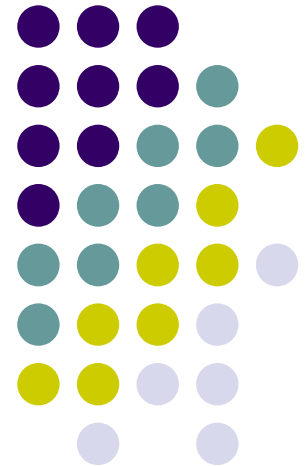


Aula 10:

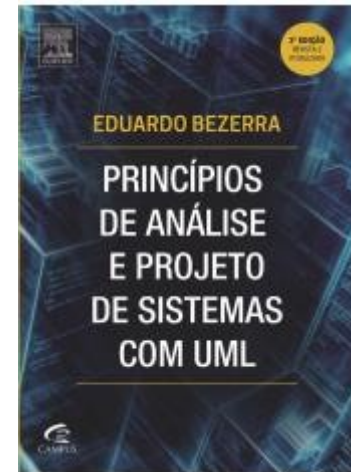
Diagramas de Interações



Prof. Fabiano Cutigi Ferrari
2º semestre de 2022

Notas Iniciais

- Preparado com base nos materiais a seguir*:
 - Slides disponibilizados em conjunto com o livro
 - Eduardo BEZERRA: Princípios de Análise e Projeto de Sistemas com UML, 3ª ed., Campus/Elsevier (2015).
 - Notas de aula e slides elaborados pelo professor, e outros materiais disponíveis na Web

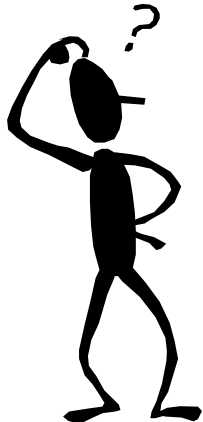


* Notas de rodapé ajudam a identificar os slides produzidos por Bezerra (2015).

- Modelo de Interações
- Diagrama de Sequência
- Diagrama de Comunicação
- Modularização de Interações
- Construção do Modelo de Interações
- Exemplos

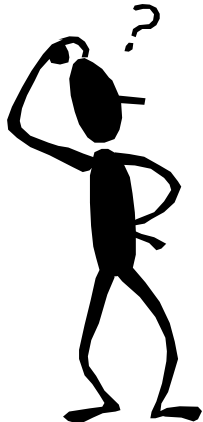
Contexto

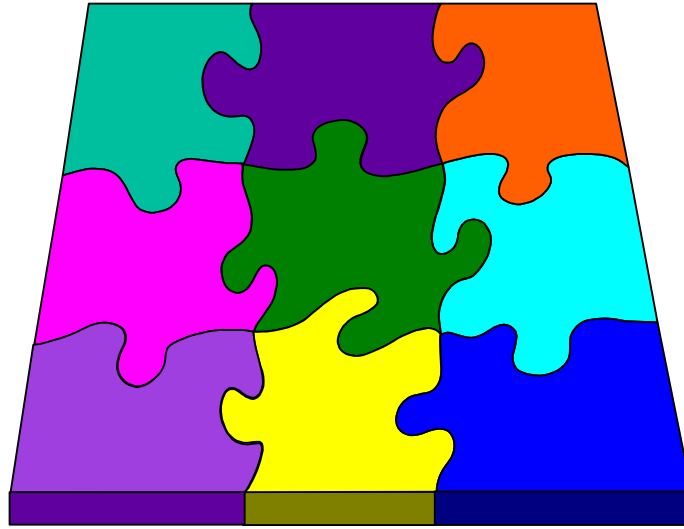
- O objetivo dos modelos de análise é fornecer um entendimento do problema correspondente ao software a ser desenvolvido.
- Entretanto, esses modelos deixam algumas perguntas sem respostas.
- No modelo de casos de uso:
 - Quais são as operações que devem ser executadas internamente ao sistema?
 - A que classes estas operações pertencem?
 - Quais objetos participam da realização deste caso de uso?



Contexto

- No modelo de classes de análise:
 - De que forma os objetos colaboram para que um determinado caso de uso seja realizado?
 - Em que ordem as mensagens são enviadas durante esta realização?
 - Que informações precisam ser enviadas em uma mensagem de um objeto a outro?
 - Será que há responsabilidades ou mesmo classes que ainda não foram identificadas?





Modelo de Interações

Modelo de Interações

- Para responder às questões anteriores, o **modelo de interações** deve ser criado.
- Esse modelo representa mensagens trocadas entre objetos para a execução de cenários dos casos de uso do sistema.
- A construção dos **diagramas de interação** é uma consolidação do entendimento dos aspectos dinâmicos do sistema.
- A modelagem de interações é uma parte da **modelagem dinâmica** de um software OO.

Diagramas de interação representam como o sistema age internamente para que um ator atinja seu objetivo na realização de um caso de uso. A modelagem de um software OO normalmente contém diversos diagramas de interação. O conjunto de todos os diagramas de interação de um sistema constitui o seu **modelo de interações**.

Objetivos do Modelo de Interações

- Obter informações adicionais para completar e aprimorar outros modelos (principalmente o modelo de classes)
 - Quais as operações de uma classe?
 - Quais os objetos participantes da realização de um caso de uso (ou cenário deste)?
 - Para cada operação, qual a sua assinatura?
 - Uma classe precisa de mais atributos?
- Fornecer aos programadores uma visão detalhada dos objetos e mensagens envolvidos na realização dos casos de uso.

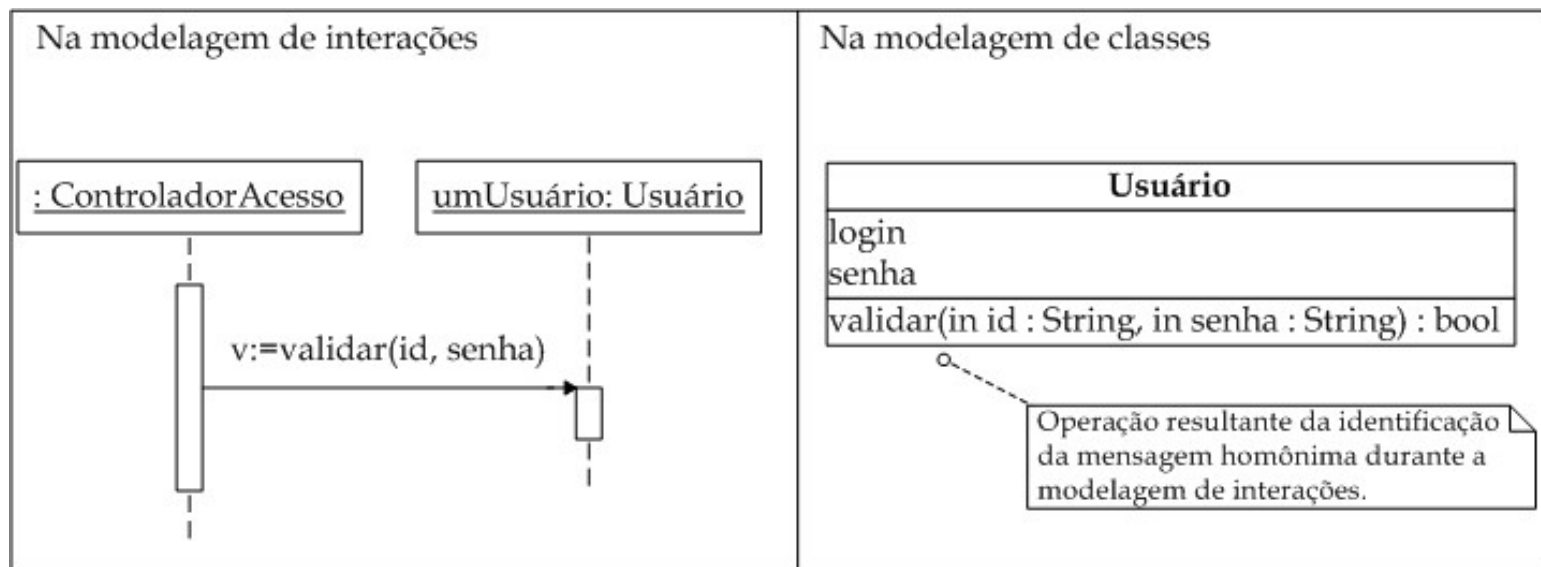
- O conceito básico da interação entre objetos é a *mensagem*.
- Um sistema OO é uma rede de objetos que trocam mensagens.
 - Funcionalidades são realizadas pelos objetos, que só podem interagir através de mensagens.
 - Um objeto envia uma mensagem para outro objeto quando o primeiro deseja que o segundo realize alguma tarefa.
- O fato de um objeto “precisar de ajuda” indica a necessidade de este enviar mensagens.

- Na construção de diagramas de interação, mensagens de um objeto a outro implicam em operações que classes devem ter.

Uma mensagem representa a requisição de um objeto remetente a um objeto receptor para que este último execute alguma operação definida para sua classe. Essa mensagem deve conter informação suficiente para que a operação do objeto receptor possa ser executada.




