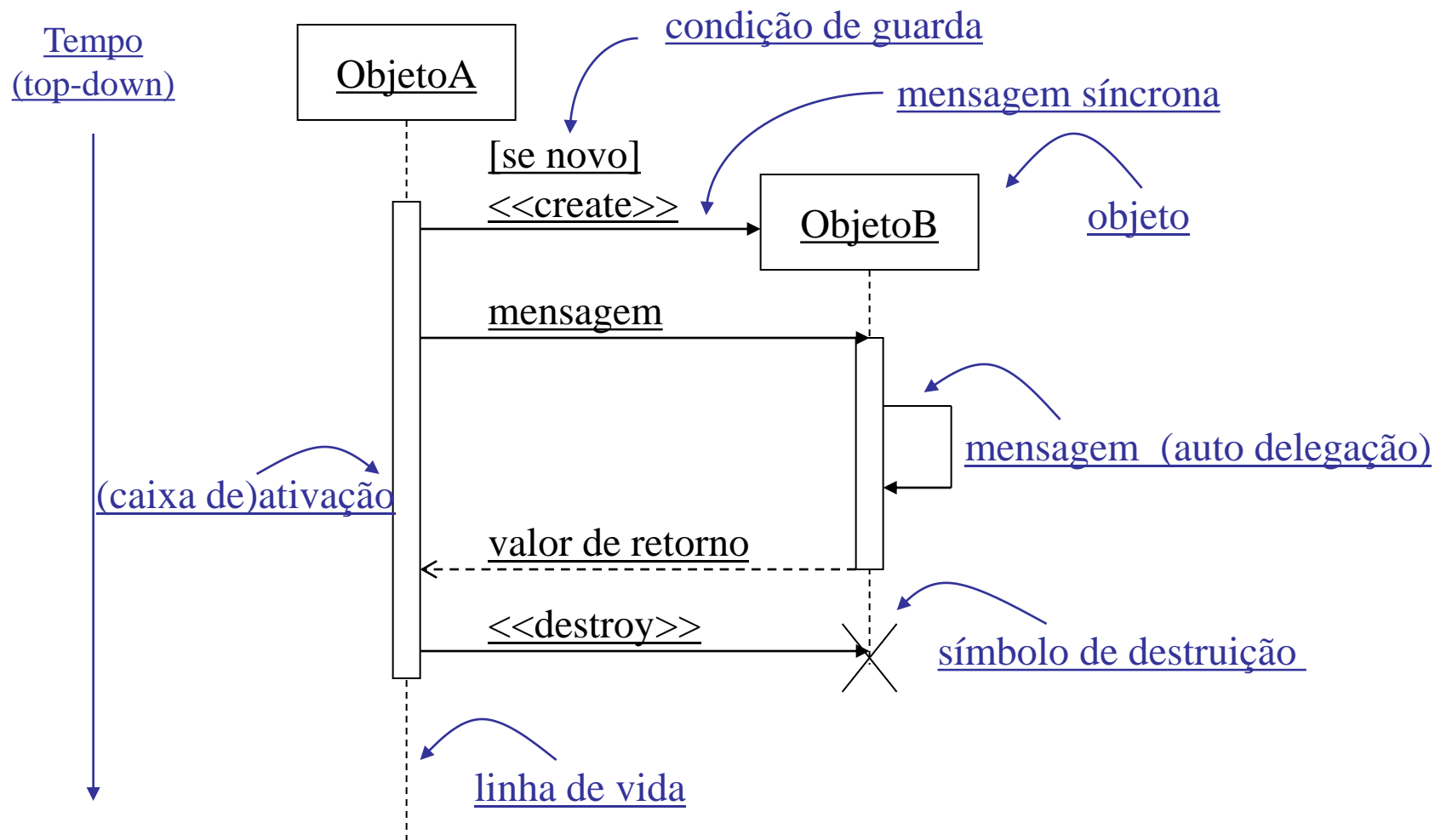
Mensagem =

<u>Ident. Objeto</u>	<u>Ident. Operação</u>	<u>Parâmetros</u>
----------------------	------------------------	-------------------

- Mensagem
 - Recepção de mensagem por um objeto
 - Considerado instância de evento
 - Decorrência da passagem de uma mensagem
 - Repercute ação representada por um comando executável
 - Comando Executável: abstração de procedimento computacional

- Deseja-se **representar o comportamento** de vários objetos
 - Dentro de um único caso de uso
 - A partir das **mensagens** que são passadas entre eles
- Objetivo
 - Definir um **contexto** de caso de uso
 - Estabelecer os **objetos** que interagem e seus **relacionamentos**
- Termo genérico que se aplica a dois tipos de diagramas que enfatizam interações entre objetos
 - **Diagrama de Seqüência**
 - **Diagrama de Colaboração**

- Informações bastante similares mas de maneira diferente
 - Diagrama de Seqüência
 - Interação enfatizando o **tempo de seqüência**
 - Mostra objetos participando em interações de acordo com suas linhas de vida e as mensagens que trocam
 - Diagrama de Colaboração
 - Interação enfatizando o **relacionamento** entre os objetos



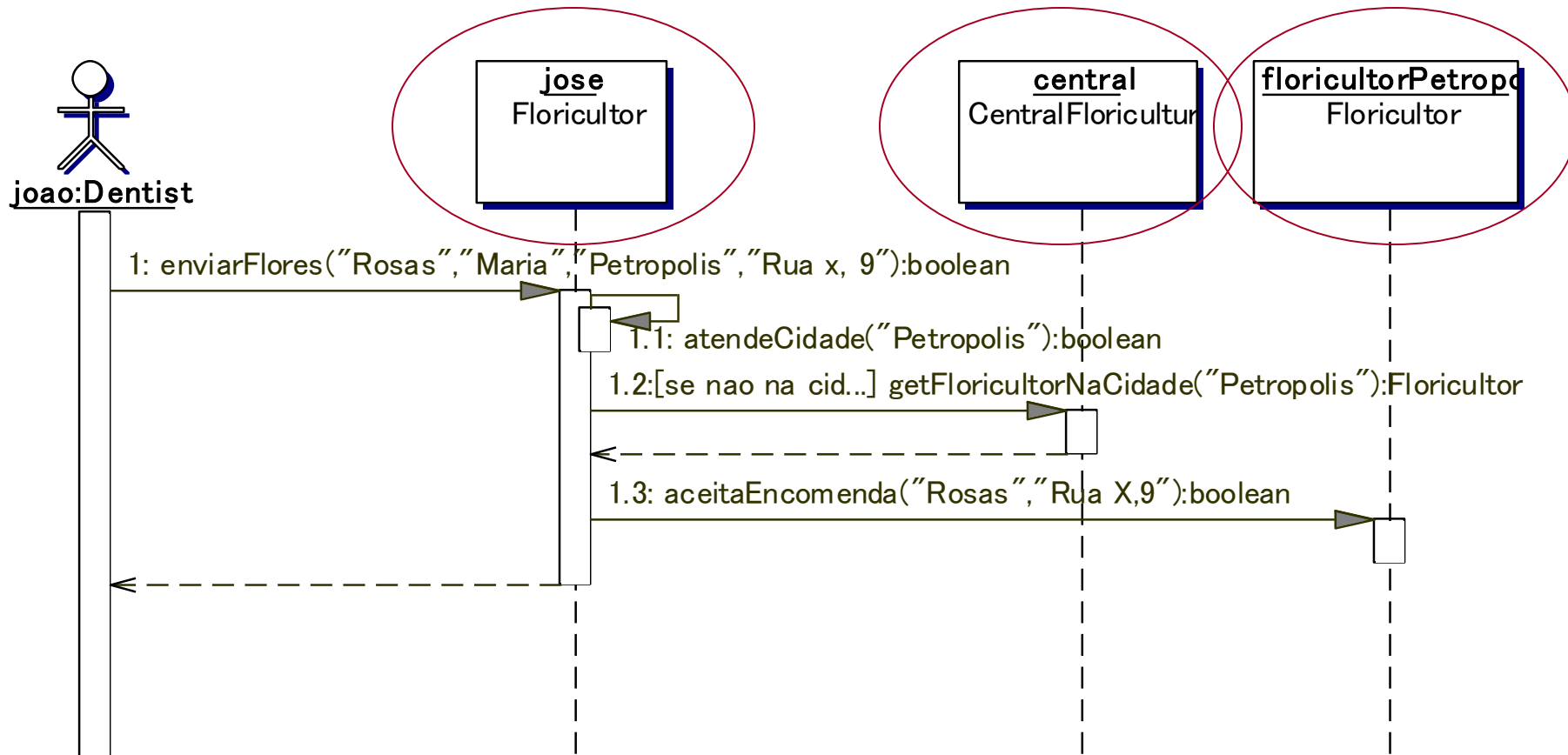
- Objetos
- Linhas de vida
- Mensagens
- Focos de controle

