Software Specification and Design - Week4

By Keeratipong Ukachoke

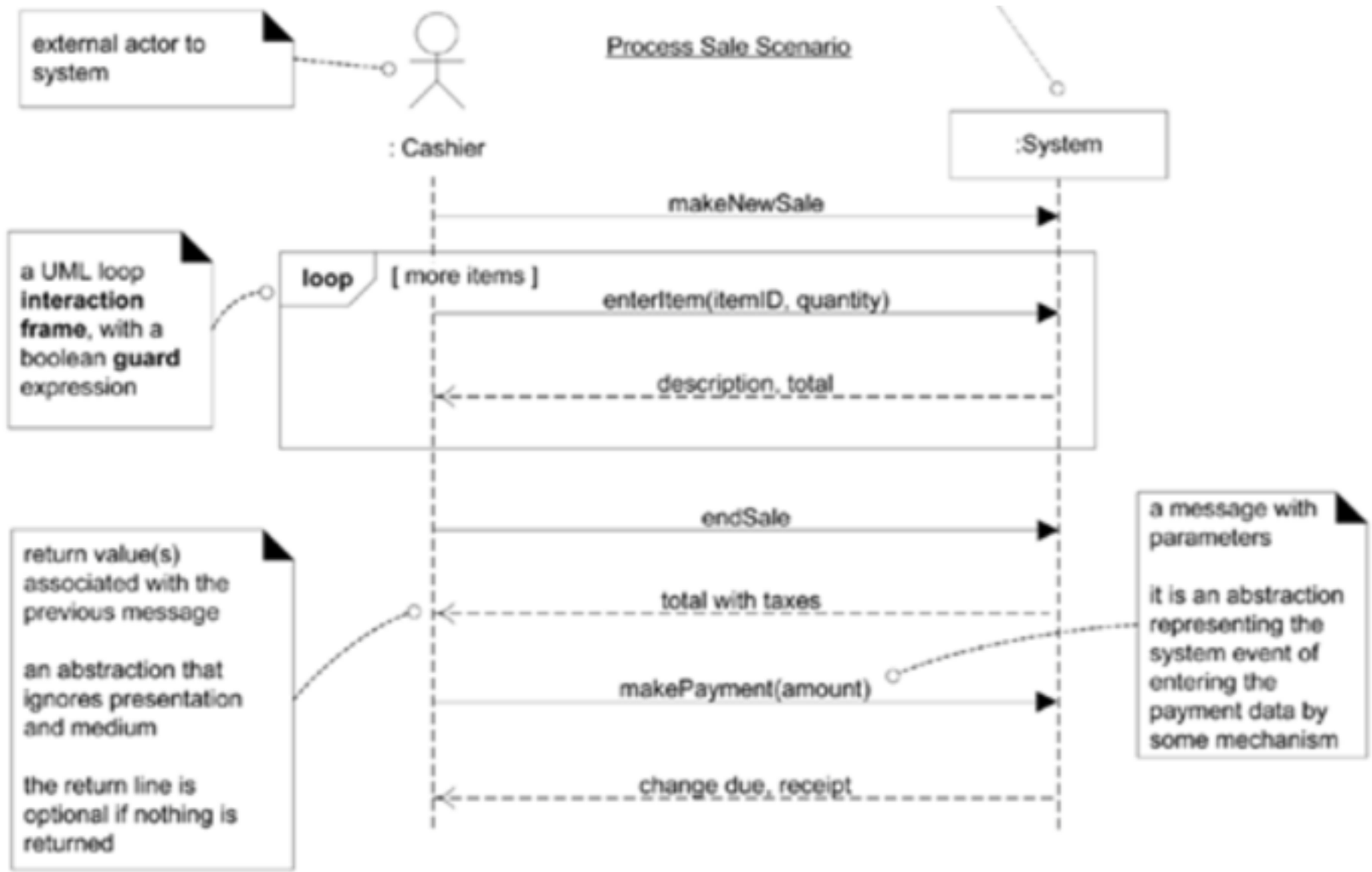
Today Topics

- System Sequence Diagram [basics]
- Operational Contract
- UML Interaction Diagram [basics]
- UML Class Diagrams [basics]
- **Finish the 1st (the easy part) of the course!!!**

System Sequence Diagram

- Illustrate input/output events of the system
- Act as an input of operational contracts
- Will be used in Object design

SSD Example


external actor to system
Process Sale Scenario
: Cashier
:System
makeNewSale
a UML loop interaction frame, with a boolean guard expression
loop
[more items]
enterItem(itemID, quantity)
description, total
endSale
a message with parameters
it is an abstraction representing the system event of entering the payment data by some mechanism
return value(s) associated with the previous message
an abstraction that ignores presentation and medium
the return line is optional if nothing is returned
total with taxes
makePayment(amount)
change due, receipt

What is SSD

- A system sequence diagram is a picture that shows **one scenario** of a use case, the events that **external actor** generate, their **order**, **inter system** events.

Why draw SSD

- What are the inputs of the system
- Higher level
- Black box behaviour

SSD and Use Case

1. Customer arrives at POS with items
2. Cashier starts a new sale
3. Cashier enters item id
4. System records sale line item and present item description, price, total

- - - Cashier repeat steps 3-4 until done

5. System presents total with taxes calculated
6. Cashier tells customer the total, and asks for payment
7. Customer pays and system handles payment

SSD Naming

- Should be expressed at the abstract level
- Because we focus on the intent and event not UI
- Similar to use case
- Example, ‘confirm’ is better than ‘click ok button’

Operational Contract

What is operational contract

- Provide more details to each operation
- Talk about preconditions and postconditions

Operational Contract Example

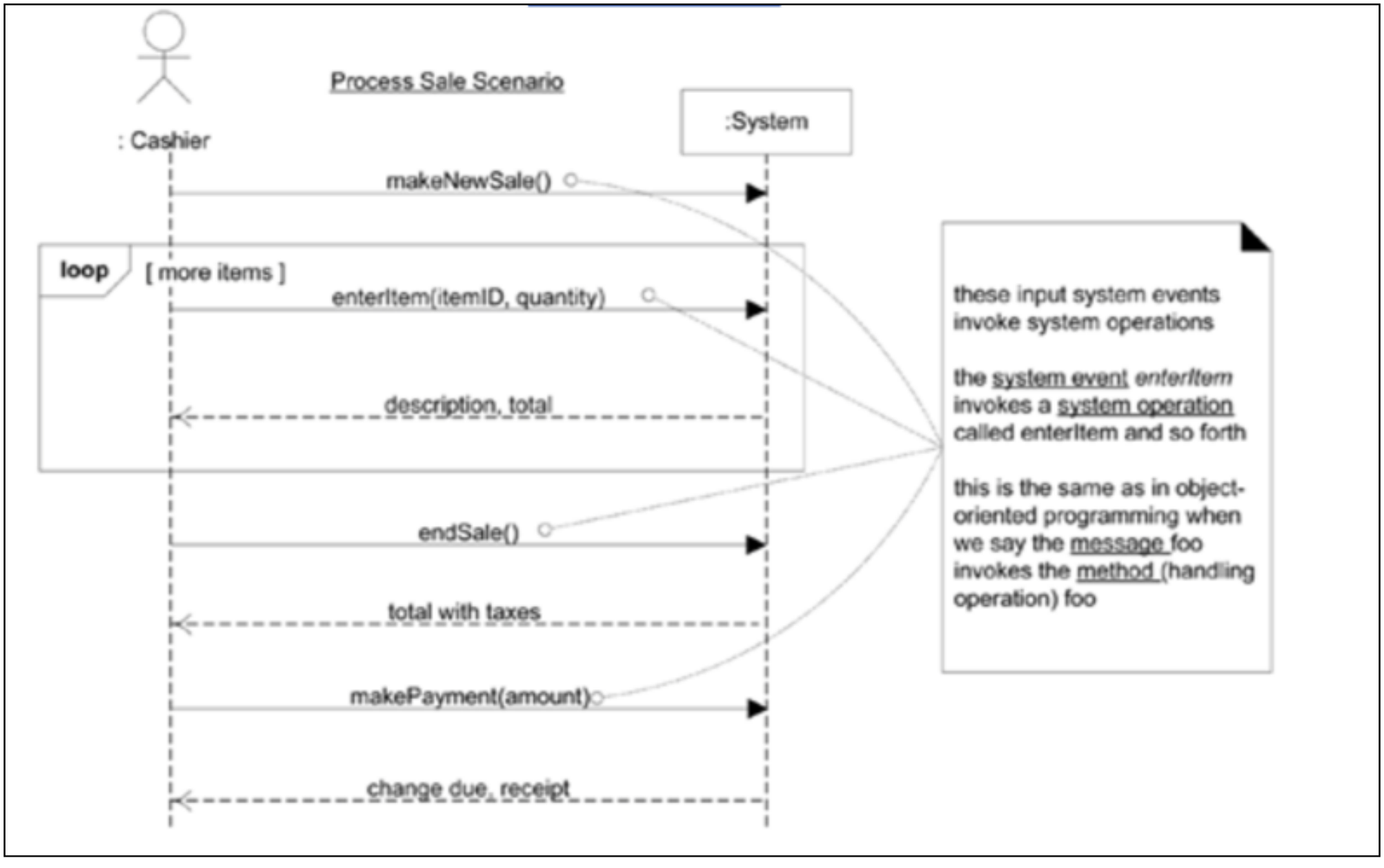
Contract CO2: enterItem

Operation:	enterItem(itemID: ItemID, quantity: integer)
Cross References:	Use Cases: Process Sale
Preconditions:	There is a sale underway.
Postconditions:	- A SalesLineItem instance sli was created *(instance creation)*. - sli was associated with the current Sale *(association formed)*. - sli.quantity became quantity *(attribute modification).* - sli was associated with a ProductDescription, based on itemID match *(association formed).*

Operational Contract Sections

- Operation : Name, parameters
- Cross References: Use cases this operation occur
- Preconditions: What must be fulfilled before the operation begins
- Postconditions: What must be fulfilled after the operation is performed

OC with SSD


Process Sale Scenario
: Cashier
:System
makeNewSale()
loop
[more items]
enterItem(itemID, quantity)
description, total
endSale()
total with taxes
makePayment(amount)
change due, receipt
these input system events invoke system operations
the system event enterItem invokes a system operation called enterItem and so forth
this is the same as in object-oriented programming when we say the message foo invokes the method (handling operation) foo

Post conditions

- Don’t be confused. It’s the action. But things that change after operations.
- Instance creation or deletion
- Attributes change of value
- Association formed and broken

How to write post conditions

- Express postconditions in past tense to emphasize they are observations about stat changes.
- Example
    - A SaleLineItem was created - Good
    - Create a SaleLineItem - Worse
    - A SaleLineItem is created - Worse

In class question

- Let’s write operation contract of ‘makeNewSale’

In class question

- Let’s write operation contract of ‘endSale’

In class question

- Let’s write operation contract of ‘makePayment’

Modify domain model

- Look at ‘endSale()’ operational contract
- There should be something to indicate that the sale is ended.
- Let’s add isComplete as a boolean to the model

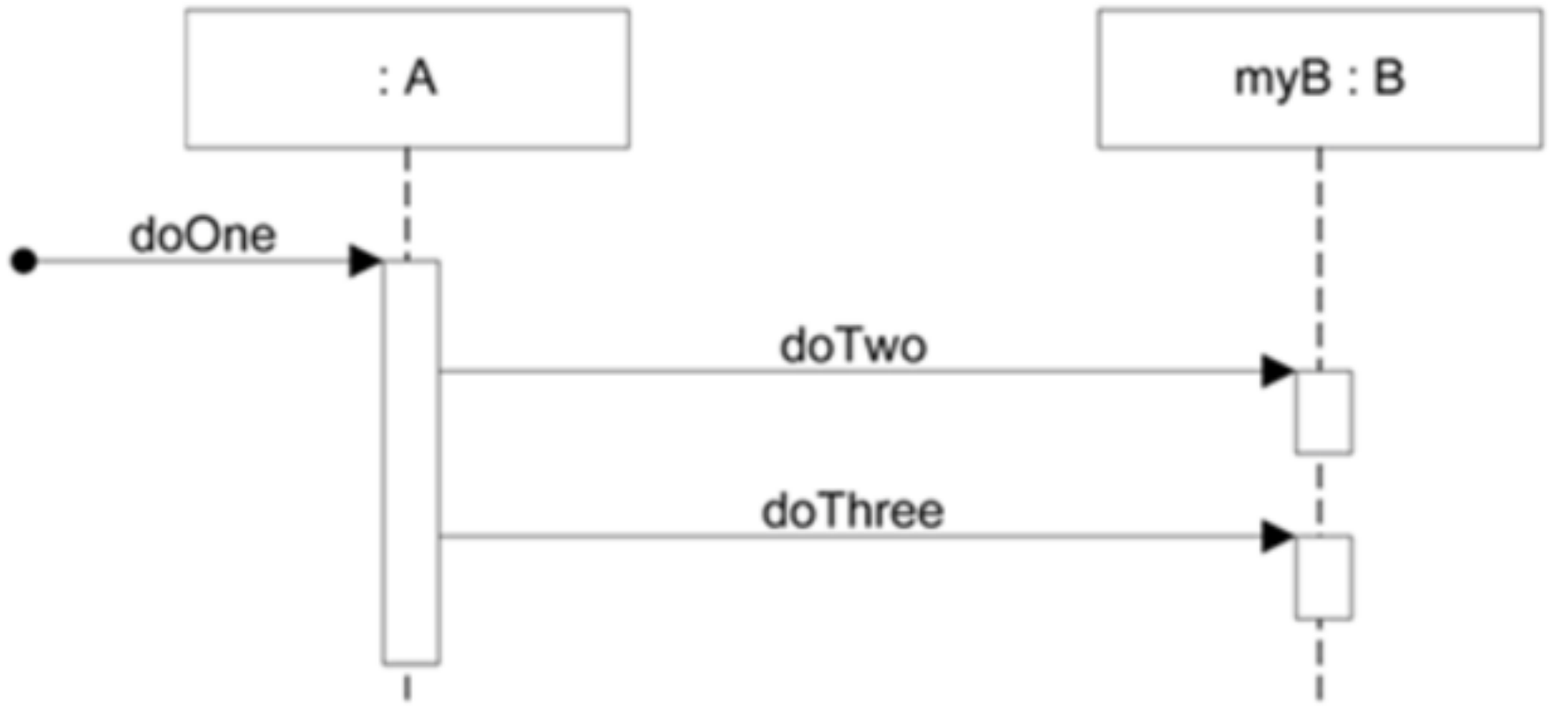
UML Interaction Diagrams

What are interaction diagrams?

