UML Interaction diagrams

Sequence and Communication (Collaboration) Diagrams

COMP 3831

Larman: Chapter 15

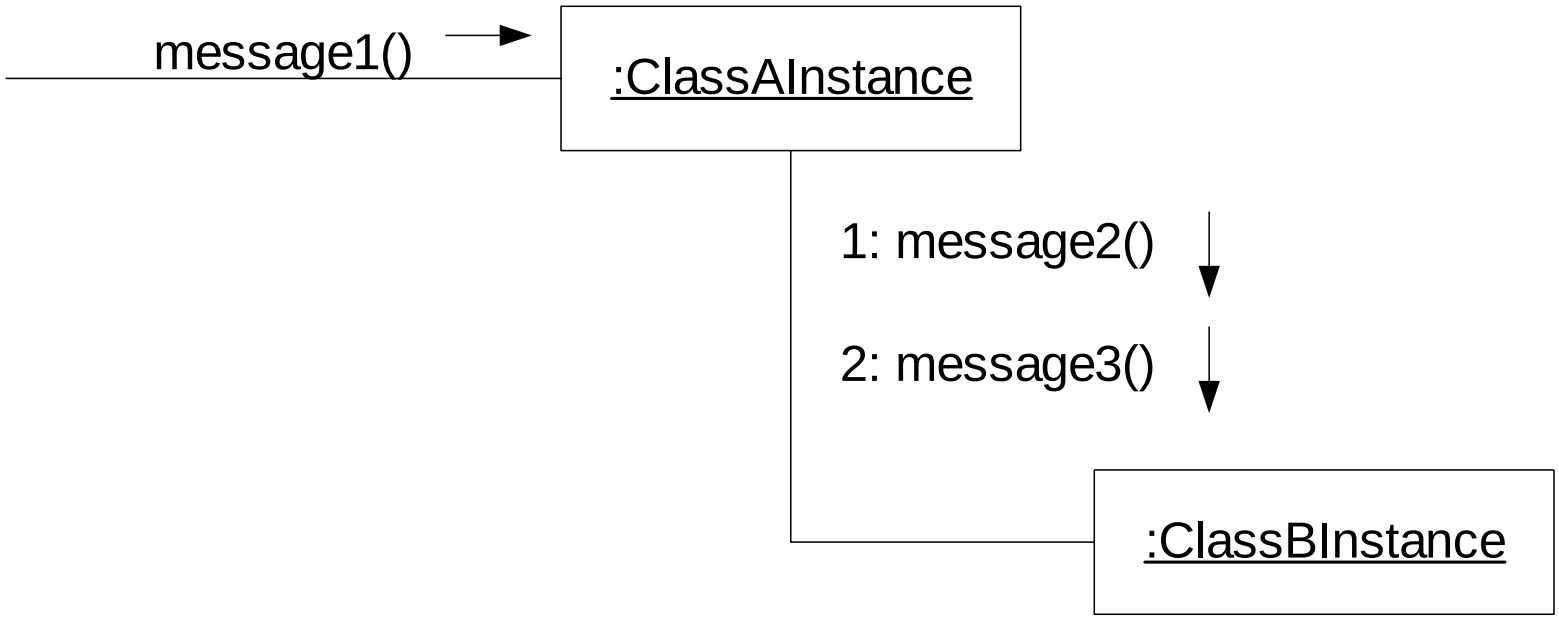
Interaction Diagrams

- Illustrate how objects interact via messages
- Interaction diagram is a generalization of two more specialized UML diagram types:
  1. Collaboration diagrams
  2. Sequence diagrams

Both express similar message interactions

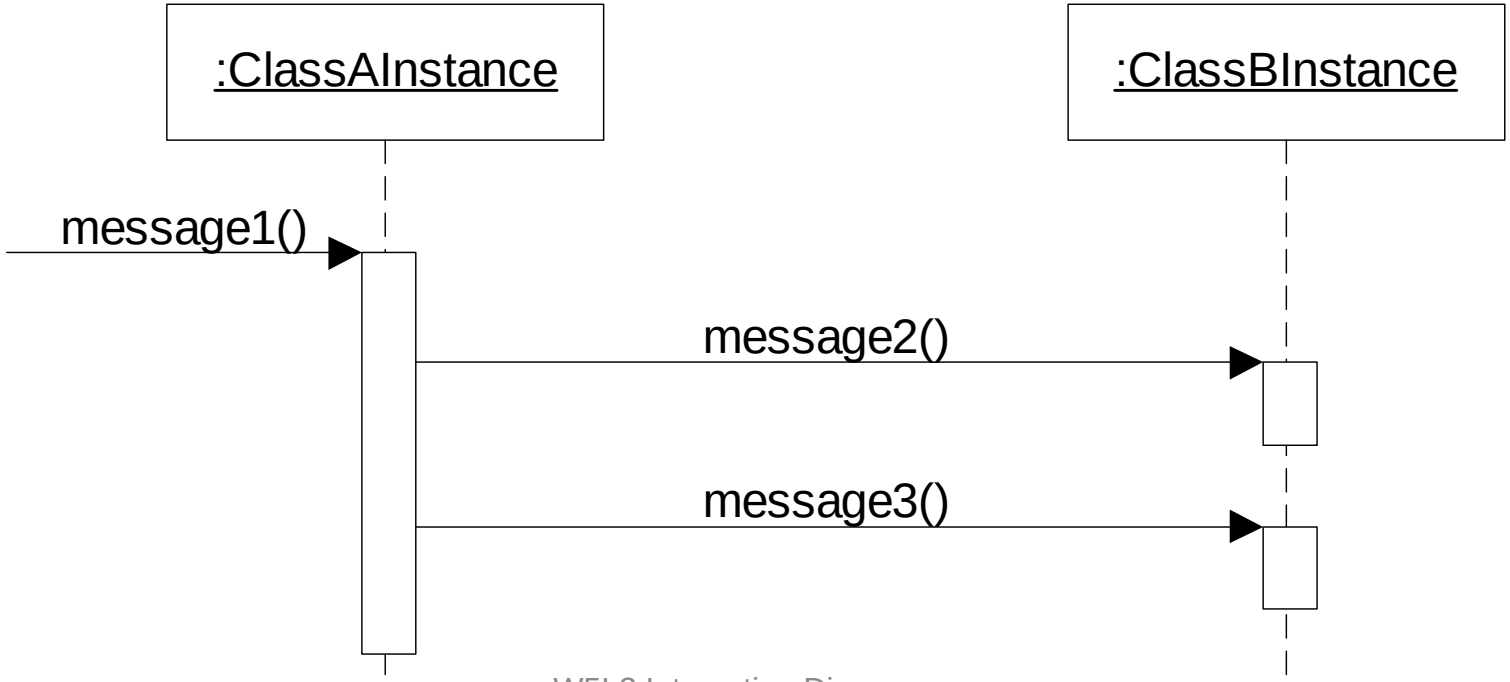
Collaboration (Communication) diagrams

- Illustrate object interactions in a network format, in which objects can be placed anywhere on the diagram.



Sequence Diagrams

- Illustrate interactions in a kind of fence format, in which each new object is added to the right.
- Unlike collaboration diagrams, sequence diagrams do not show links.