- Apresentados na **dimensão horizontal** do diagrama
- **Ordem** dos objetos não é considerada
 - Dispô-los de forma a tornar o diagrama “mais legível”
- Objetos tem nomes
 - **obj:Classe**

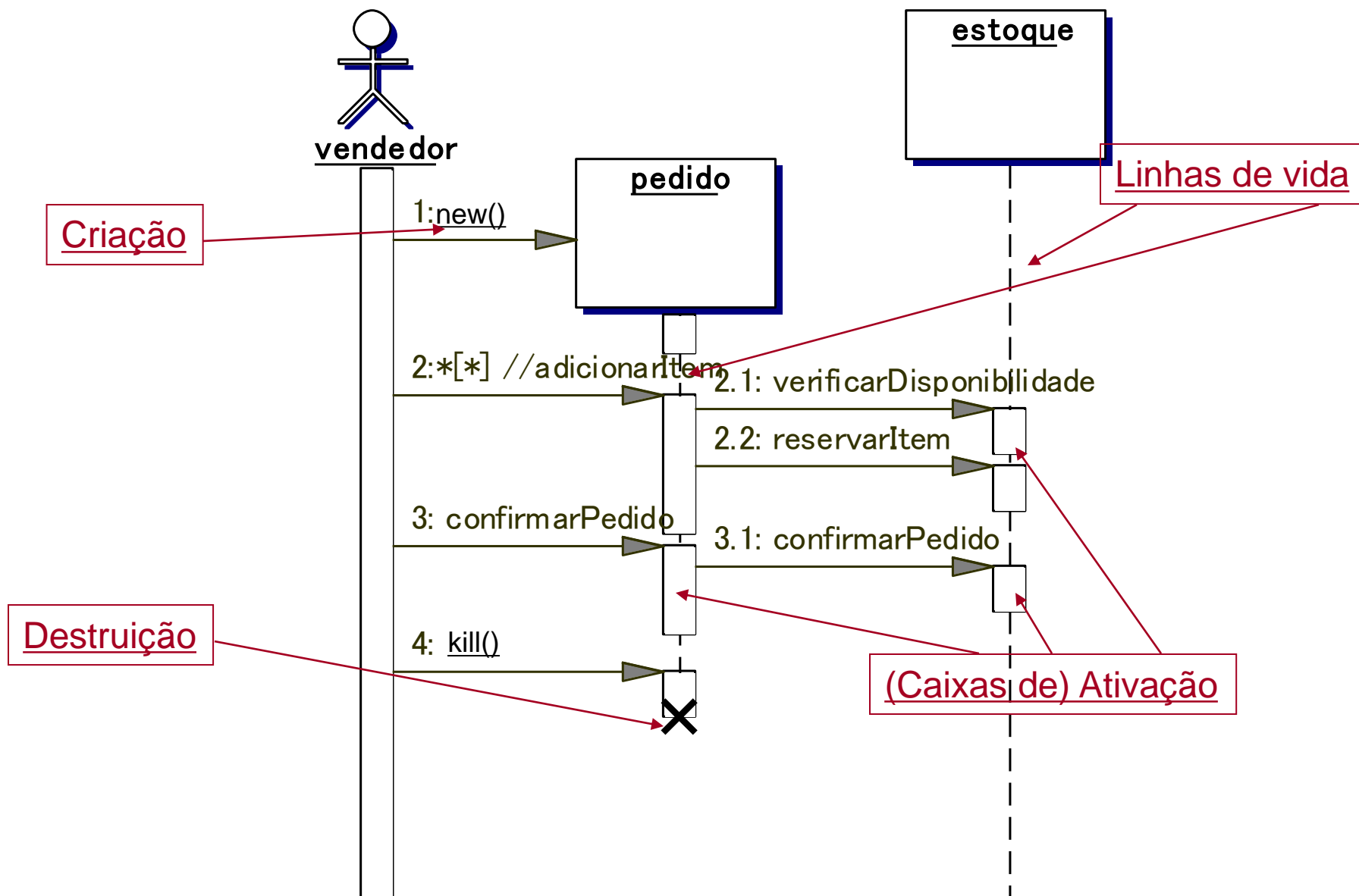
Ex.: joão:Dentista

:Floricultor (um objeto floricultor não identificado)

obj1: (um objeto obj1 sem classe definida)



- **Dimensão vertical** do diagrama
- Apresentam o **tempo de vida** dos objetos
- Pode apresentar a **ativação** ou a **desativação** dos objetos
 - Indicam que os objetos estão executando algo
 - Foco de controle
 - Caixas de ativação podem ser empilhadas
 - Indica chamada de método do próprio objeto
 - Objeto jose no slide anterior
- Podem representar a **criação** e a **destruição** de objetos

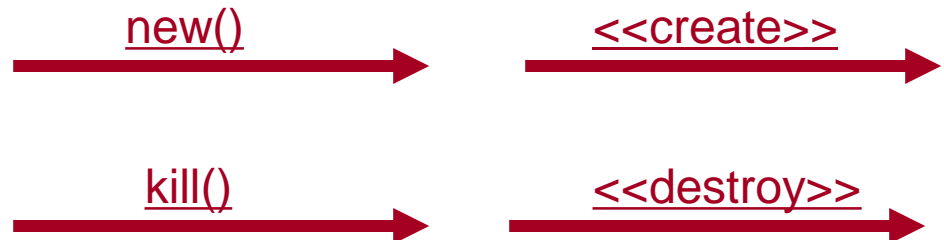





- Objetos interagem através da troca de mensagens
 - Setas sólidas que vão do objeto solicitante para o solicitado
 - Para o próprio objeto: auto-delegação
 - Rotulados com os nomes dos estímulos mais os argumentos (ou valores dos argumentos) do estímulo
- Sintaxe