- Diagrams we use to show how objects interact via messages.
- Dynamic object modeling
- Two types:
    - Sequence diagrams
    - Communication diagrams

Sequence Diagrams

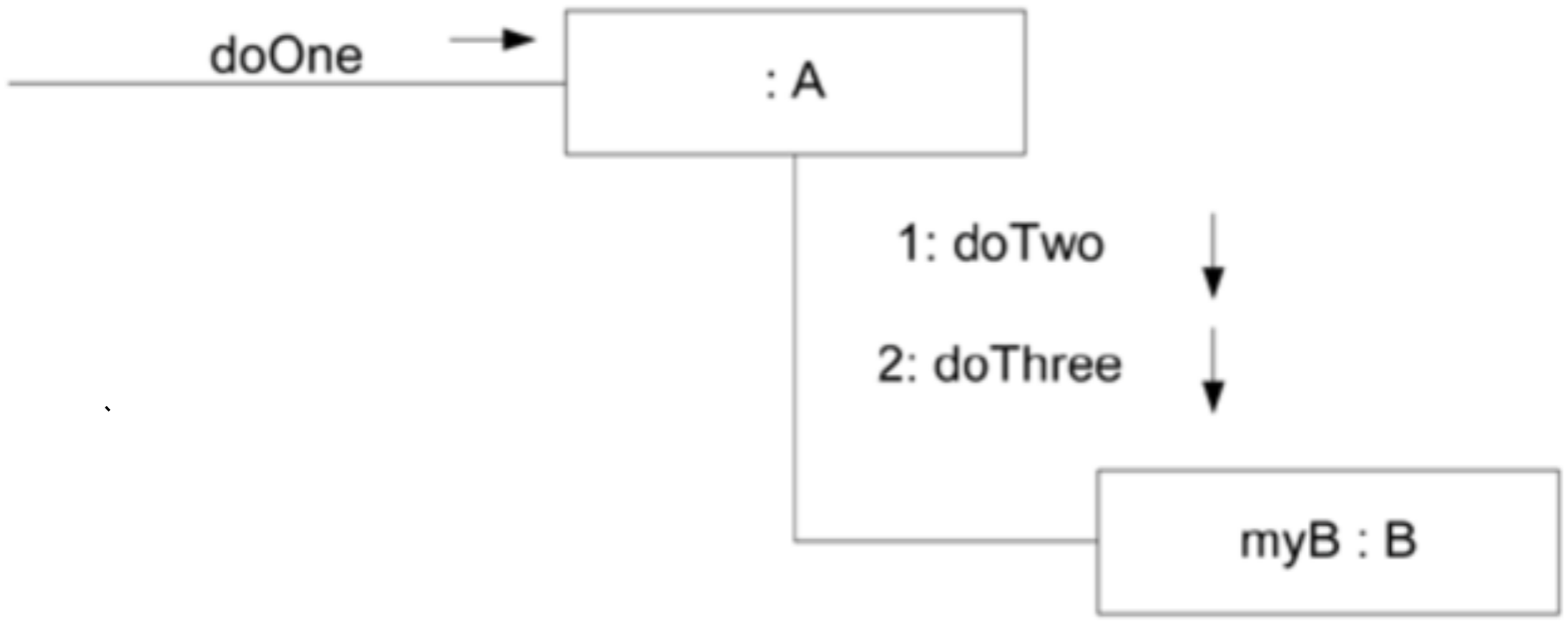
Figure 15.1. Sequence diagram.



How can we turn this into code?

Communication Diagram

Figure 15.2. Communication diagram.



The diagram represents the same code.

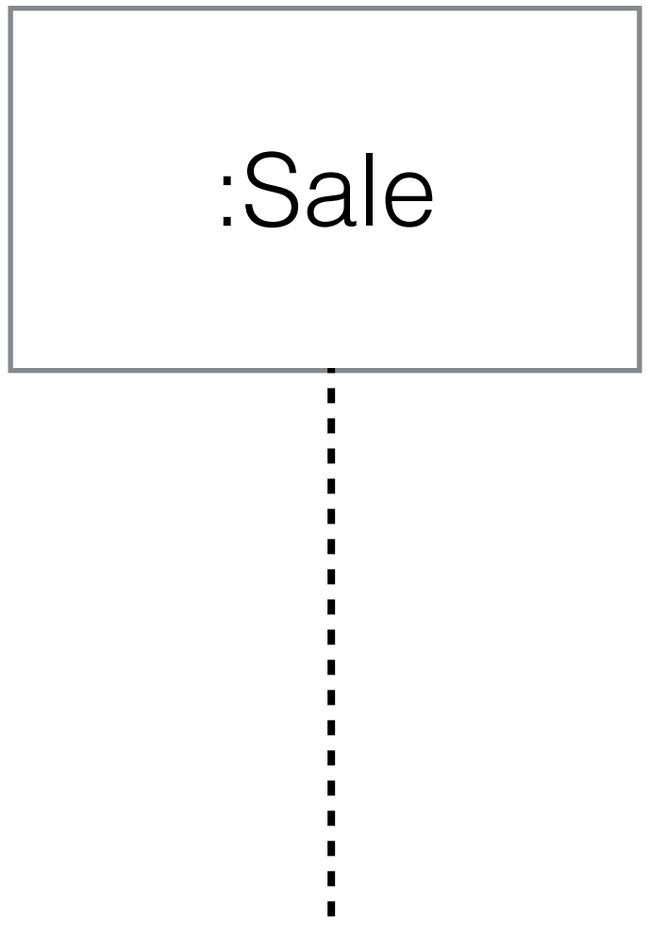
Comparison

- Sequence diagrams
    - The sequence/time of events are clearly demonstrated.
    - Require a lot of space
    - Richer notations
- Communication diagram
    - The sequence/time of events are hard to see
    - Space economical
    - Fewer notations

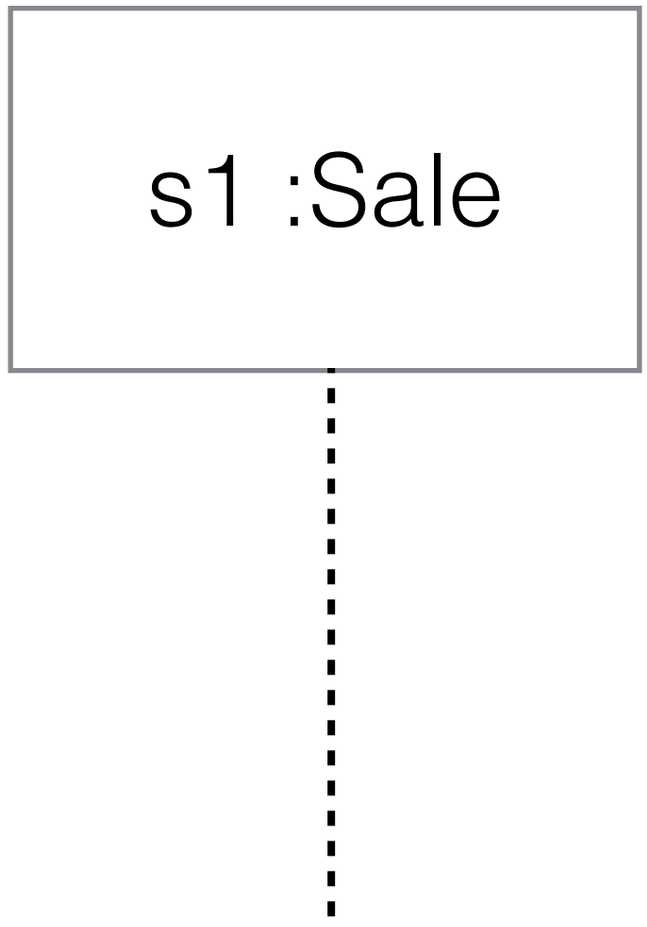
Why interaction diagram is important?

WHY?