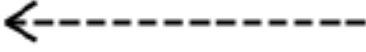
Mensagens versus Responsabilidades

- Qual o objetivo da construção dos diagramas de interação?
 - Identificar **mensagens** e, em última análise, **responsabilidades** (**operações e atributos**)



Uma mensagem implica na existência de uma operação no objeto receptor. A resposta do objeto receptor ao recebimento de uma mensagem é a execução da operação correspondente.

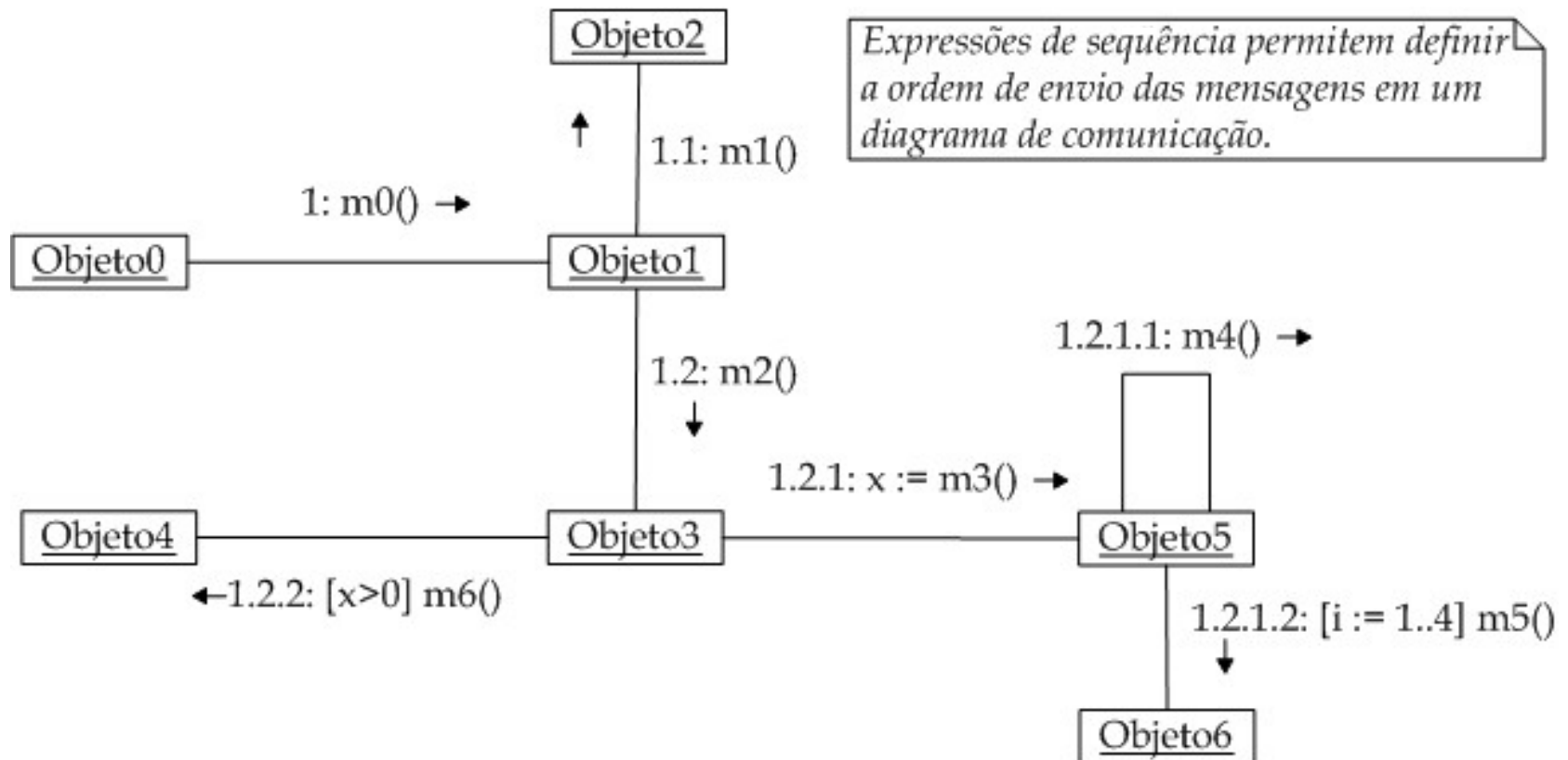
Tipos de Mensagens

 Synchronous	 Asynchronous
 Creation	 Reply

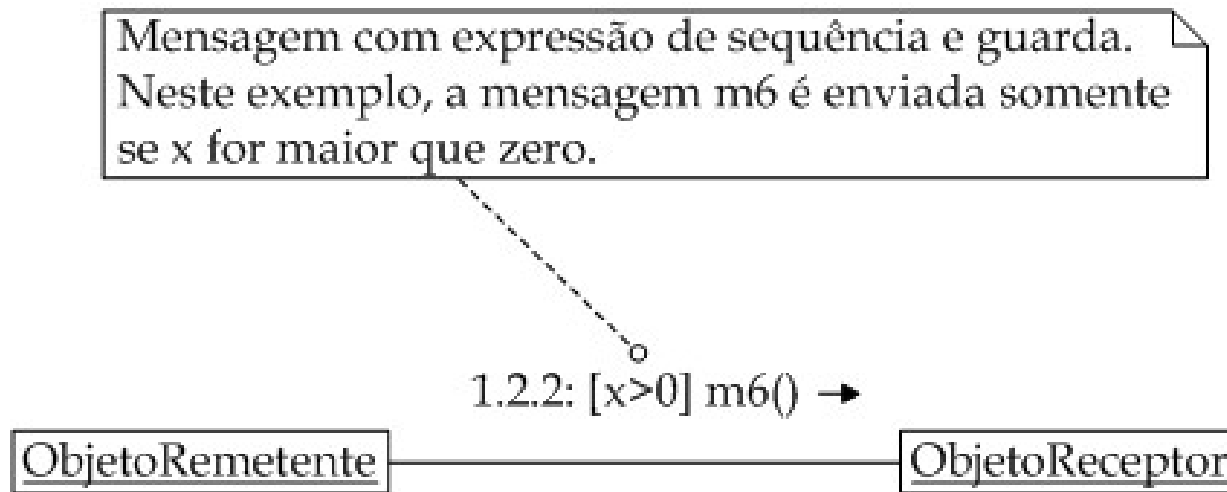
Sintaxe UML para mensagens e exemplos

- O único termo obrigatório corresponde ao **nome** da mensagem.
- Exemplos:
 - Mensagem simples, sem cláusula alguma.
1: adicionarItem(item)
 - Mensagem com cláusula de condição.
3: [a > b] trocar(a, b)
 - Mensagem com cláusula de iteração e com limites indefinidos.
2*: desenhar()
 - Mensagem com cláusula de iteração e com limites definidos.
2: [i := 1..10] figuras[i].desenhar()
 - Mensagem aninhada com retorno armazenado na variável x.
1.2.1: x := selecionar(e)

Exemplos (sintaxe UML para Mensagens)



Exemplos (sintaxe UML para Mensagens)

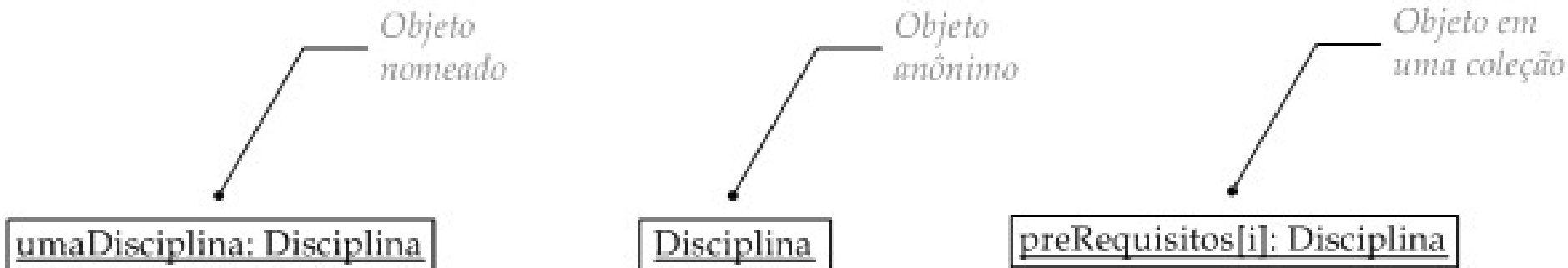


Notação para objetos

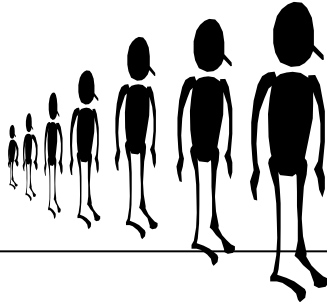
- Objetos são representados em um diagrama de interação como sendo objetos anônimos ou objetos nomeados, dependendo da situação.
- Elementos de uma coleção também podem ser representados.