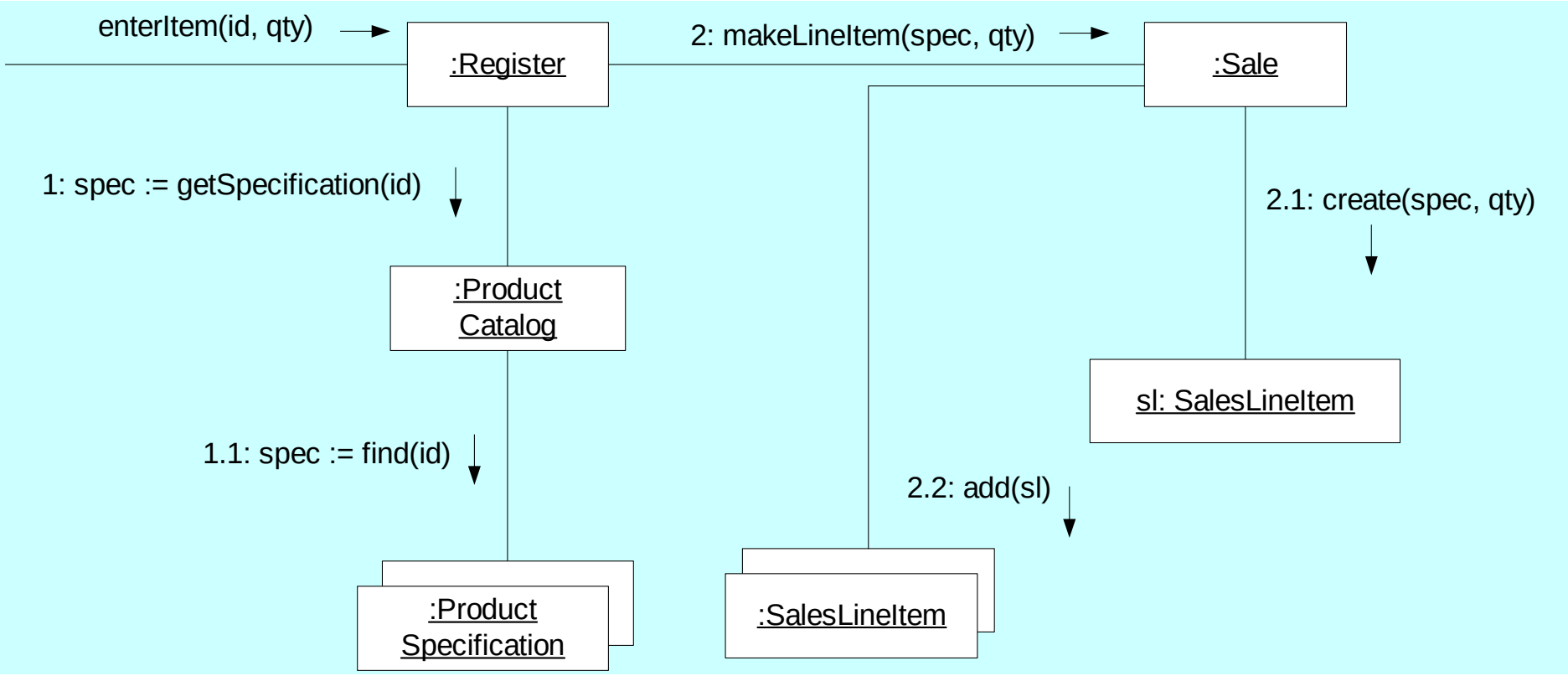


Sequence vs. Collaboration diagrams

Type	Strengths	Weaknesses
Sequence	clearly shows sequence or time ordering of messages simple notation	forced to extend to the right when adding new objects – consumes horizontal space
Collaboration (Communication)	space economical – flexibility to add new objects in two dimensions better to illustrate complex branching, iteration, and concurrent behavior	difficult to see sequence of messages more complex notation

Collaboration diagrams

- represent interactions between objects as a series of sequenced messages.
- describe both the static structure and the dynamic behaviour of a system.
- Unlike sequence diagrams, collaboration diagrams do not have an explicit way to denote time and instead number messages in order of execution.

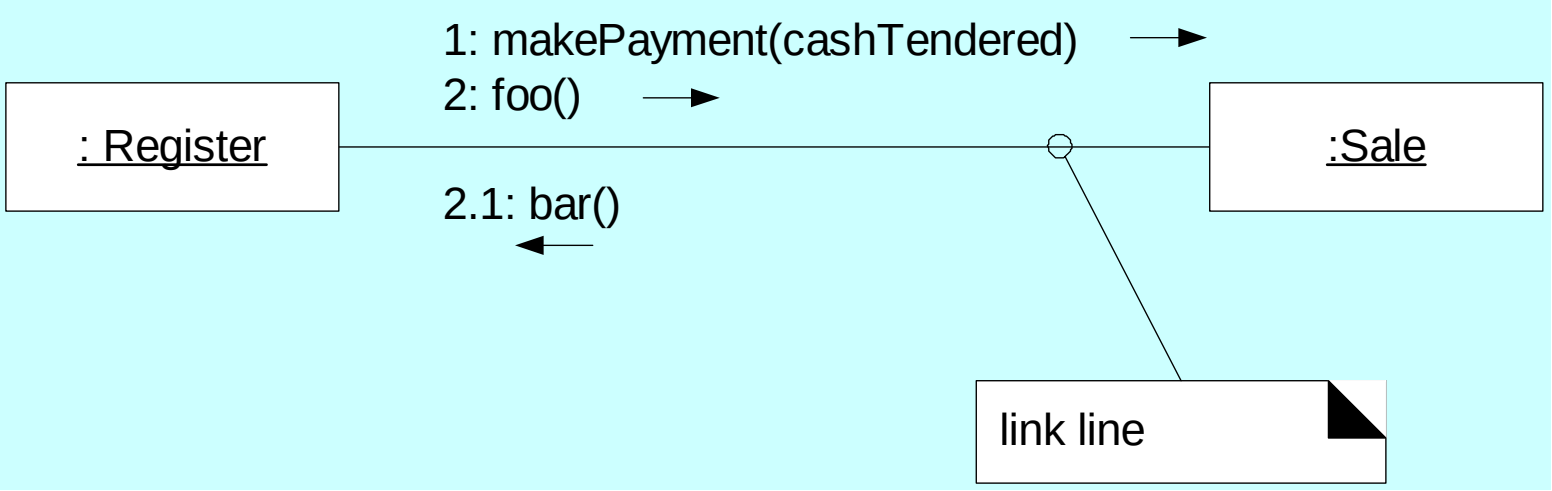


Symbols & Notations

Collaboration diagrams

Link:

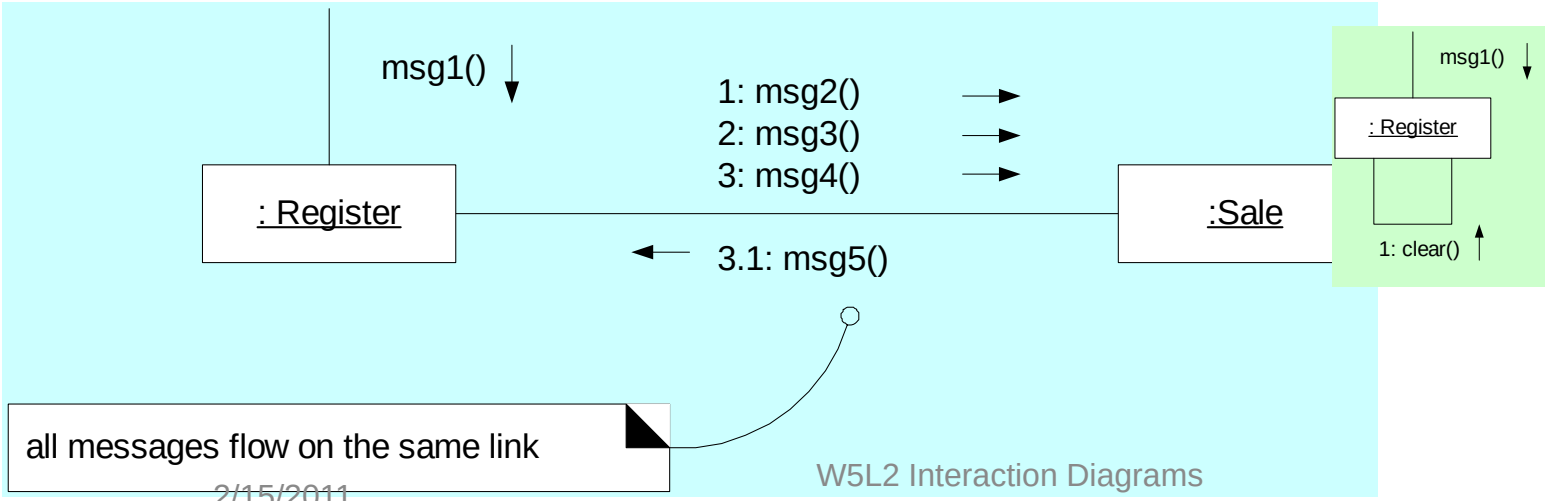
- A connection path between two objects
- Indicates some form of navigation and visibility between the objects is possible
- Multiple messages can flow along the same single link



Symbols & Notations (continued …)

