De Requisitos para Design/Projeto

Notações para design da interação entre objetos

A transição para atividades de design

- Aprendendo a fazer a coisa certa
  - Requisitos e Análise OO
- Aprendendo a fazer certo a coisa
  - Trabalho de design
  - Projetando habilidosamente uma solução para satisfazer os requisitos para esta interação

Design Orientado a Objeto

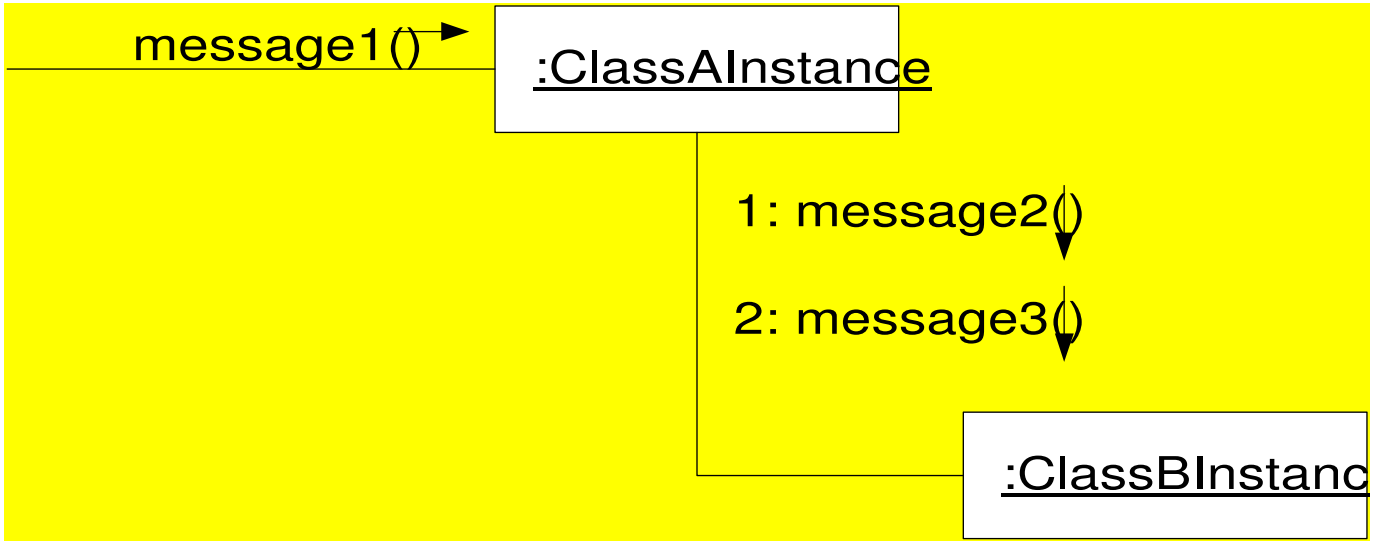
- Uma solução lógica baseada no paradigma OO é desenvolvida
- No núcleo dessa solução estão os **Diagramas de Interação**
  - Como objetos colaboram para satisfazer requisitos
- Em paralelo **Diagramas de Classe** são desenhados
- Estes artefatos são parte do **Modelo de Design**

Diagrama de Interação

- Linguagem usada para apoio ao trabalho de design/projeto
- **Diagramas de Interação**
  - Ilustram como objetos interagem via mensagens
  - Generalização de 2 tipos de diagramas UML mais especializados :
    - Diagramas de Colaboração
    - Diagramas de Seqüência

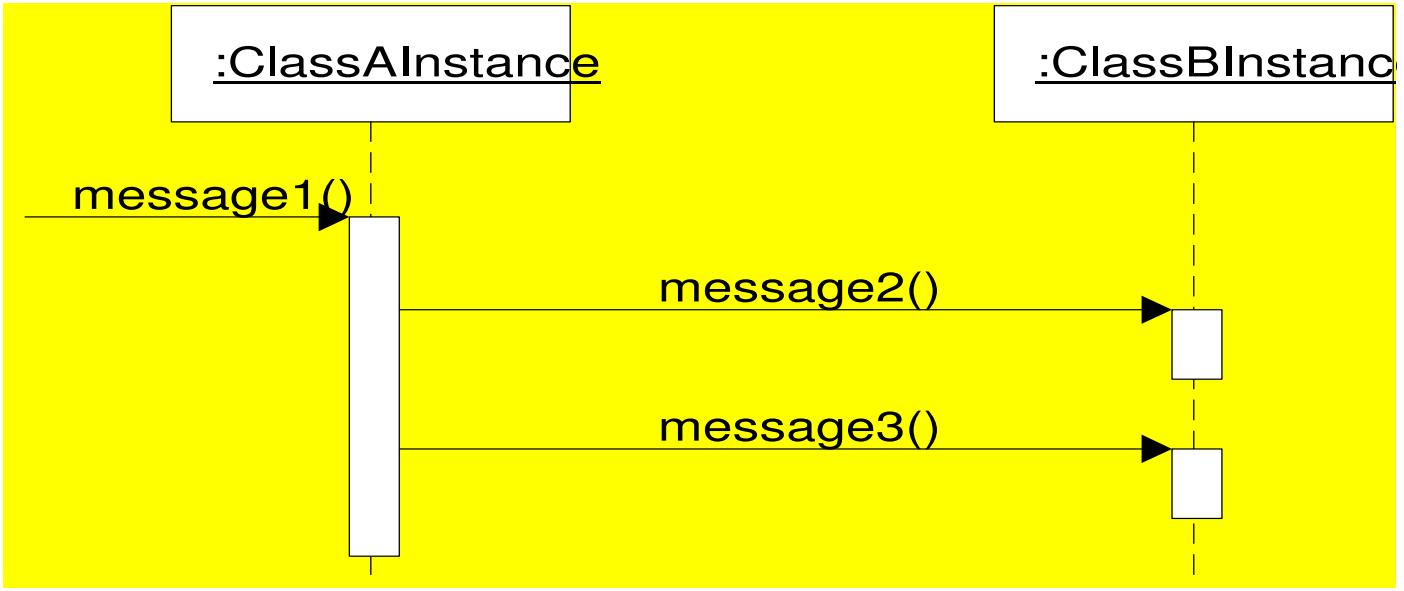
Diagramas de Colaboração

Ilustram interações entre objetos em formato de grafo ou rede [objetos podem ser colocados em qqr lugar no diagrama]



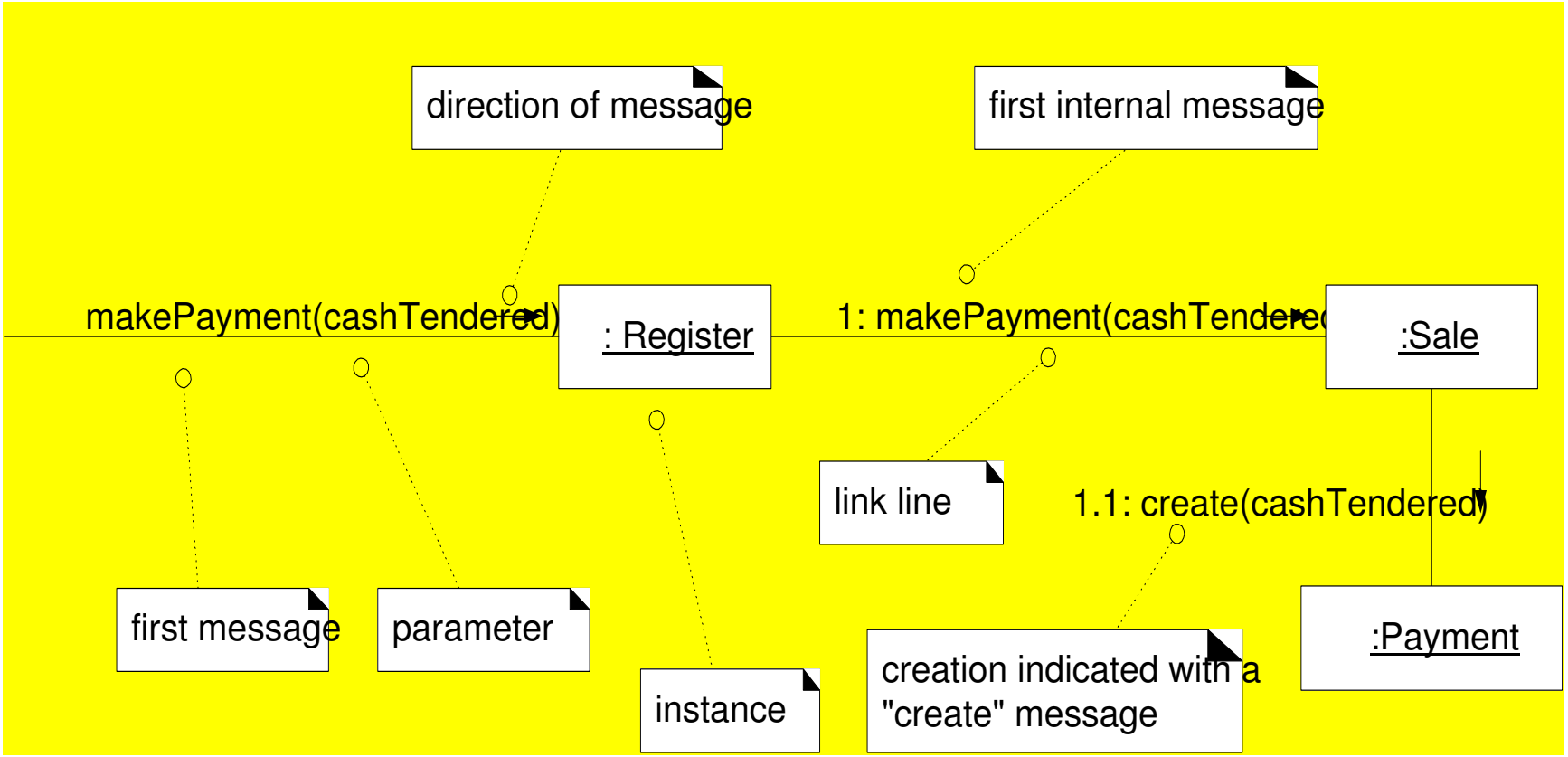
Diagramas de Seqüência

Ilustram interações nas quais cada objeto é adicionado `a direita [formato de cerca]



