UNIVERSITY of GUELPH

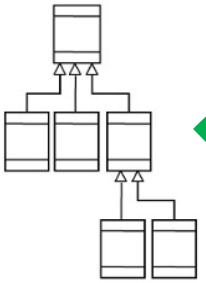
ENGINEERING

Sequence Diagrams

Fattane Zarrinkalam

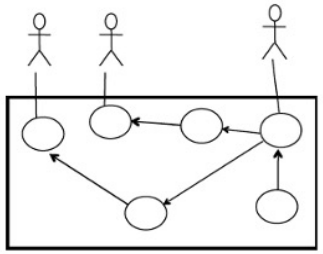
Fowler: UML Distilled: A Brief Guide to the Standard Object Modeling Lang. 3rd Ed. (Chapter 4)
K. Hamilton and R. Miles “Learning UML 2.0. O'Reilly”, ISBN-10: 0- 596-00982-8, 2006. (Chapter 7)

UML Diagrams



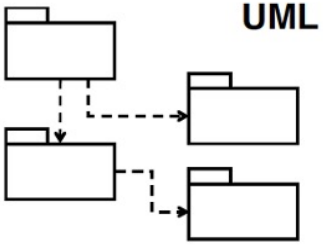
UML Class Diagrams

- information structure
- relationships between data items
- modular structure for the system



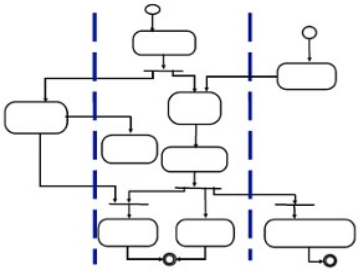
Use Cases

- user's view
- Lists functions
- visual overview of the main requirements



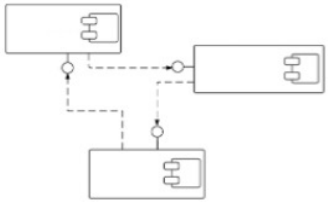
UML Package Diagrams

- Overall architecture
- Dependencies between components



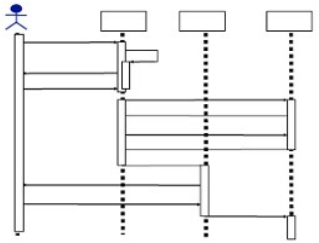
Activity diagrams

- business processes;
- concurrency and synchronization;
- dependencies between tasks;



UML Component Diagrams

- static implementation view of a system
- Interfaces between components



UML Sequence Diagrams

- individual scenario
- interactions between users and system
- Sequence of messages