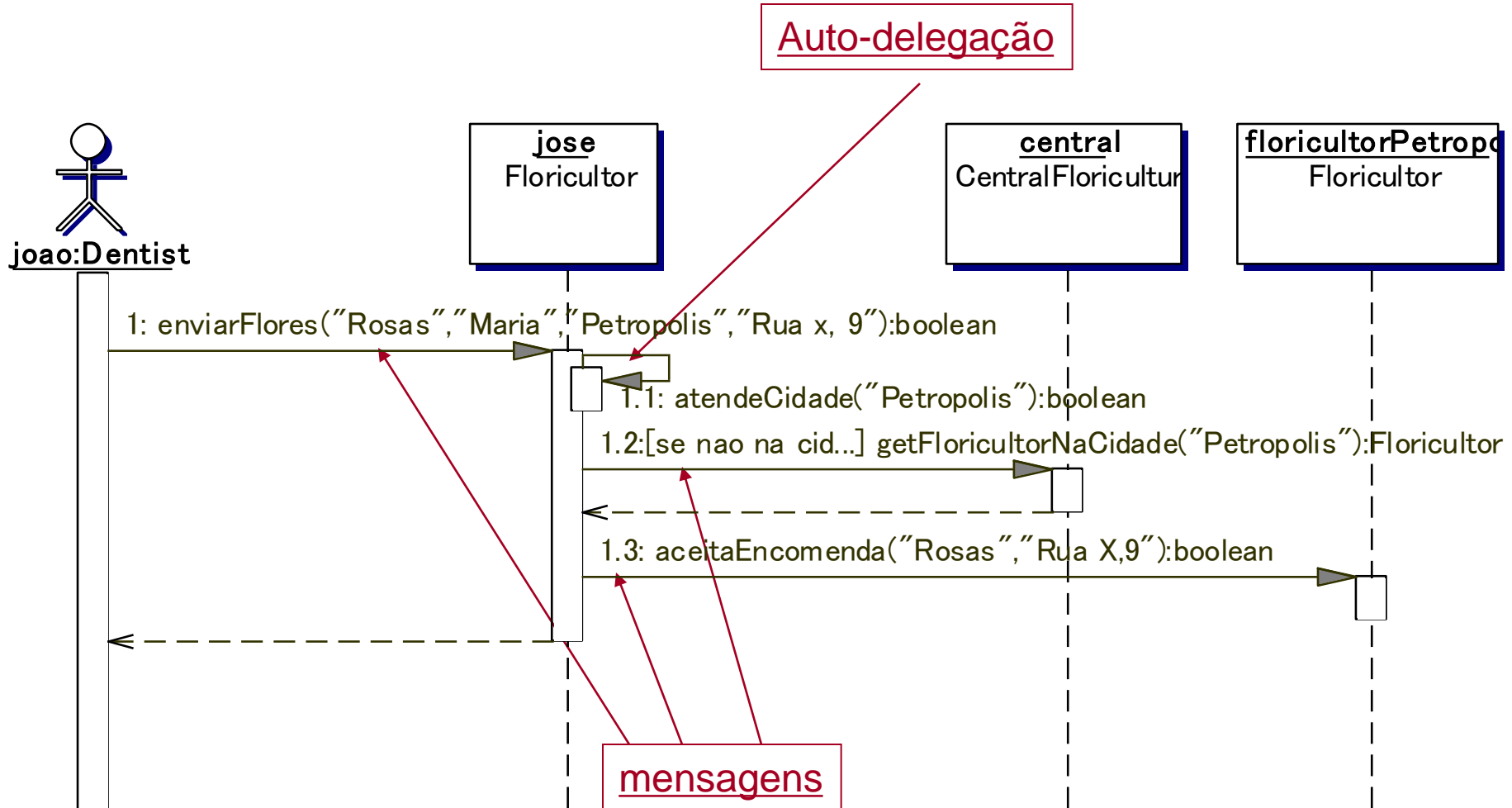
return := message(parameter:parameterType):returnType

- onde
 - **return** é o nome do valor de retorno
 - **message** é o nome da mensagem
 - **parameter** é o nome de um parâmetro da mensagem
 - **parameterType** é o nome do tipo desse parâmetro
 - **returnType** é o tipo do valor de retorno

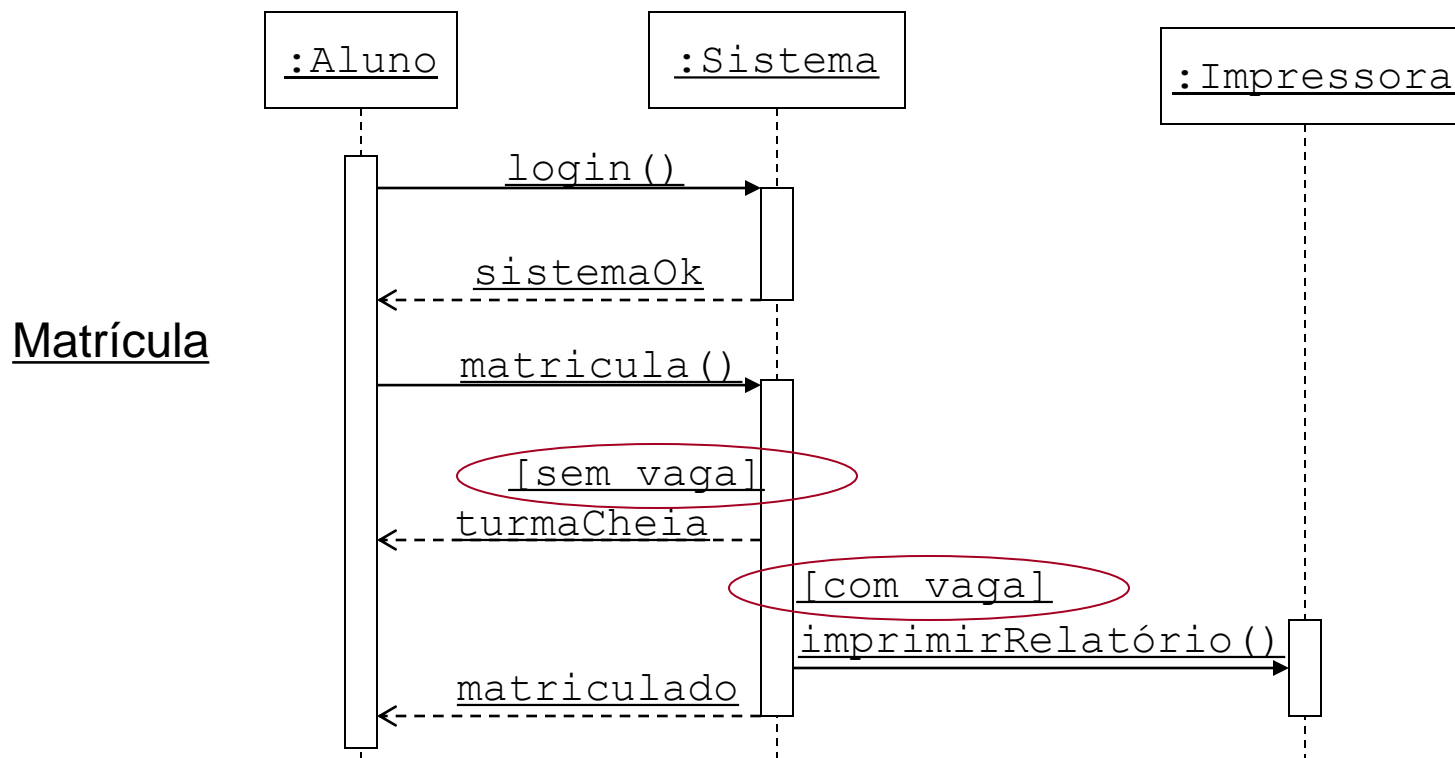
- Tipos de ação que uma mensagem pode representar
 - call
 - Invoca uma operação sobre um objeto
 - Objeto pode mandar uma chamada para si próprio
 - » Resultando na execução local de uma operação
 - return
 - Representa o retorno de um valor para o objeto que chamou a operação
 - Opcional
 - create
 - Criação de um objeto
 - destroy
 - Eliminação de um objeto