Based on C. Larman, 2002

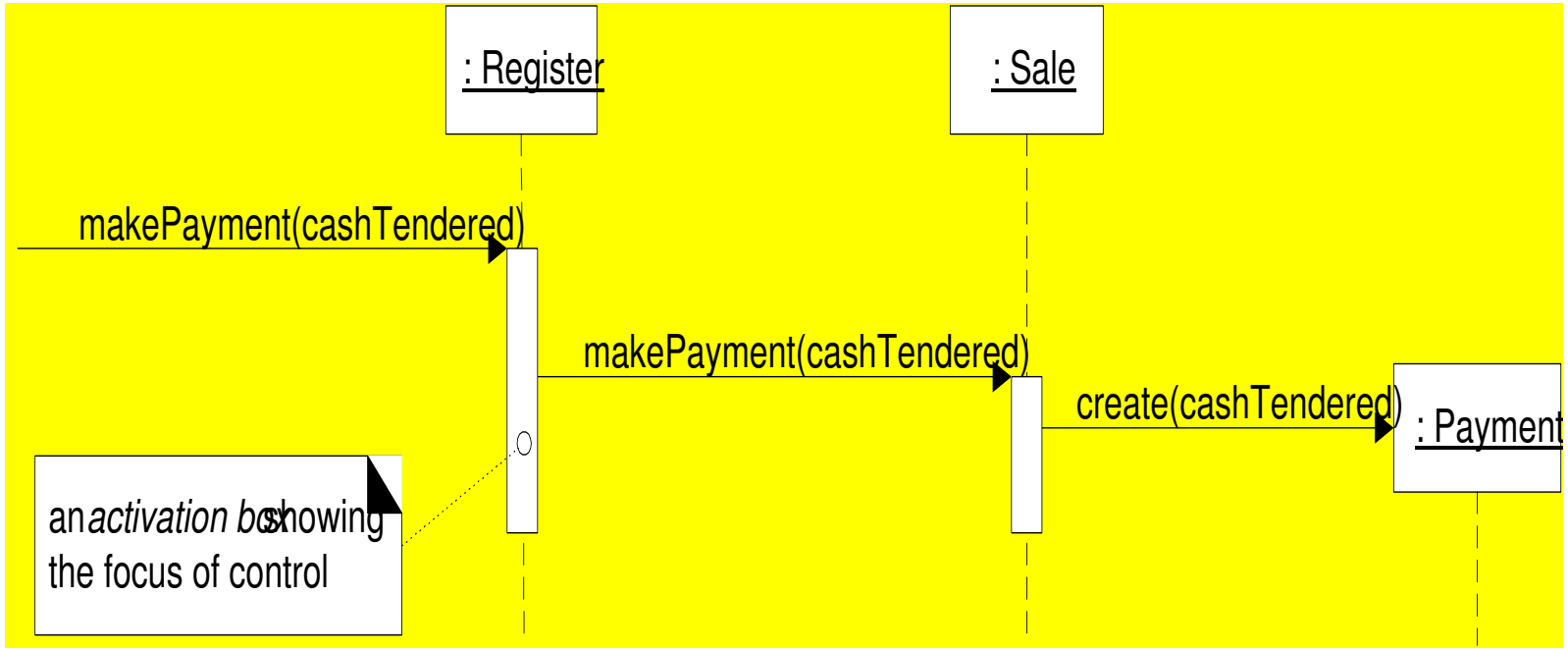
Ex. de Diagrama de Colaboração: makePayment



Lendo o Diagrama de Colaboração: makePayment

- 1. A mensagem *makePayment* é enviada para uma instância de *Register*. O sender não é identificado
- 2. A instância *Register* envia a mensagem *makePayment* para uma instancia de *Sale*
- 3. A instância *Sale* cria uma instância de um *Payment*

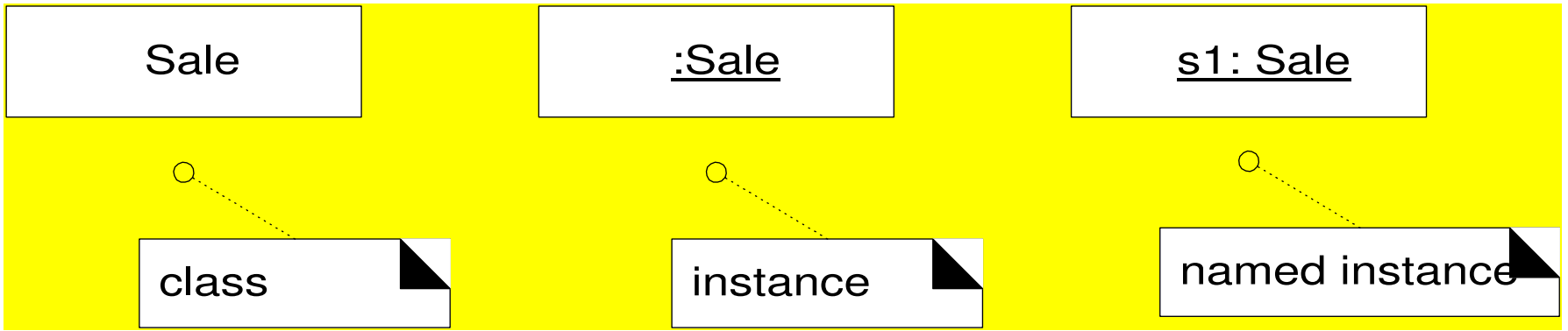
Ex. de Diagrama de Seqüência: makePayment



Pontos Fortes e Fracos

- **Diagrama de Seqüência:**
  - Mostra claramente a seqüência e ordem das mensagens no tempo. Notação simples
  - Consome espaço horizontal
- **Diagrama de Colaboração:**
  - Flexível para adicionar novos objetos. Econômico no espaço. Melhor para ilustrar ramificações complexas, iteração e comportamento concorrente
  - Difícil ver a seqüência de mensagens. Notação mais complexa

I. Diagrama de Interação: Notação



Sintaxe para a expressão de Mensagens

return := message(parameter : parameterType) : returnType

spec := getProductSpec(id)

spec := getProductSpec(id : ItemID)

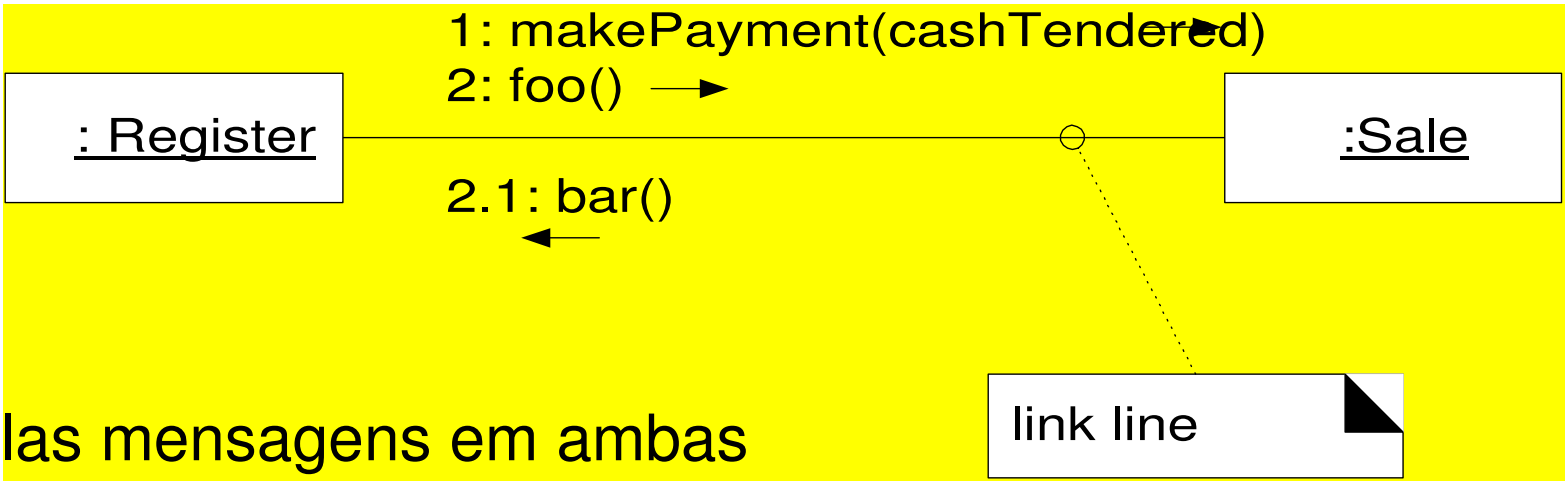
spec := getProductSpec(id : ItemID) : ProductSpecifi

Um Link

É um caminho de conexão entre dois objetos: indica alguma forma de navegação e visibilidade entre os objetos