Notação para objetos

- Classes também podem ser representadas.
 - Para o caso de mensagens enviadas para a classe.
 - Uma mensagem para uma classe dispara a execução de uma *operação estática*.
 - A representação de uma classe em um diagrama de sequência é a mesma utilizada para objetos, porém o nome da classe não é sublinhado

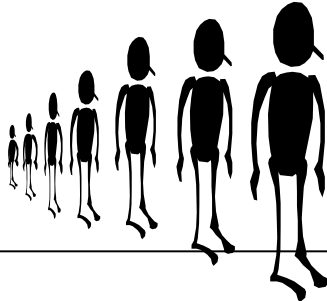


Multiobjetos



- Um **multiobjeto** é o nome que a UML dá para uma *coleção* de objetos de uma mesma classe. Pode ser utilizado para:
 - representar o lado muitos de uma associação de conectividade um para muitos.
 - representar uma lista (temporária ou não) de objetos sendo formada em uma colaboração.

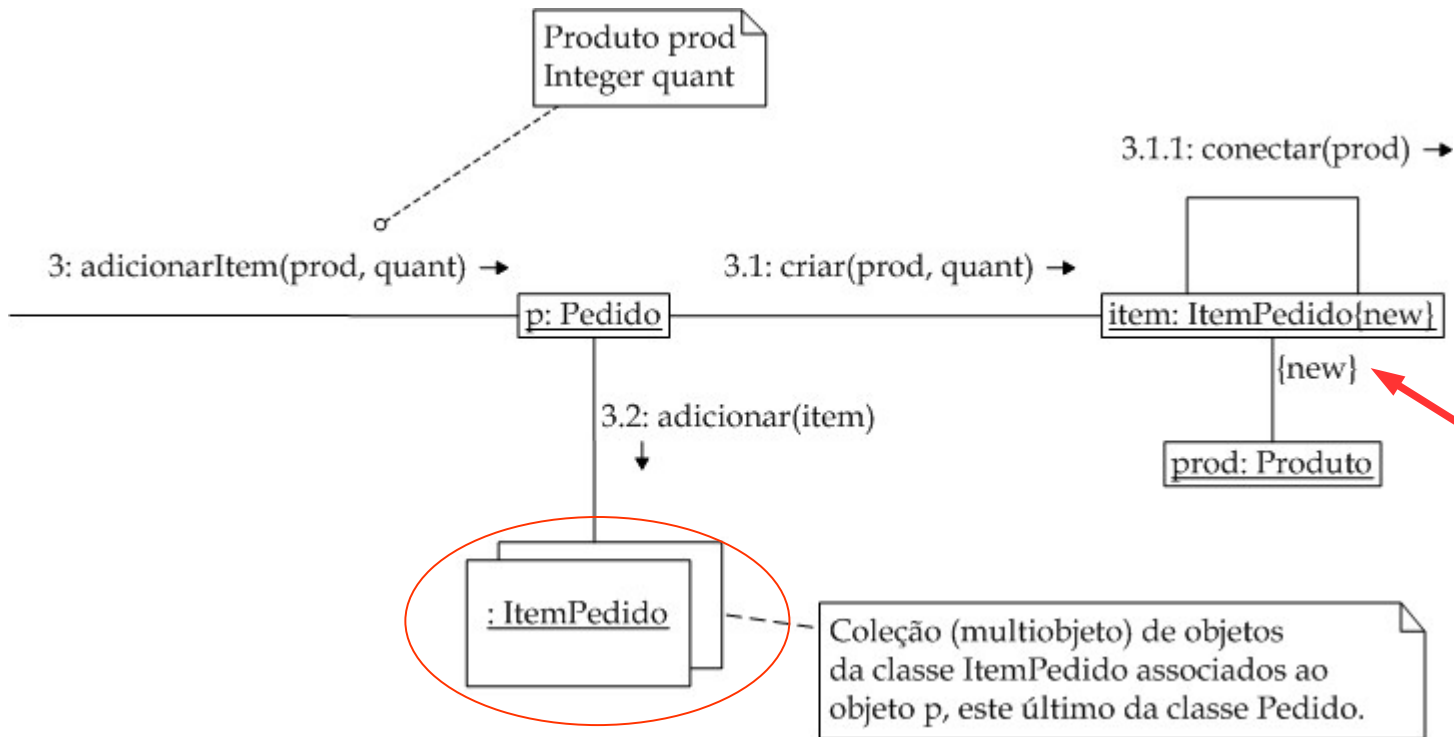
Multiobjetos



- Um multiobjeto é representado na UML através de dois retângulos superpostos.
 - A superposição dos retângulos evita a confusão com a notação usada para objetos.
 - O nome do multiobjeto é apresentado no retângulo que fica por cima e segue a mesma nomenclatura utilizada para objetos.
 - Convenção: usar o nome da classe de seus elementos para nomear o multiobjeto.

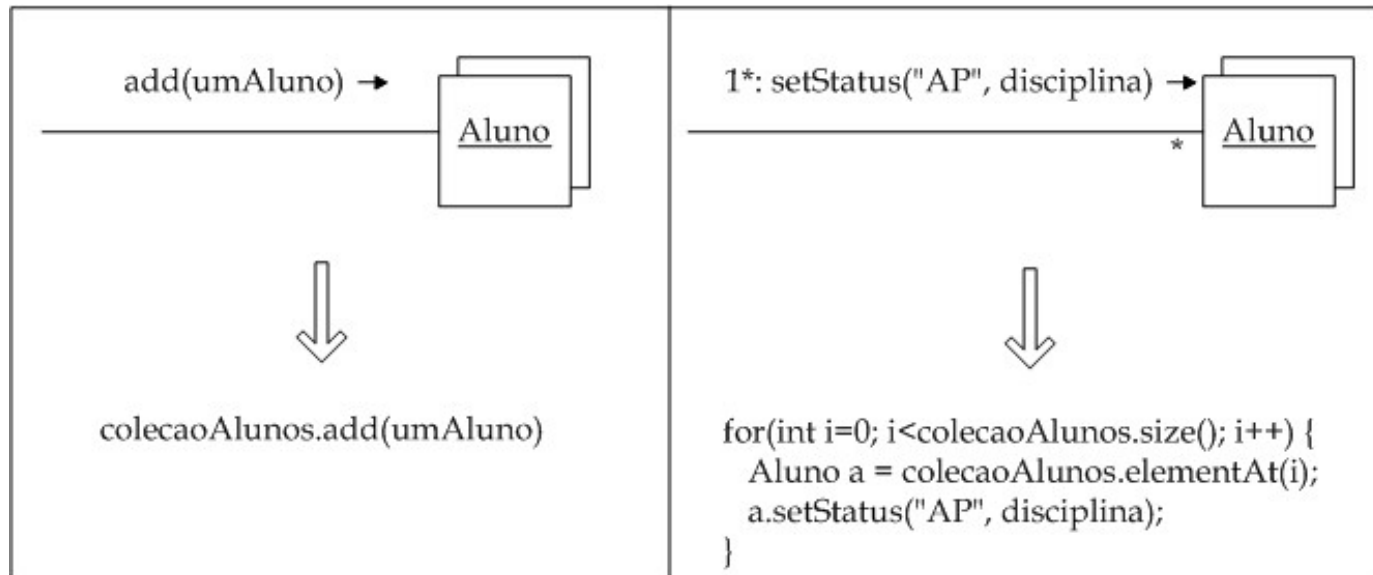
Notação para multiobjetos

- Um multiobjeto é representado graficamente na UML através de dois retângulos superpostos.



Mensagens para Objetos/Coleção

- Uma mensagem pode ser enviada para um multiobjeto, ou pode ser enviada para um único objeto (elemento) do multiobjeto.
- Quando o símbolo de iteração não é usado, convencionou-se que a mensagem está sendo enviada para o próprio multiobjeto.
- Exemplo:



Implementação de multiobjetos

- Multiobjetos são normalmente implementados através de alguma estrutura de dados que manipule uma coleção.
- Portanto, algumas mensagens típicas que podemos esperar que um multiobjeto aceite são as seguintes:
 - Posicionar o cursor da coleção no primeiro elemento.
 - Retornar o i-ésimo objeto da coleção.
 - Retornar o próximo objeto da coleção.
 - Encontrar um objeto de acordo com um identificador único.
 - Adicionar um objeto na coleção.
 - Remover um objeto na coleção.
 - Obter a quantidade de objetos na coleção.
 - Retornar um valor lógico que indica se há mais objetos a serem considerados.

Implementação de multiobjetos (cont)

- A interface **List** da linguagem Java apresenta operações típicas de um multiobjeto.

```
public interface List<E> extends Collection<E> {  
    E get(int index);  
    E set(int index, E element);  
    boolean add(E element);  
    void add(int index, E element);  
    E remove(int index);  
    abstract boolean addAll(int index, Collection<? extends E> c);  
    int indexOf(Object o);  
    int lastIndexOf(Object o);  
    ListIterator<E> listIterator();  
    ListIterator<E> listIterator(int index);  
    List<E> subList(int from, int to);  
}
```

Tipos de diagrama de interação

- Há três tipos de diagrama de interação na UML 2.0: **diagrama de sequência**, **diagrama de comunicação** e **diagrama de visão geral da interação**.
 - O diagrama de sequência e o diagrama de comunicação são equivalentes.

Diagrama de sequência: foco nas mensagens enviadas no decorrer do tempo.