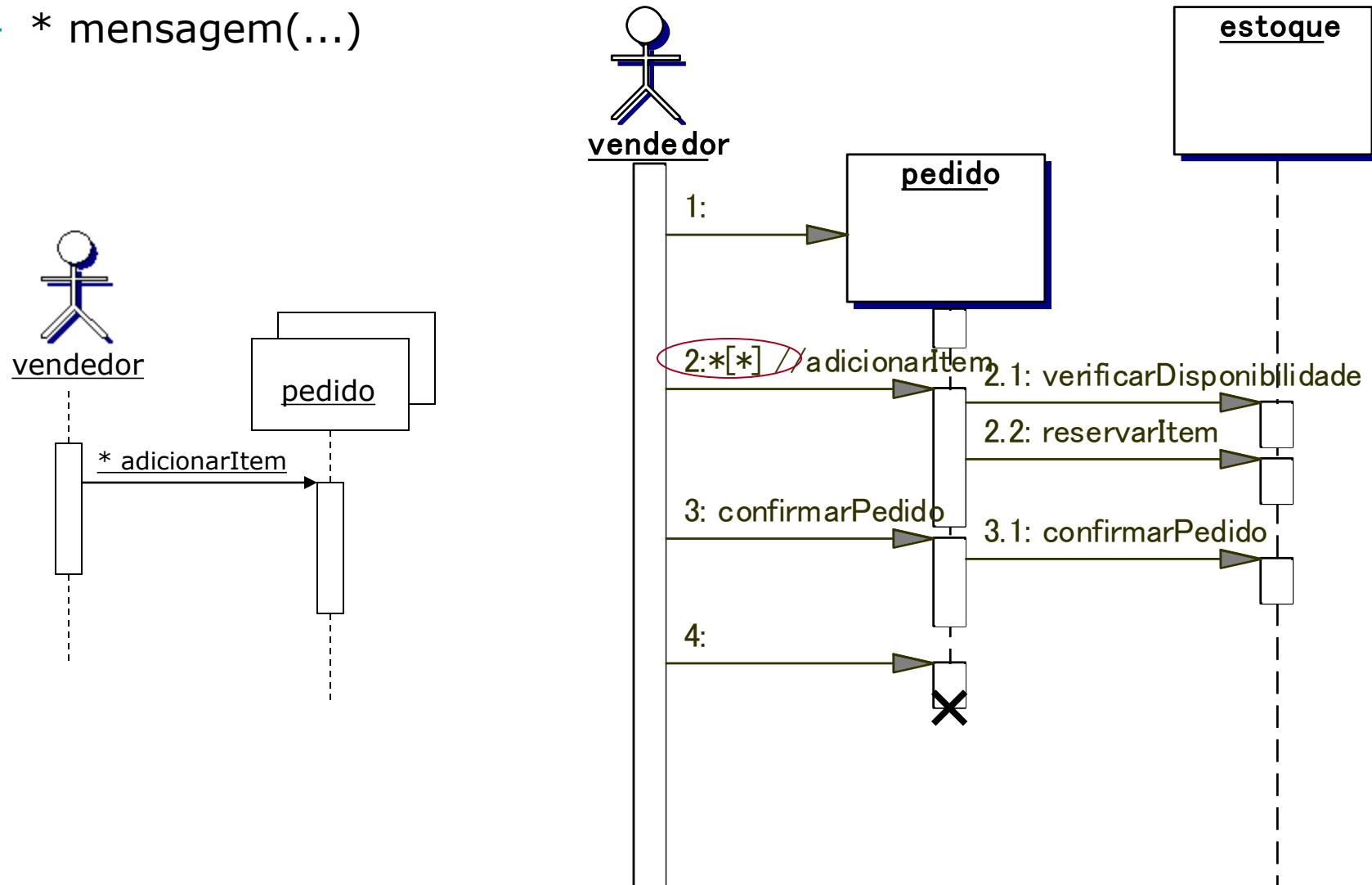
Símbolo	Significado
	Mensagem síncrona
	Mensagem assíncrona
	Mensagem de retorno (opcional)



- Mensagens podem apresentar condições de guarda
 - condições em que a mensagem é enviada
 - [condição de guarda]



- Uma mensagem pode ser enviada repetidas vezes
 - * mensagem(...)



- Período de tempo que o objeto executa uma ação
- Relação de controle entre ativação e o responsável pela sua invocação

- Escolher um **caso de uso**
- Identificar os **objetos** que fazem parte da **interação**
- Identificar o objeto que **começa** a interação
- Identificar as **mensagens** trocadas entre os objetos
- Identificar a **seqüência** destas mensagens

Diagrama de Atividade

- O objetivo do diagrama de atividades é mostrar o fluxo de atividades em um único processo. O diagrama mostra como uma atividade depende uma da outra.
- Enquanto os diagramas de sequência dão ênfase ao fluxo de controle de um objeto para outro, os diagramas de atividades dão ênfase ao fluxo de controle de uma atividade para outra;
- Representa os fluxos conduzidos por processamentos.
- É essencialmente um gráfico de fluxo, mostrando o fluxo de controle de uma atividade para outra. Comumente isso envolve a modelagem das etapas sequenciais em um processo computacional.

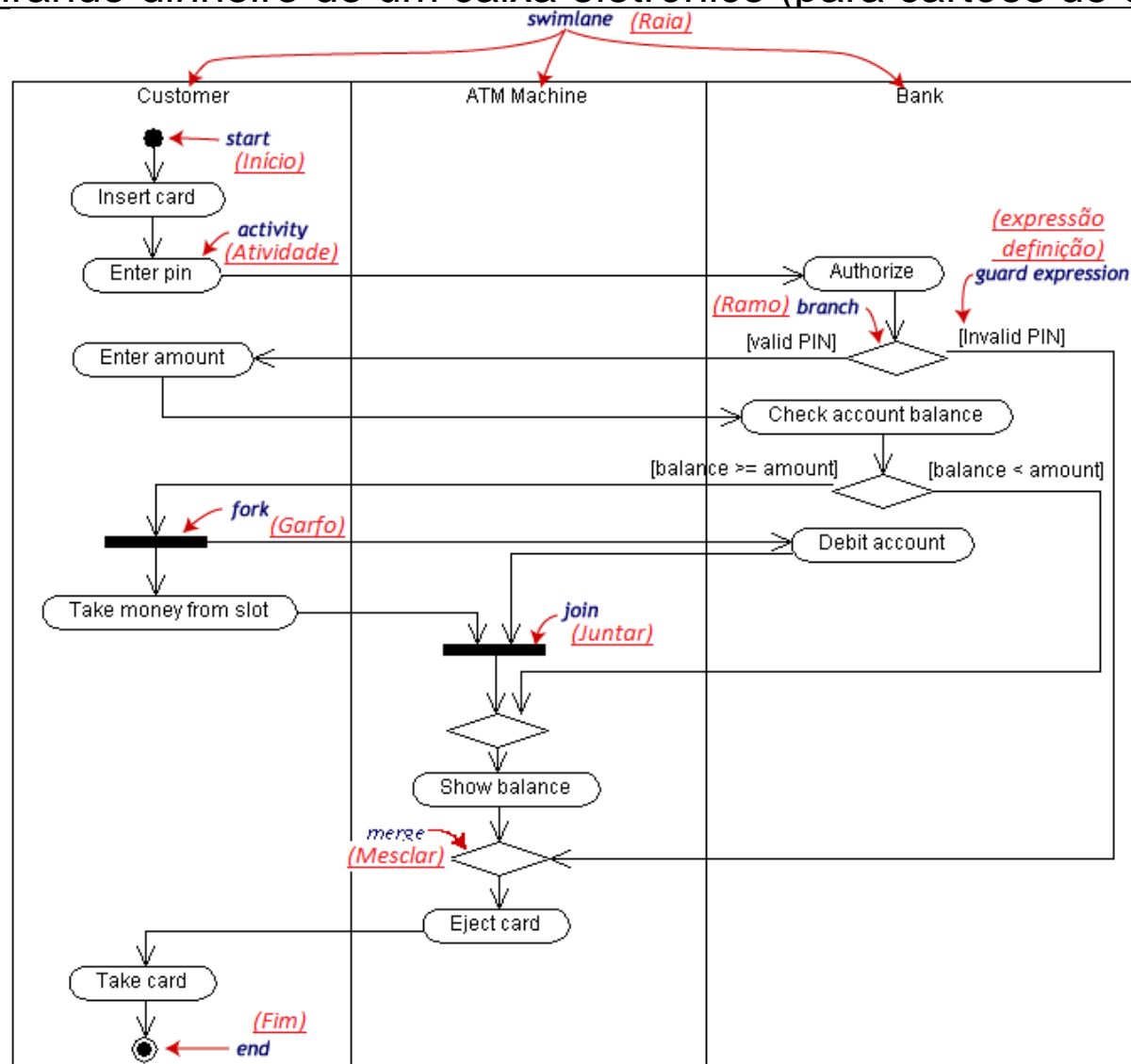
(Fonte: <https://www.purainfo.com.br/artigos/uml-diagrama-de-atividades/>)

- Atividades: Comportamento a ser realizado.
- Sub-atividade: Execução de uma sequência não atômica de atividades.
- Transição: Fluxo de uma atividade para outra.
- Ação: Transformação.
- Decisão: Dependendo de uma condição, mostra as diferentes transições.
- Raia: Diferenciação de unidades organizacionais.
- Bifurcação (Fork): Separa uma transição em várias transições executadas ao mesmo tempo.
- Sincronização (Join): Concatenação de transições vindas do Fork.
- Objecto: O [objecto](#) da atividade.
- Envio de sinal: Transição pra um meio externo, por exemplo, um [hardware](#).
- Recepção de sinal: Recepção do envio.
- Região: Agrupamento de uma ou mais atividades.
- Exceção: Atividades que ocorrerem em decorrência de uma exceção.