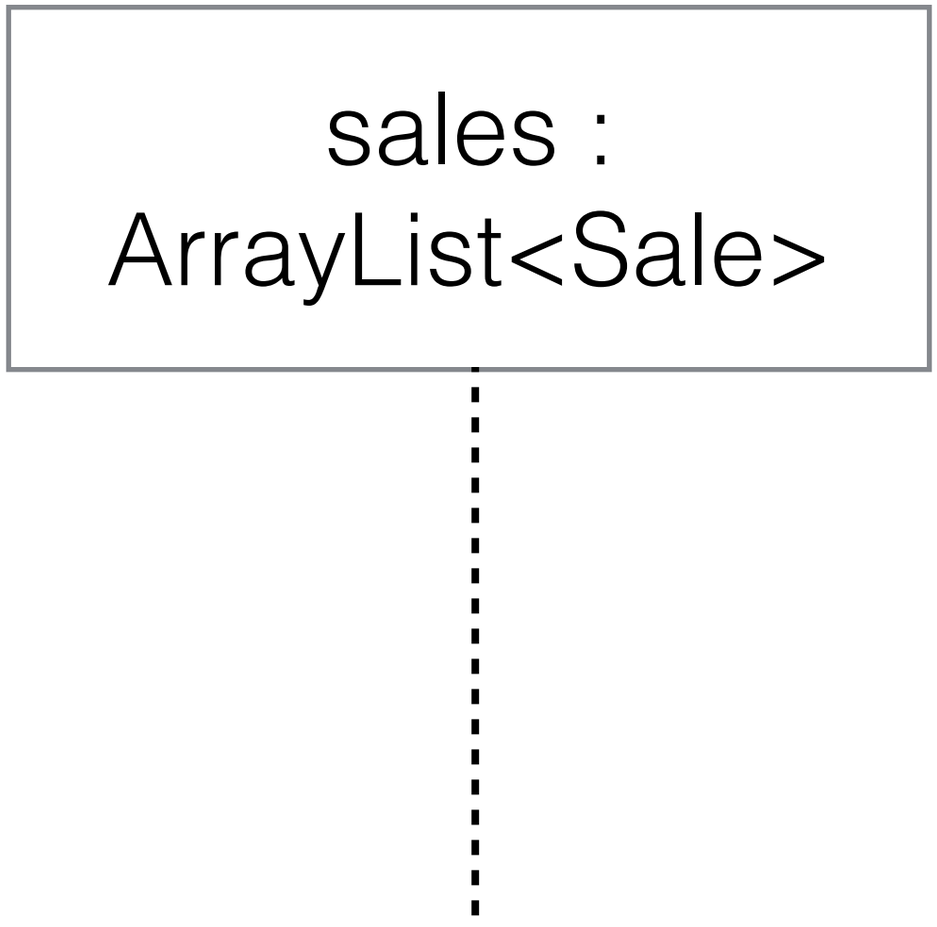
Notations - Life line box



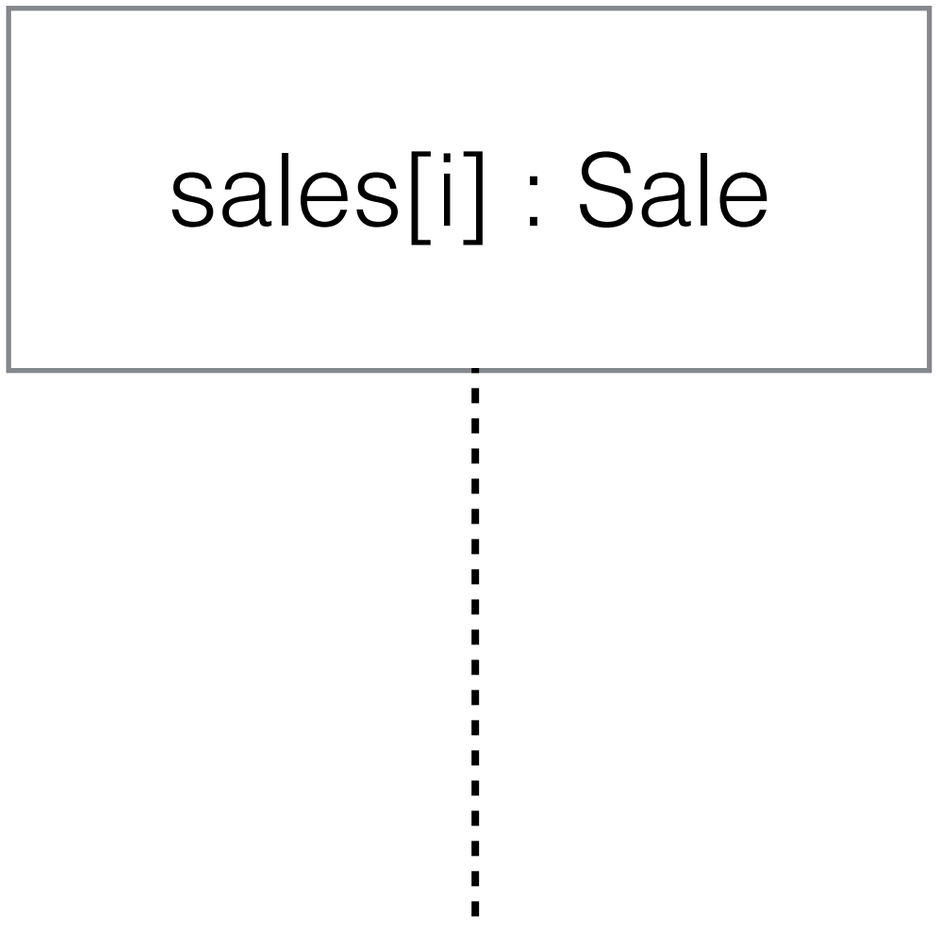
Notations - Life line box



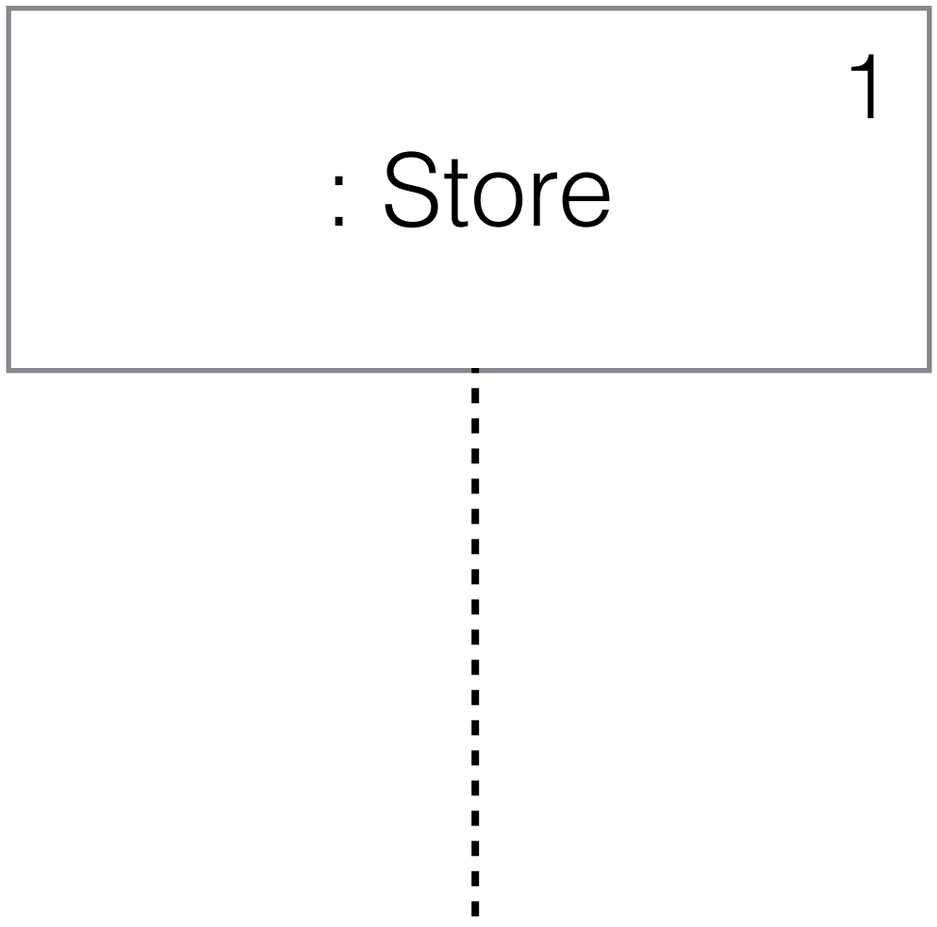
Notations - Life line box



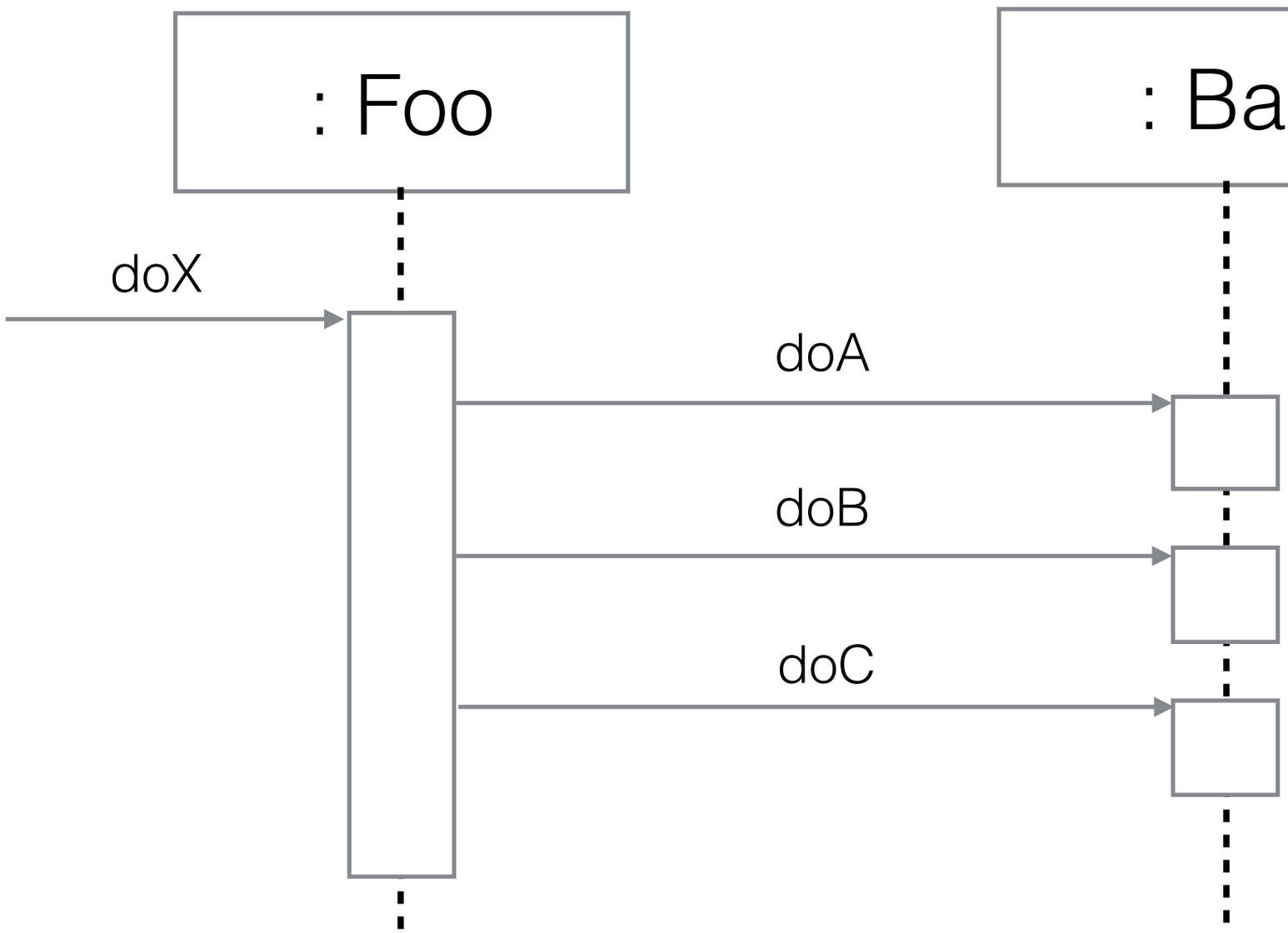
Notations - Life line box



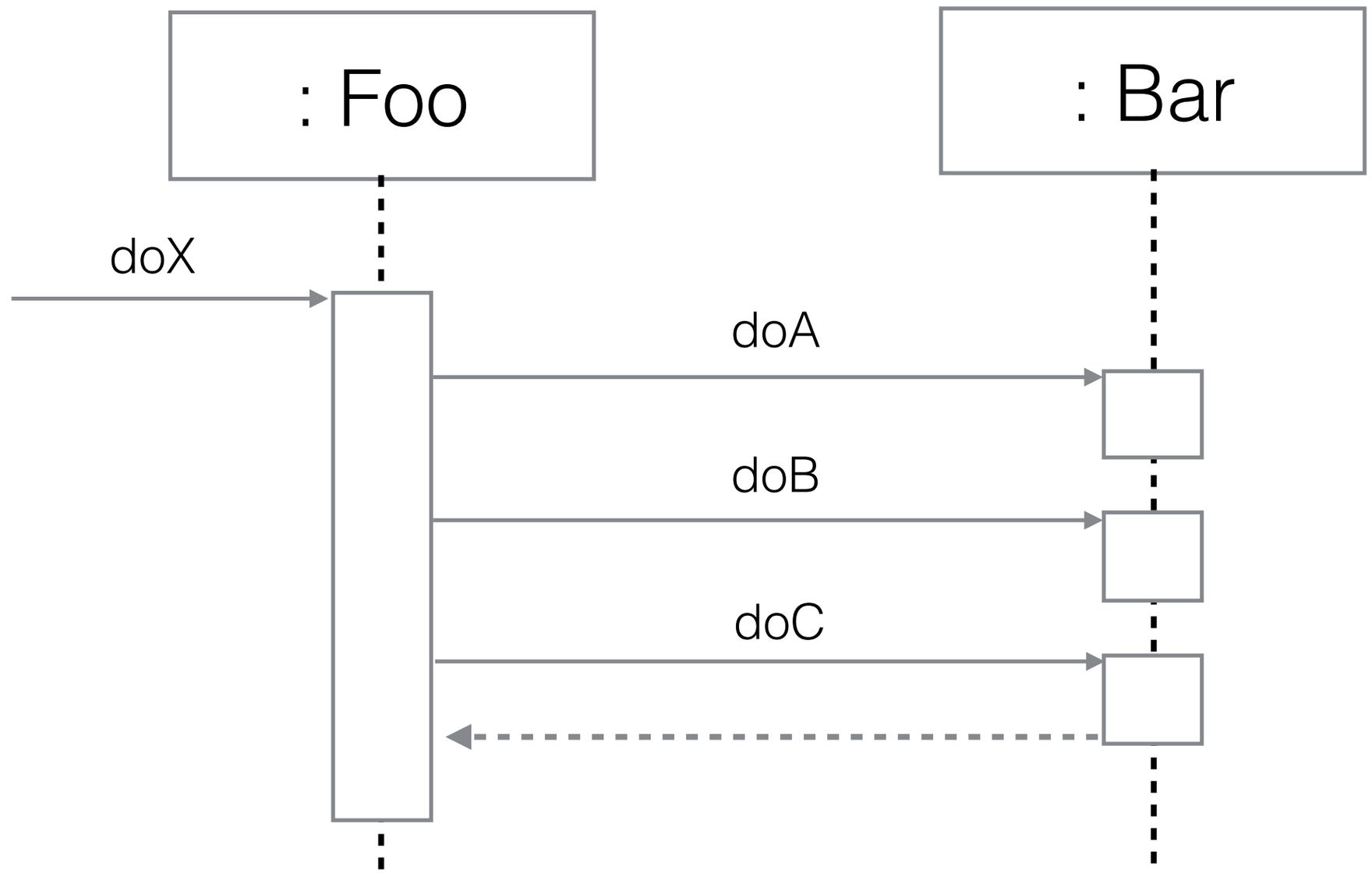