Diagrama de comunicação: foco nas mensagens enviadas entre objetos que estão relacionados.

Diagrama de visão geral de interação. Pode ser utilizado para apresentar uma visão geral de diversas interações entre objetos, cada uma delas representada por um diagrama de interação. Esse diagrama é útil para ***modularizar*** a construção do diagramas de sequência (ou de comunicação).

Tipos de diagrama de interação

- Há três tipos de diagrama de interação na UML 2.0: **diagrama de sequência**, **diagrama de comunicação** e **diagrama de visão geral da interação**.
 - O diagrama de sequência e o diagrama de comunicação são equivalentes.

Diagrama de sequência: foco nas mensagens enviadas no decorrer do tempo.

Diagrama de comunicação: foco nas mensagens enviadas entre objetos que estão relacionados.

Diagrama de visão geral de interação. Pode ser utilizado para apresentar uma visão geral de diversas interações entre objetos, cada uma delas representada por um diagrama de interação. Esse diagrama é útil para *modularizar* a construção do diagramas de sequência (ou de comunicação).

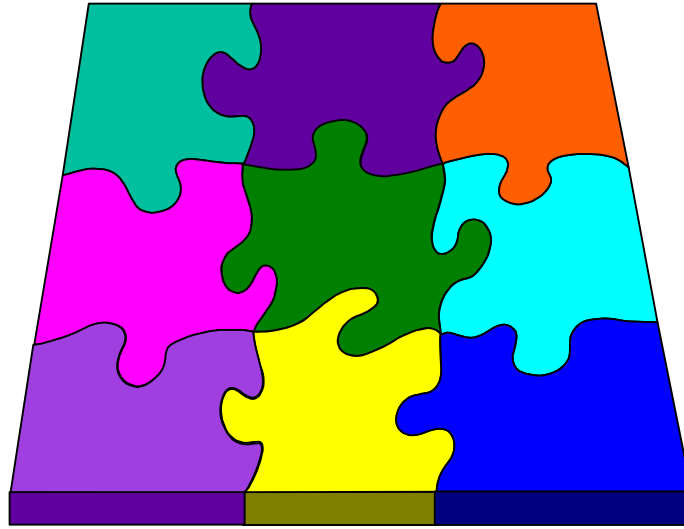


Diagrama de Sequência

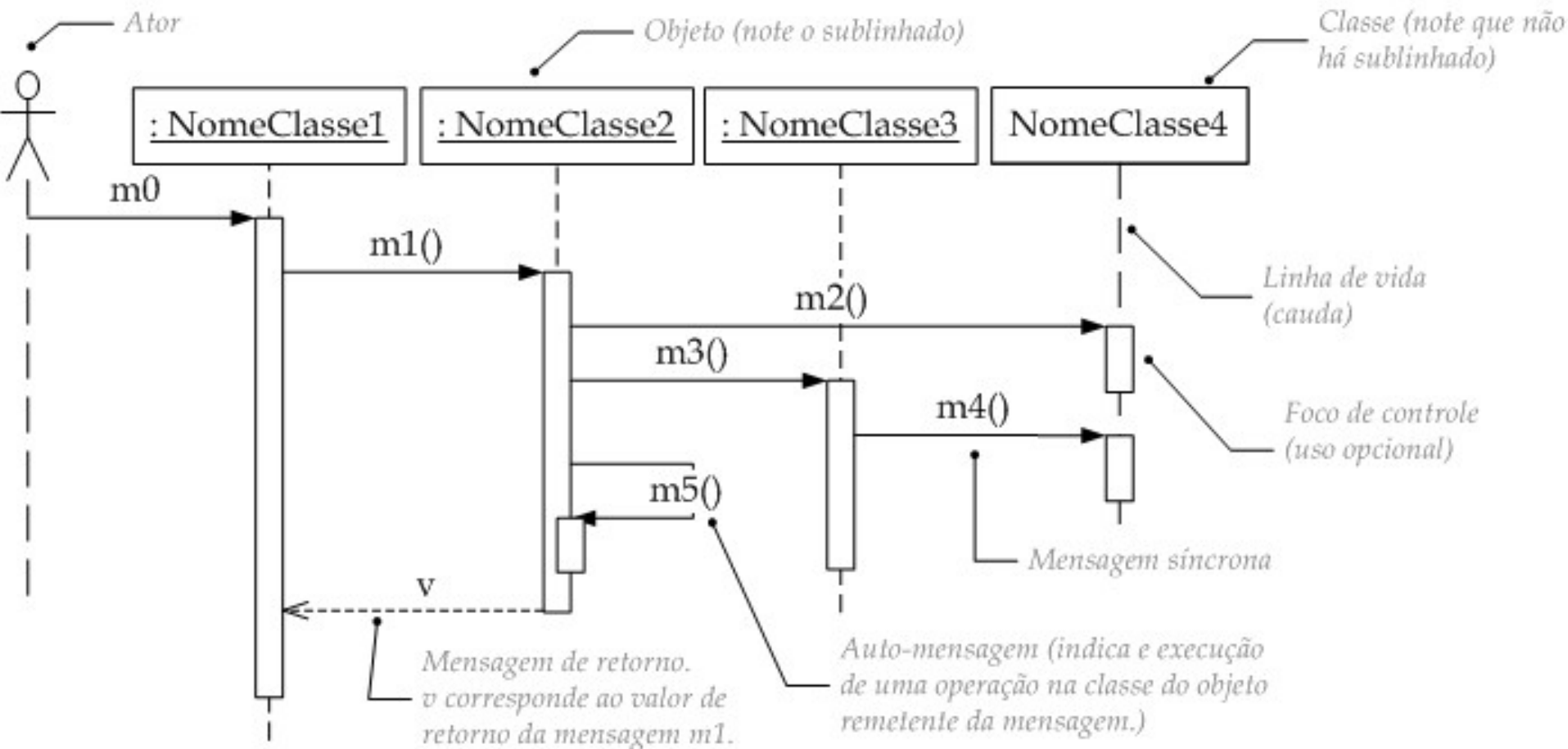
Diagrama de Sequência

- Os objetos participantes da interação são organizados na horizontal.
- Abaixo de cada objeto existe uma linha (linha de vida)
- Cada linha de vida possui o seu foco de controle.
 - Quando o objeto está fazendo algo.
- As mensagens entre objetos são representadas com linhas horizontais rotuladas partindo da linha de vida do objeto remetente e chegando a linha de vida do objeto receptor.
- A posição vertical das mensagens permite deduzir a ordem na qual elas são enviadas.
- A ordem de envio de mensagens em um diagrama de sequência pode ser deduzida a partir das expressões de sequência.
- A Criação e a destruição de objetos podem ser representadas.

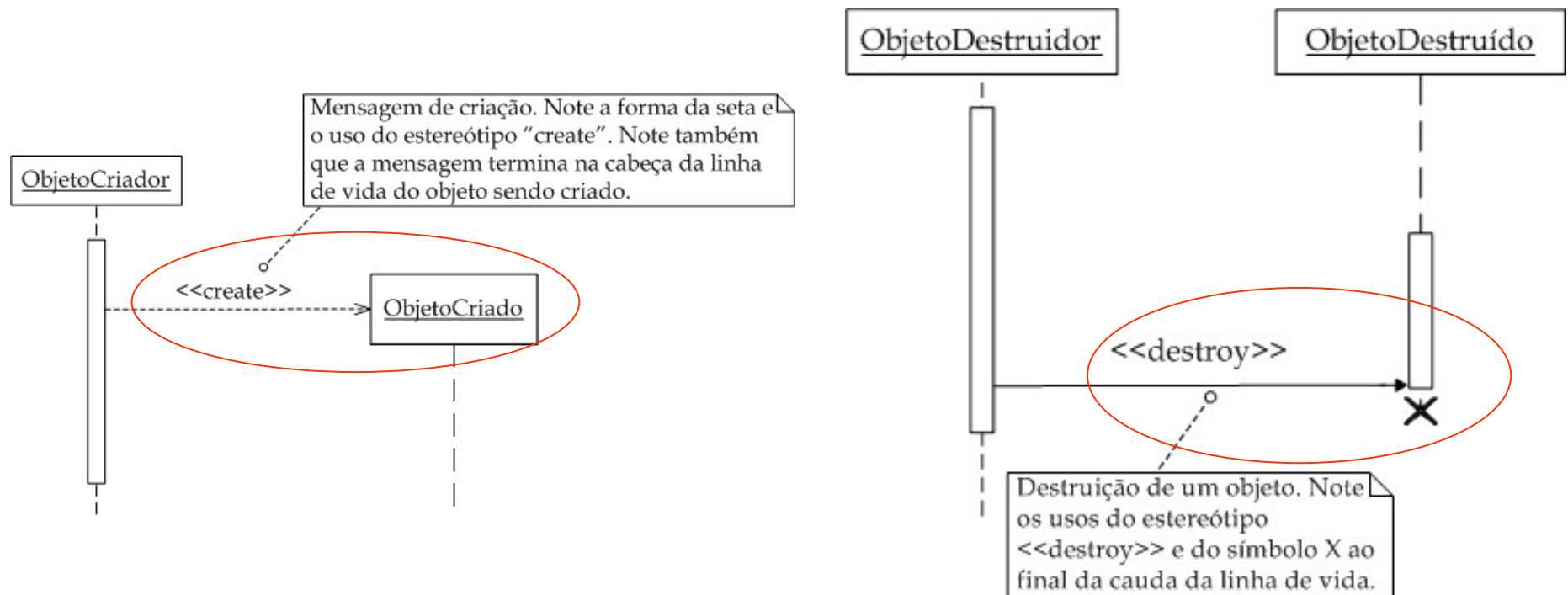
Elementos Gráficos de um DS

- Elementos básicos em um diagrama de sequência:
 - Atores
 - Objetos, multiobjetos e classes
 - Mensagens
 - Linhas de vida e focos de controle
 - Criação e destruição de objetos
 - Iterações