(Fonte: wikipedia.org)

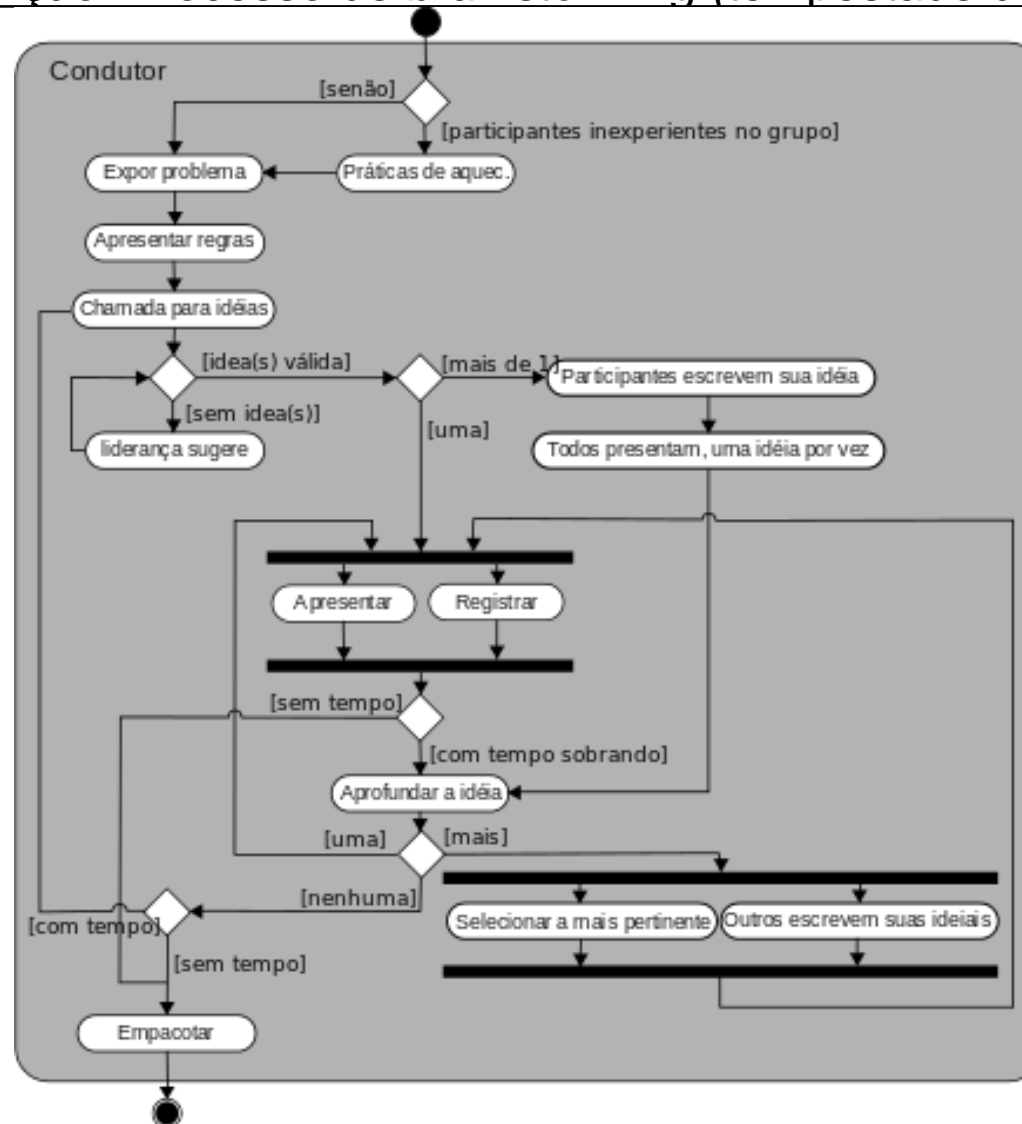
Diagrama de Atividade - exemplo

Descrição: Retirando dinheiro de um caixa eletrônico (para cartões de crédito).



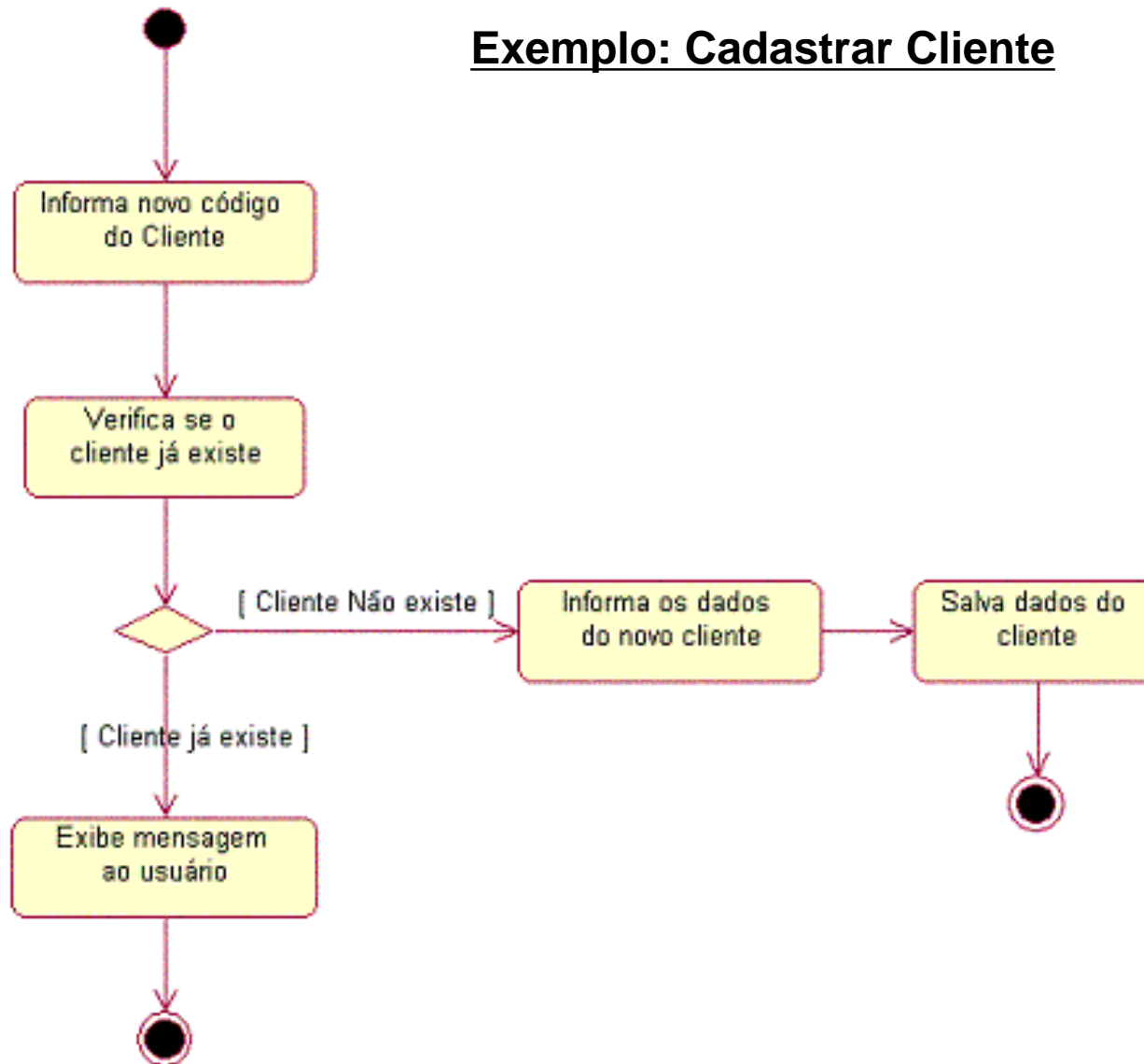
- Um diagrama de atividades é essencialmente um fluxograma que dá ênfase à atividade que ocorre ao longo do tempo. Você pode considerar um diagrama de atividades como um diagrama de sequência cujo interior é revelado;
- Um diagrama de sequência observa os objetos que passam mensagens;
- Um diagrama de atividade observa as operações passadas entre os objetos;
- Mostra o fluxo de uma atividade para outra;
- Uma atividade é uma execução em andamento;
- As atividades resultam em uma ação;
- As ações abrangem a chamada a outras operações, enviando um sinal, criando ou destruindo um objeto.