Notations - Singleton



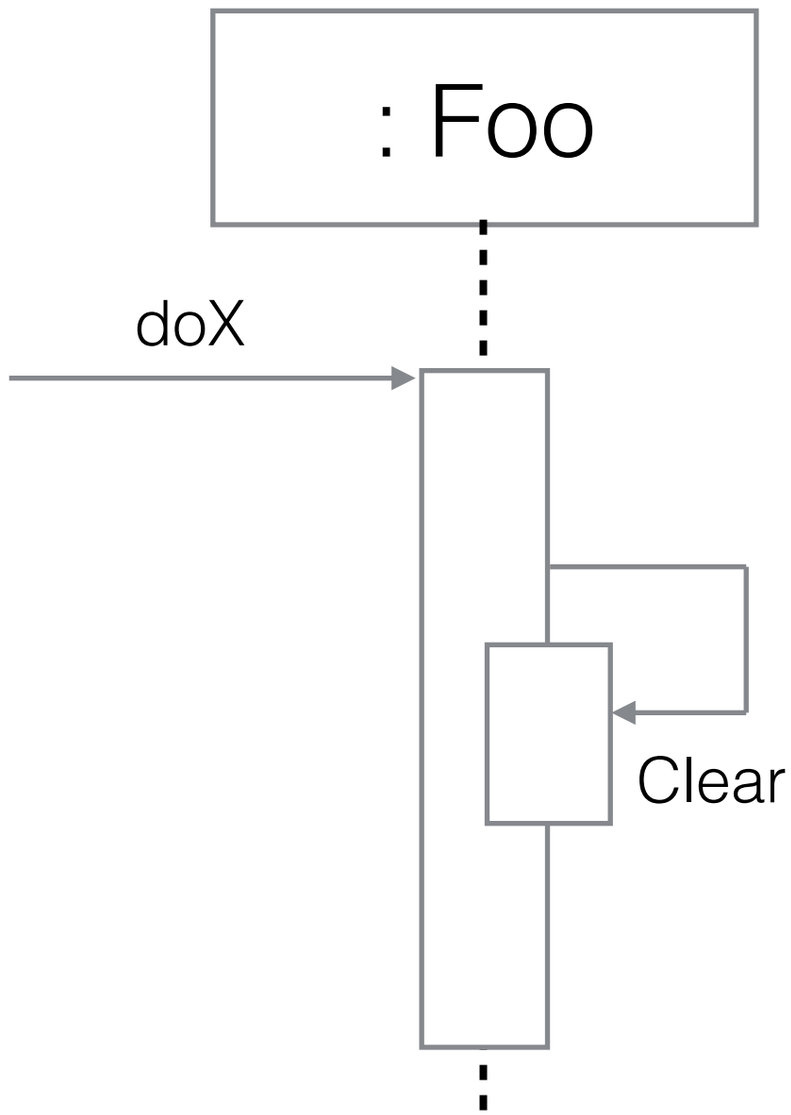
Notations - Message



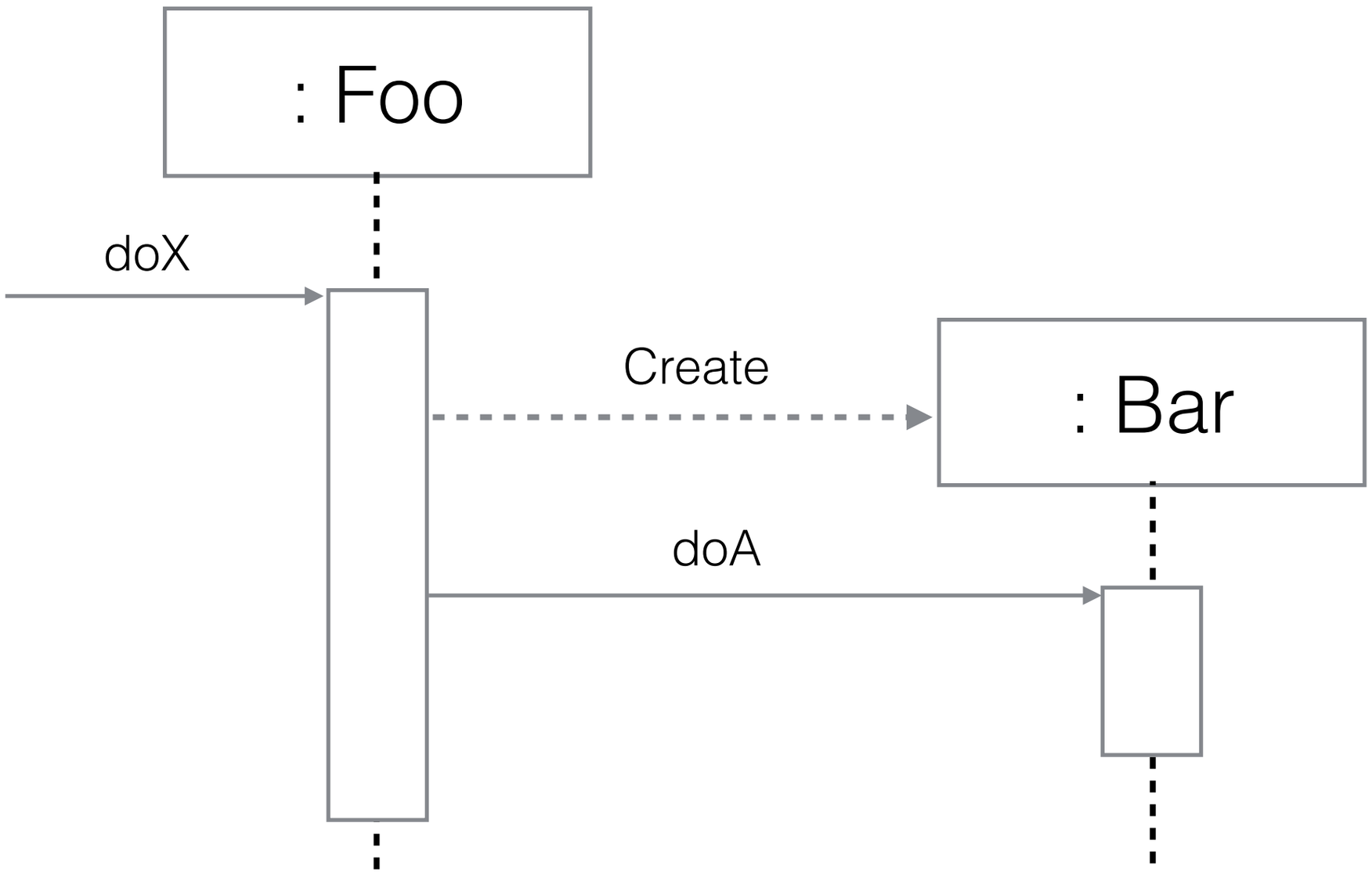
Notations - Returns



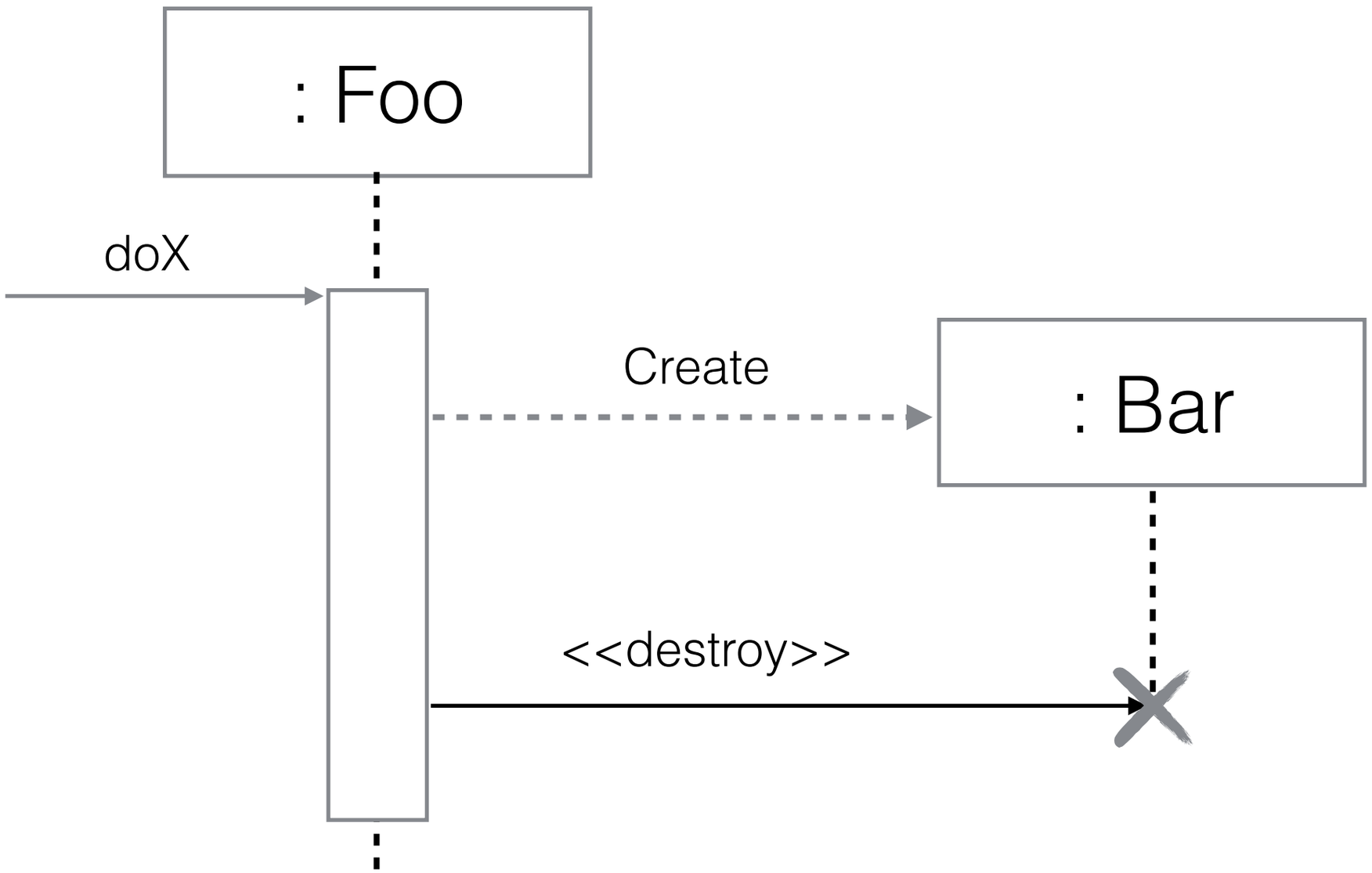
Notations - self/this



Notations - Creation

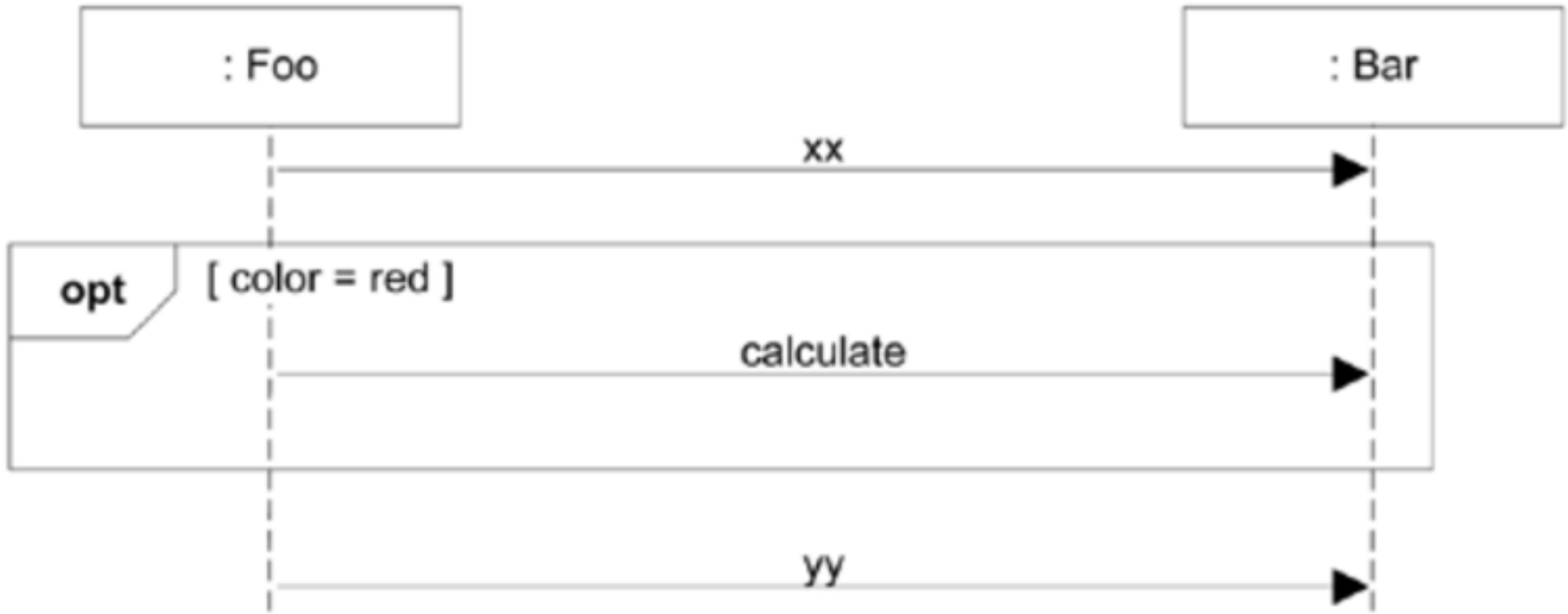


Notations - Termination

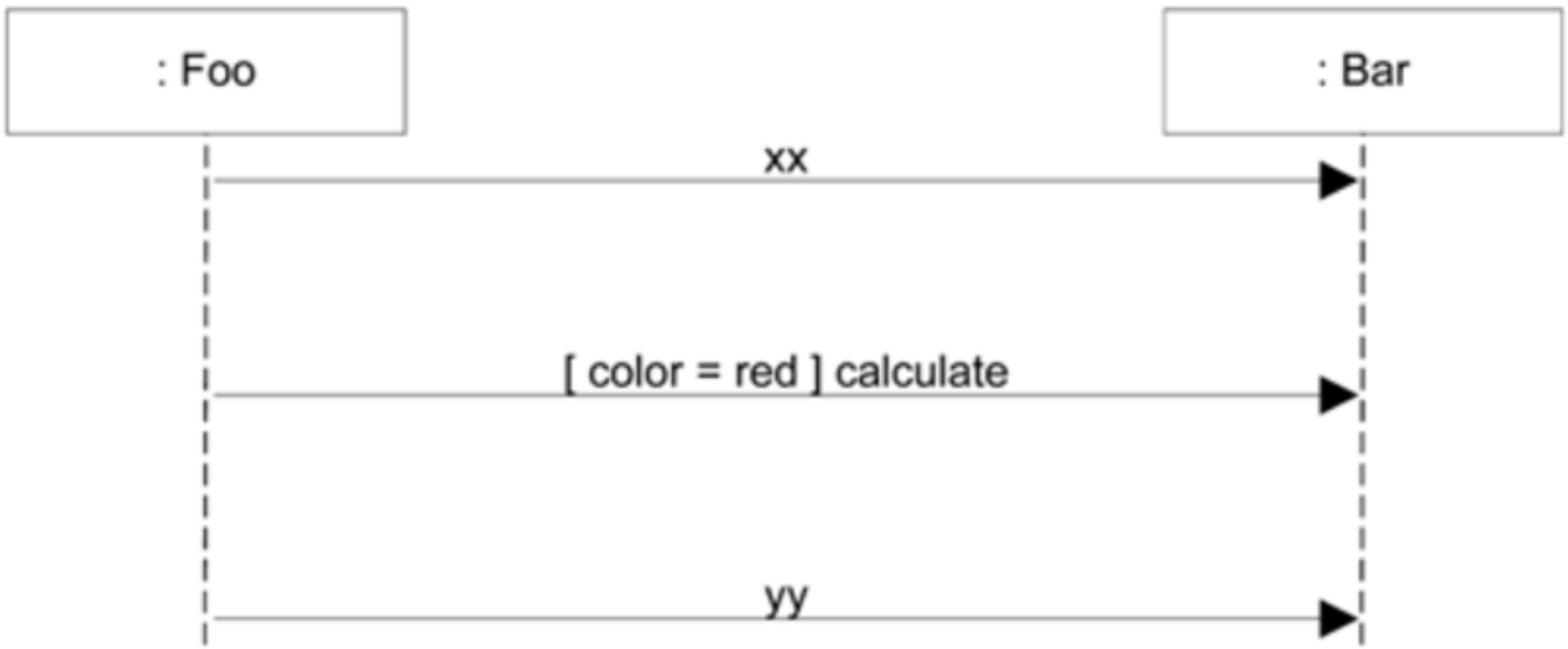