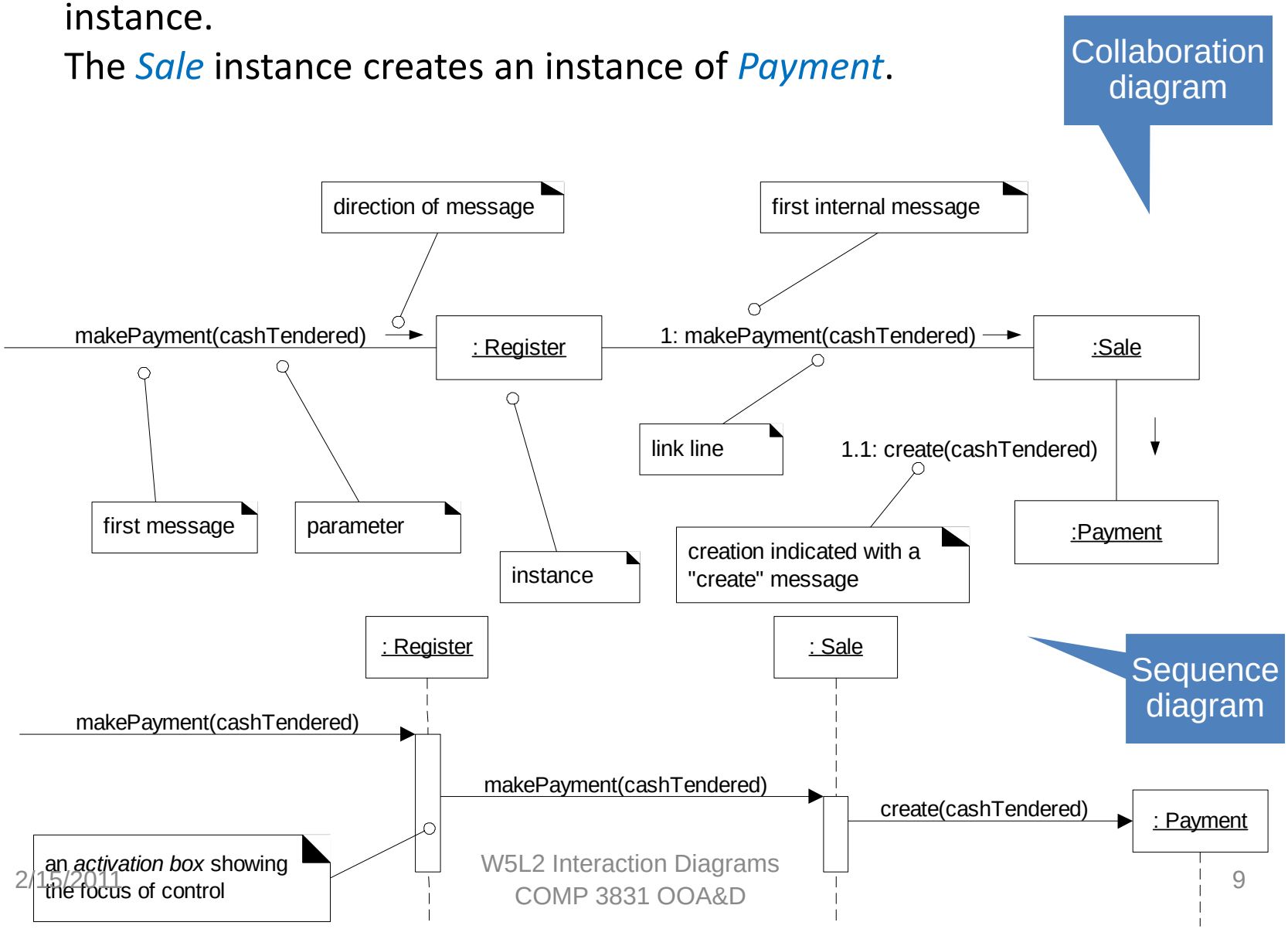
Messages (continued)

- Represented with a message expression and small arrow indicating direction
- All messages flow on same link
- Message can be sent from an object to itself



Example interaction diagram: makePayment

1. The message *makePayment* is sent to an instance of *Register*.
2. The *Register* instance sends the *makePayment* message to a *Sale* instance.
3. The *Sale* instance creates an instance of *Payment*.



Suggestions

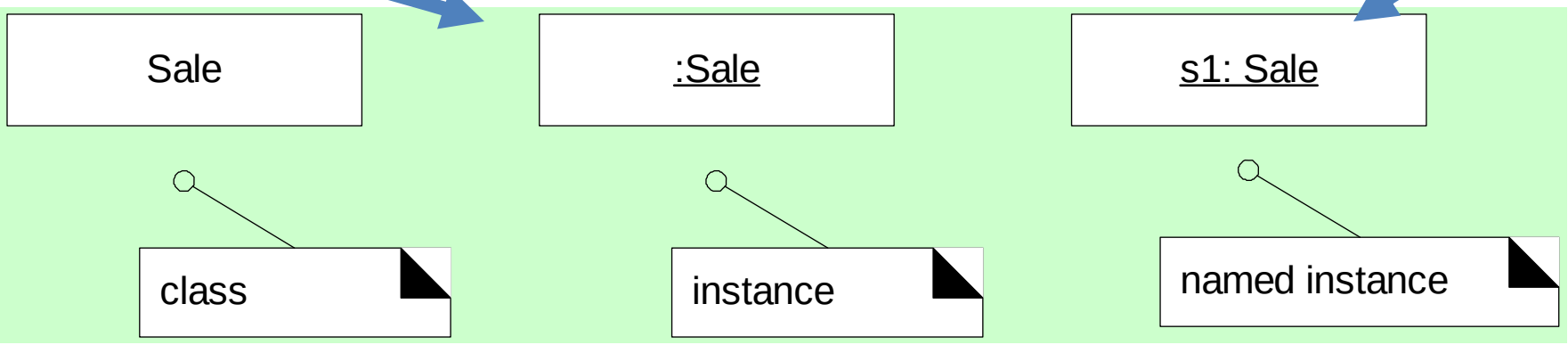
Collaboration diagrams

- During the elaboration phase, about a day at the start of an iteration, should be spent on creation of interaction and class diagrams.
  - Even though diagrams may be imperfect
- Diagrams (interaction and/or class diagrams) are necessary before proceeding to programming.
- Create interaction diagrams in pairs. Design will turn out much better and partners will learn quickly from each other.

Illustrating Classes & Instances

Collaboration diagrams

- UML approach to illustrate instances vs. classifiers
  - Instance has designator string underlined. Note that a ":" precedes the class name.
  - A name can be used to uniquely identify the instance. Again, note that a ":" precedes the class name.



UML message expression syntax

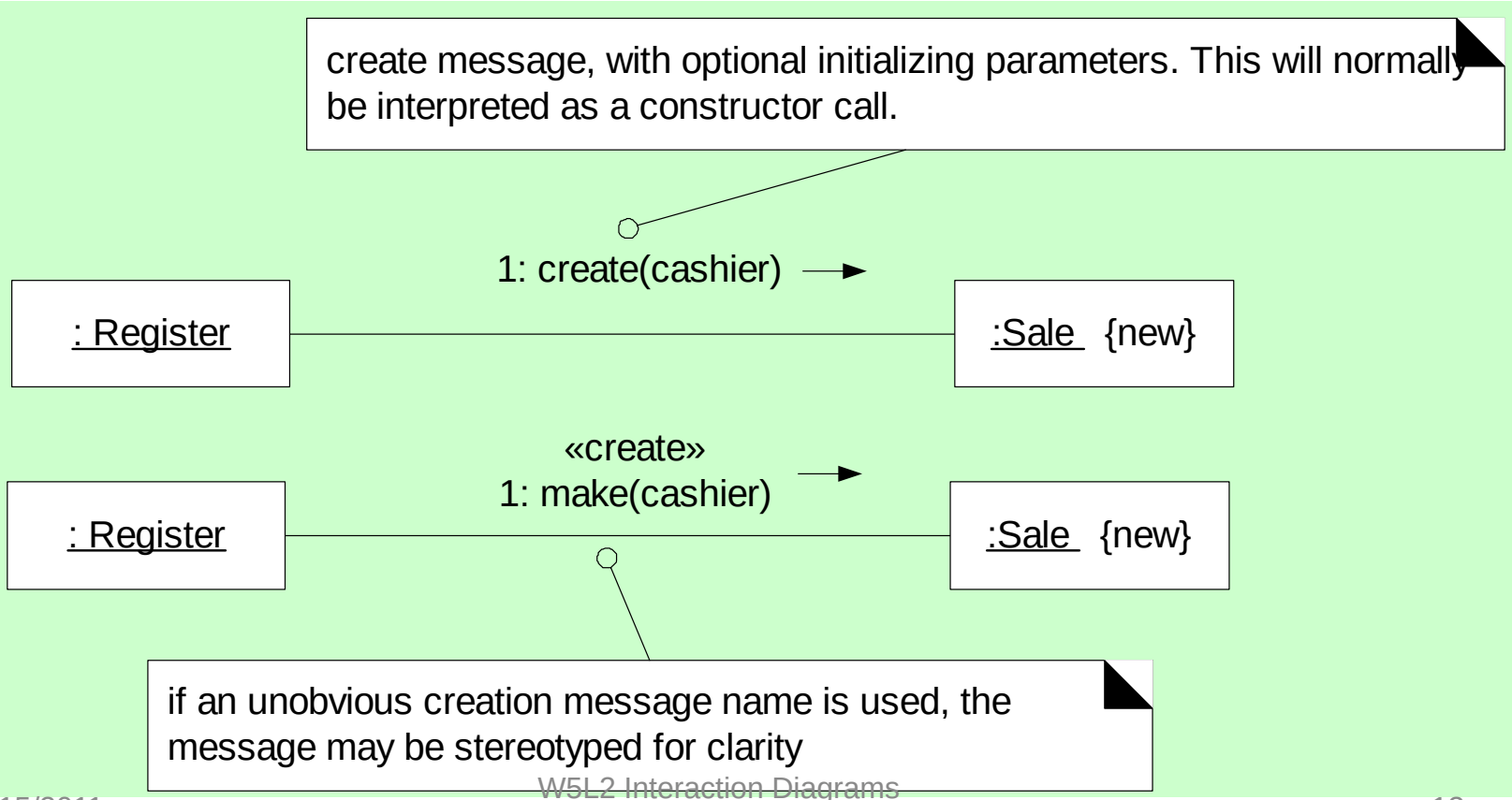
return := message(parameter : parameterType) : returnType

```
Examples:

spec := getProductSpect( id )
spec := getProductSpect( id:ItemID )
spec := getProductSpect( id:ItemID ) : ProductSpecification
```