Um link é uma instância de uma associação

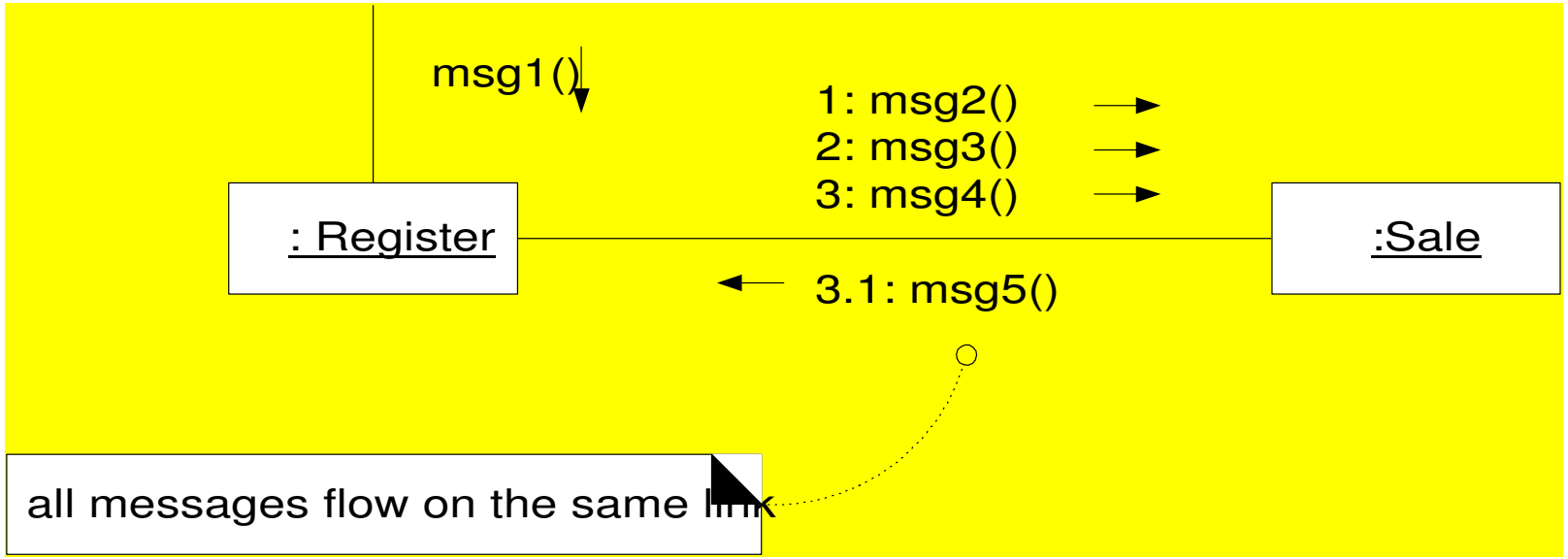
Based on C. Larman, 2002



Múltiplas mensagens em ambas as direções podem fluir no mesmo link

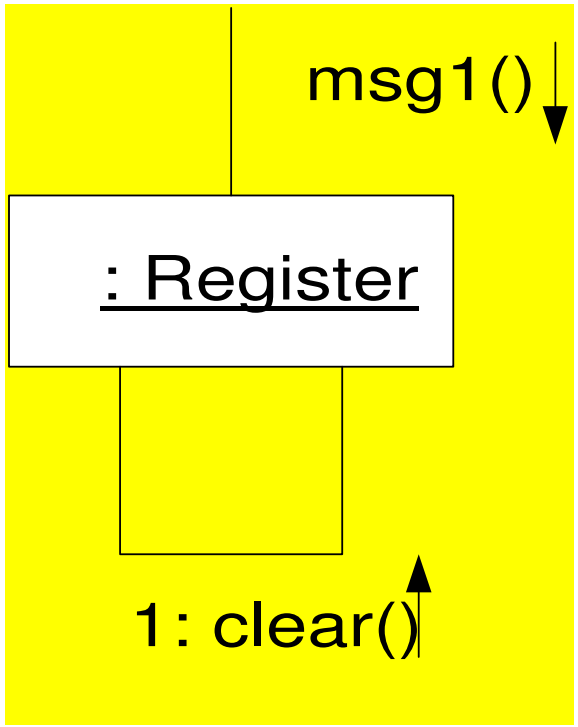
Mensagens

São representadas por uma expressão e uma pequena seta indicando a direção. Um número de seqüência é adicionado

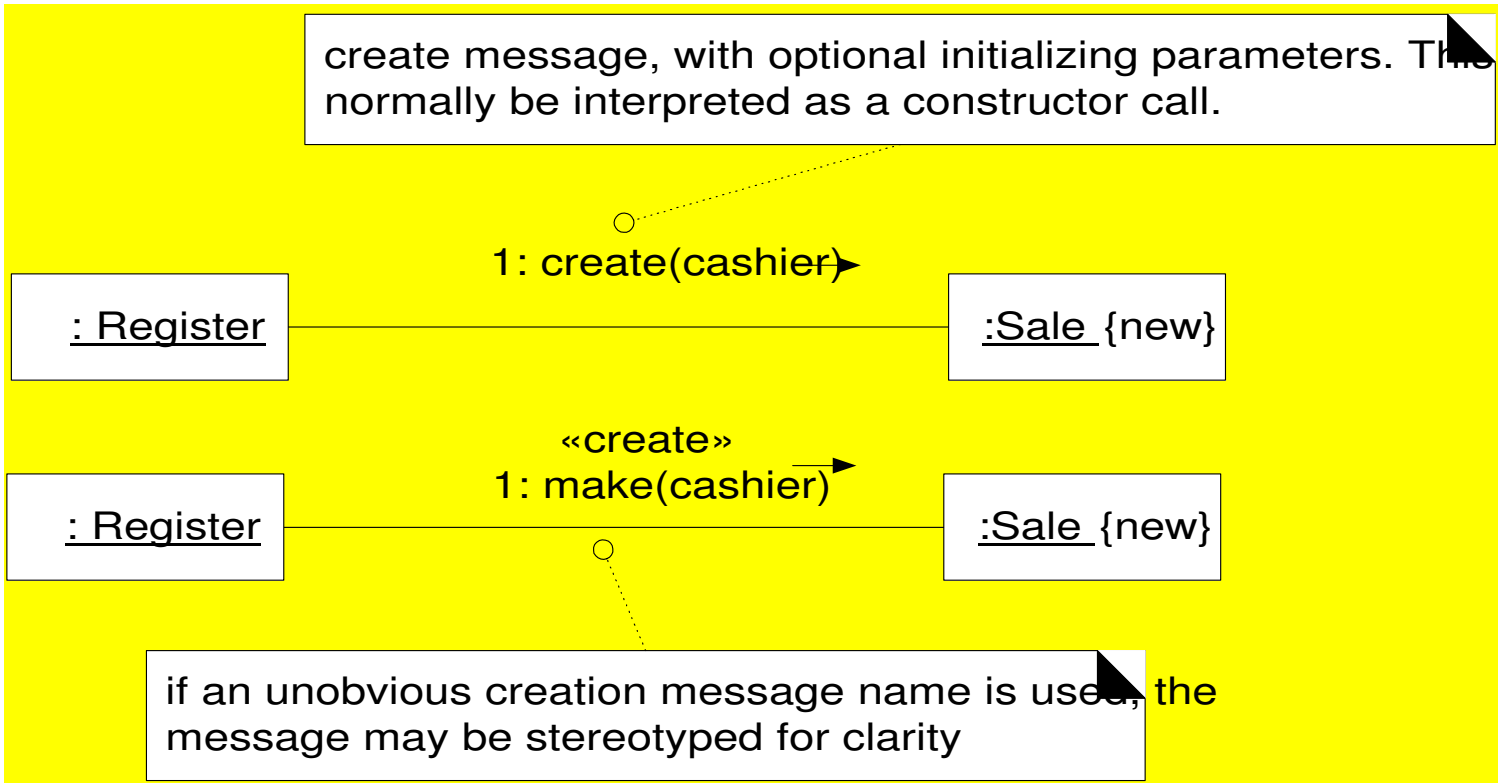


Based on C. Larman, 2002

Mensagens para “self” or “this”