Elementos gráficos de um DS



Criação/Destruição de Objetos em um DS



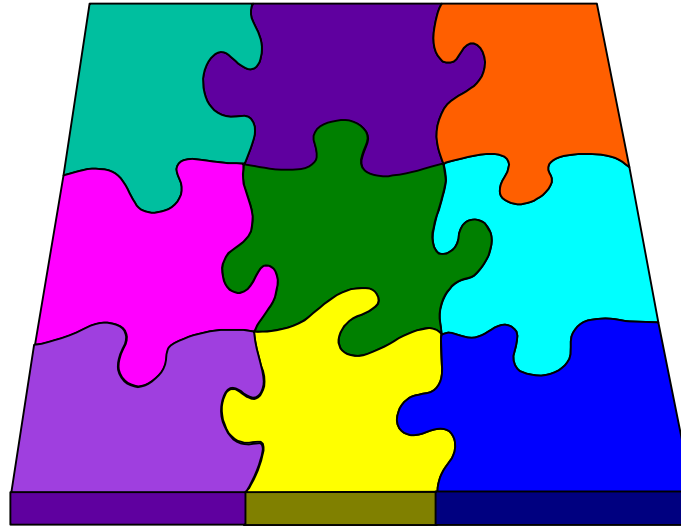


Diagrama de Comunicação

(chamado de Diagrama de Colaboração na UML 1.*)

Diagrama de Comunicação (DC)



- As ligações (linhas) entre objetos correspondem a relacionamentos existentes entre os objetos.
 - Deve haver consistência com o diagrama de classes...

Diagrama de Comunicação (DC)

- Os objetos estão distribuídos em duas dimensões
 - Vantagem: normalmente permite construir desenhos mais legíveis comparativamente aos diagramas de sequência.
 - Desvantagem: não há como saber a ordem de envio das mensagens a não ser pelas expressões de sequência.
- Direção de envio de mensagem é indicada por uma seta próxima ao rótulo da mensagem.

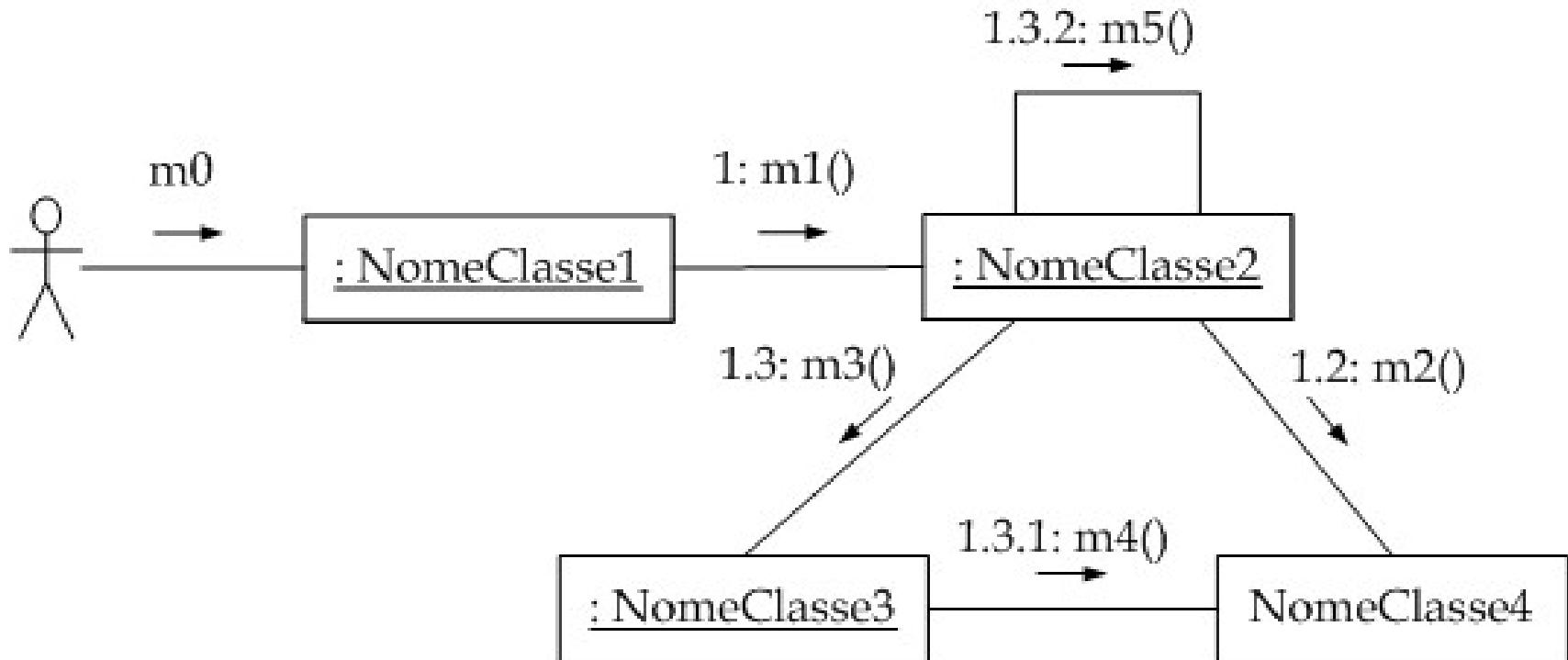
Elementos Gráficos de um DC

- Elementos básicos em um diagrama de comunicação:
 - Atores
 - Objetos, multiobjetos e classes
 - Mensagens
 - **Ligações entre objetos**
 - Criação e destruição de objetos
 - Iterações

Em um diagrama de sequência...

- Atores
- Objetos, multiobjetos e classes
- Mensagens
- **Linhas de vida e focos de controle**
- Criação e destruição de objetos
- Iterações

Elementos Gráficos de um DC



Criação de objetos em um DC

- Durante a execução de um cenário de caso de uso, objetos podem ser criados e outros objetos podem ser destruídos.
- Alguns objetos podem sobreviver à execução do caso de uso (se conectando a outros objetos); outros podem nascer e morrer durante essa execução.