Notations - Loop

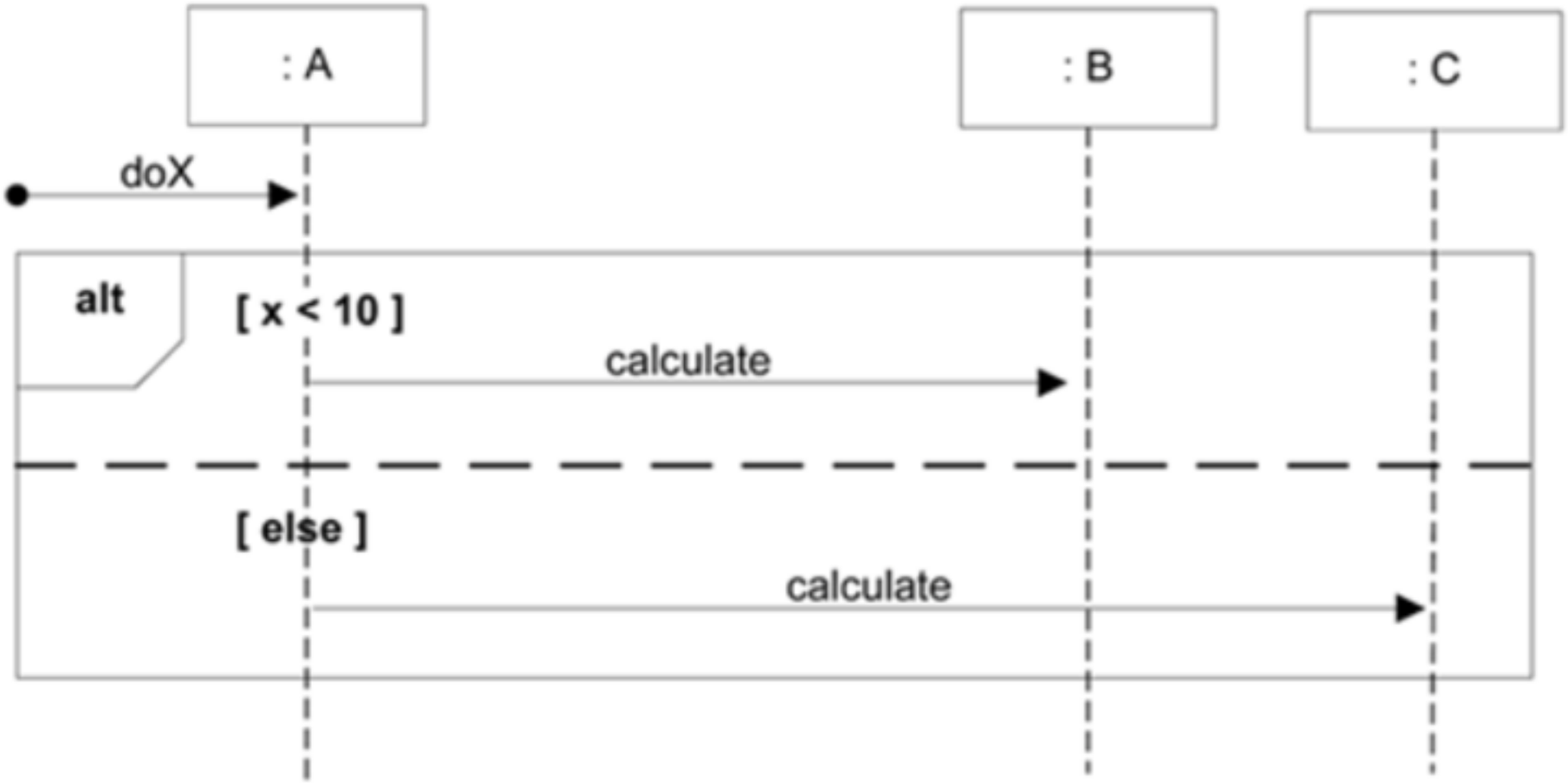
Notations - Conditional



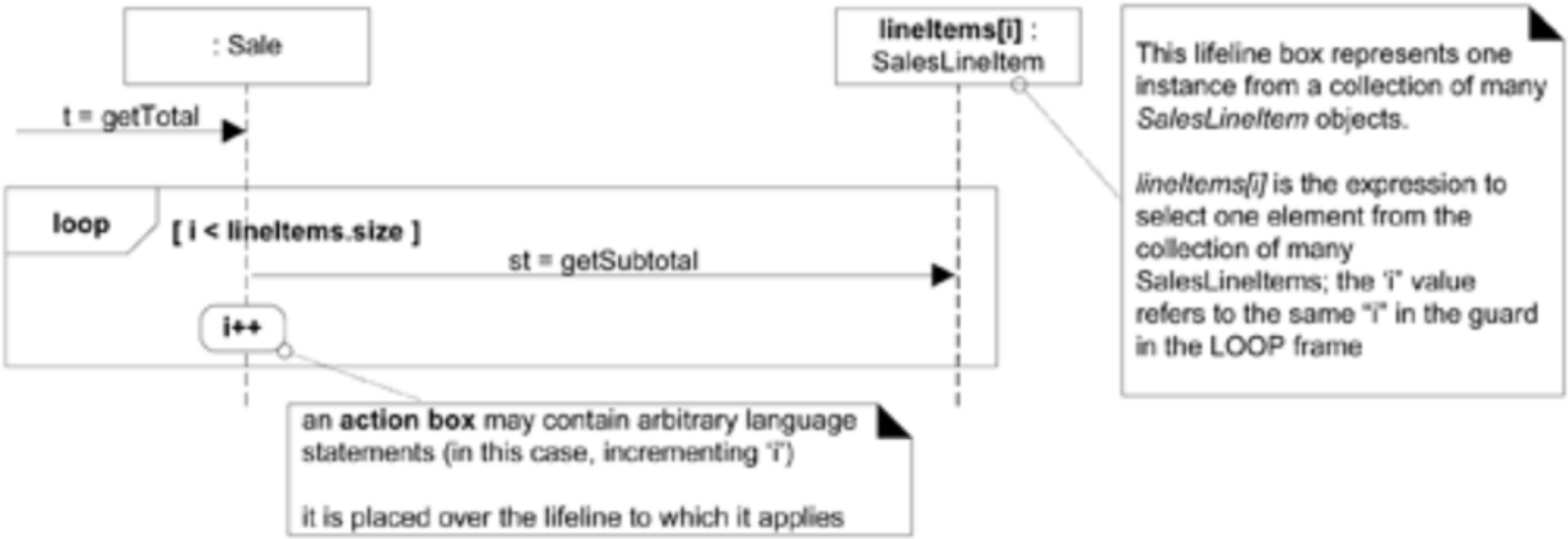
Notations - Conditional (2)



Notations - Conditional (3)

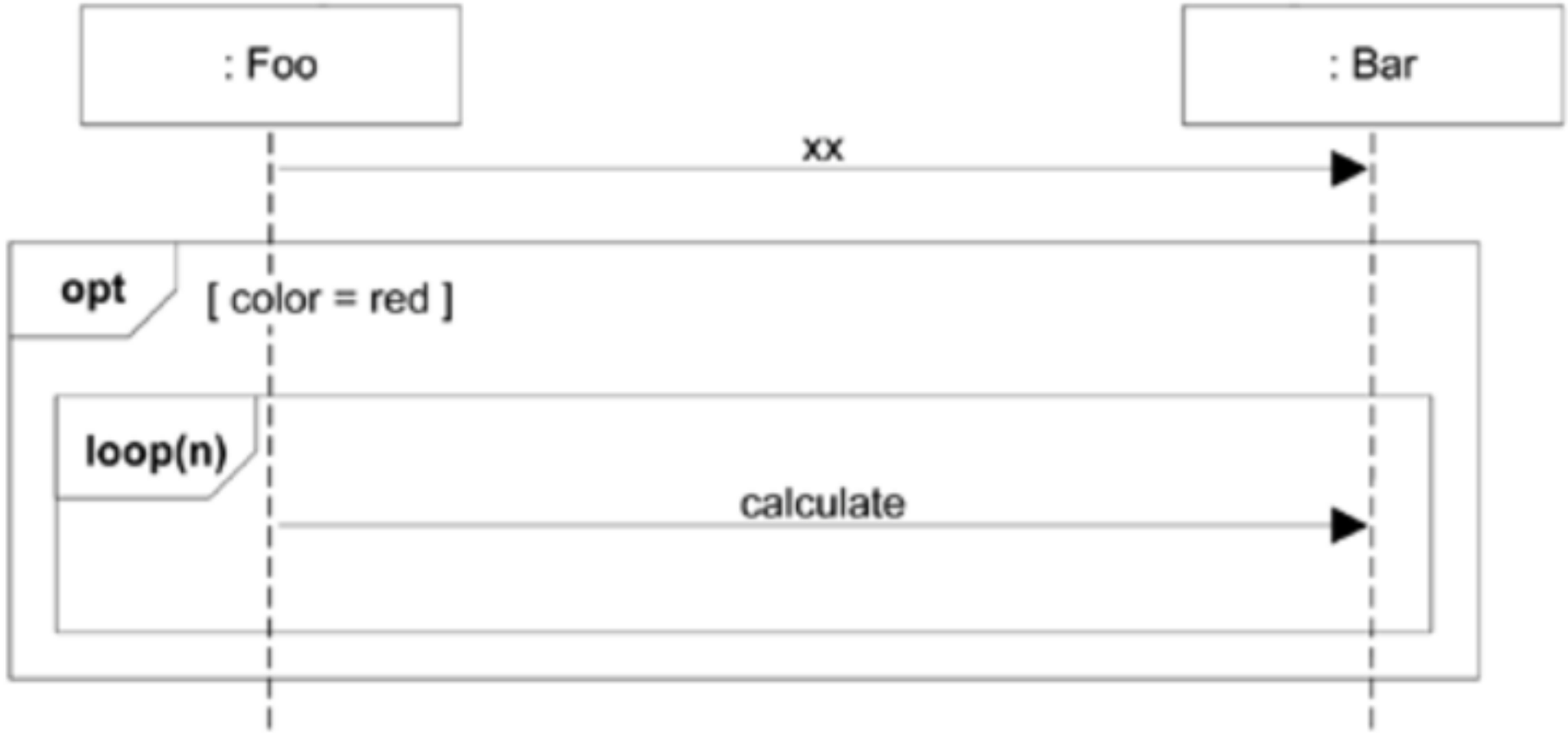


Notations - Iteration

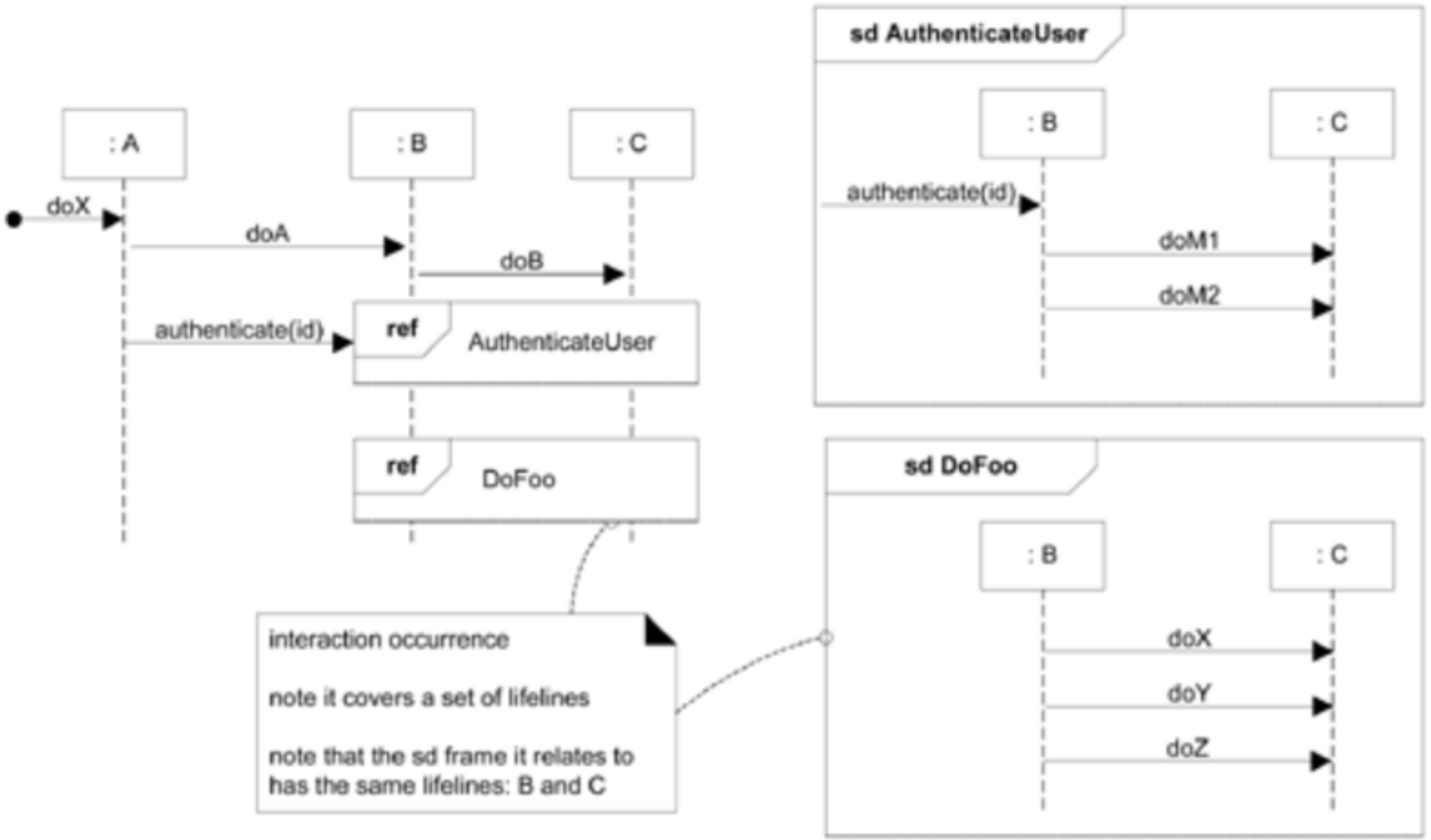


Notations - Iteration (2)