Descrição: Processo de *brainstorming* (tempestade de idéias)



Fonte: https://pt.wikipedia.org/wiki/Diagrama_de_atividade (31/7/18)

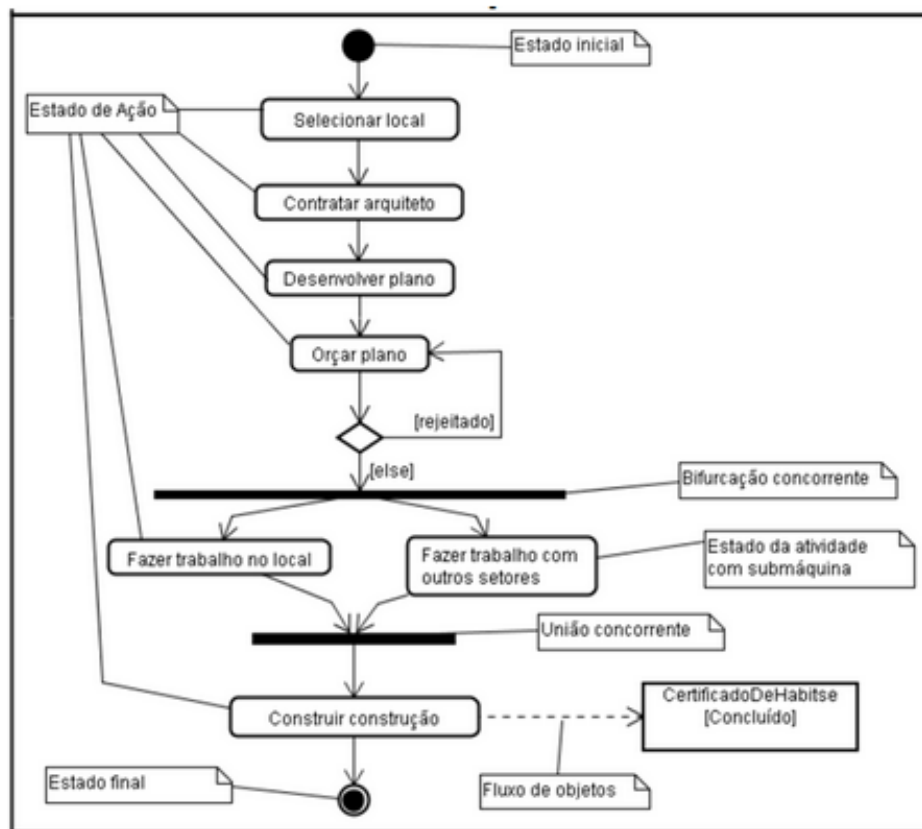
Exemplo: Cadastrar Cliente



(Fonte: <https://www.purainfo.com.br/artigos/uml-diagrama-de-atividades/>)

Considere o fluxo de trabalho associado à construção de uma casa. Primeiro, você seleciona um local. A seguir, contrata um arquiteto para projetar sua casa. Uma vez definida a planta, seu desenvolvedor determina os custos da casa. Após concordar com um preço e com uma forma de pagamento, a construção pode começar. As licenças são tiradas, o terreno é cavado, a fundação é cimentada, as estruturas são erguidas e assim por diante até tudo ficar pronto. Você então recebe as chaves e um certificado de habite e toma posse da casa.

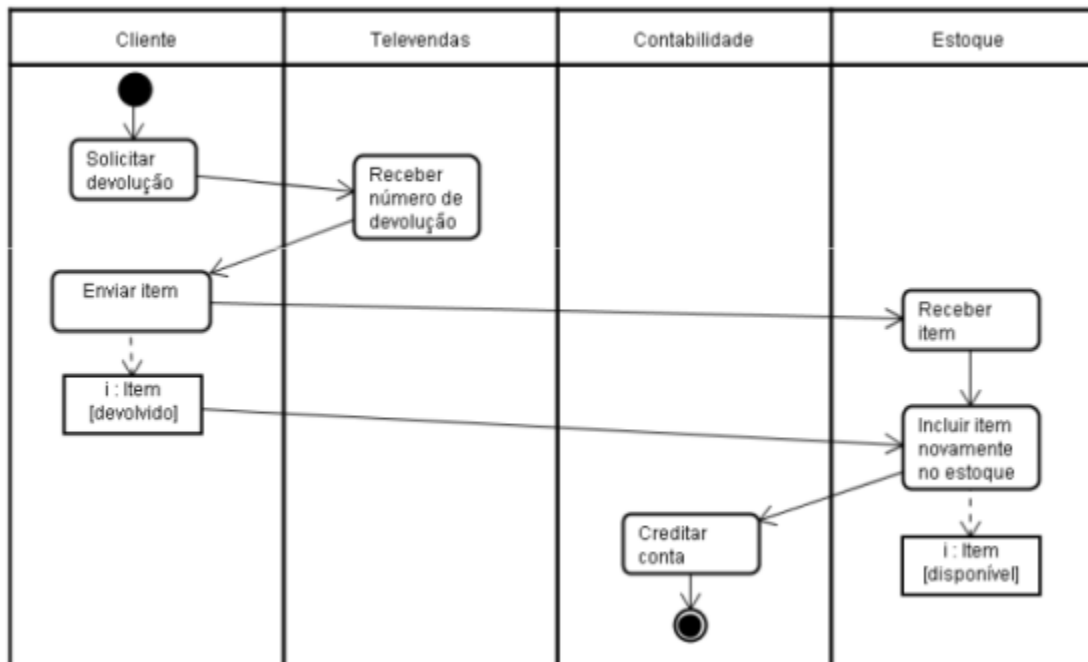
Embora seja uma grande simplificação do que realmente acontece em um processo de construção, essa descrição capta o percurso crítico do fluxo de trabalho correspondente:



(Fonte: <https://www.purainfo.com.br/artigos/uml-diagrama-de-atividades/>)

Por exemplo, o diagrama a seguir mostra um diagrama de atividades para uma empresa de varejo, que especifica o fluxo de trabalho envolvido quando um cliente devolve um item de um pedido postal. O trabalho começa com a ação solicitar devolução do (mero de devolução), retorna ao cliente (item e depois Incluir item novamente no estoque (creditar conta).

Conforme o diagrama, acompanha o item devolvido para o estado disponível.