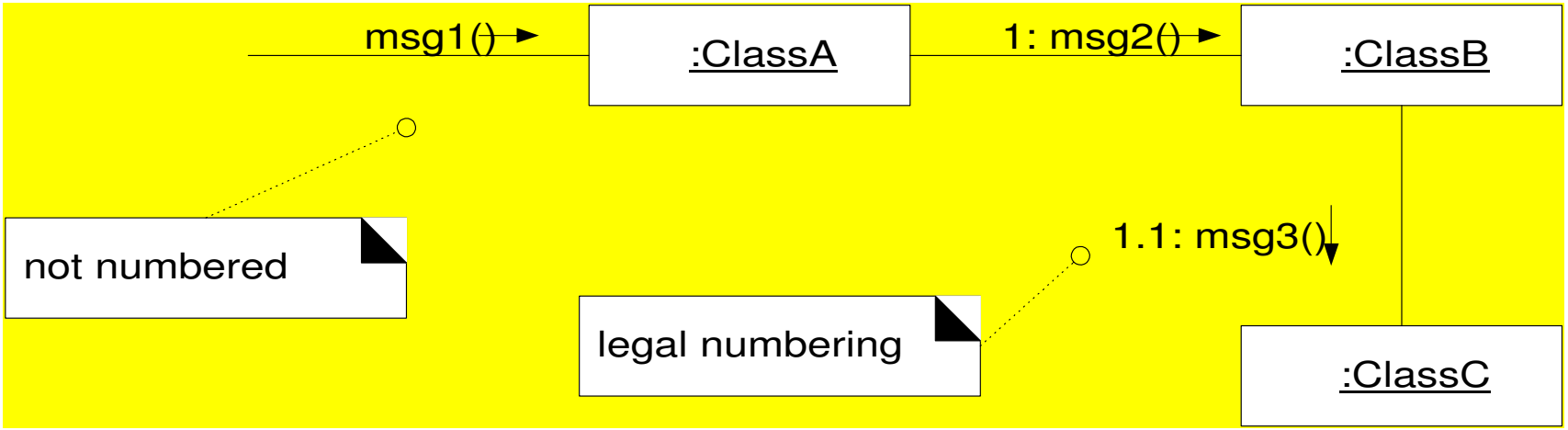


Criação de Instâncias

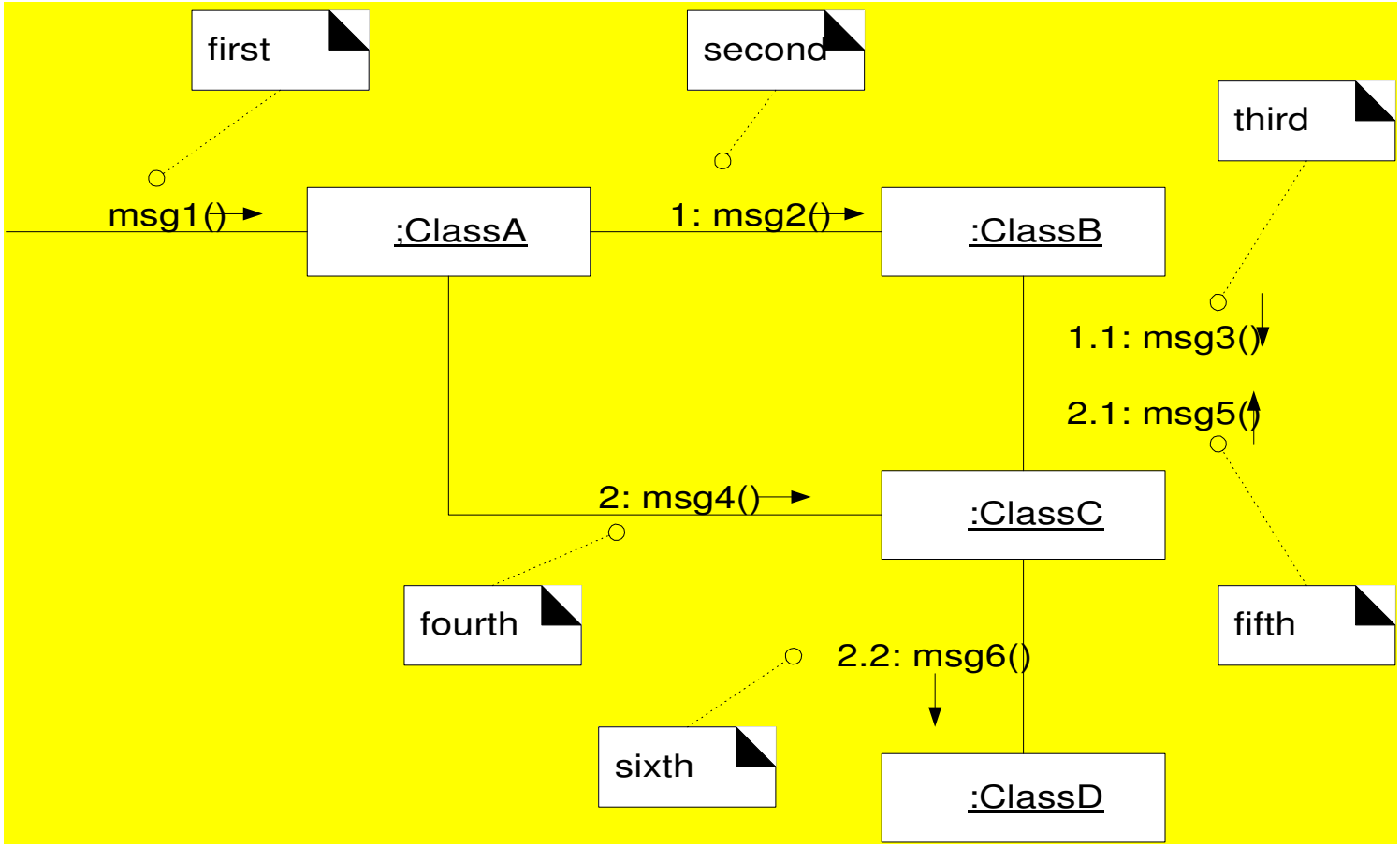


Sequenciamento de Mensagens

A primeira mensagem não é numerada. Aninhamento é permitido

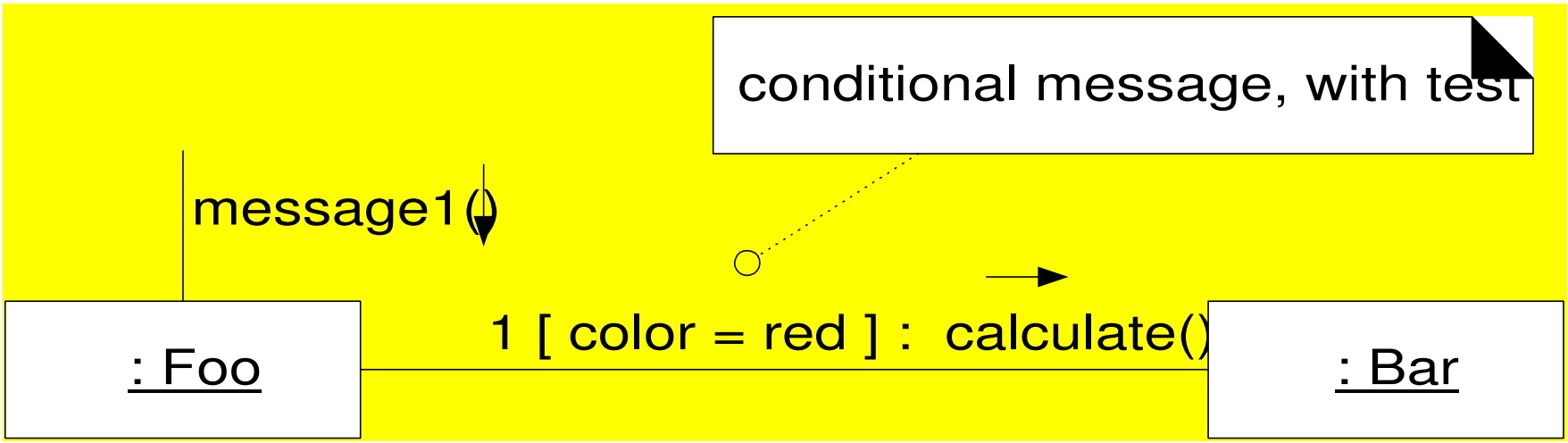


Numeração de seqüência complexa



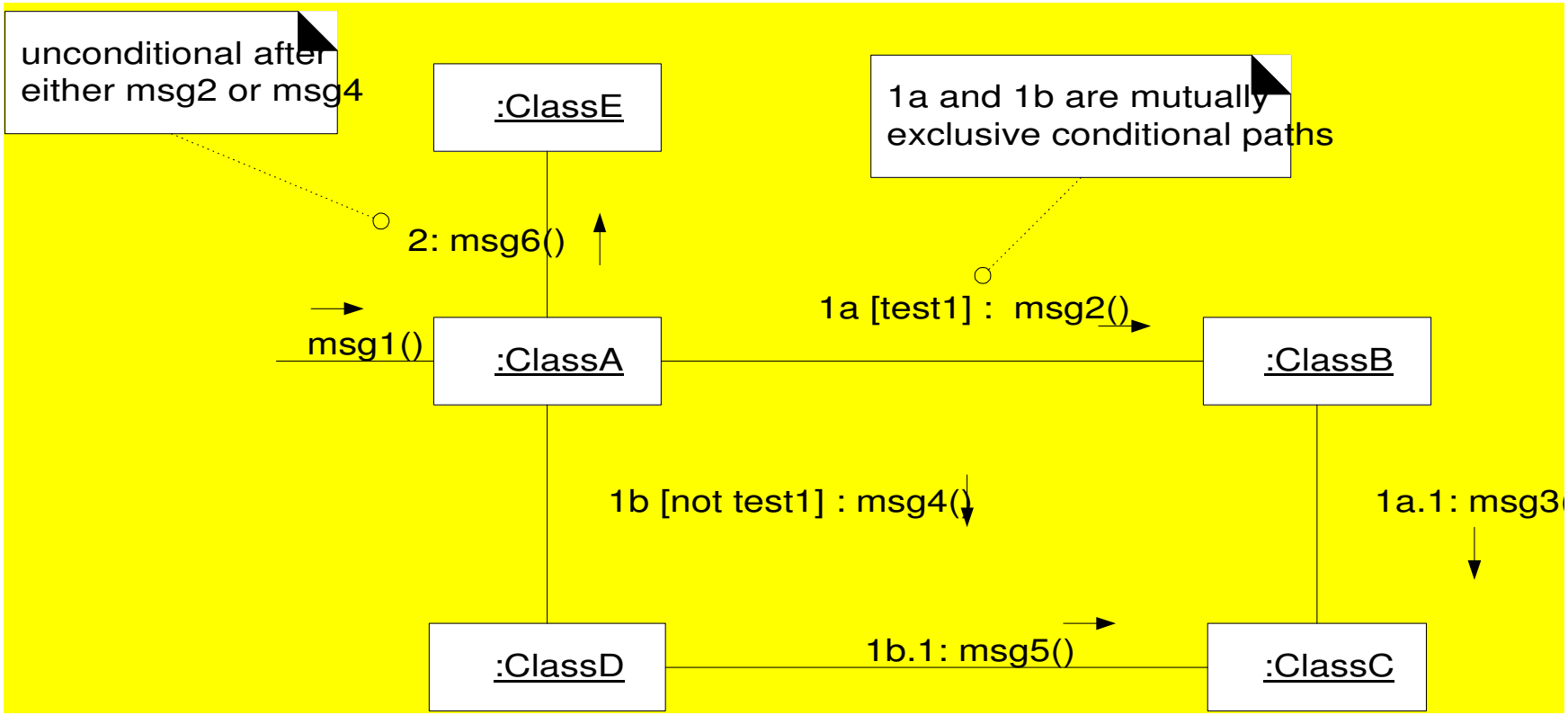
Based on C. Larman, 2002

Mensagens Condicionais



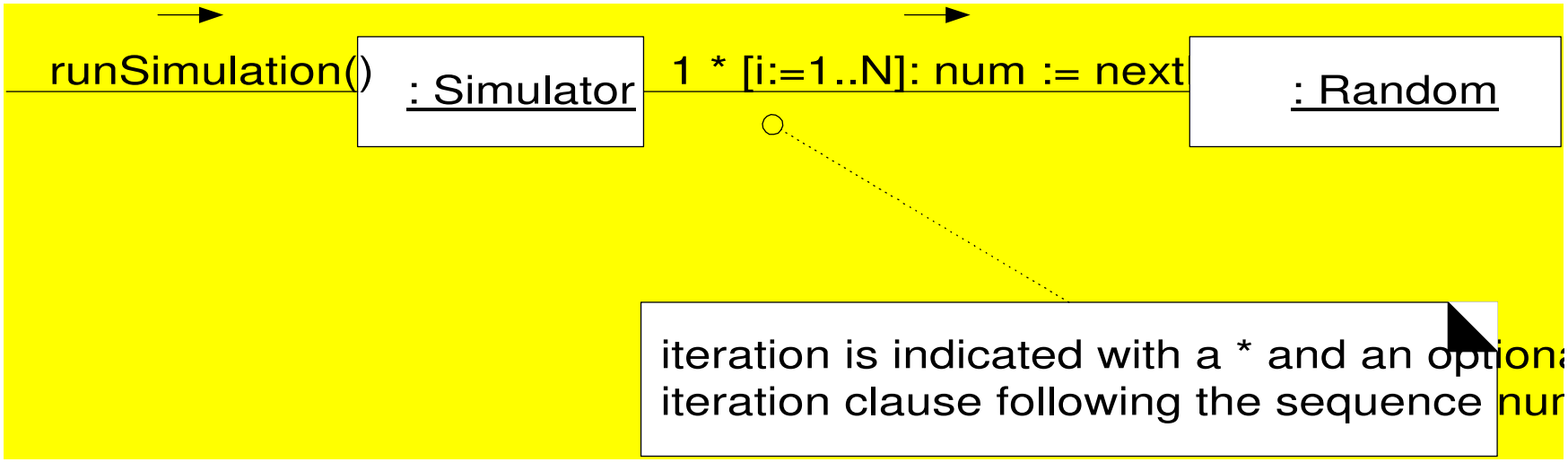
A mensagem só é enviada se a cláusula resultar true