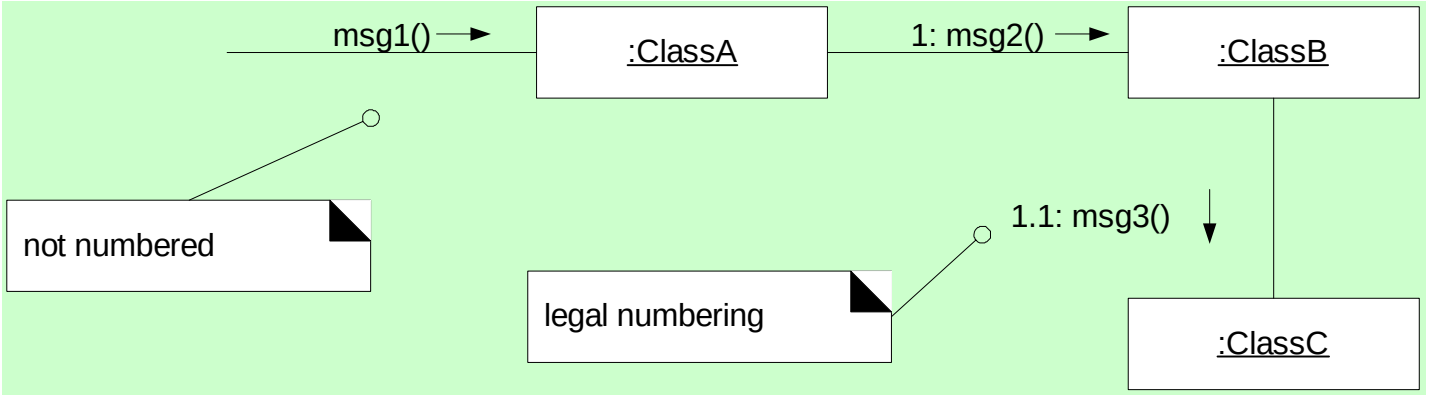
Creation of instances

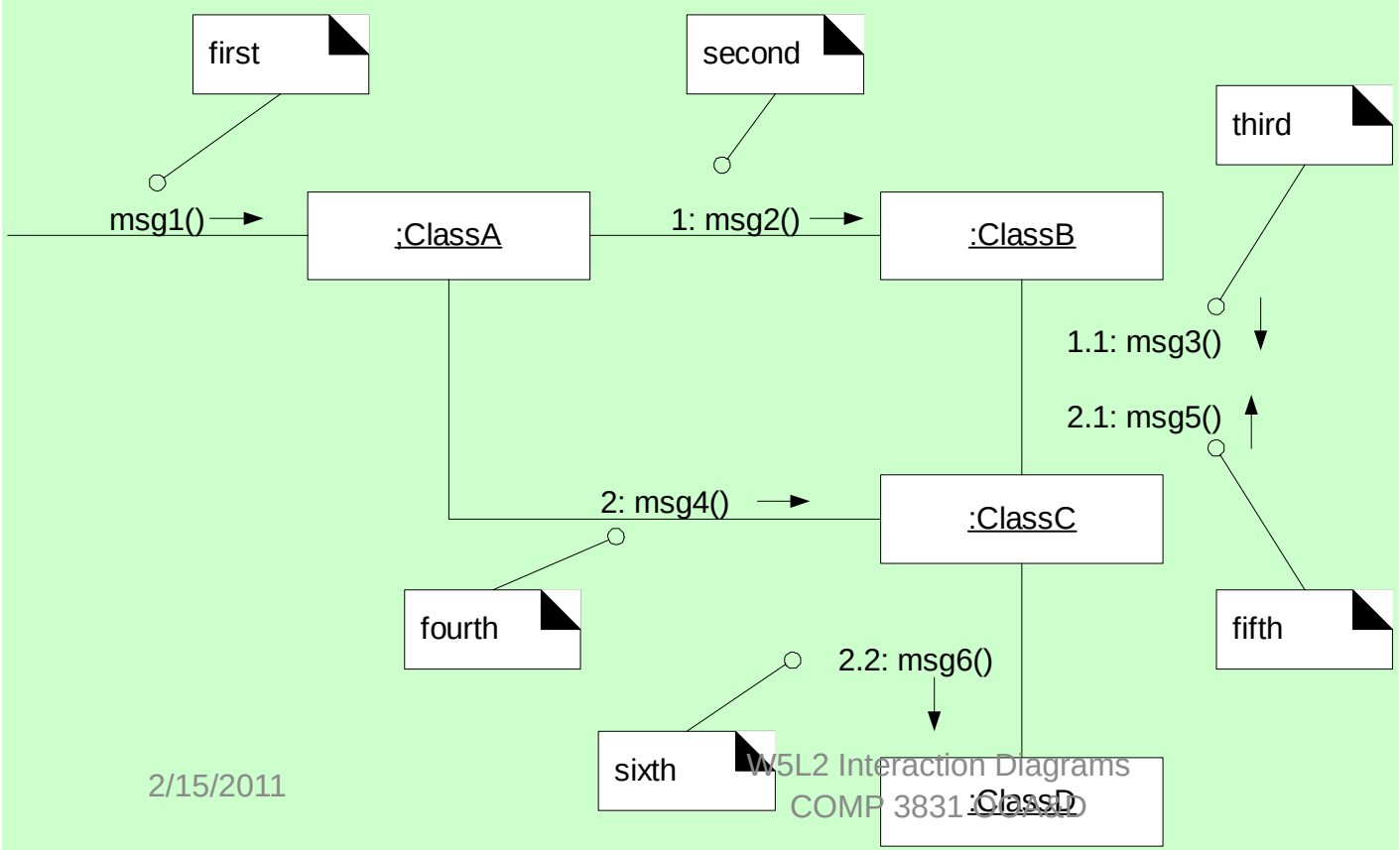
- There is a convention in UML to use a message named *create* to create an instance
  - Create message may include parameters
- If another message is used, it may be annotated with *<<create>>* stereotype
- *{new}* may be optionally added to the instance box to highlight creation