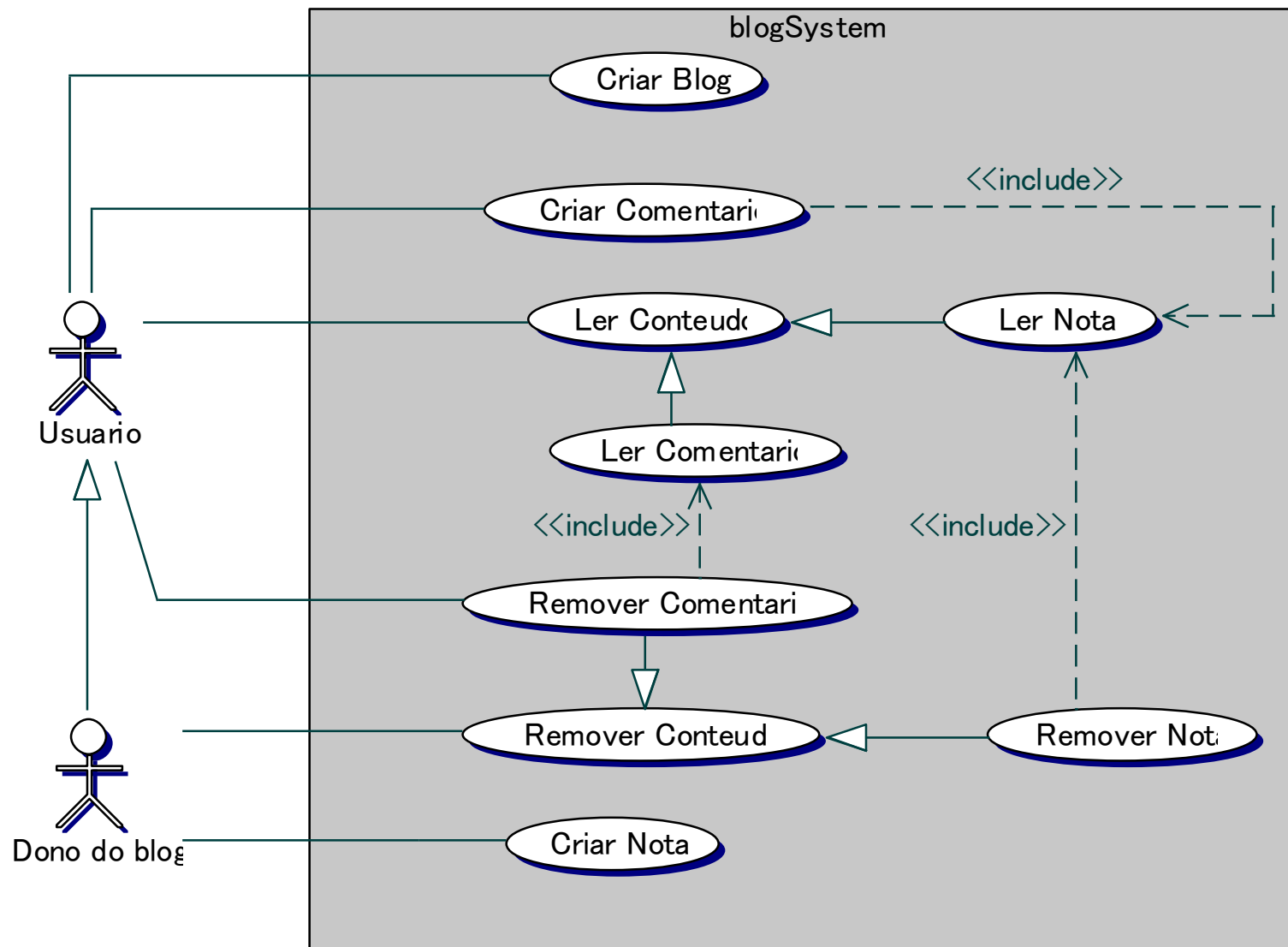


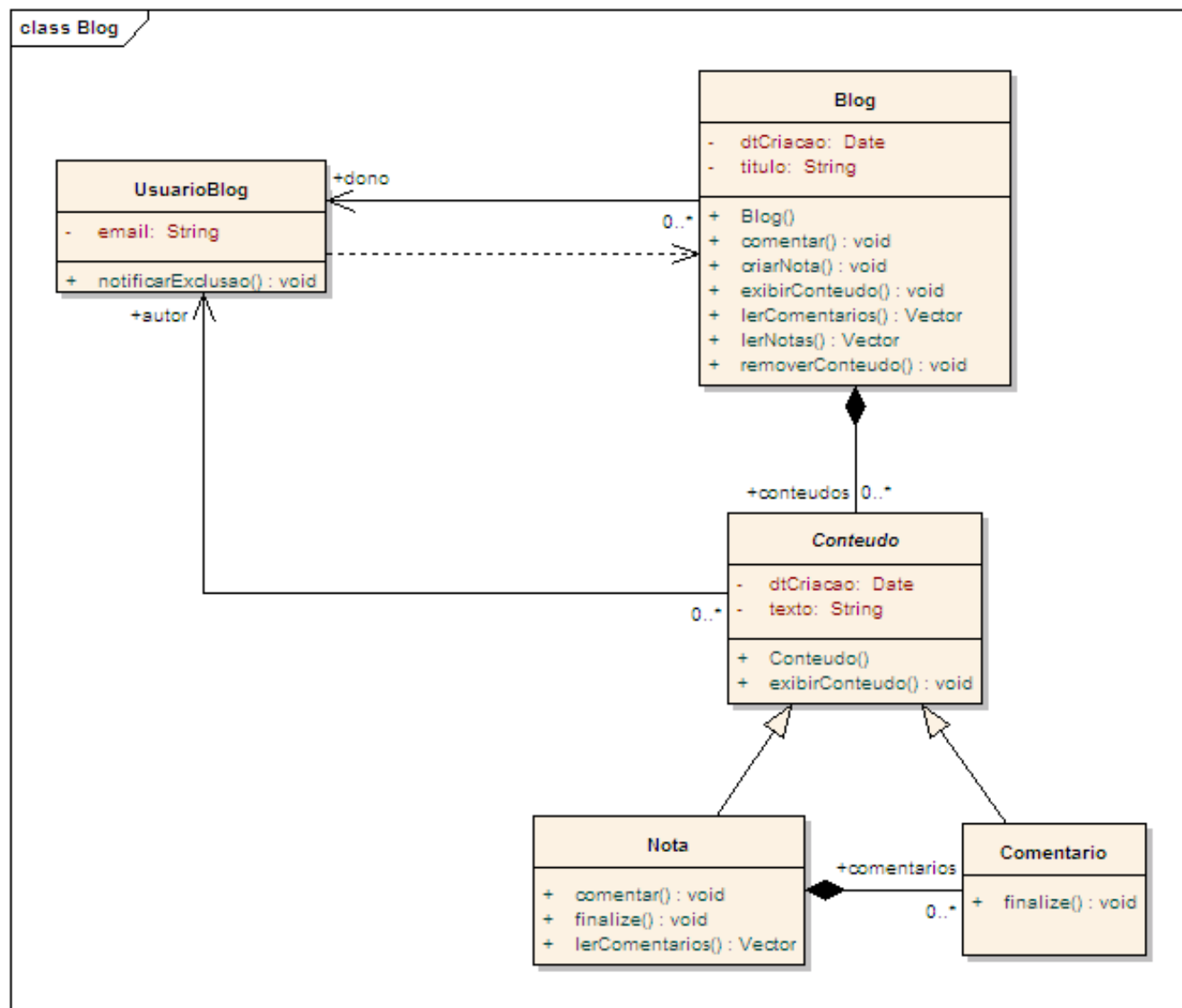
(Fonte: <https://www.purainfo.com.br/artigos/uml-diagrama-de-atividades/>)