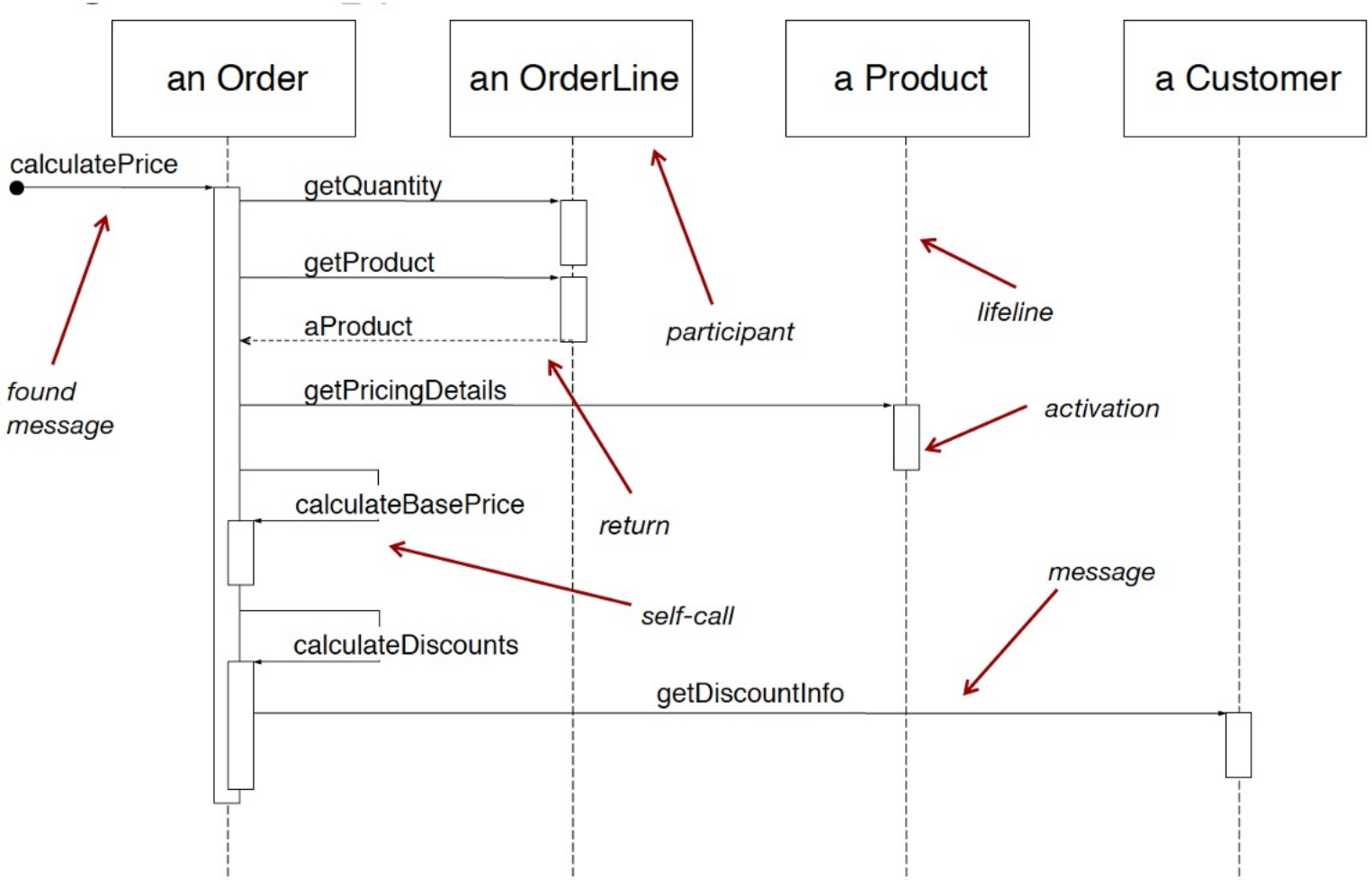
Sequence Diagram

- Depict the dynamic behavior of elements that make up a system as they interact over time.
- We can use this diagram to determine how best to develop a system.
- This allows architects, designers, and developers to ensure that the system, when implemented, satisfies its requirements.
- Sequence diagrams allow you to focus more on the timeline in which events occur.

Sample Scenario

We have an order and are going to invoke a command on it to calculate its price. To do that, the order needs to look at all the line items on the order and determine their prices, which are based on the pricing rules of the order line's products. Having done that for all the line items, the order then needs to compute an overall discount, which is based on rules tied to the customer.