Message numbering sequence

Collaboration diagrams

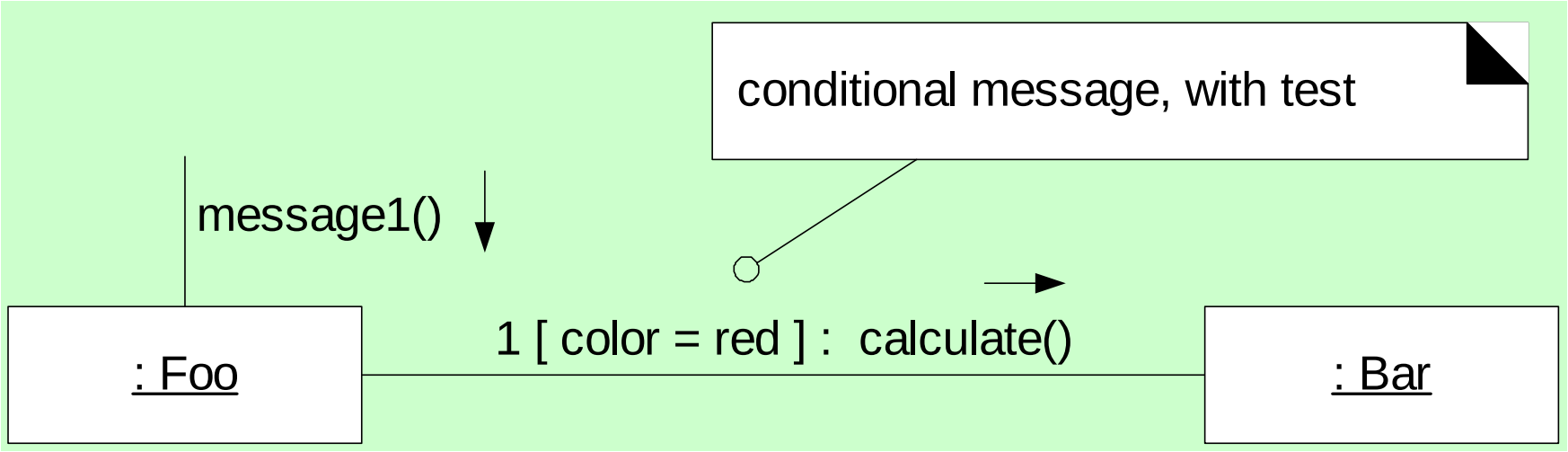




- First message is not numbered
- Sequence numbering can become nested using legal numbering (the Dewey decimal system). For example, nested messages under the first message are labeled 1.1, 1.2, 1.3, and so on.

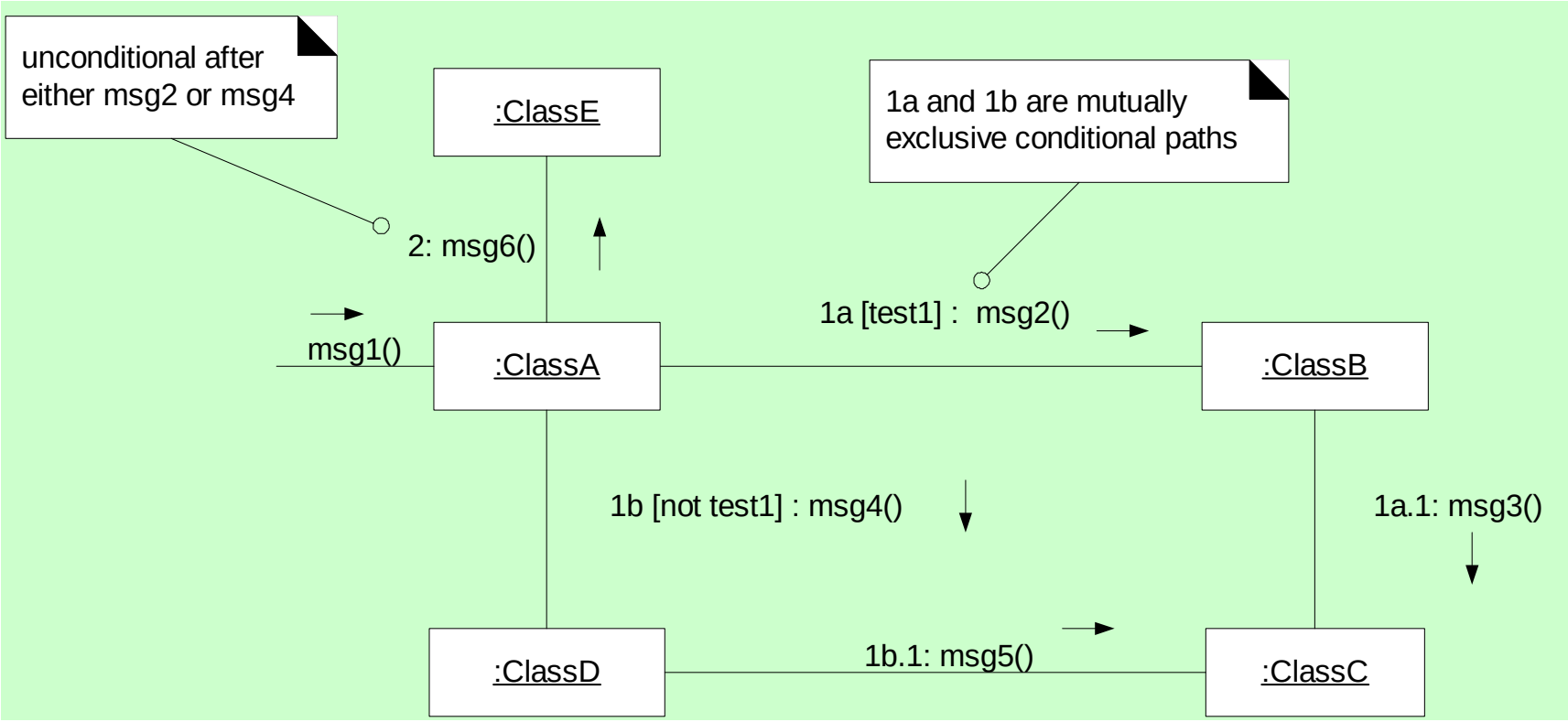
Conditional messages

- The condition for a message is usually placed in square brackets immediately following the sequence number.
- The message is only sent if the conditional clause evaluates to true.



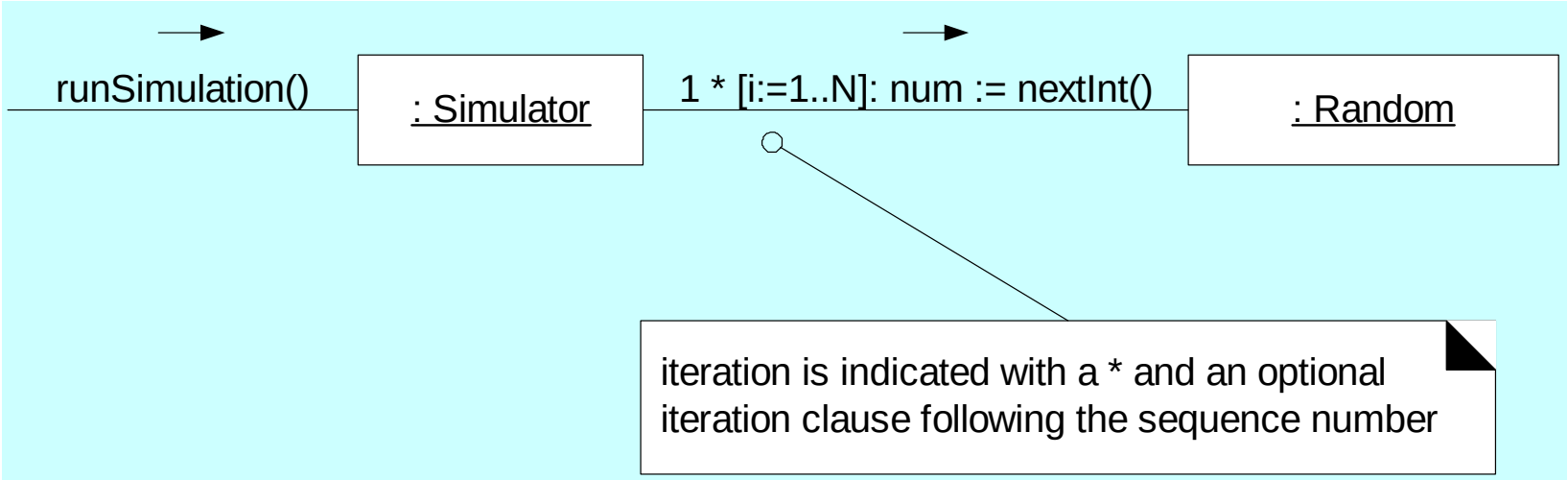
Mutually exclusive conditional paths

- Sequence expression modified with a conditional path expression.
- First letter is *a* by convention
  - In drawing below, either *1a* or *1b* could execute after *msg1*.
  - Subsequent nested messages consistently prepended with their outer message sequence (*1b.1* is nested within *1b*).