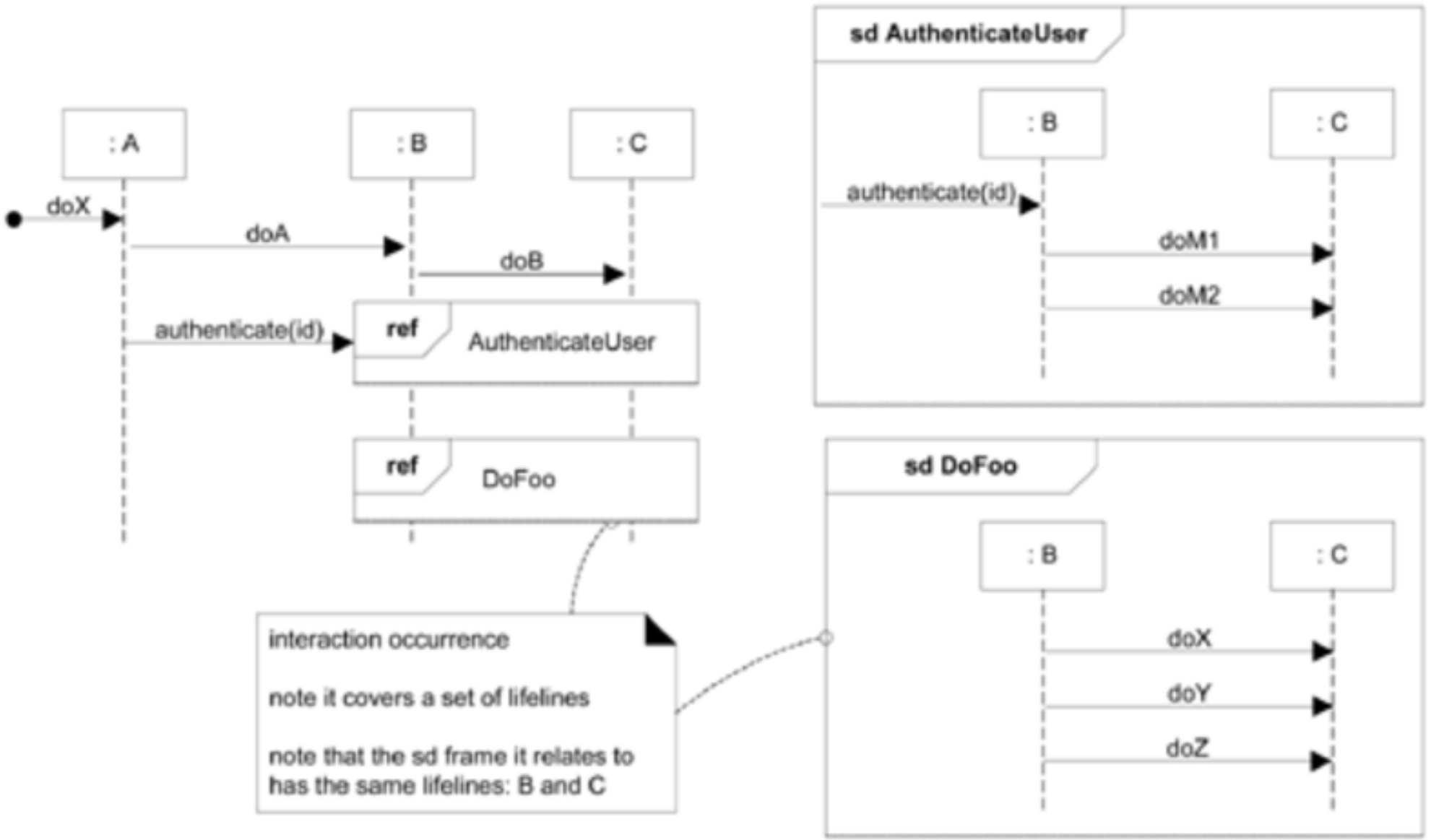
Notations - Nesting



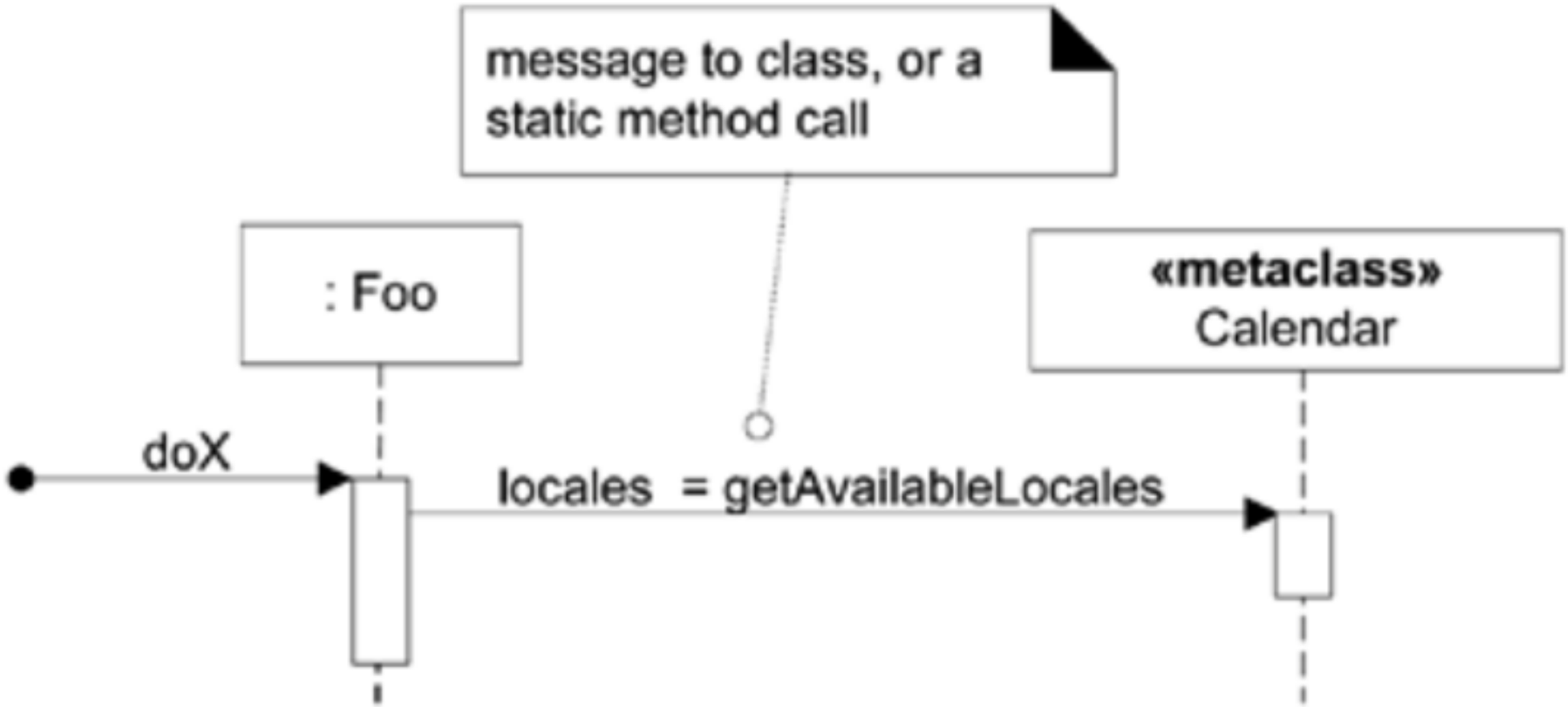
Notations - Reference



Notations - Reference

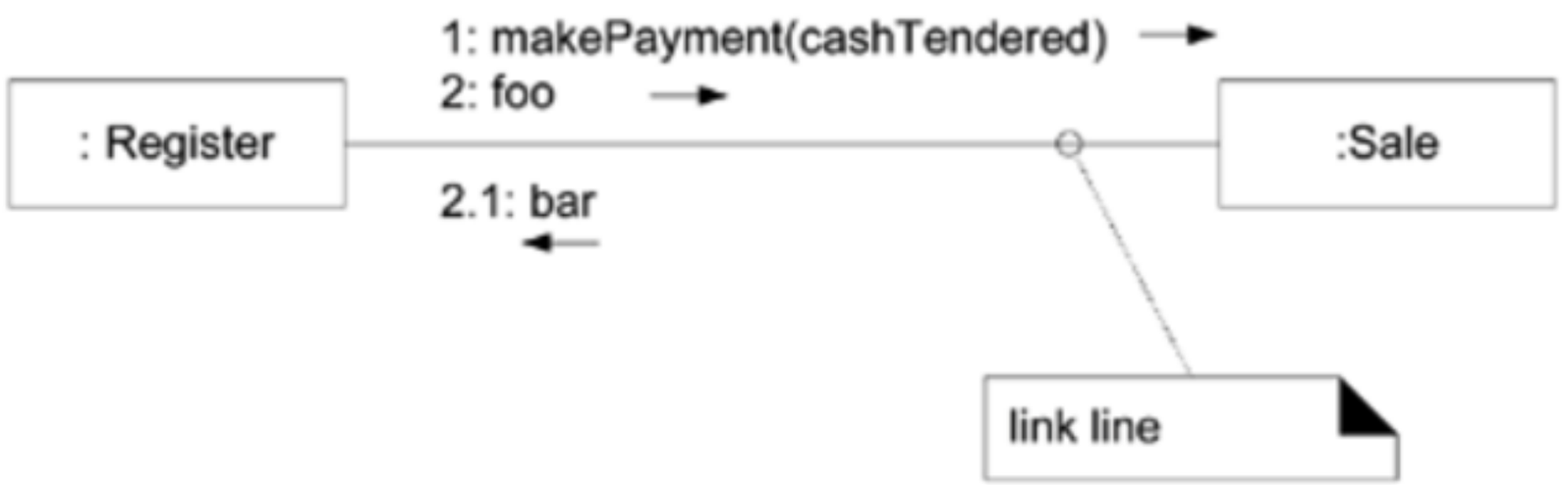


Notations - Static method

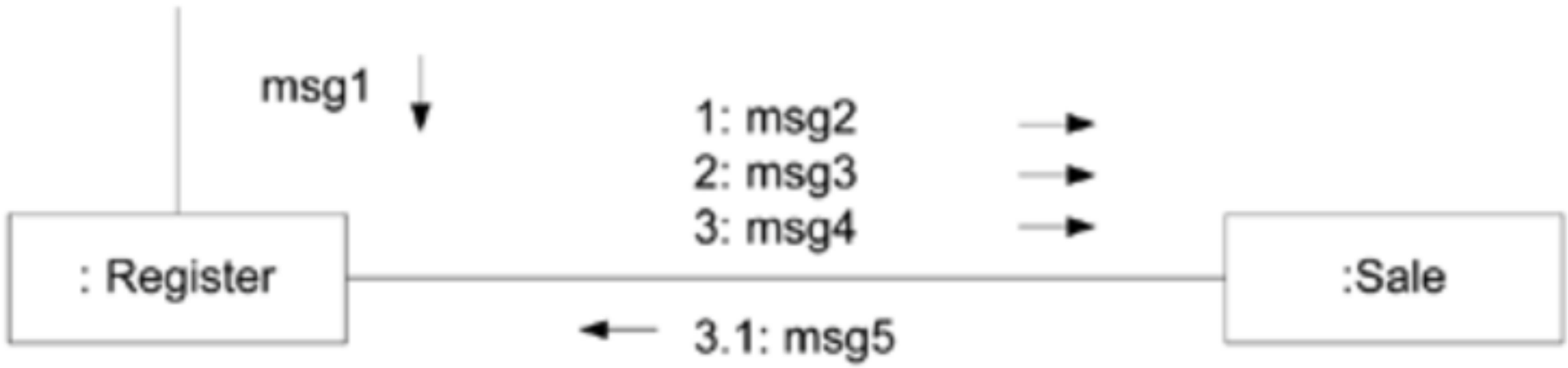


Communication diagram

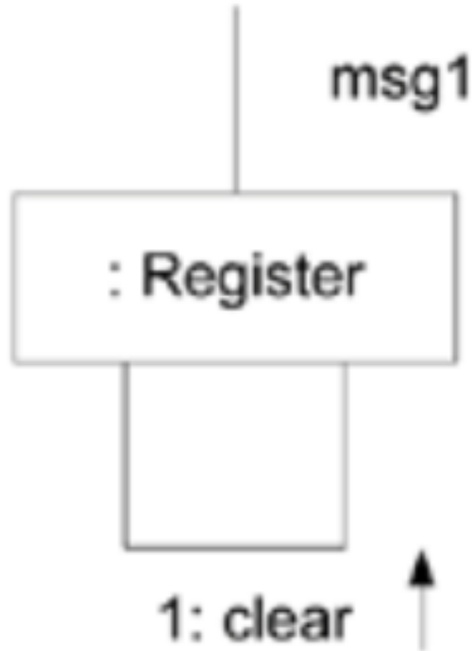
Notation - Links



Notation - Messages

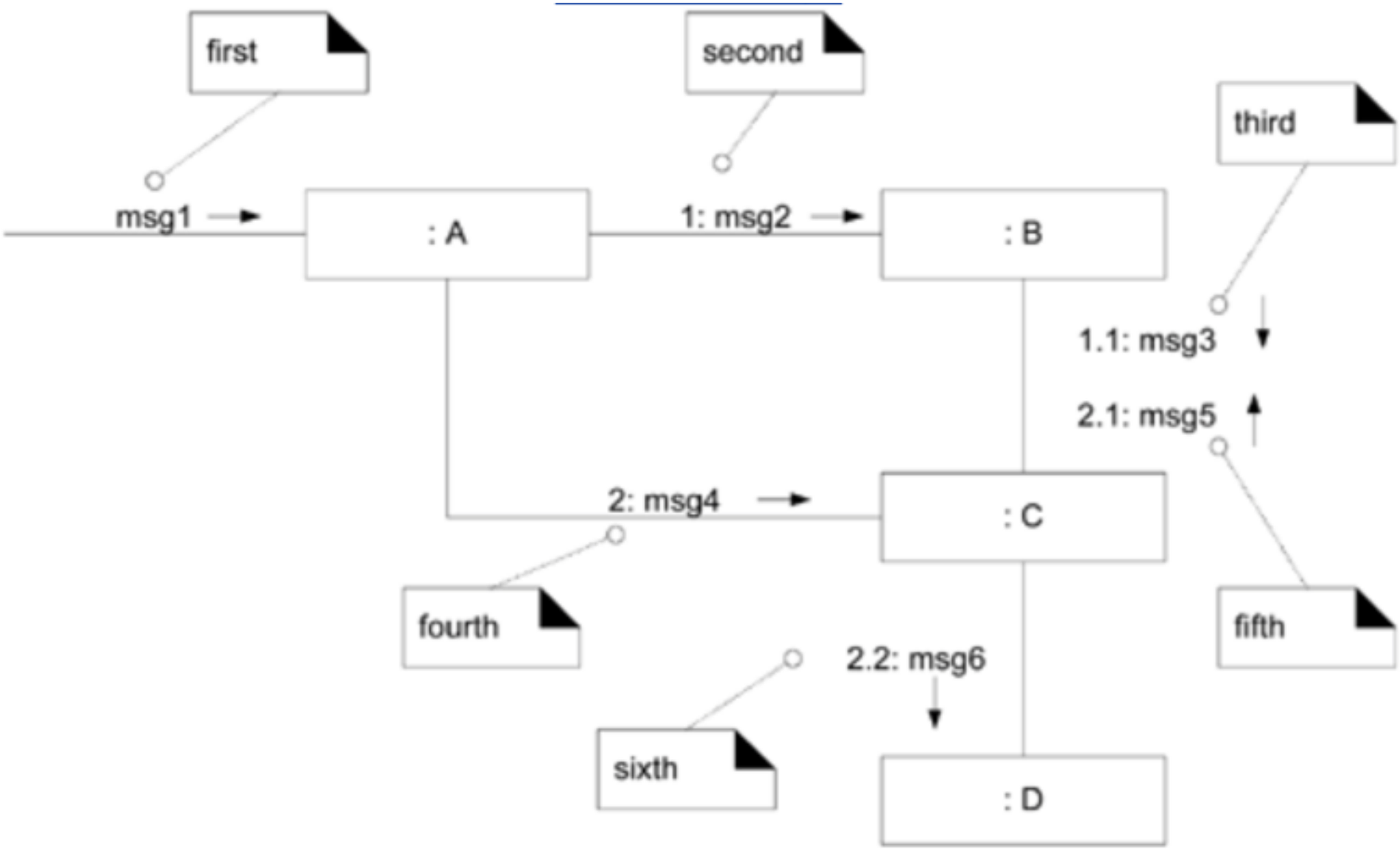


Notation - self/this

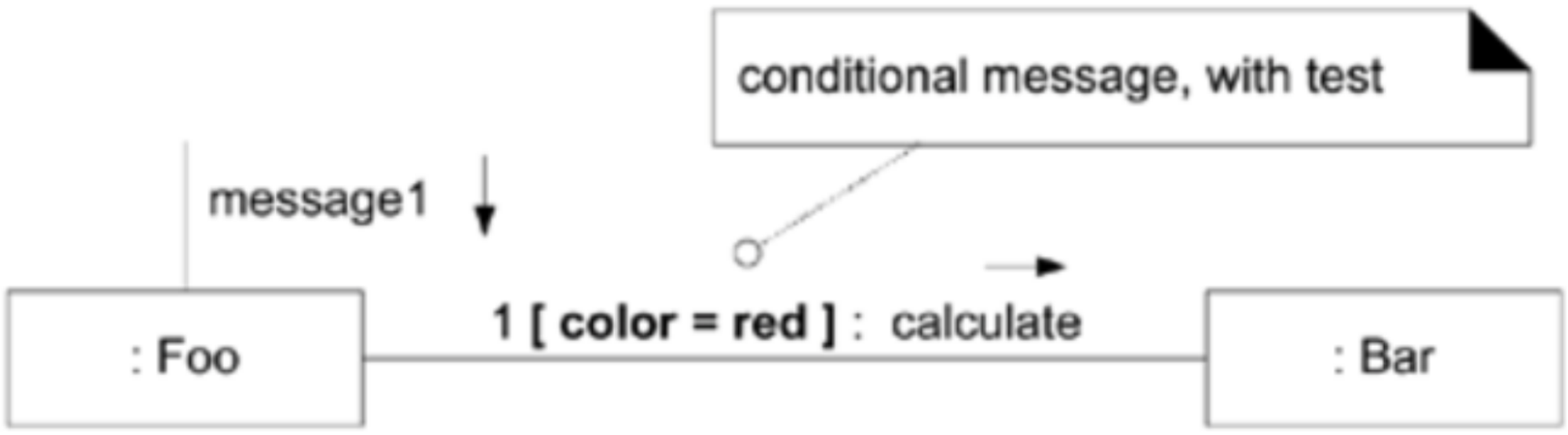


Notation - Number Sequence

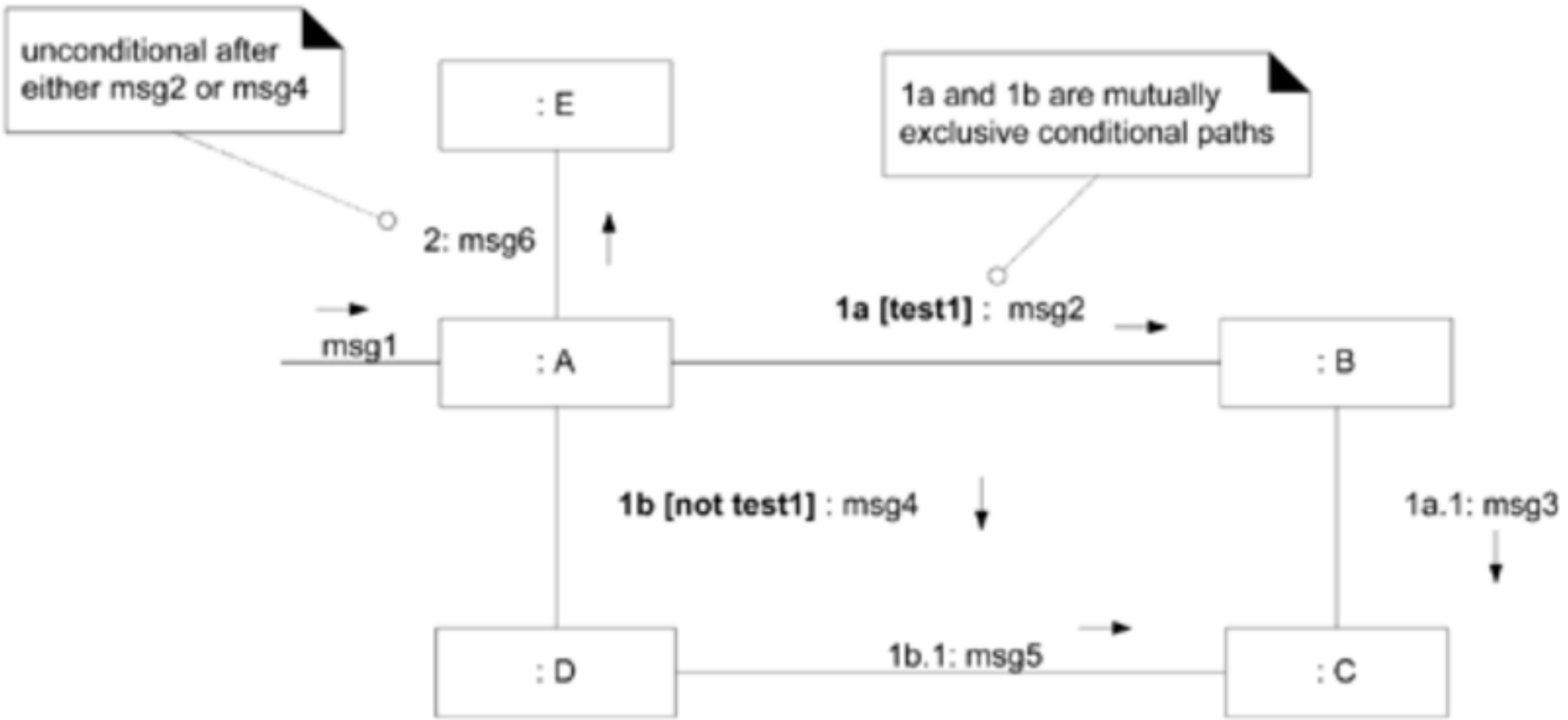
Notation - Number Sequence



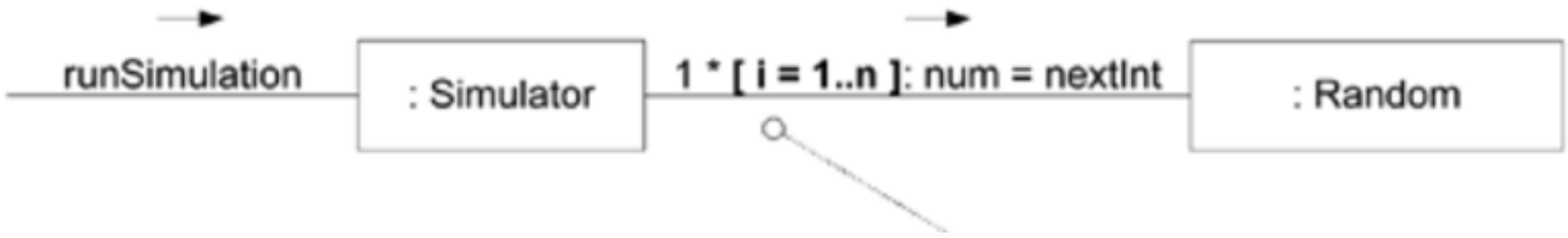