Sequence Diagram



Participant

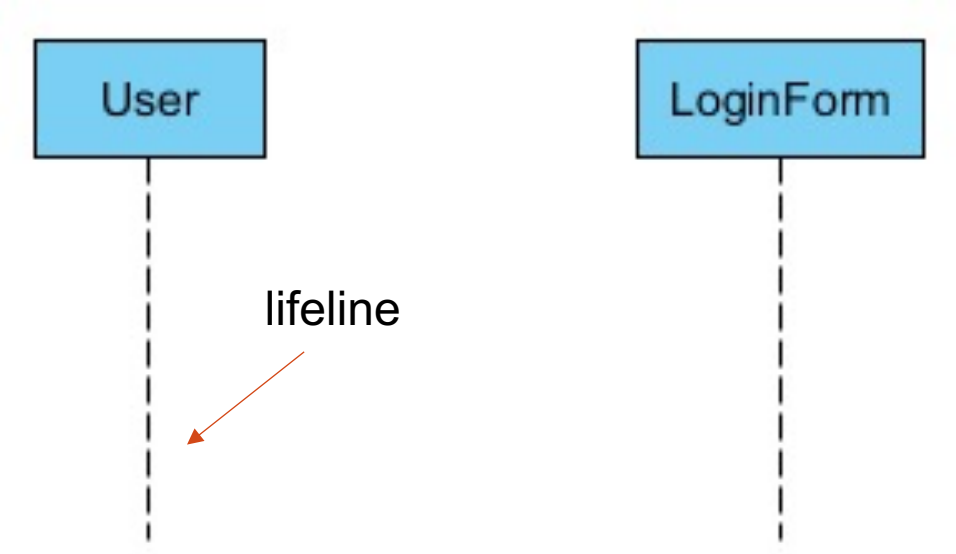
- The parts of your system that interact with each other during the sequence

name : class_name

Example participant name	Description
admin	A part is named `admin`, but at this point in time the part has not been assigned a class.
: ContentManagementSystem	The class of the participant is `ContentManagementSystem`, but the part currently does not have its own name.
admin : Administrator	There is a part that has a name of `admin` and is of the class `Administrator`.