Caminhos condicionais mutuamente exclusivos



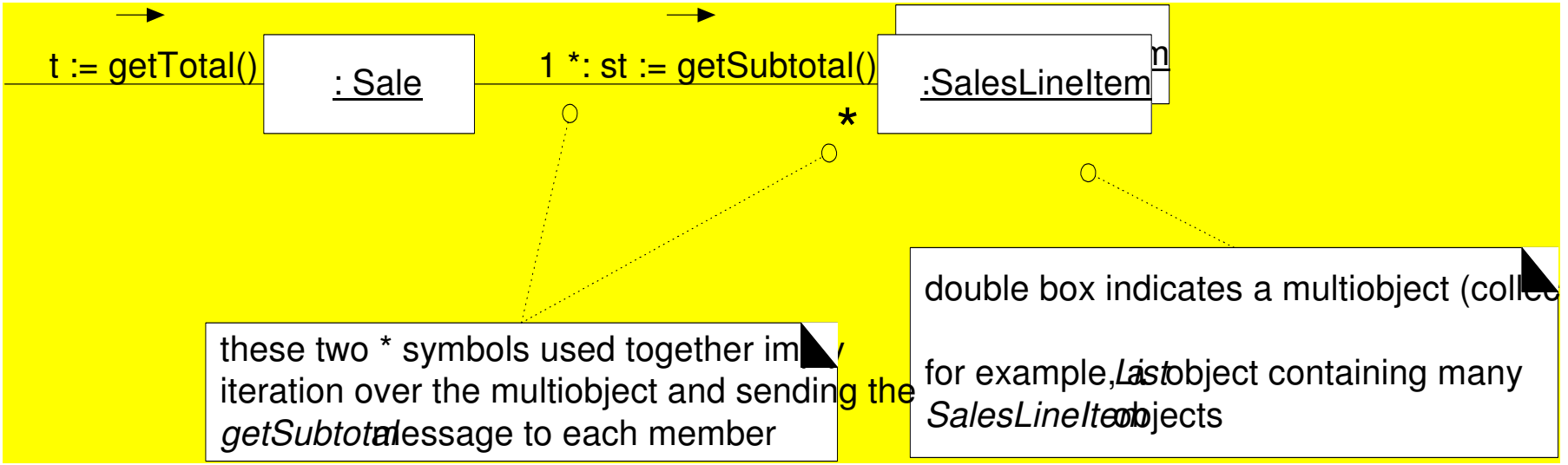
Note o caminho condicional letras a e b

Iteração ou Looping



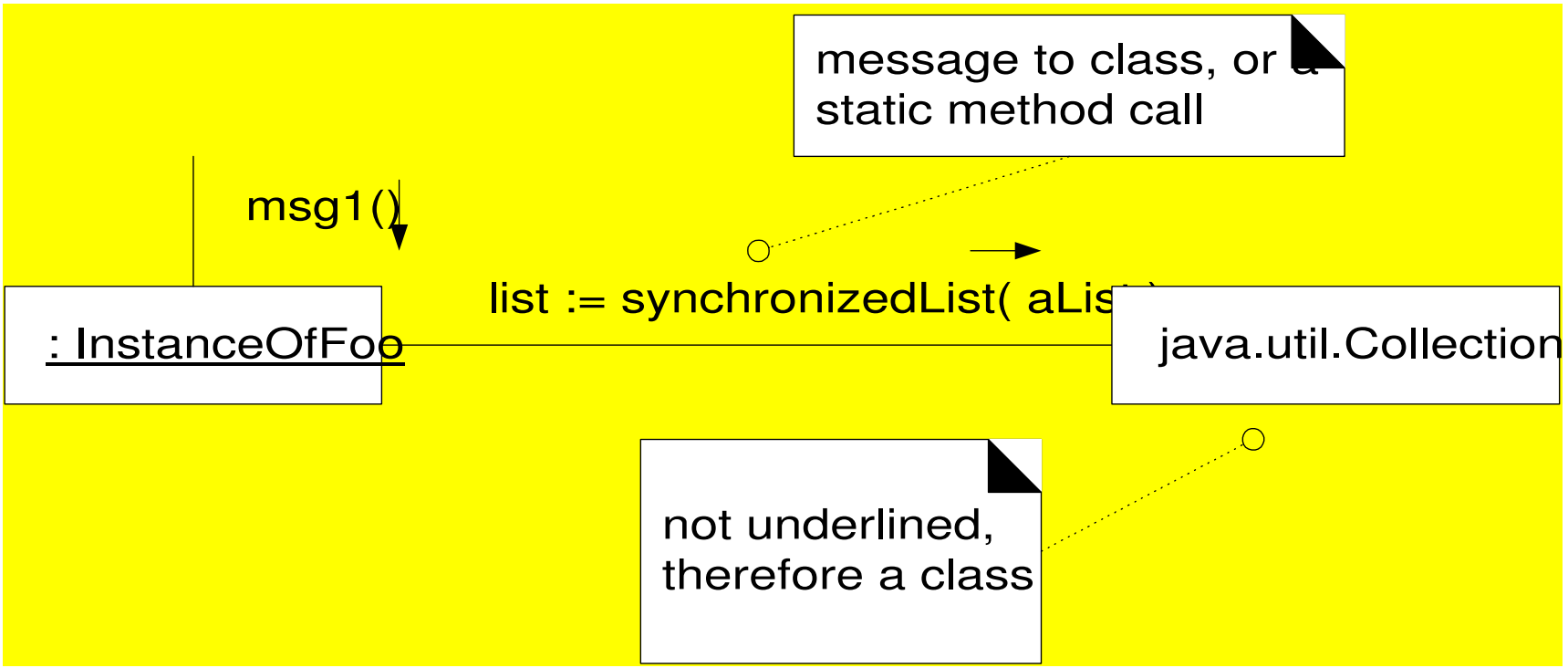
Iteração sobre uma coleção

Ex. Enviando mensagem a cada elemento de uma lista



Mensagens para uma classe de objetos

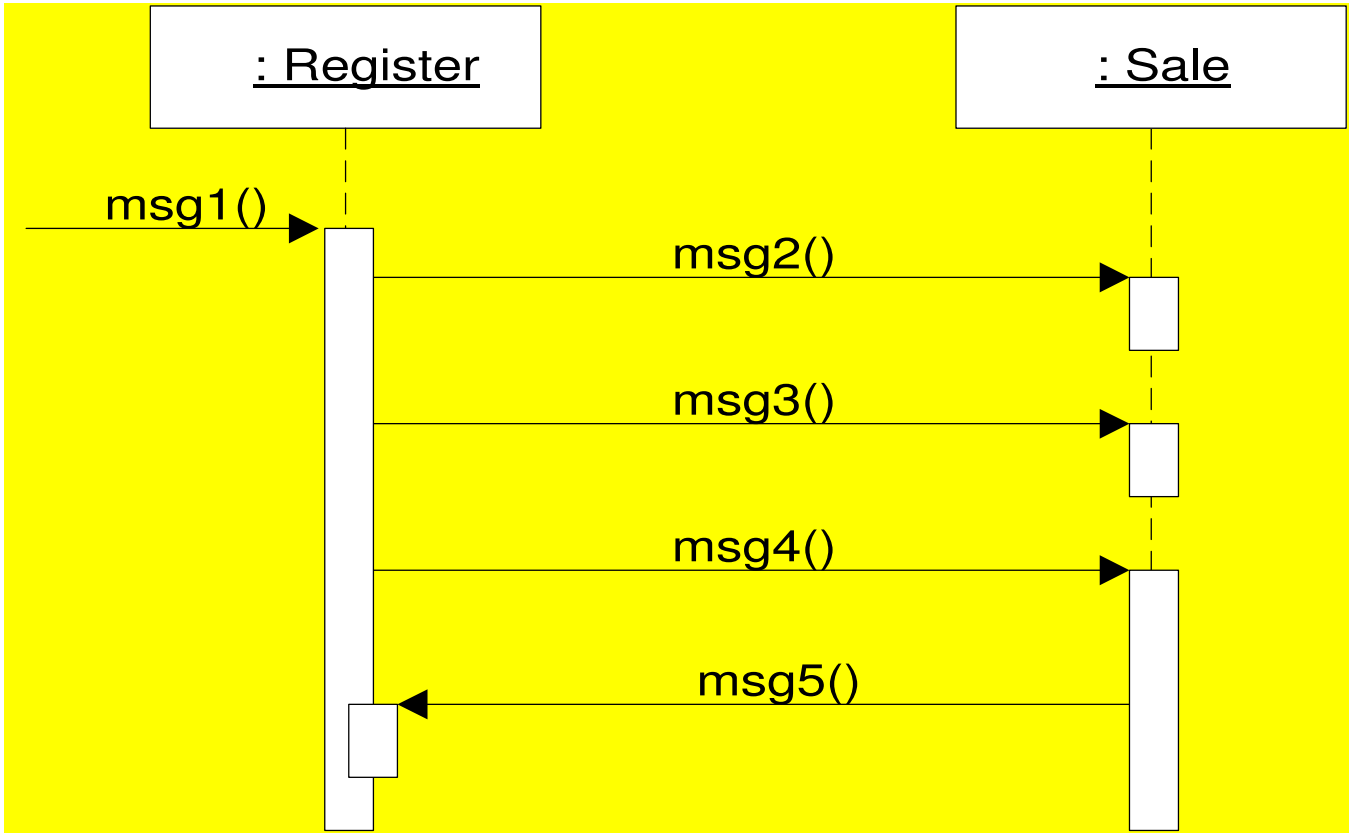
Em vez de para instância de objeto