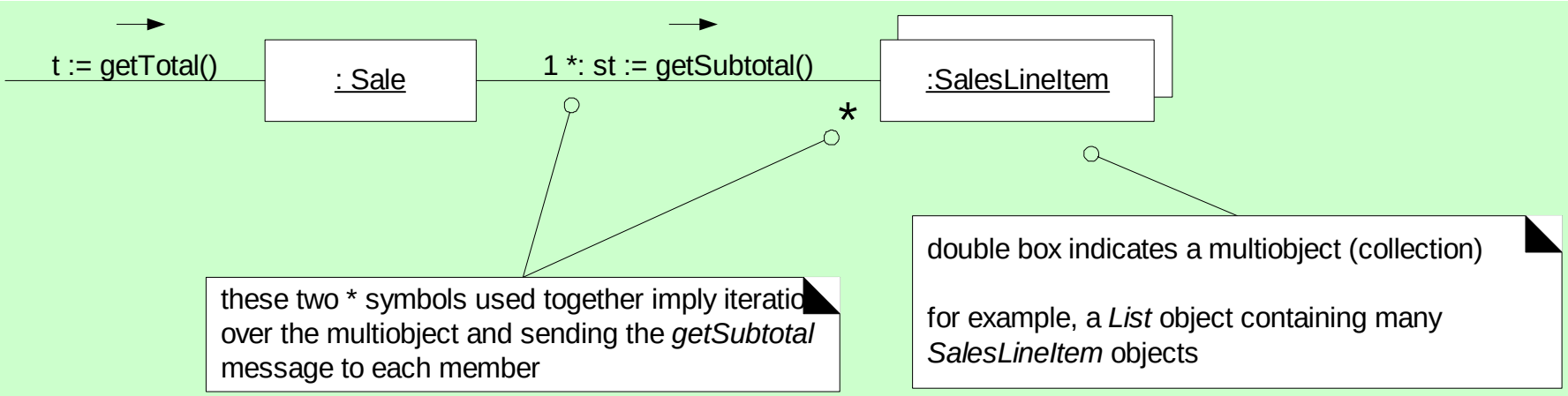
Iteration (looping)

- A simple '*' is used with optional iteration clause following the sequence number



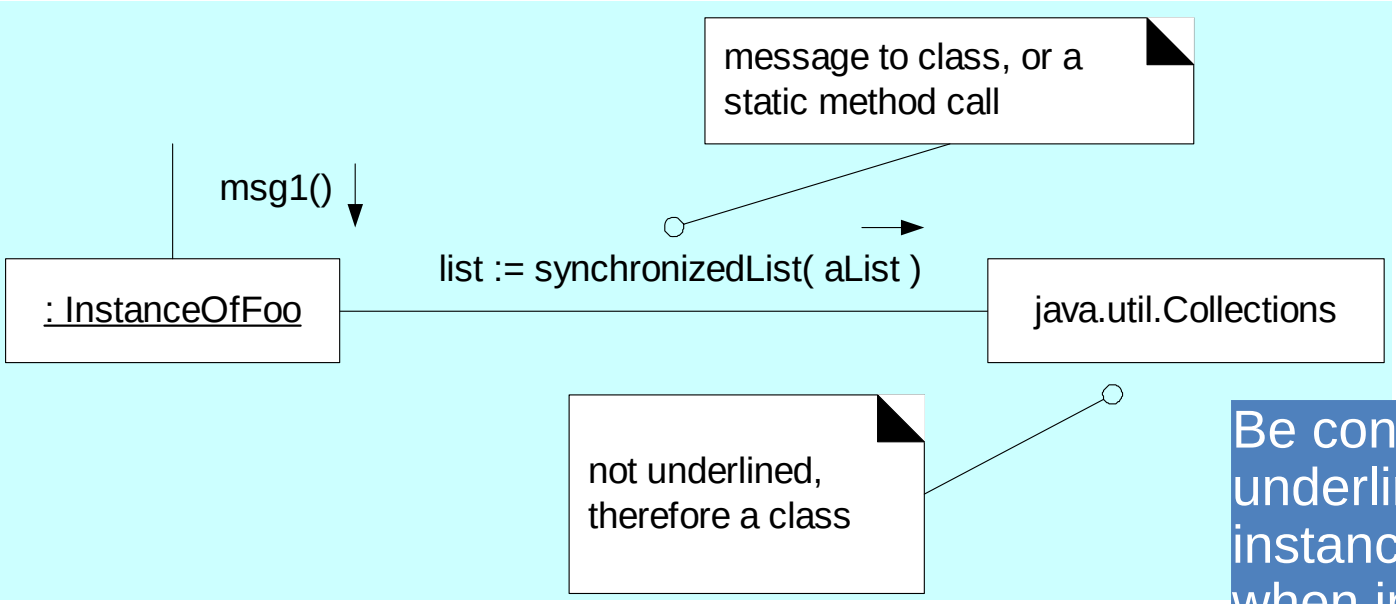
Iteration (looping) over a collection

- UML term *multiobject* is used to denote collection
- A '*' multiplicity marker at end of link is used to indicate that message is being sent to each element of collection



Messages to a Class object

- A Class Object contains static methods.
- This is shown with name not underlined, indicating that message is being sent to a class rather than an instance

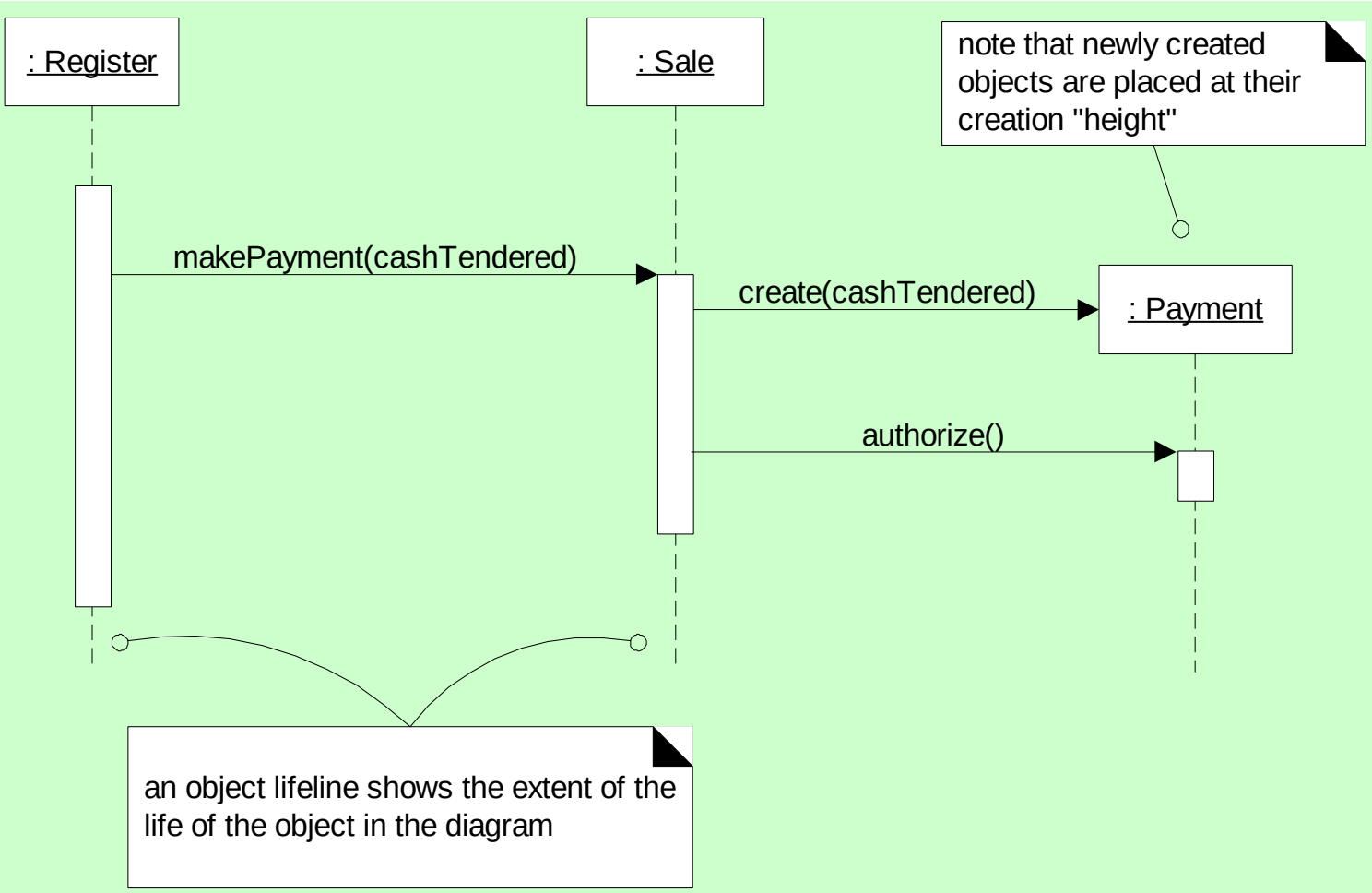


Be consistent in underlining instance names when instance is intended.