Notation - Conditional



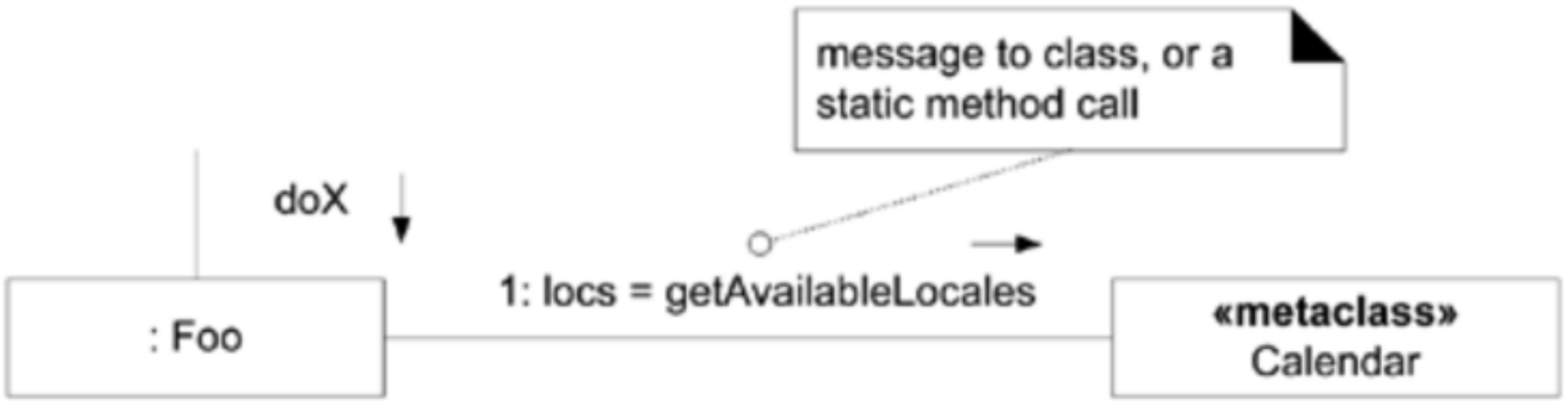
Notation - Conditional



Notation - Iteration



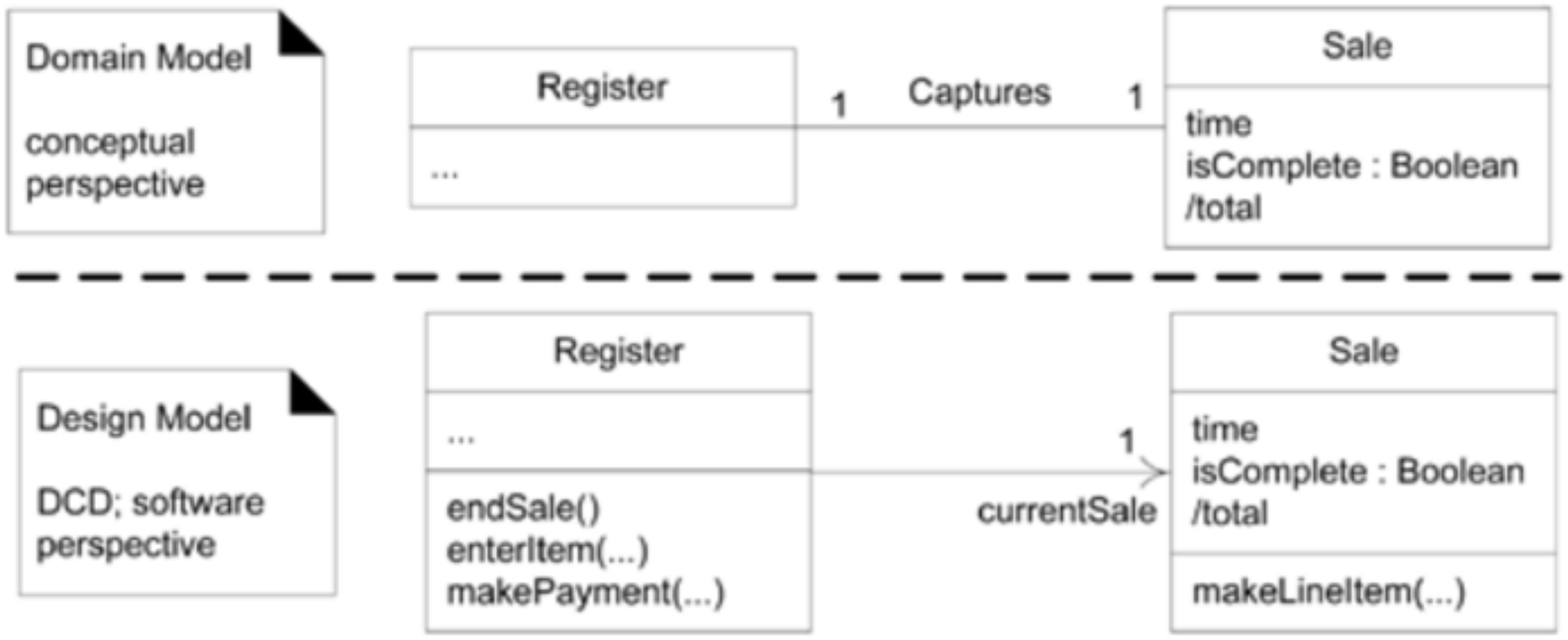
Notation - Static



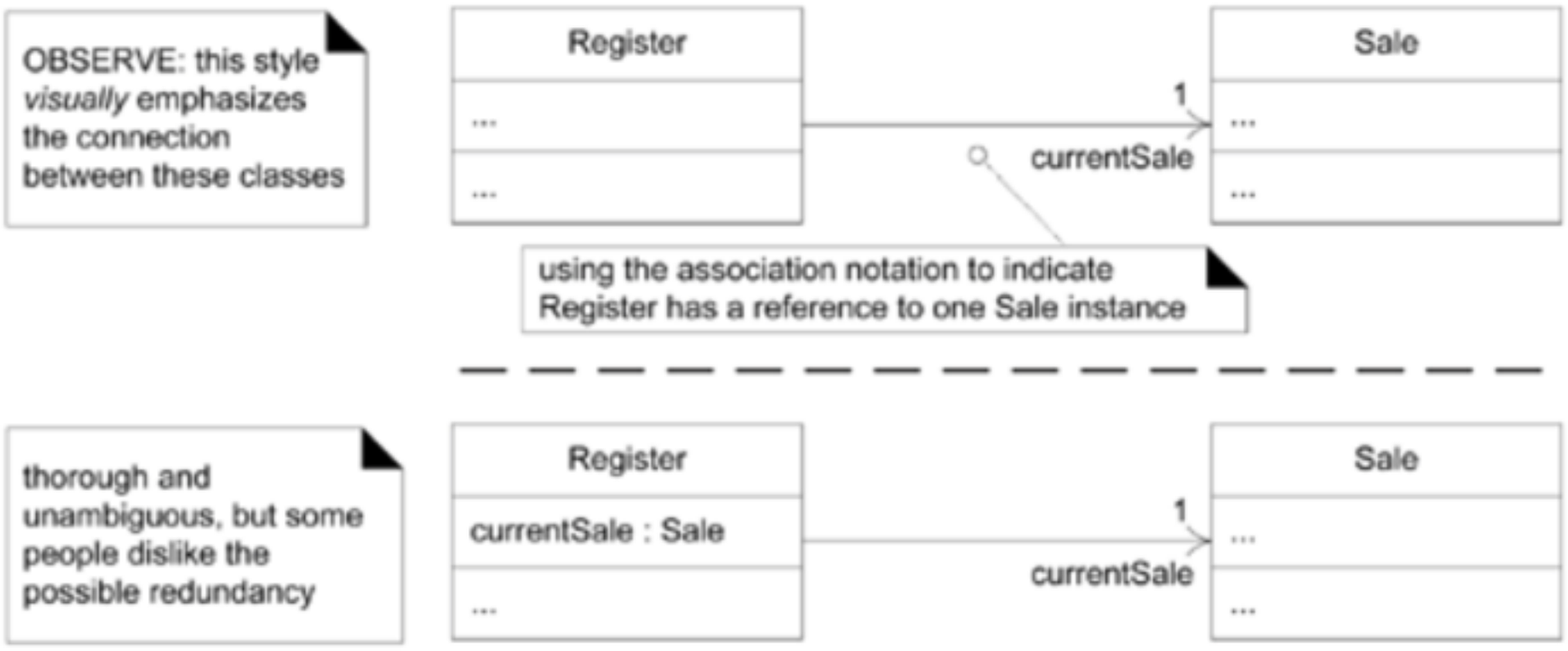
UML Class Diagram

UML Class Diagram

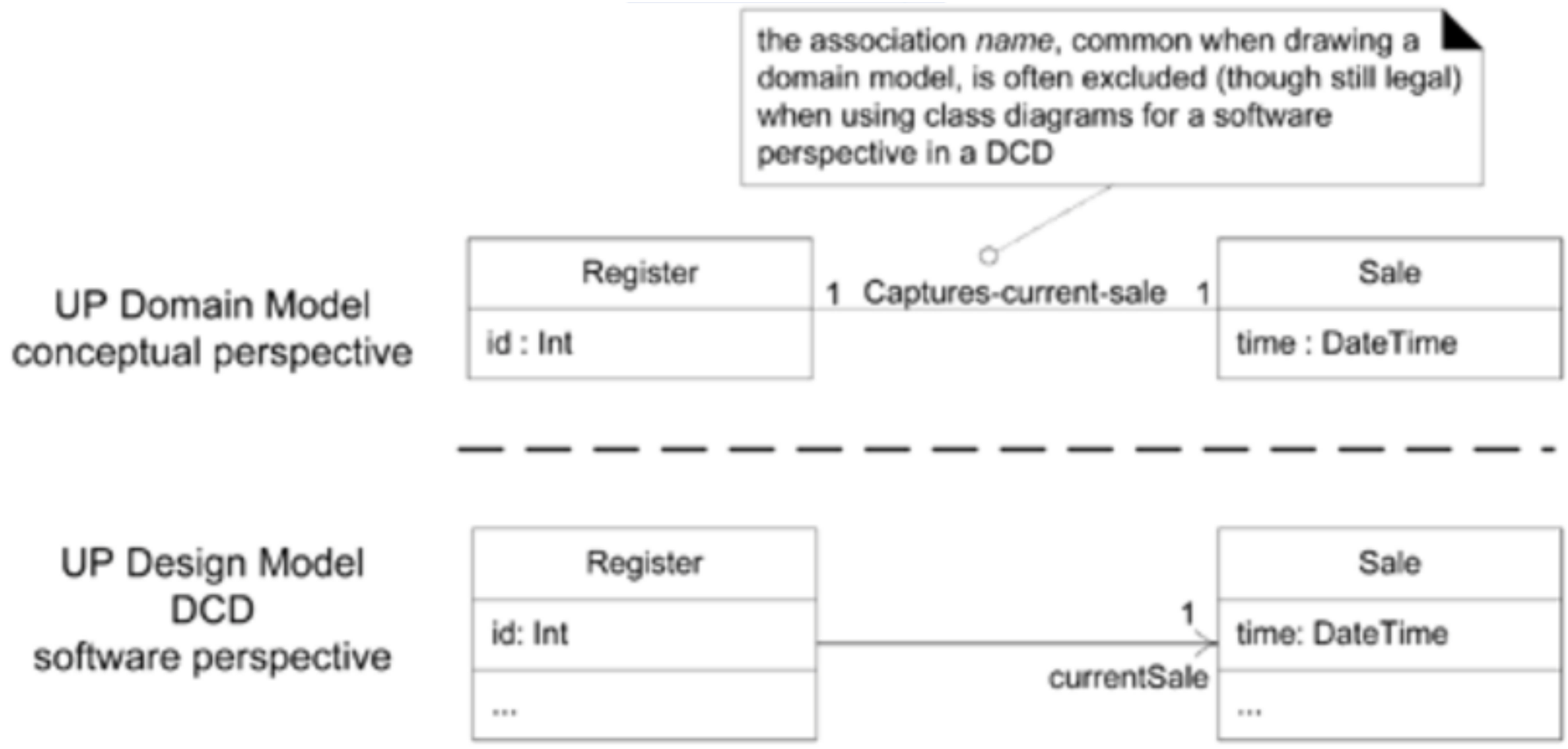
- Static object modeling



UML Attributes



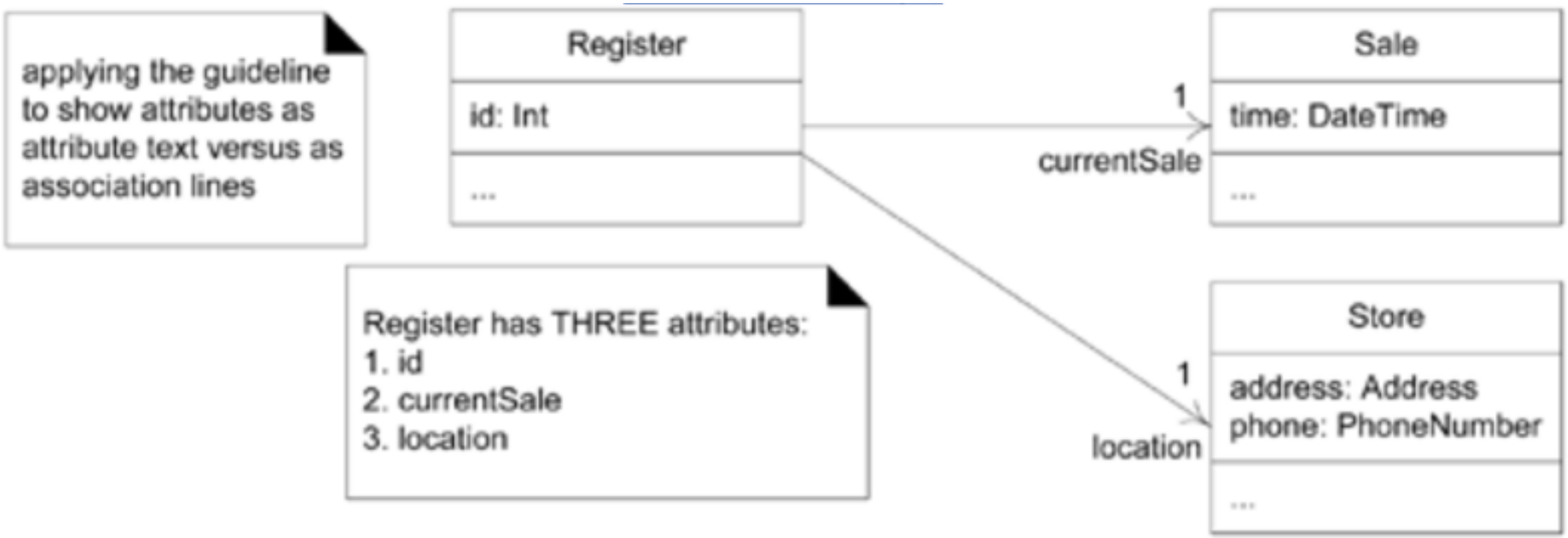
Excluding association



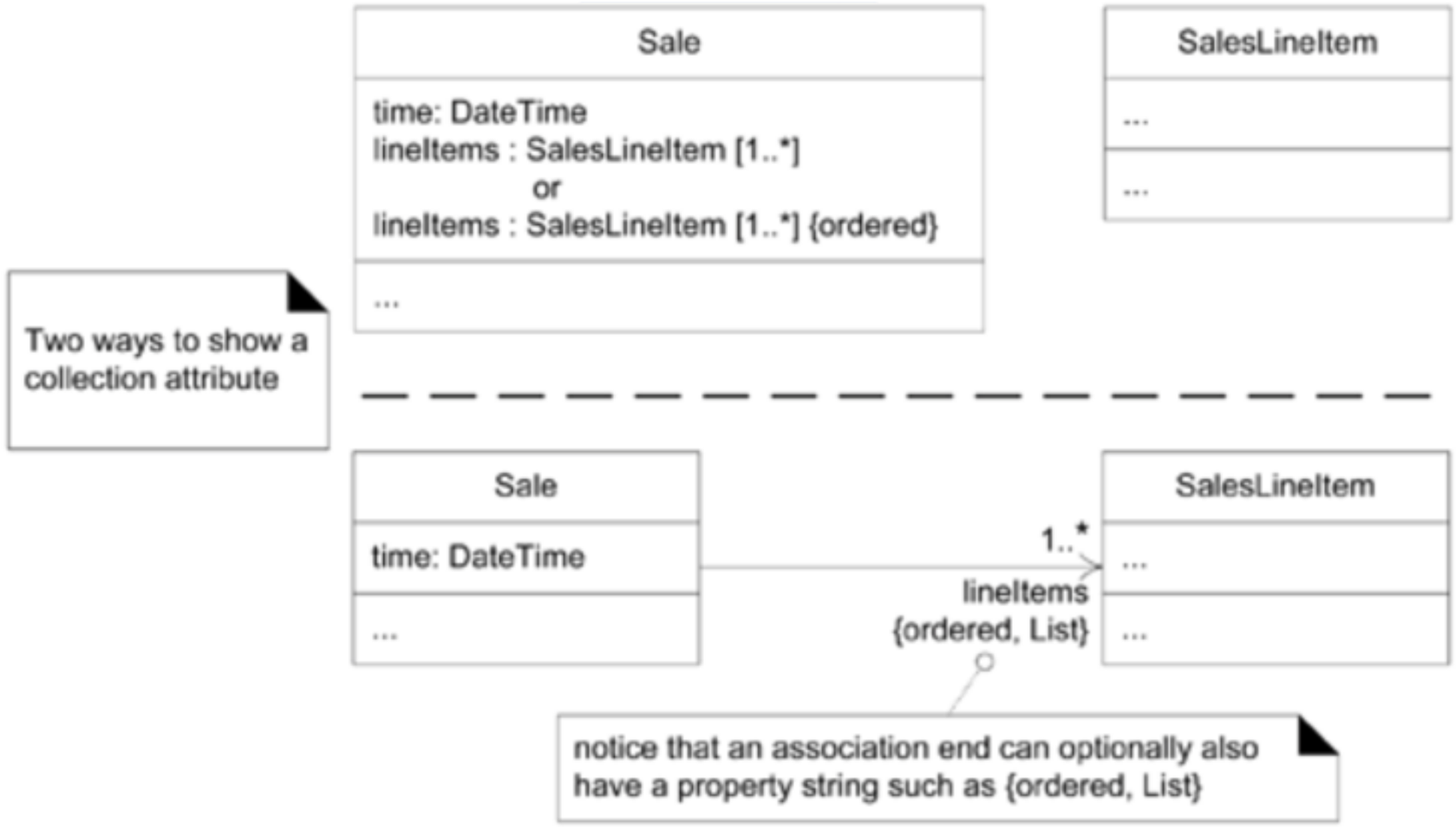
Attribute guideline

- Use text notation for data type object
- Association for complex objects
    - Give visual emphasis

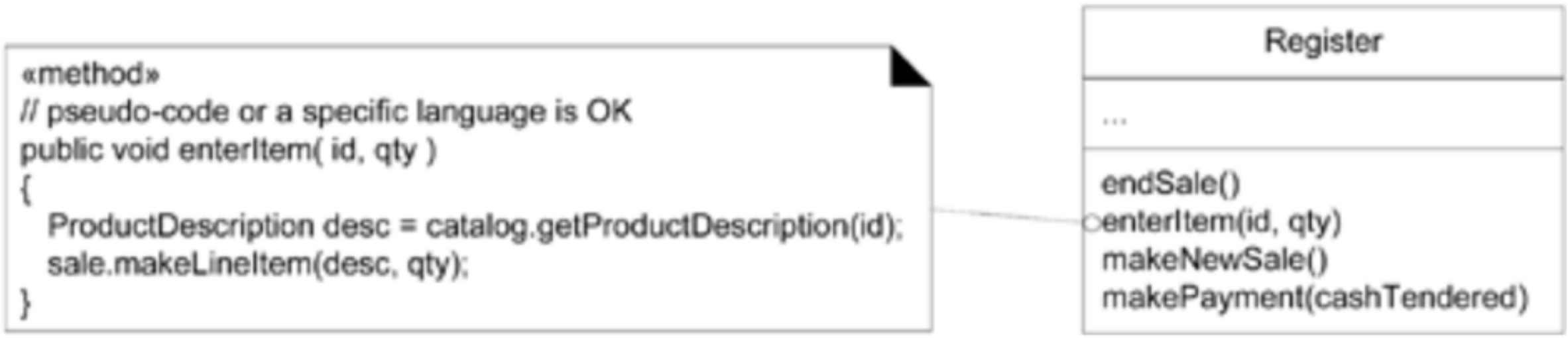
Attribute Guide