Based on C. Larman, 2002

II. Diagramas de Seqüência

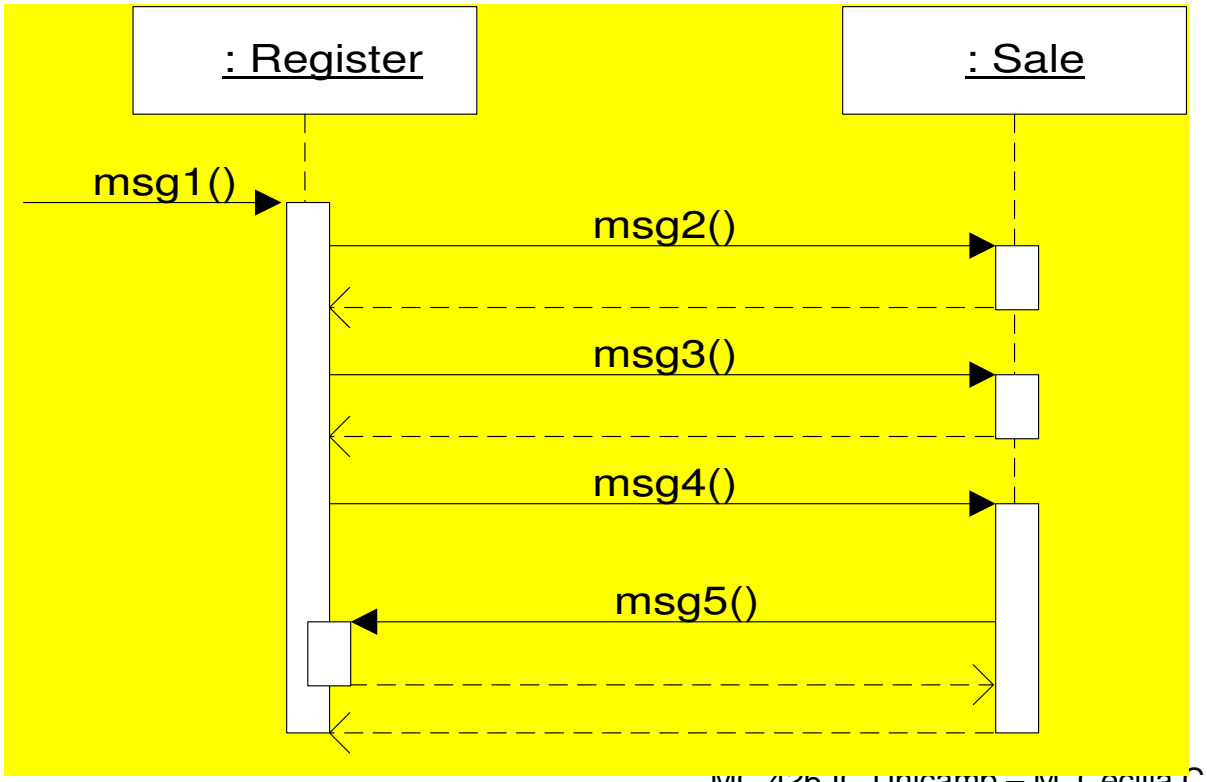
Mensagens e foco no controle



Based on C. Larman, 2002

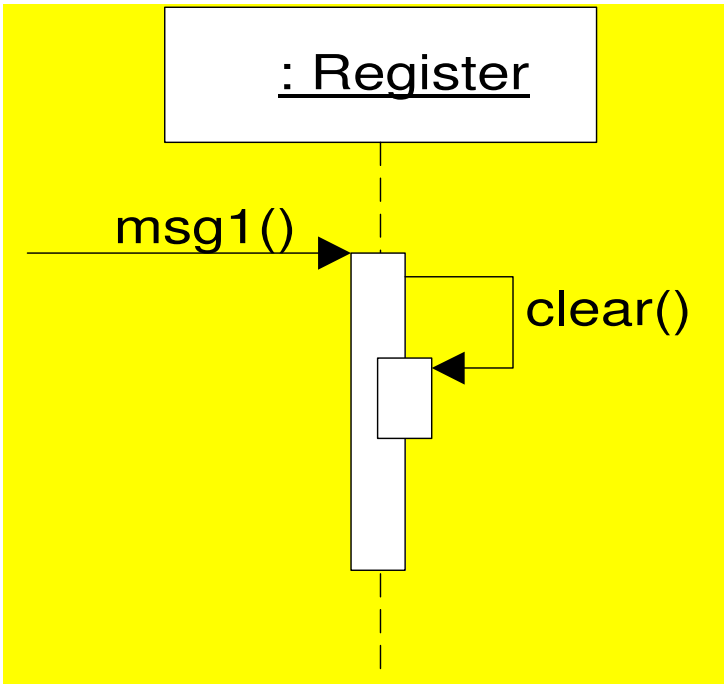
Ilustrando retornos

Linhas pontilhadas no final de uma caixa de ativação

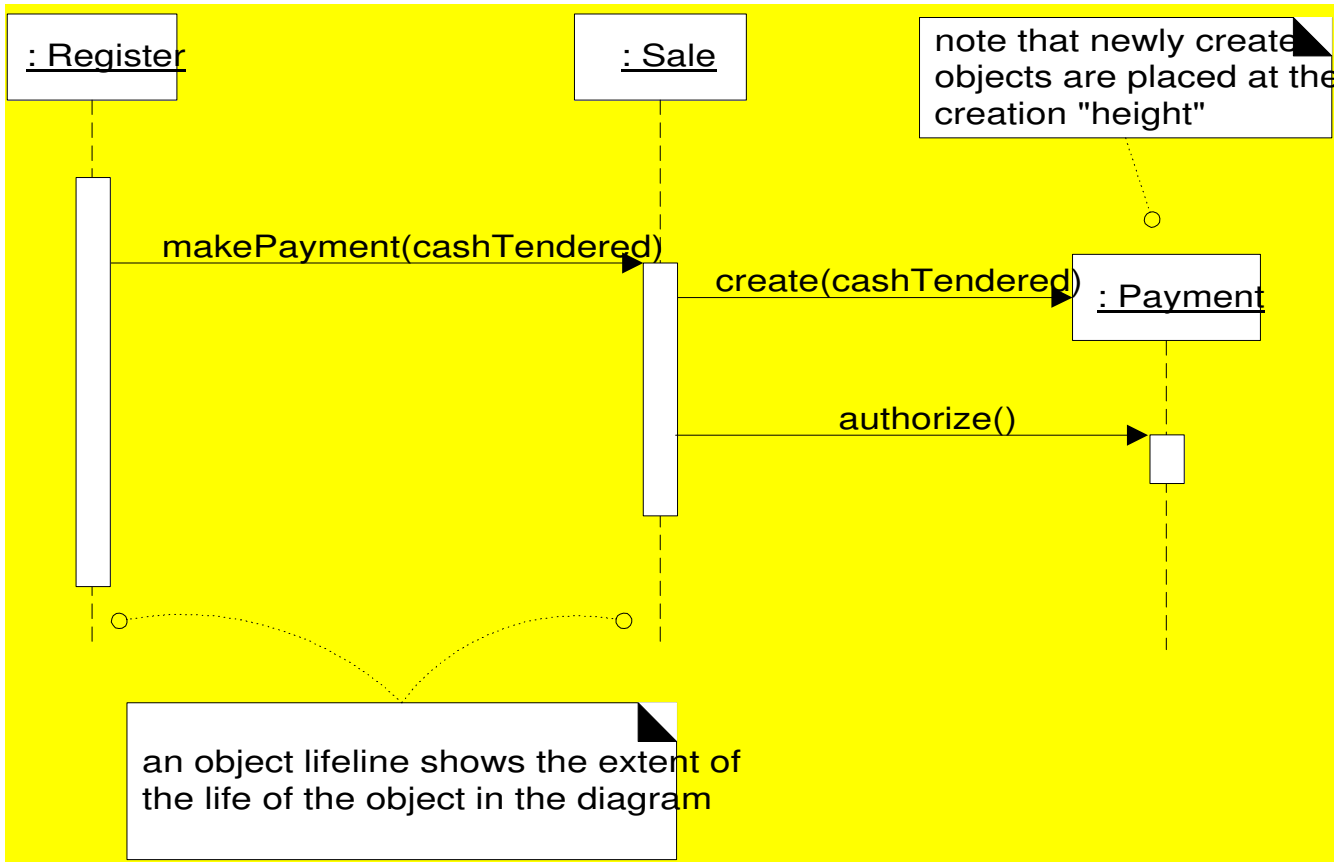


Based on C. Larman, 2002

Mensagens para “self” or “this”