More on Sequence Diagram notation

Sequence diagrams

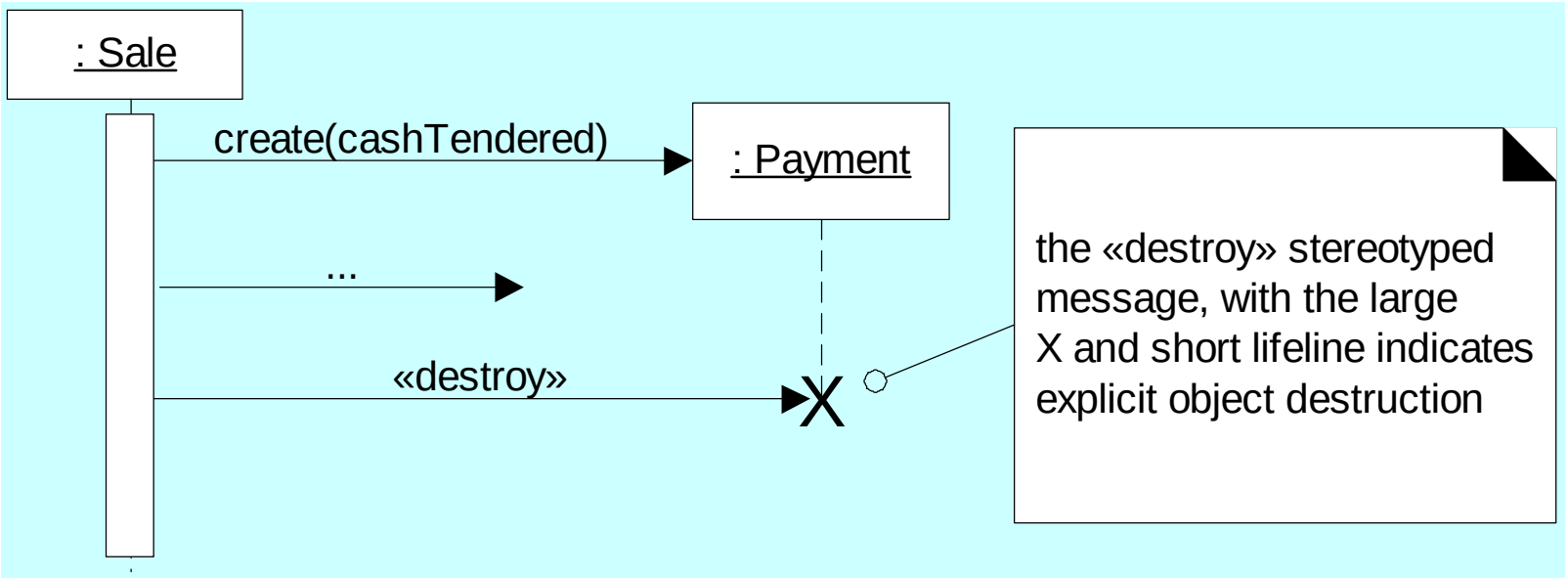
Creation of instances:



Object lifelines and object destruction

Sequence diagrams

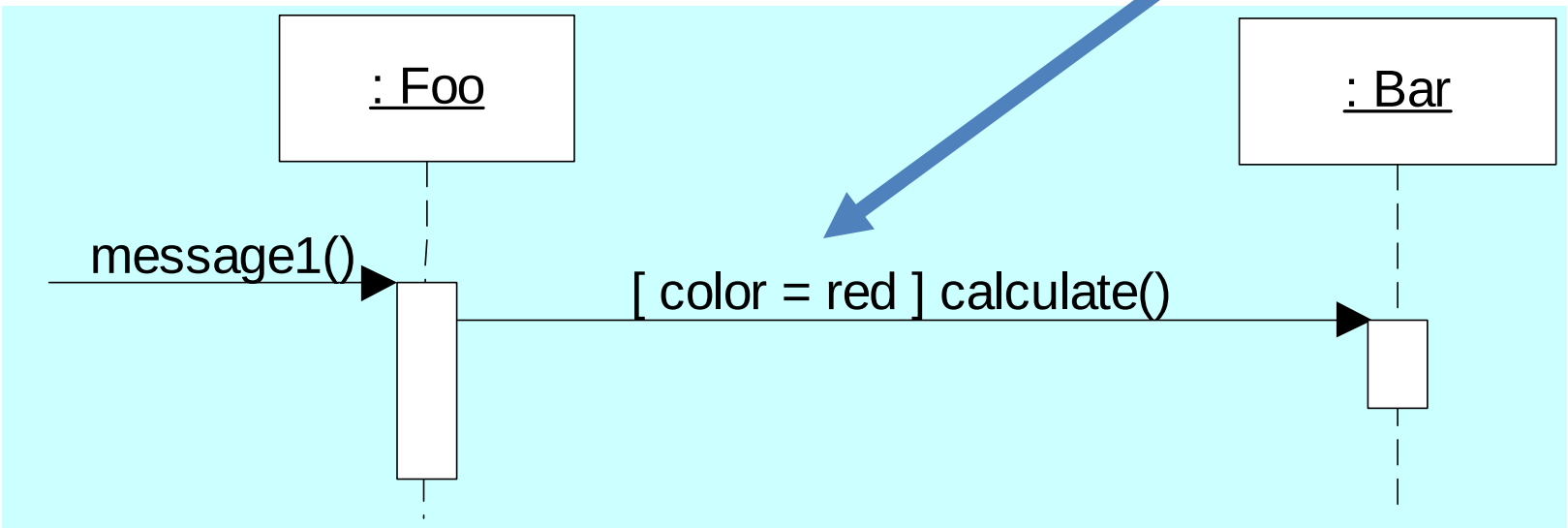
- Object lifelines:
  - The vertical dashed lines underneath the objects
  - These indicate extent of the life of the object
- Explicit destruction of an object is expressed as shown below.



Conditional messages

Sequence diagrams

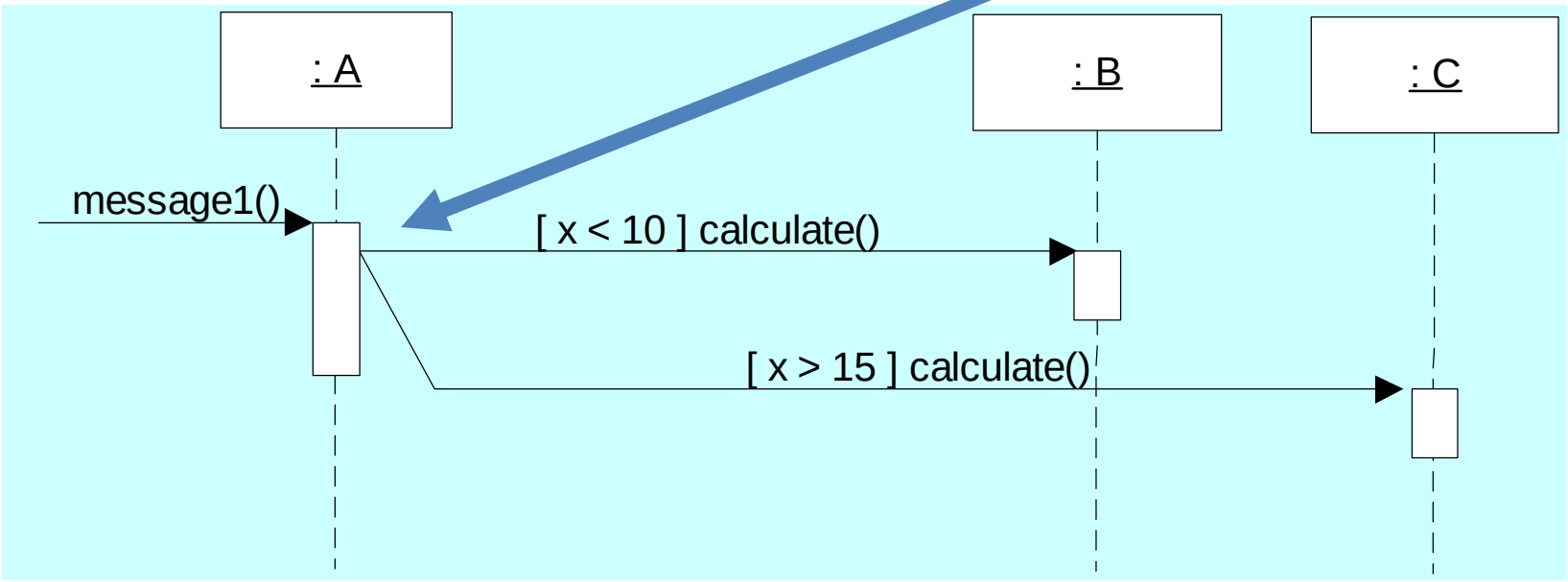
- In sequence diagrams, a conditional message is as shown below.