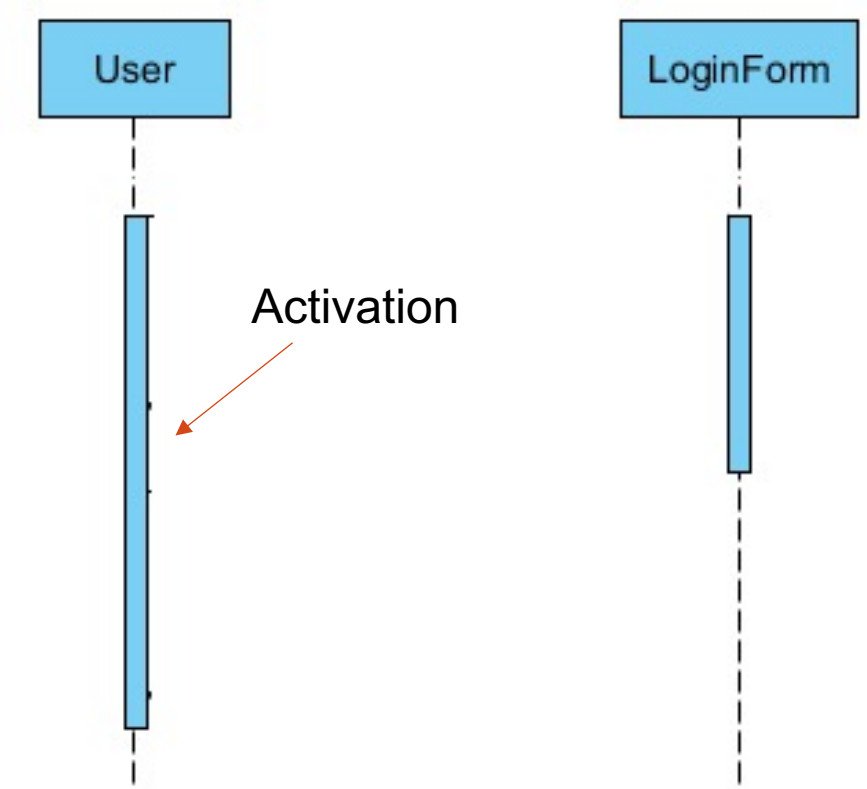
Lifeline

- The vertical line for attaching incoming and outgoing messages



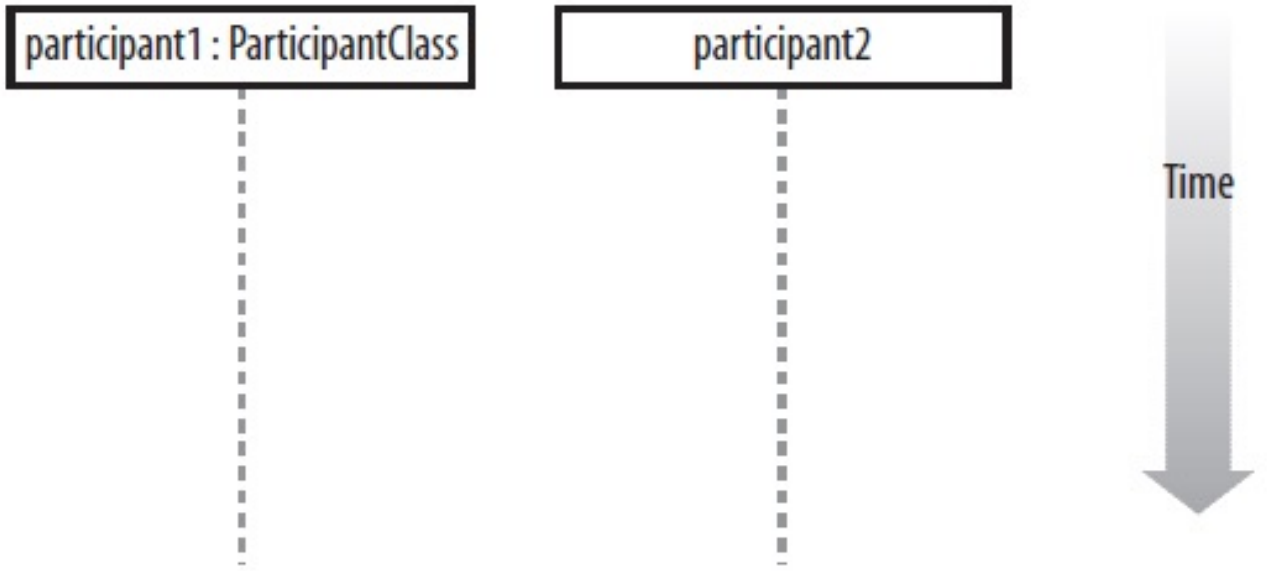
Activation

- Represents the span of an object's lifetime
  - Optional



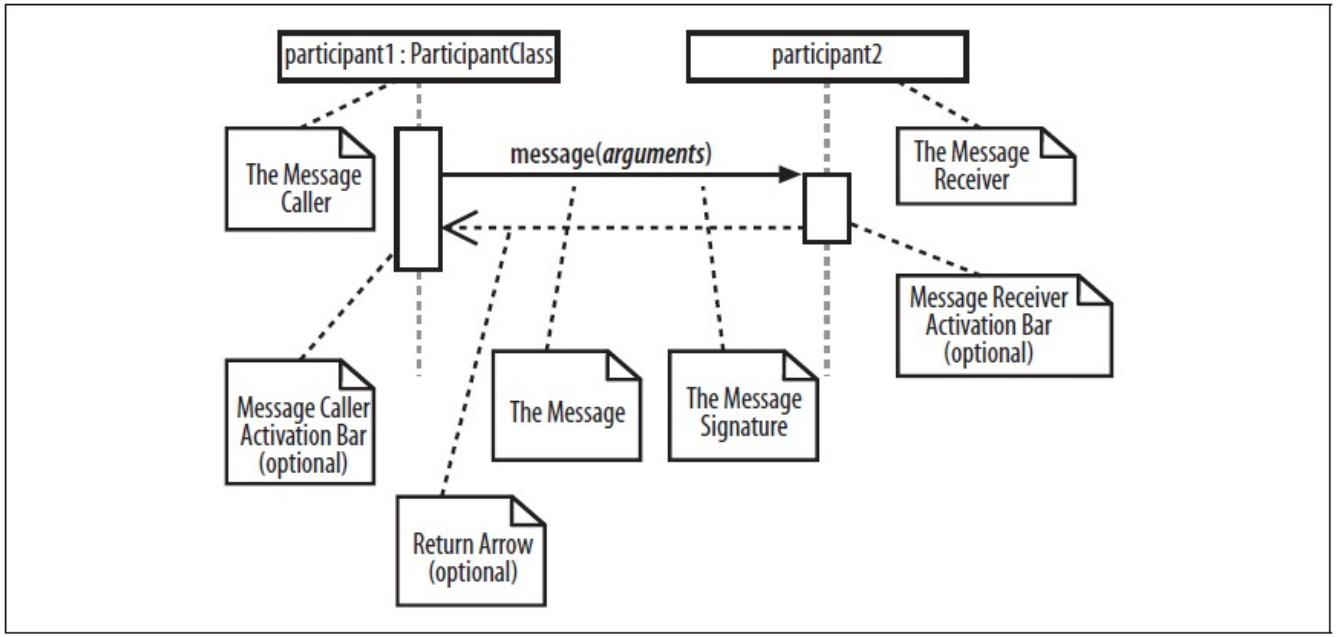
Time

- Time is an important factor
- Starts at the top of the page and then progresses down the page



Messages

- Communication from a sender class to a receiver class via an association is called a message.
- The sender object (or class) is known as the client, and the receiver object (or class) is known as the supplier.