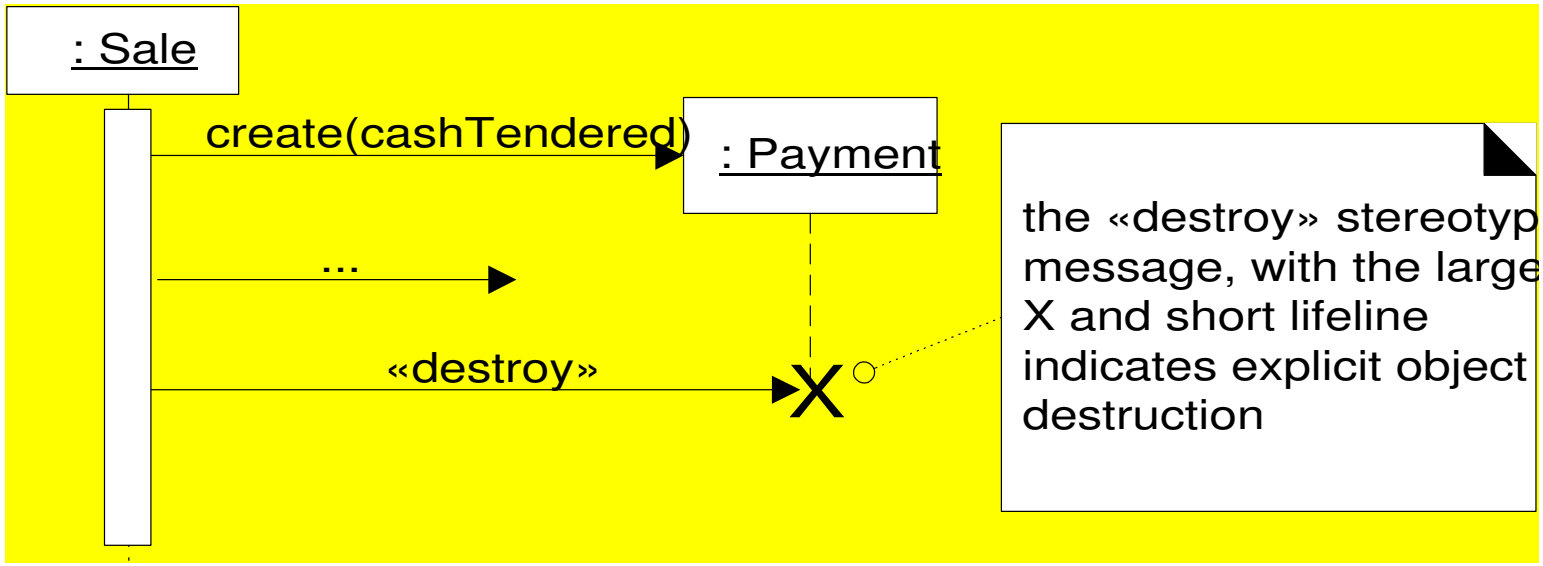


Criação de Instâncias



Based on C. Larman, 2002

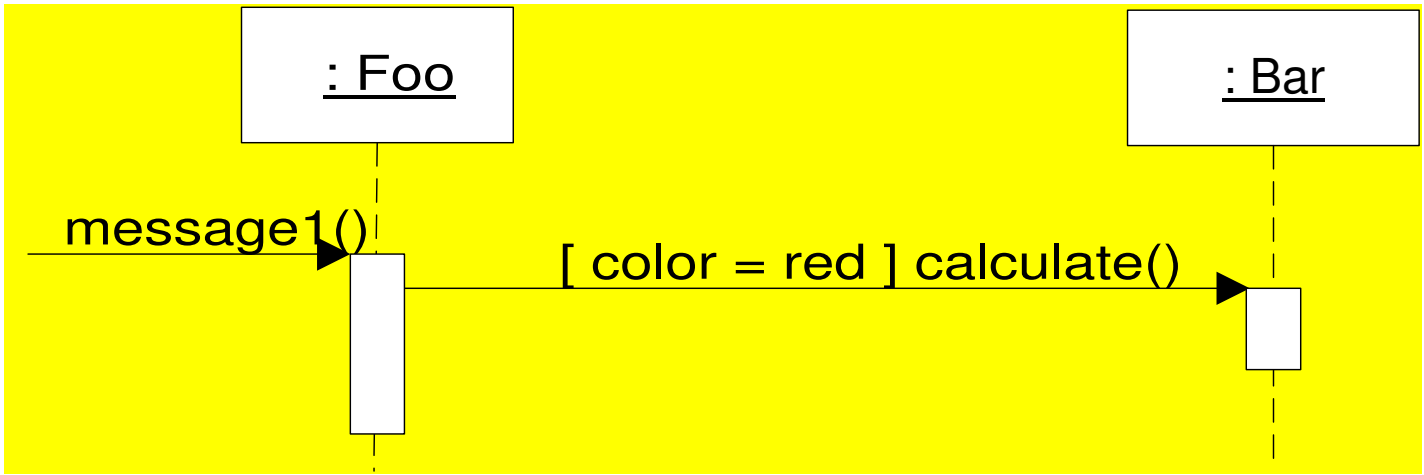
Destruição de Objeto



Based on C. Larman, 2002

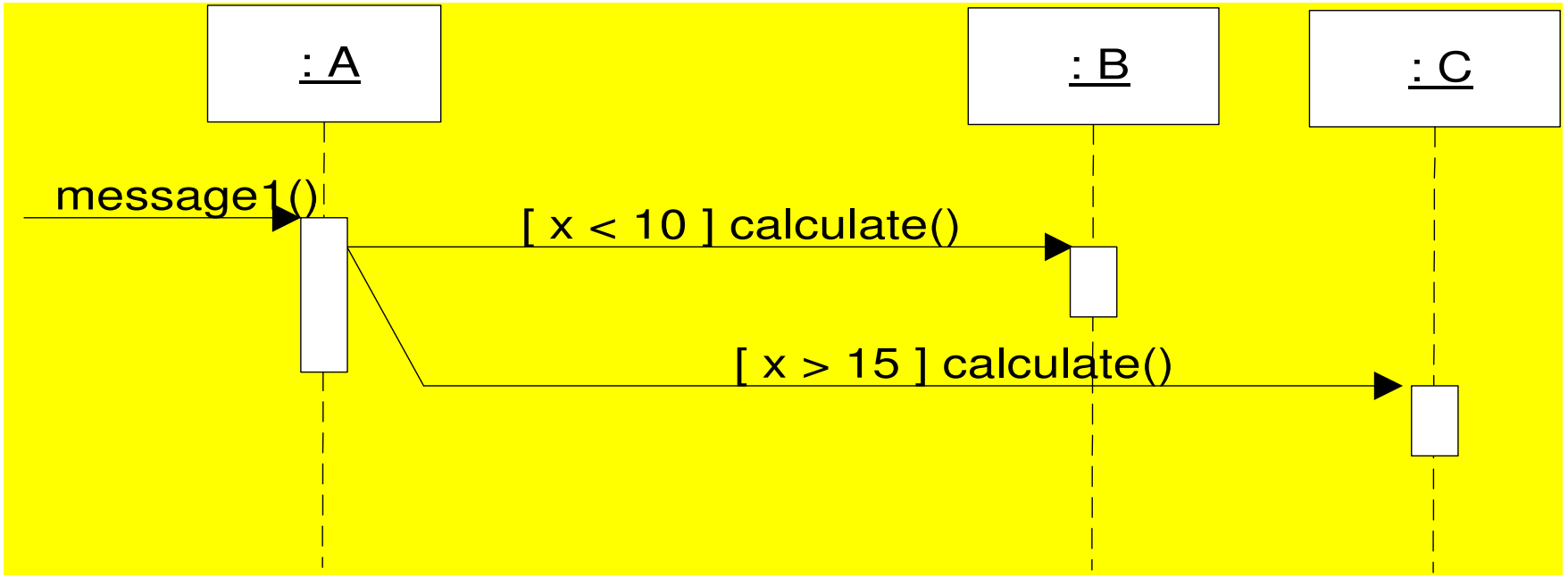
Linhas pontilhadas na vertical abaixo de objetos mostram a vida do objeto. Mostrar a destruição é importante especialmente quando a linguagem não faz garbage collection [ex. C++]