Mutually exclusive conditional messages

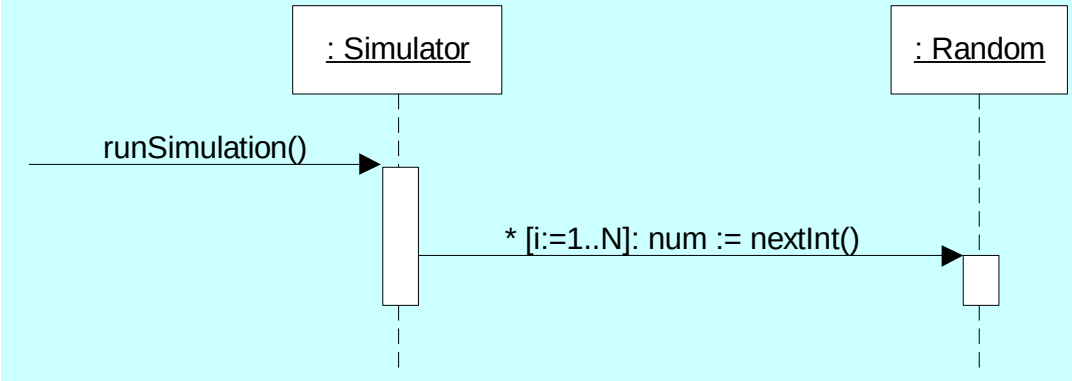
Sequence diagrams

- Mutually exclusive conditional messages are illustrated with a kind of angled line emerging from a common point, as shown.

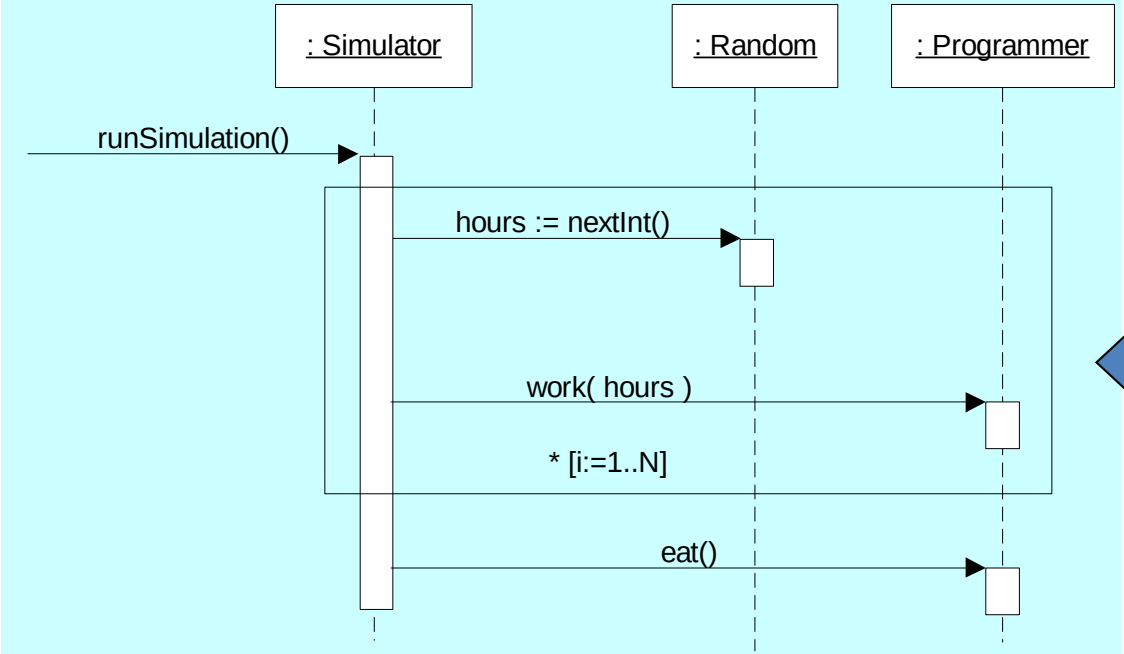


Iteration

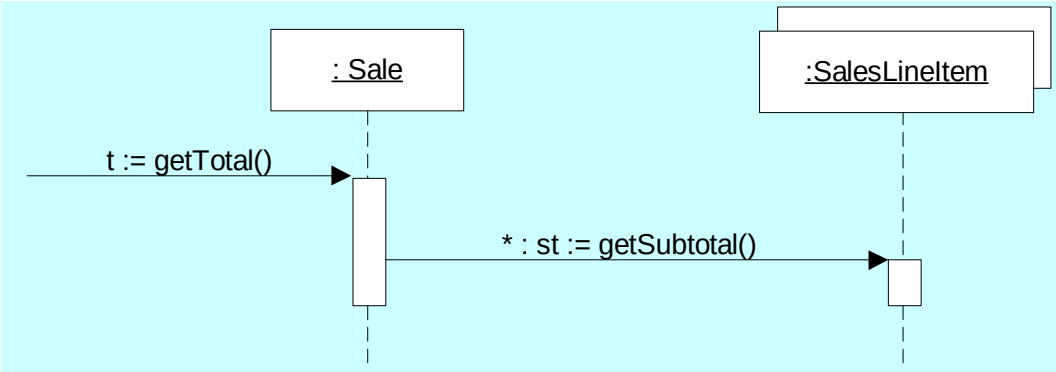
Sequence diagrams



Single message iteration



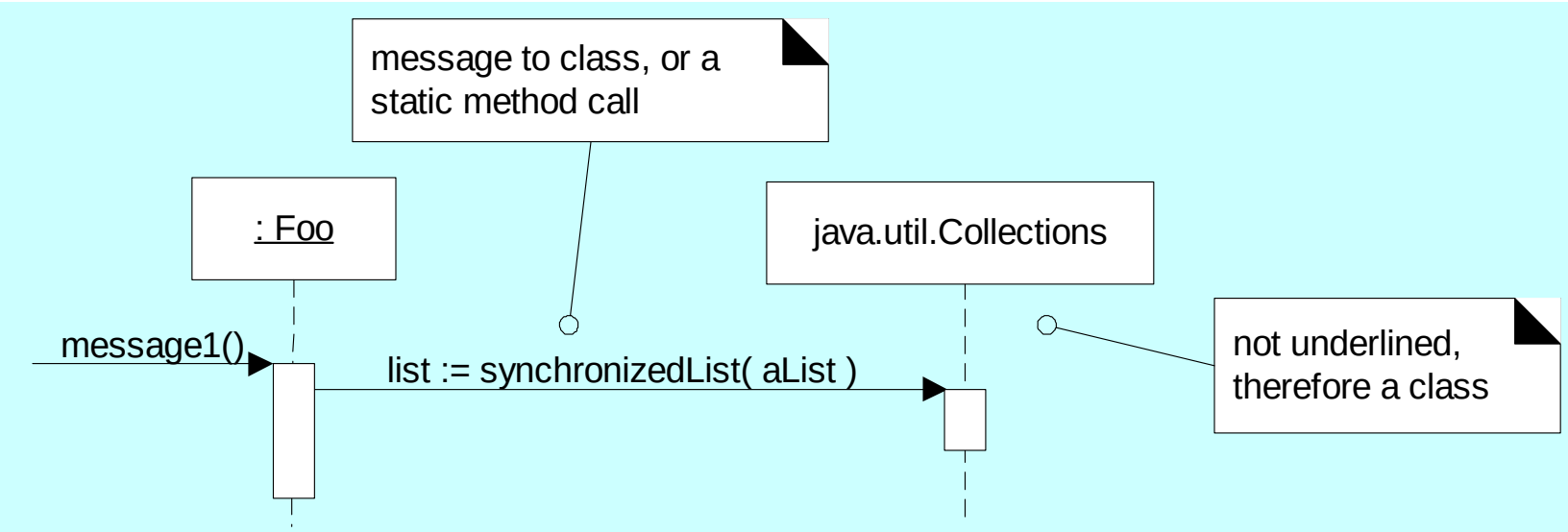
Iteration of series of messages



Iteration over a collection

Messages to Class Objects

- Class or static method calls are shown by not underlining the name of the classifier, which signifies a class object rather than an instance.



Questions and Conclusions