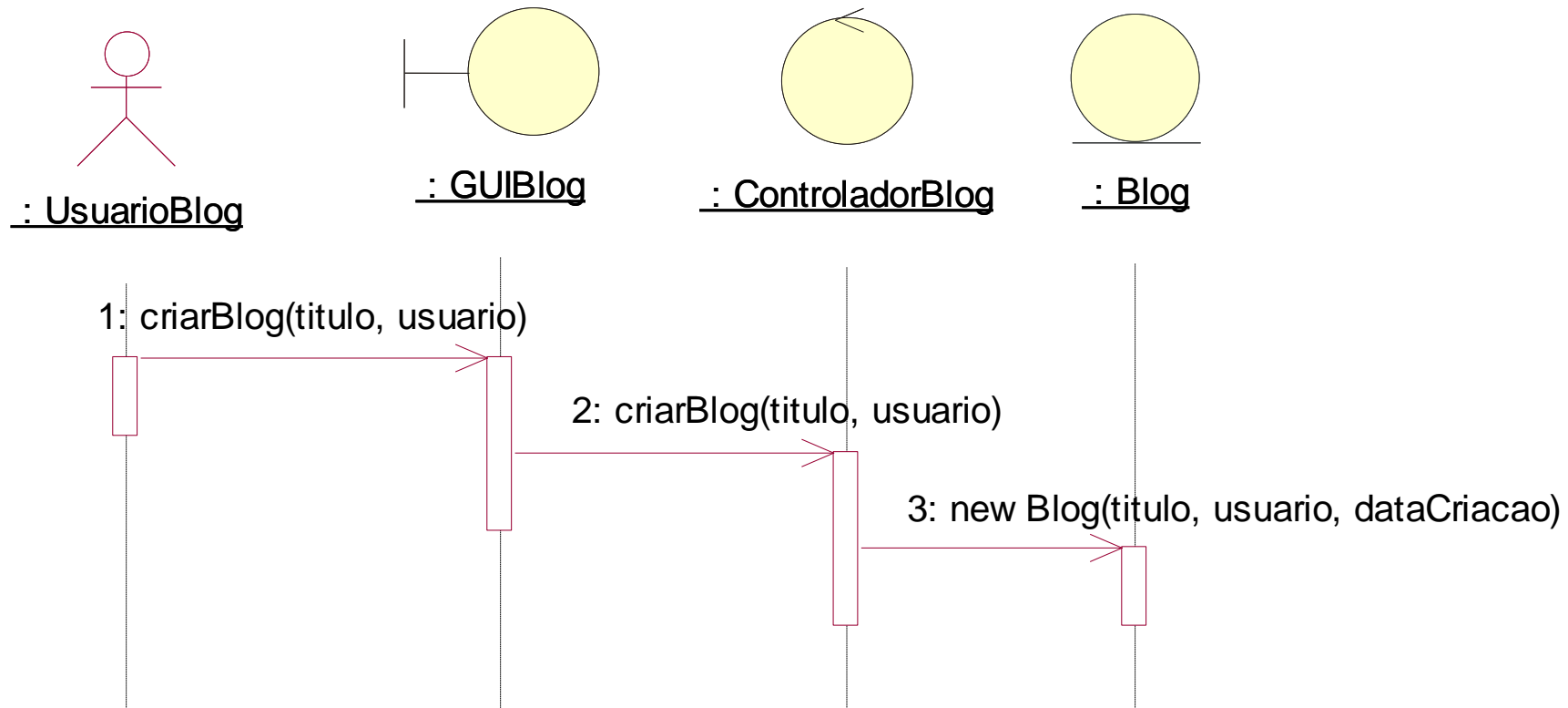
- Um *blog* tem um título e uma data de criação e além disso é um conjunto de conteúdos.
- Estes conteúdos (mensagens) podem ser notas ou comentários sobre as notas. Tanto notas quanto comentários têm características comuns como o texto e a data de sua criação.
- Todo usuário possui:
 - E-mail (deve ser único, ou seja, não há mais de um usuário com o mesmo e-mail)

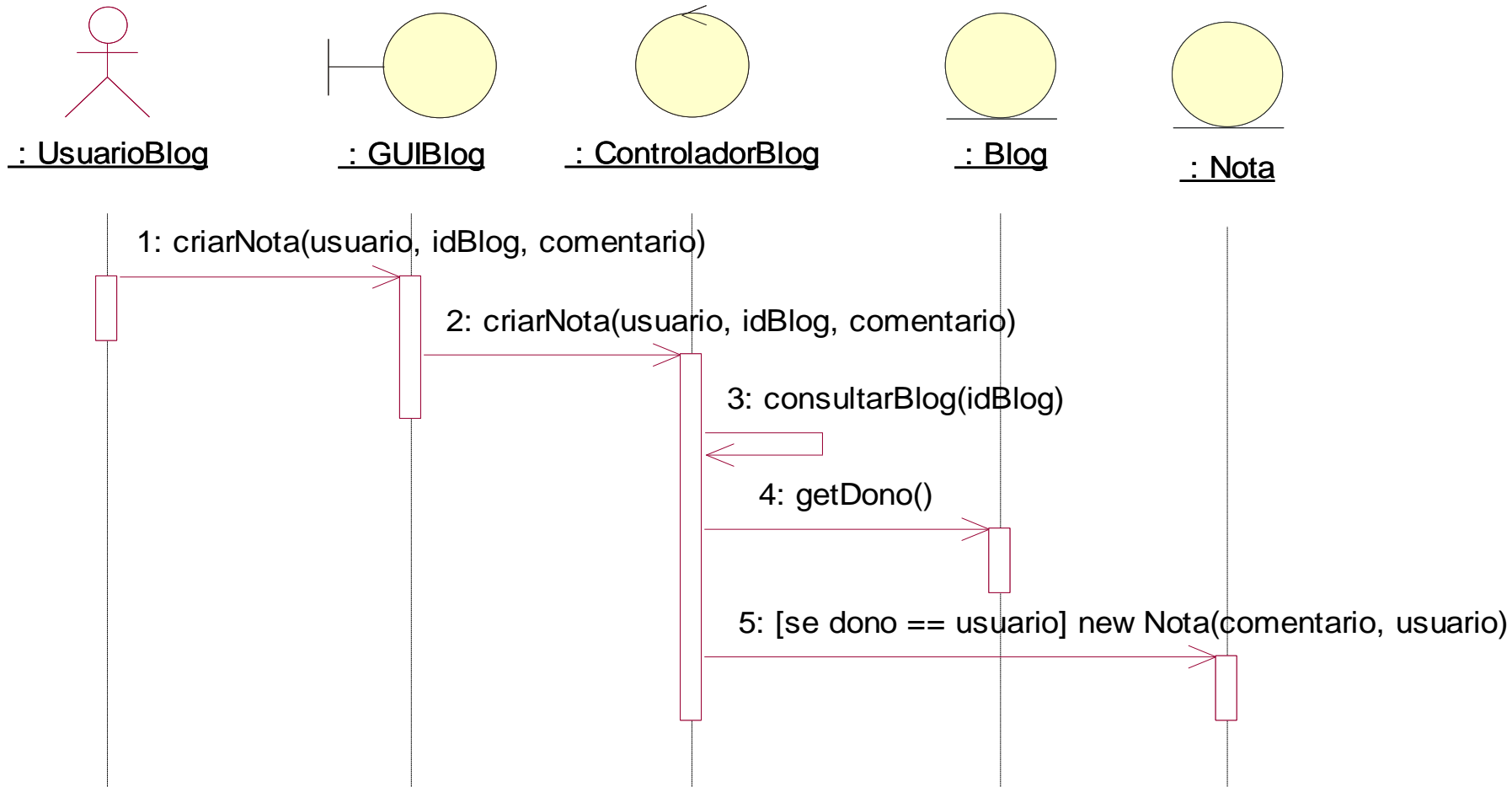
- Um *blog* tem um título e uma data de criação e além disso é um conjunto de conteúdos.
- Estes conteúdos (mensagens) podem ser notas ou comentários sobre as notas. Tanto notas quanto comentários têm características comuns como o texto e a data de sua criação.
- Todo usuário possui:
 - E-mail (deve ser único, ou seja, não há mais de um usuário com o mesmo e-mail)
 - Permitir a criação de blogs
- Permitir a utilização de blogs
 - Qualquer usuário pode ler conteúdos
 - Somente o dono do blog pode criar notas
 - Qualquer usuário pode criar comentários. Para criar um comentário o usuário precisa ler as notas.
 - Somente o dono do blog pode remover conteúdos. Para remover um conteúdo ele precisará ler o conteúdo. Caso ele remova um comentário, o autor do comentário deve ser notificado por e-mail.

- Desenvolva:
 - CASO DE USO
 - DIAGRAMA DE CLASSES
 - DIAGRAMA DE SEQUÊNCIA









- Omondo – Plugin para Eclipse - <http://www.omondo.com/>
- Astah – <http://astah.net>
- Together - http://www.borland.com/products/downloads/download_together.html
- IBM Rational Rose - <http://www.ibm.com/software/rational>
- ...

- Slides adaptado das Notas de aula do Laboratório de Engenharia de Software da PUC – Rio, disponível em <http://wiki.les.inf.puc-rio.br/index.php/>
- **Guedes, G., UML 2.0 - Uma Abordagem Prática . Rio de Janeiro : Novatec /2009.**
- Cockburn, A., *Writing Effective Use Cases*, Addison-Wesley, 2001.
- Fowler, M e Scott, K., *UML Distilled – A Brief Guide to the standard Object Modeling Language*, Addison Wesley Longman, 2002
- Booch, G., Rumbaugh, J. and Jacobson, I., *Unified Modeling Language User Guide*, 2nd Edition, Addison-Wesley Object Technology Series, 2005.
- <https://www.youtube.com/watch?v=2aE9NYZQHYk> (Acesso em 2/8/2018)
- https://www.youtube.com/watch?v=X1tgd97_i2A (Acesso em 2/8/2018)