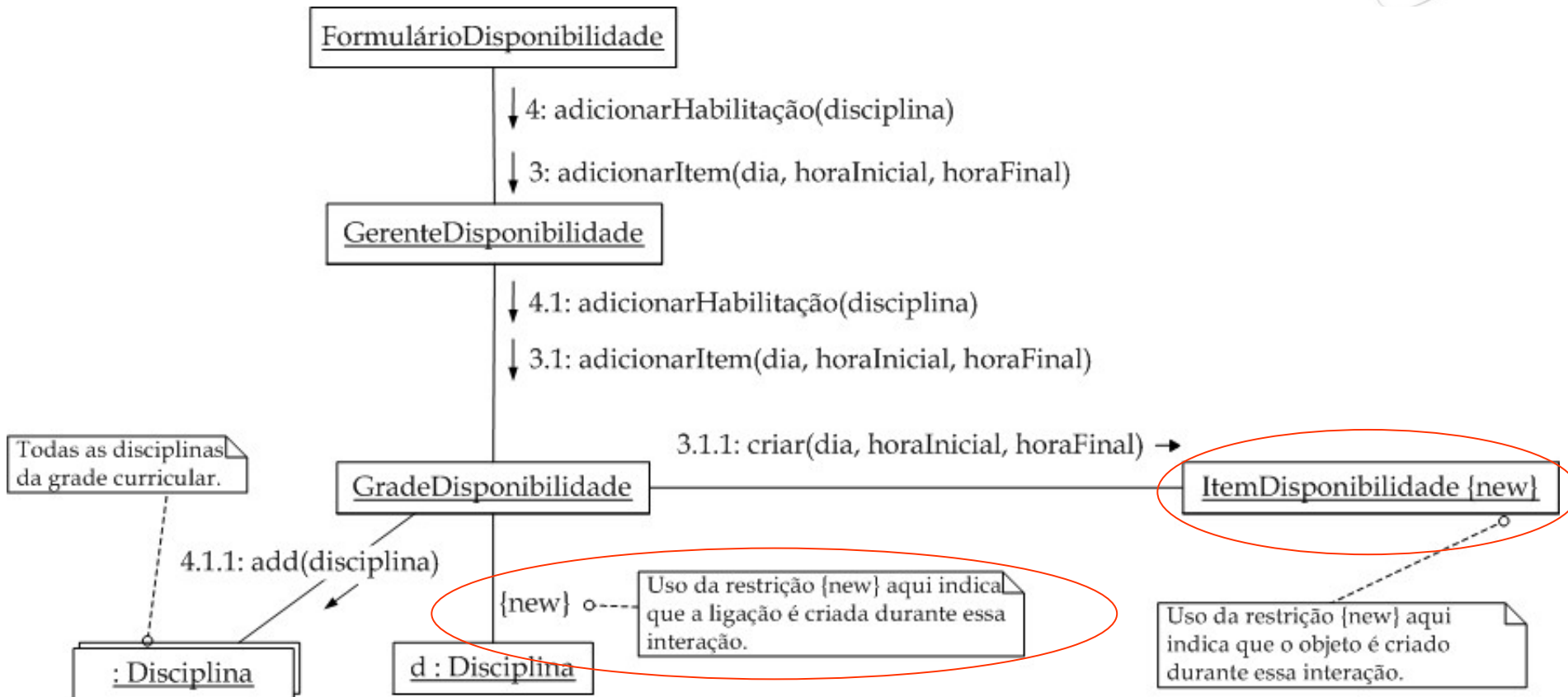


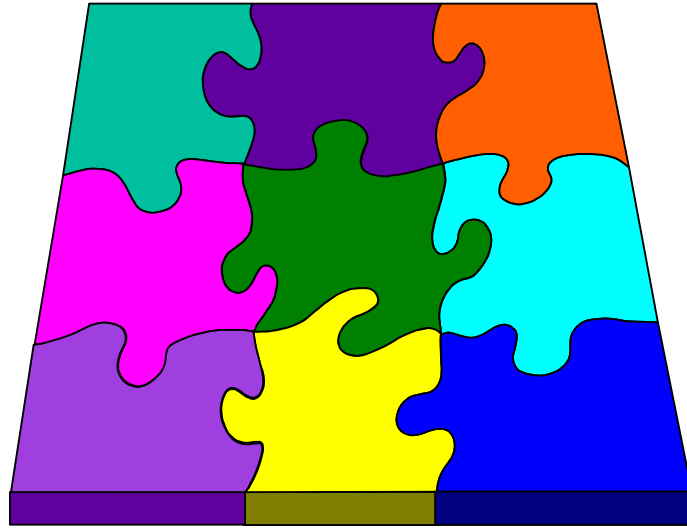
Criação de objetos em um DC

- A UML define etiquetas (tags) para criação e destruição de objetos (ou de ligações entre objetos) no diagrama de comunicação.
 - {new}: objetos ou ligações criados durante a interação.
 - {destroyed}: objetos ou ligações destruídos durante a interação.
 - {transient}: objetos ou ligações destruídos e criados durante a interação.



Criação de objetos em um DC

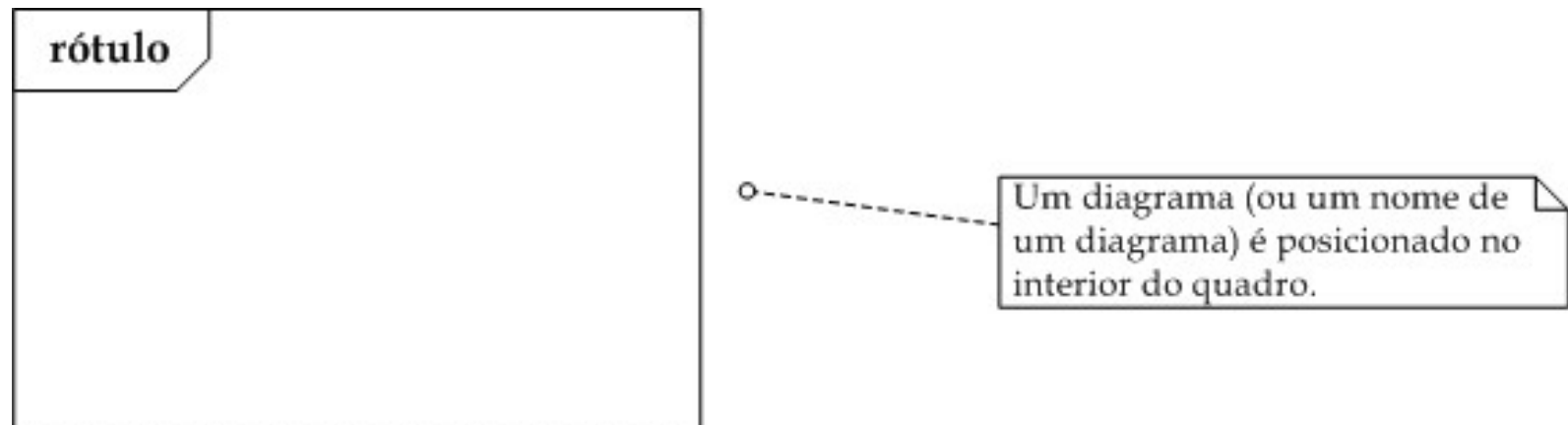




Modularização de Interações

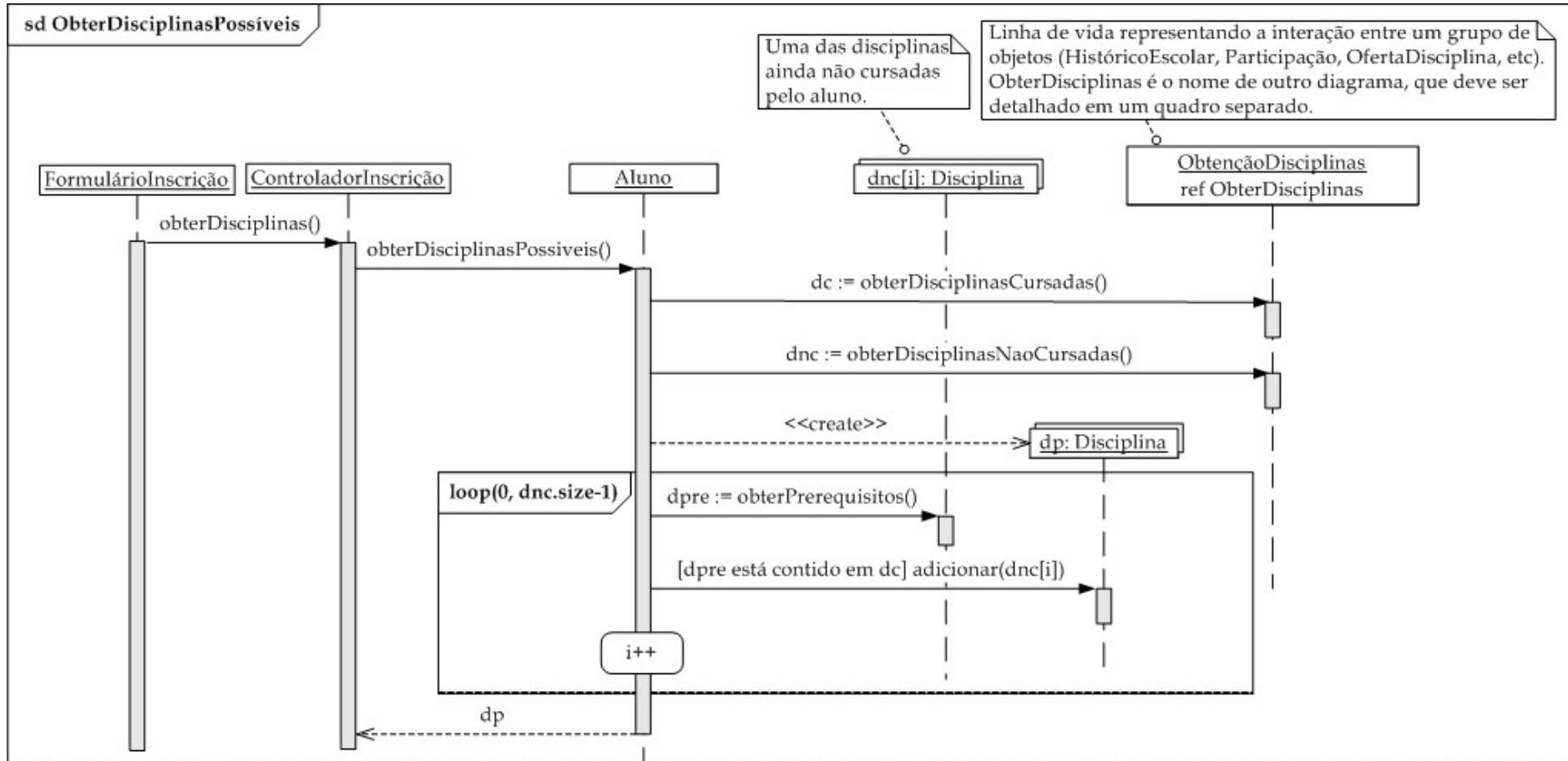
Quadros de Interação

- Elemento gráfico, que serve para modularizar a construção de diagramas de sequência (ou de comunicação).
- Objetivos específicos:
 - Dar um nome ao diagrama que aparece dentro do quadro;
 - Fazer referência a um diagrama definido separadamente; e
 - Definir o fluxo de controle da interação.
- Notação:



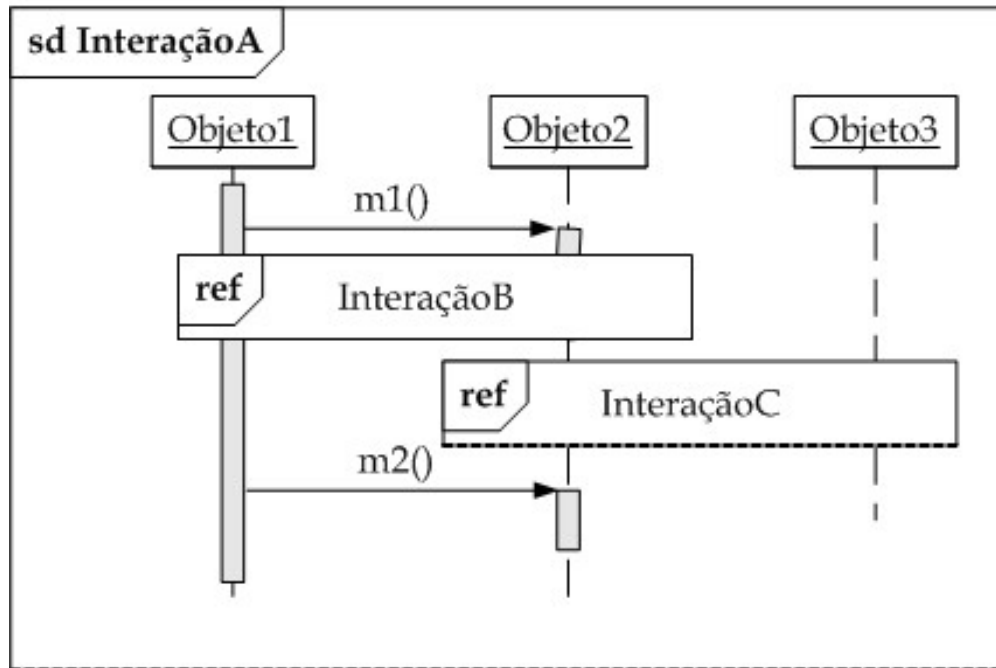
Diagramas Nomeados

Dar um nome ao diagrama que aparece dentro do quadro

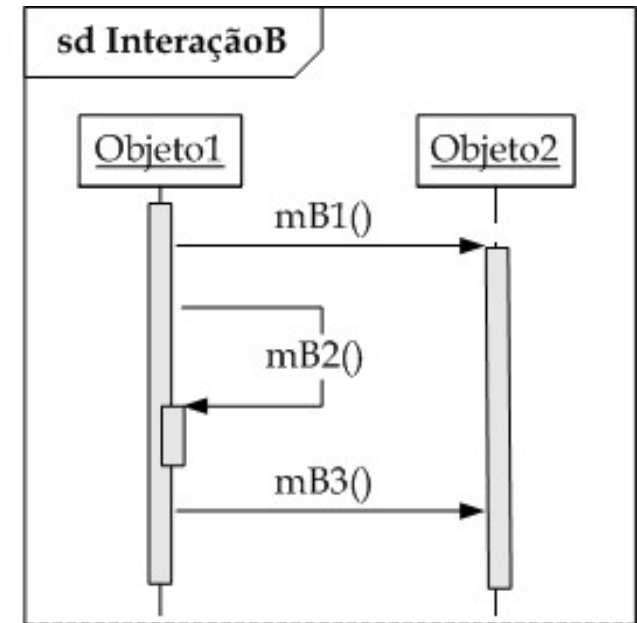


Diagramas Referenciados

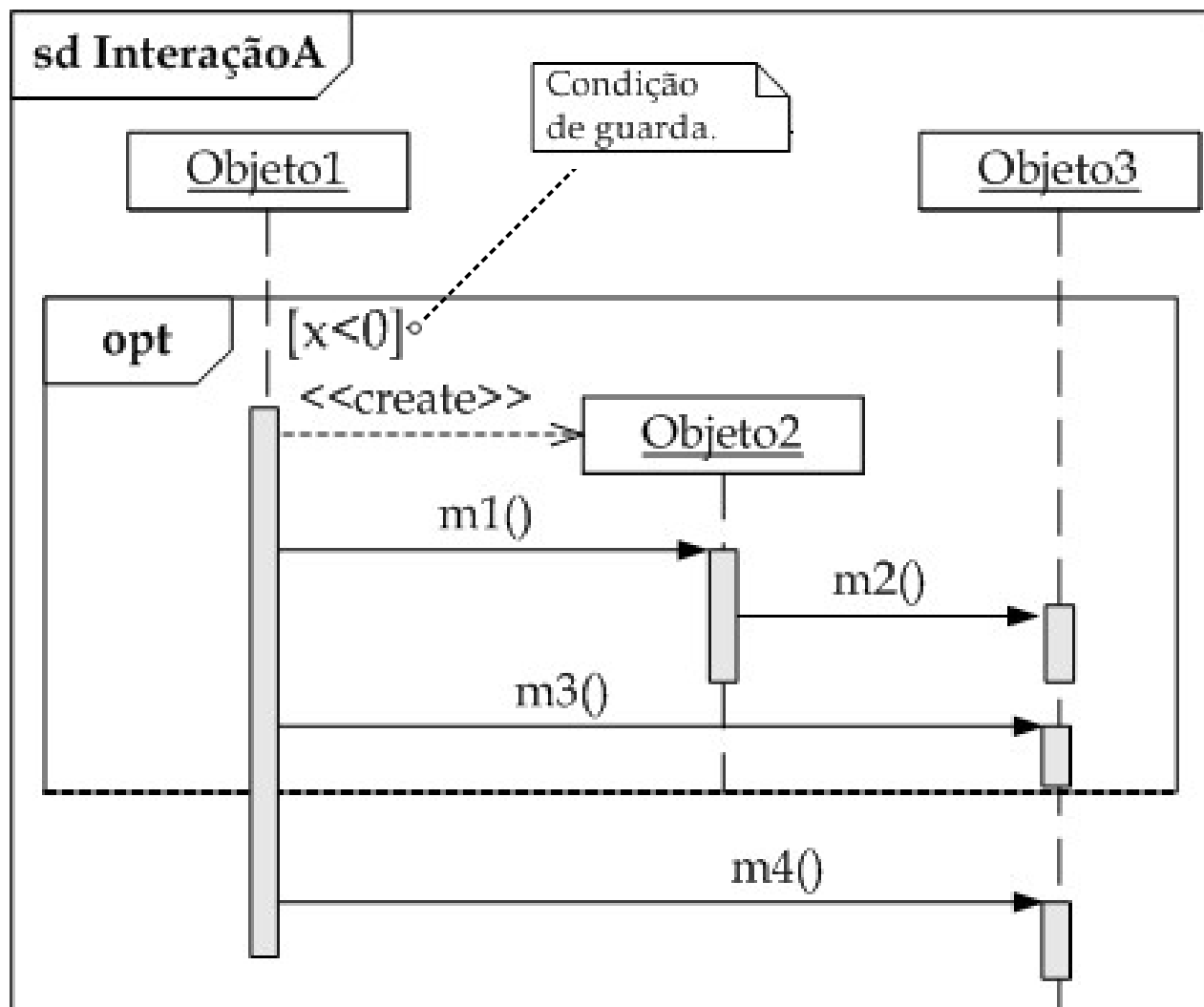
Fazer referência a um diagrama definido separadamente.



InteraçãoB e InteraçãoC são nomes de diagramas que apresentam mensagens trocadas entre os objetos Objeto1 e Objeto2. Note que os quadros correspondentes são rotulados com "ref" e posicionados sobre as linhas de vida dos objetos.