Mensagens condicionais



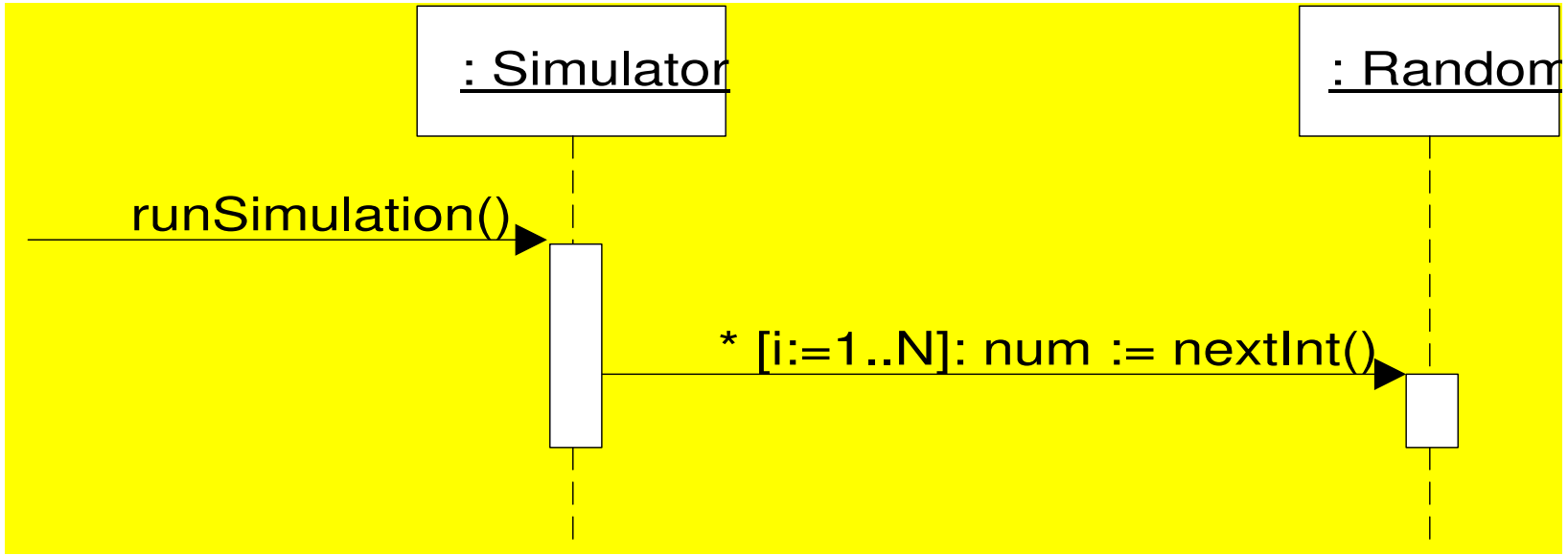
Based on C. Larman, 2002

Mensagens condicionais mutuamente exclusivas



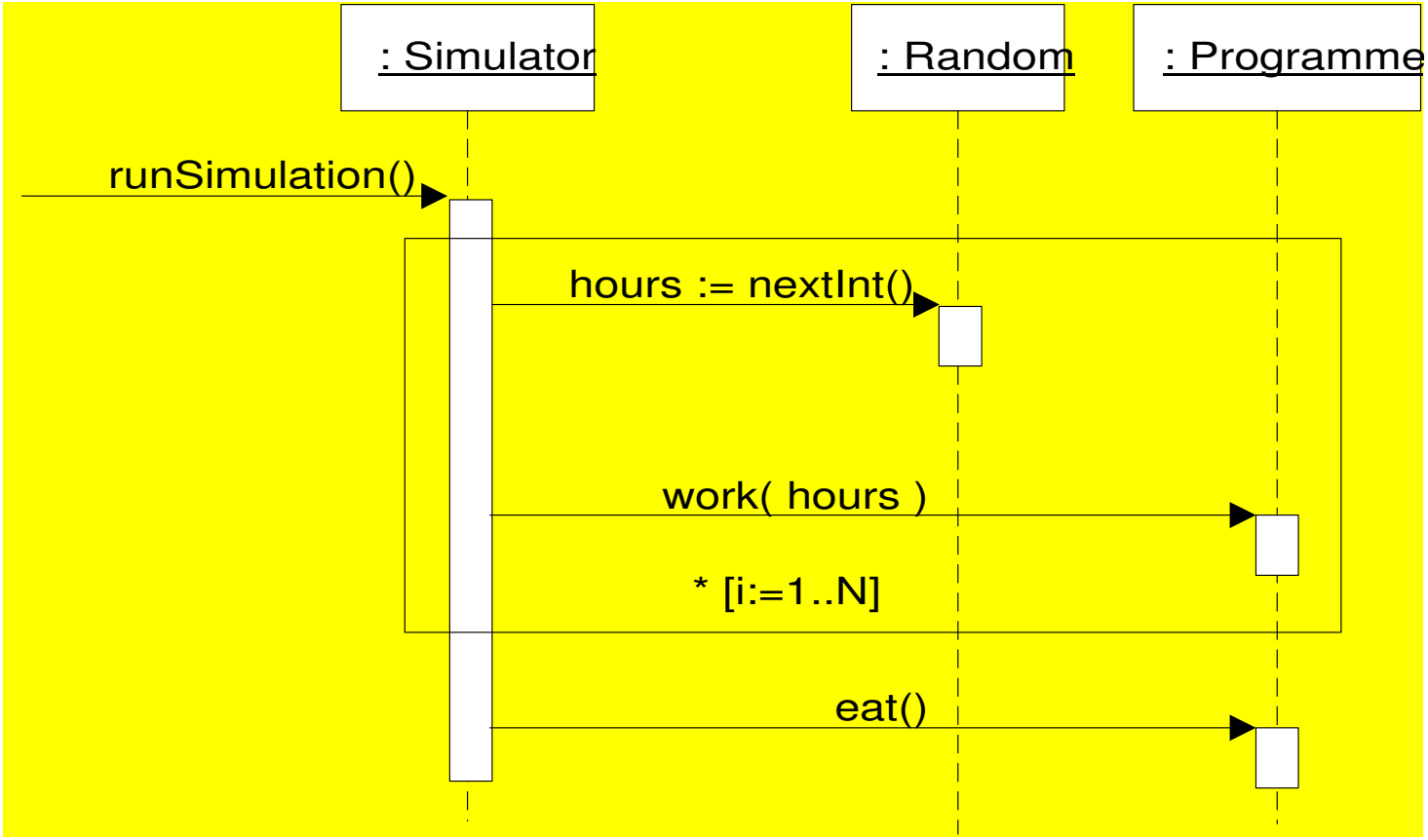
Based on C. Larman, 2002

Iteração para uma única mensagem



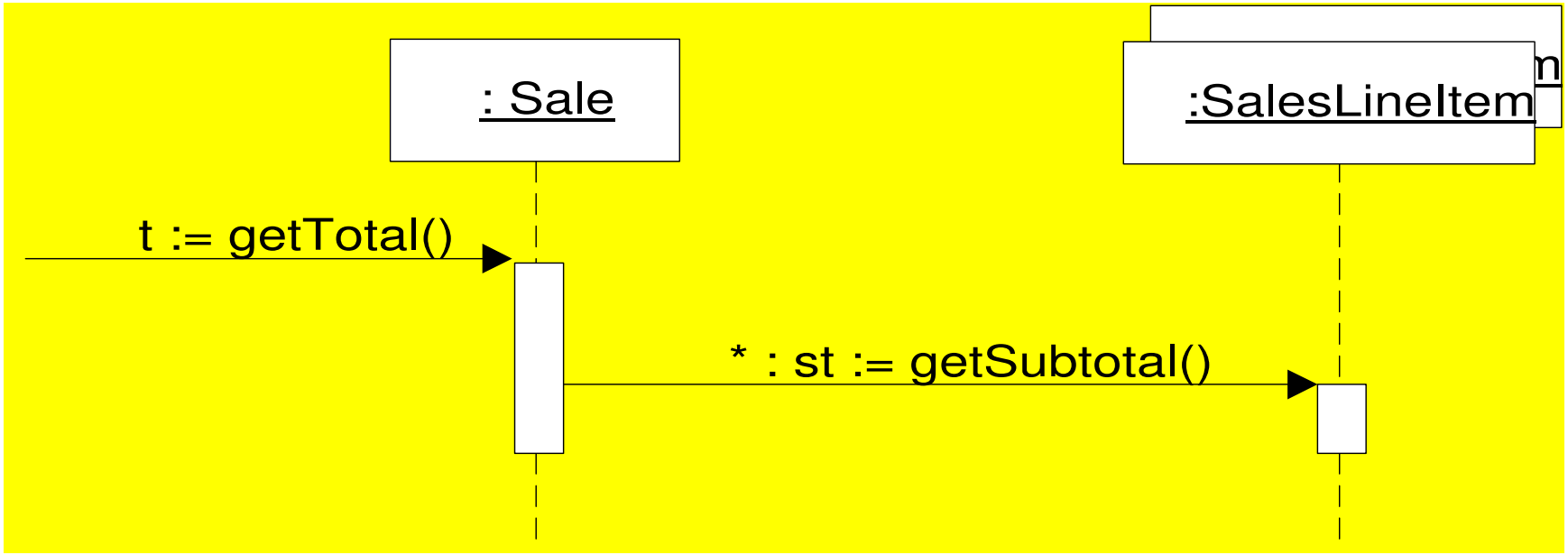
Based on C. Larman, 2002

Iteração para uma série de Mensagens

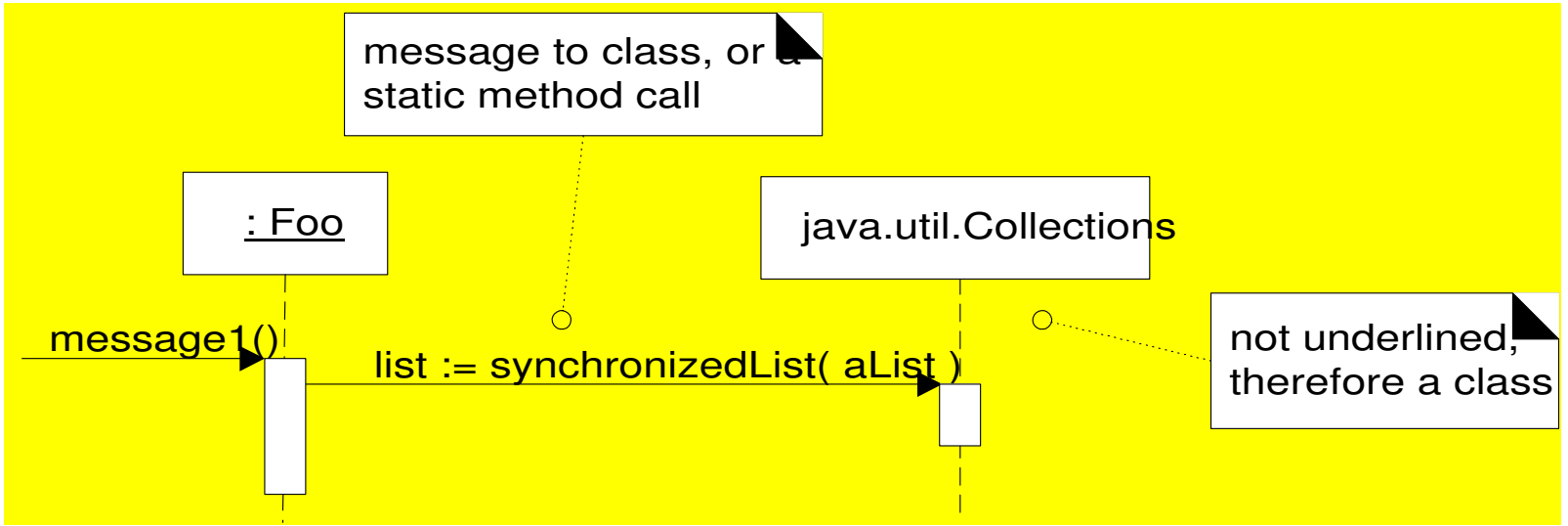


Based on C. Larman, 2002

Iteração sobre uma Coleção



Mensagens a Classe de Objetos



A seguir...

- Como pensar em e projetar com objetos
- Fundamental design de objetos requer conhecimento de:
  - Princípios de atribuição de responsabilidade
  - *design patterns*

Referências

- Larman, C. (2002) *Applying UML and Patterns – An Introduction to Object Oriented Analysis and Design and the Unified Process*, Prentice-Hall Inc.
- Muller, P.A. (1997) *Instant UML*,Wrox Press Ltd.