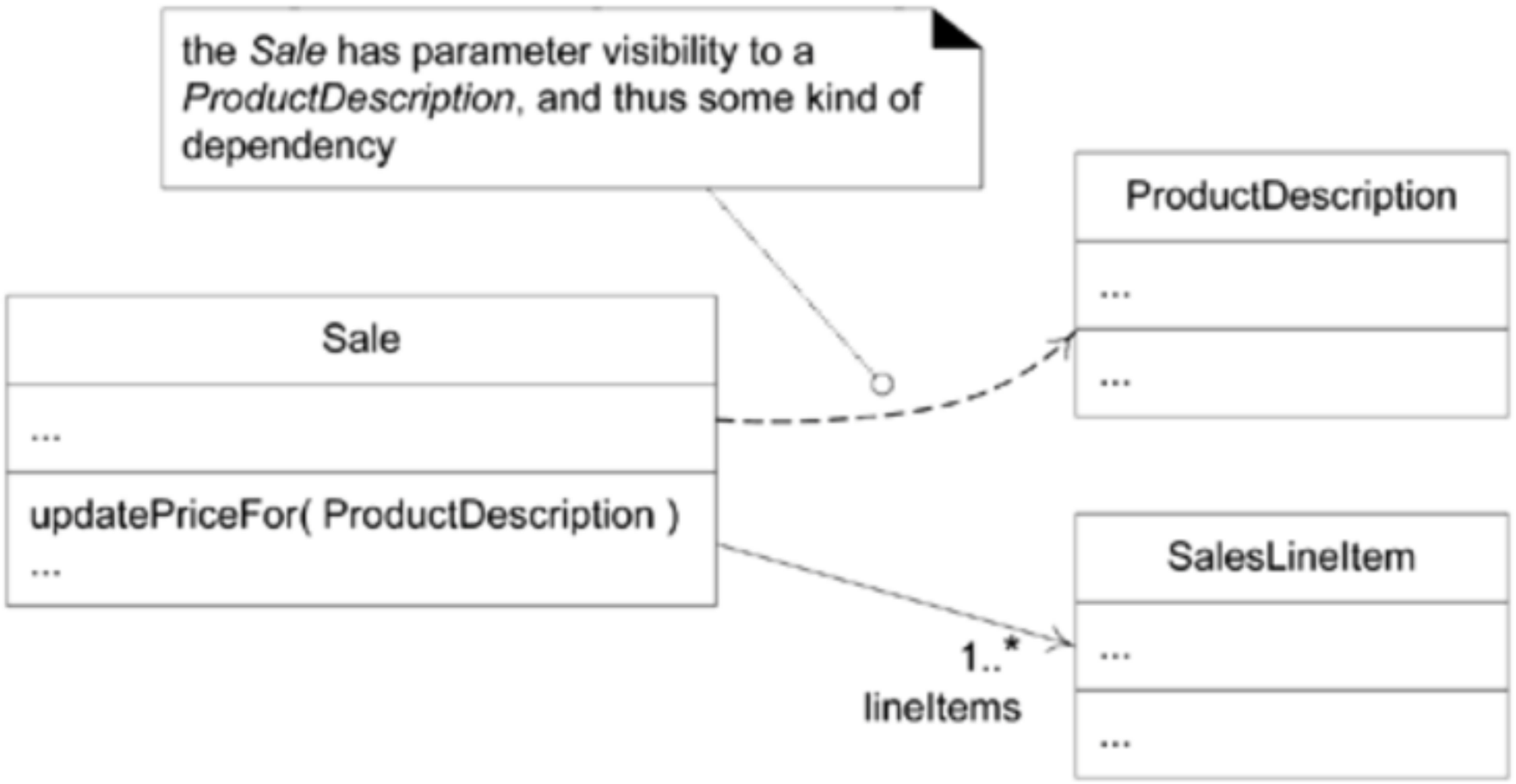
Collection Attributes



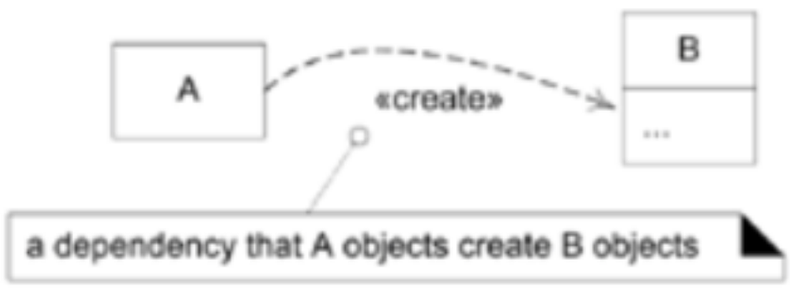
Methods


«method»
// pseudo-code or a specific language is OK
public void enterItem(id, qty)
{
ProductDescription desc = catalog.getProductDescription(id);
sale.makeLineItem(desc, qty);
}
Register
...
endSale()
enterItem(id, qty)
makeNewSale()
makePayment(cashTendered)

Dependency


the *Sale* has parameter visibility to a *ProductDescription*, and thus some kind of dependency
Sale
...
updatePriceFor(ProductDescription)
...
ProductDescription
...
...
SalesLineItem
...
...
1..*
lineItems

Dependency 2



User-defined section