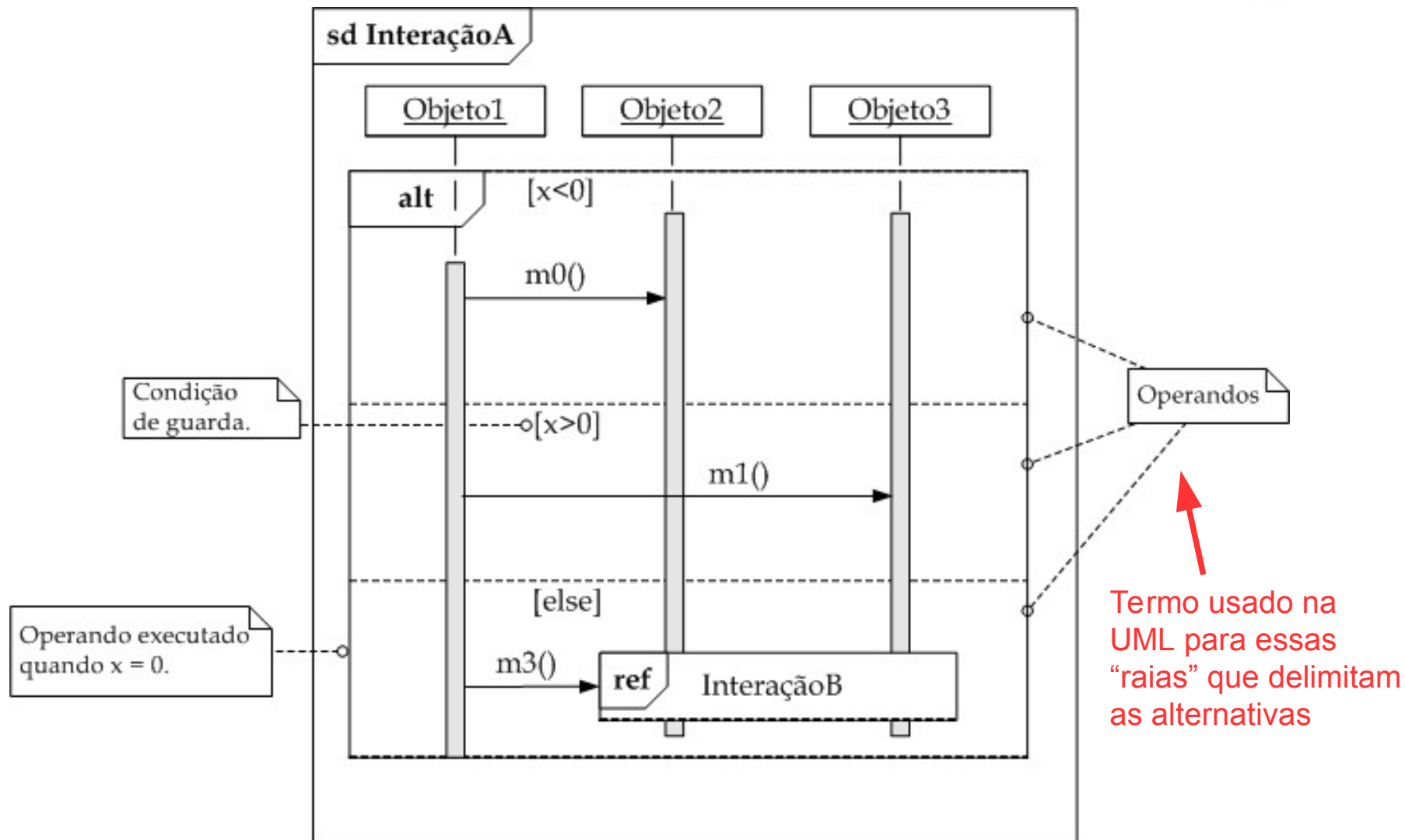


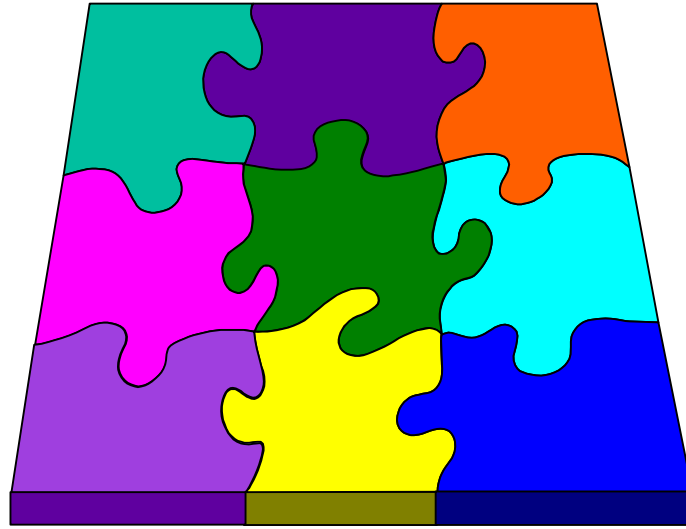
Fluxo de Controle: opções (Fluxo opcional if-then)



Fluxo de Controle: Alternativas

(Escolhas mutuamente exclusivas if-then-else)

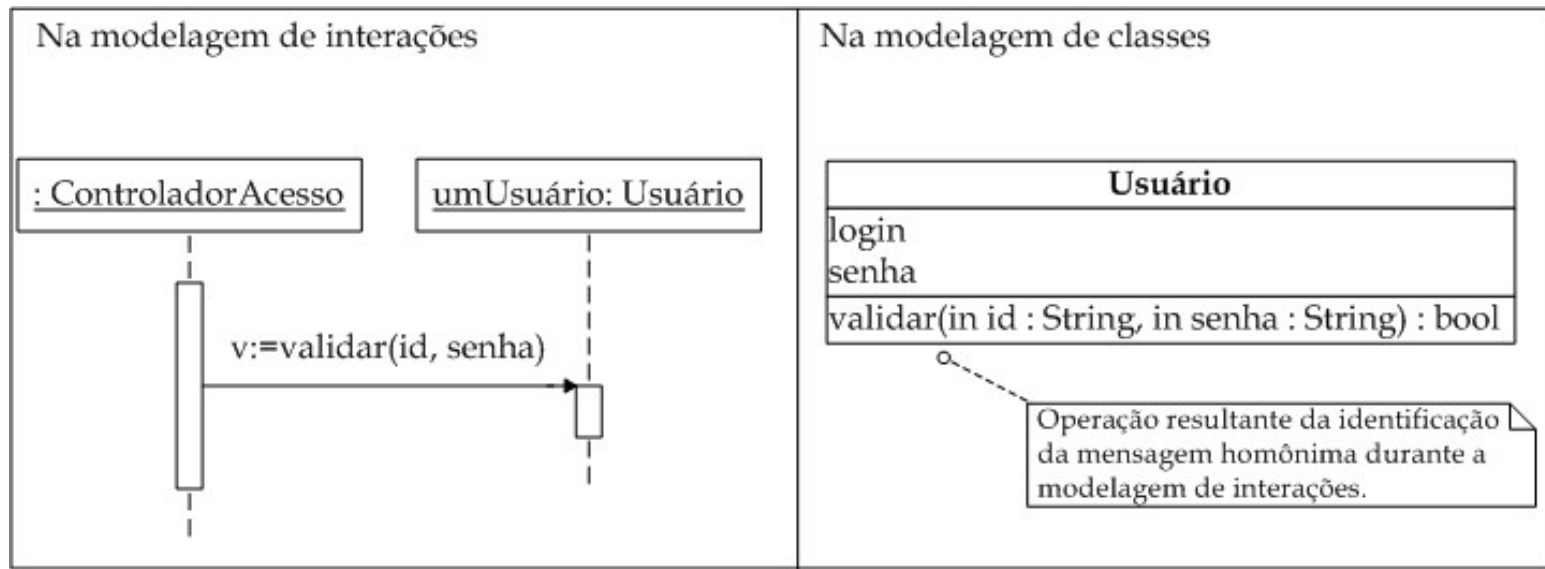




Construção do Modelo de Interações

Mensagens *versus* responsabilidades

- O objetivo da modelagem de interações é identificar **mensagens** e, em última análise, **responsabilidades**.



Uma mensagem implica a existência de uma operação no objeto receptor. A resposta do objeto receptor ao recebimento de uma mensagem é a execução da operação correspondente.

Alocação de responsabilidades

- Podemos então entender a modelagem de interações como um processo cujo objetivo final é **decompor as responsabilidades do sistema e alocá-las a classes**.
- Dado um conjunto de N responsabilidades:
 - (a) Uma possibilidade é criar uma única classe no sistema para assumir com todas as N responsabilidades.
 - (b) Outra possibilidade é criar N classes no sistema, a cada um delas sendo atribuída uma das N responsabilidades.
- Certamente, (a) e (b) são absurdas do ponto de vista prático.
 - Mas... entre as muitas maneiras possíveis de alocar responsabilidades, **como podemos saber quais delas são melhores que outras?**

Acoplamento e Coesão

- A resposta à pergunta anterior não é nenhuma receita de bolo.
 - De fato, para construirmos um bom modelo de interações, devemos lançar mão de diversos princípios de projeto.
- Dois dos principais princípios são o **acoplamento** e a **coesão**.

Coesão

- A *coesão* é uma medida do quão fortemente relacionadas e focalizadas são as responsabilidades de uma classe.
- É extremamente importante assegurar que as responsabilidades atribuídas a cada classe sejam altamente relacionadas.
 - Em outras palavras, o projetista deve definir classes de tal forma que cada uma delas tenha **alta coesão**.

Acoplamento

- O *acoplamento* é uma medida de quão fortemente uma classe está conectada a outras classes, tem conhecimento ou depende das mesmas.
- Uma classe com acoplamento fraco (baixo) não depende de muitas outras.
- Por outro lado, uma classe com acoplamento forte é menos inteligível isoladamente e menos reutilizável.
- Além disso, uma classe com alto acoplamento é mais propensa a mudanças, quando é necessário modificar as classes da qual ela depende.

Acoplamento e Coesão

Conclusão:

Criar modelos com alta coesão e baixo acoplamento deve ser um objetivo de qualquer projetista.

Referências e Outros Exemplos

- BEZERRA, E.: Princípios de Análise e Projeto de Sistemas com UML, 3ª edição, Campus - Elsevier (2015).
- Links com outros exemplos:
 - <http://www.agilemodeling.com/artifacts/communicationDiagram.htm>
 - <https://creately.com/blog/diagrams/sequence-diagram-tutorial/>