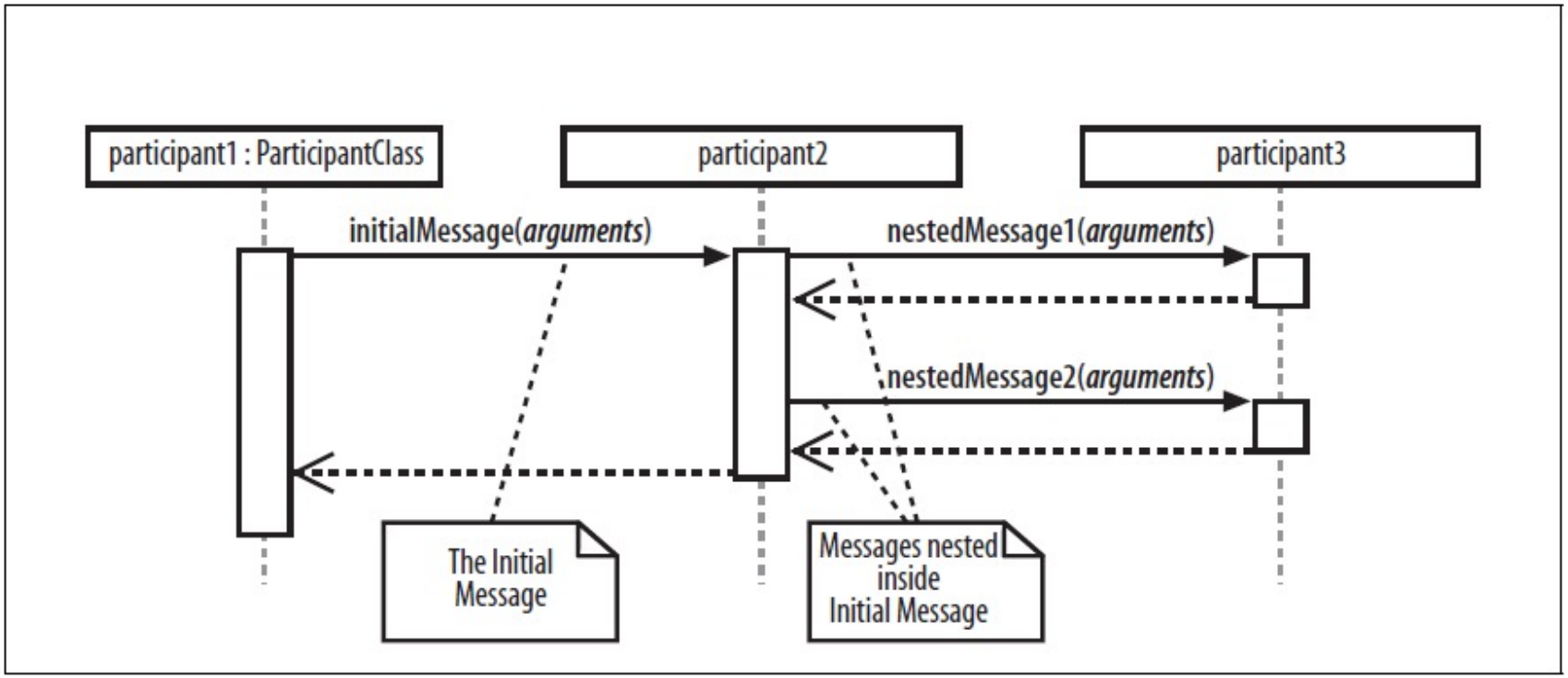


Message Signature

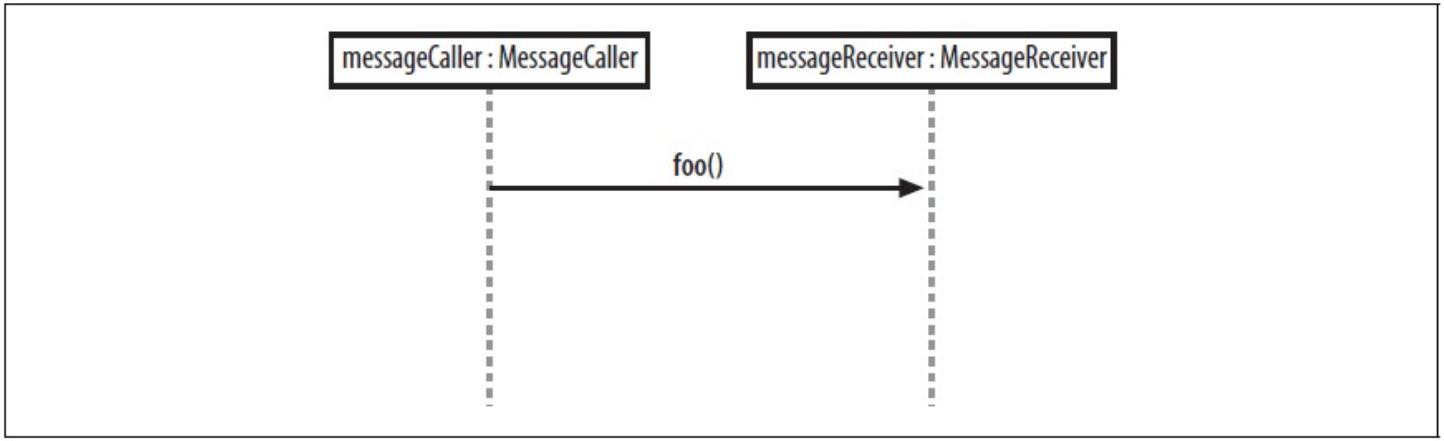
Attribute = signal_or_message_name (arguments) : return_type

Example message signature	Description
doSomething( )	The message's name is `doSomething`, but no further information is known about it.
doSomething(number1 : Number, number2 : Number)	The message's name is `doSomething`, and it takes two arguments, `number1` and `number2`, which are both of class `Number`.
doSomething( ) : ReturnClass	The message's name is `doSomething`; it takes no arguments and returns an object of class `ReturnClass`.
myVar = doSomething( ) : ReturnClass	The message's name is `doSomething`; it takes no arguments, and it returns an object of class `ReturnClass` that is assigned to the `myVar` attribute of the message caller.

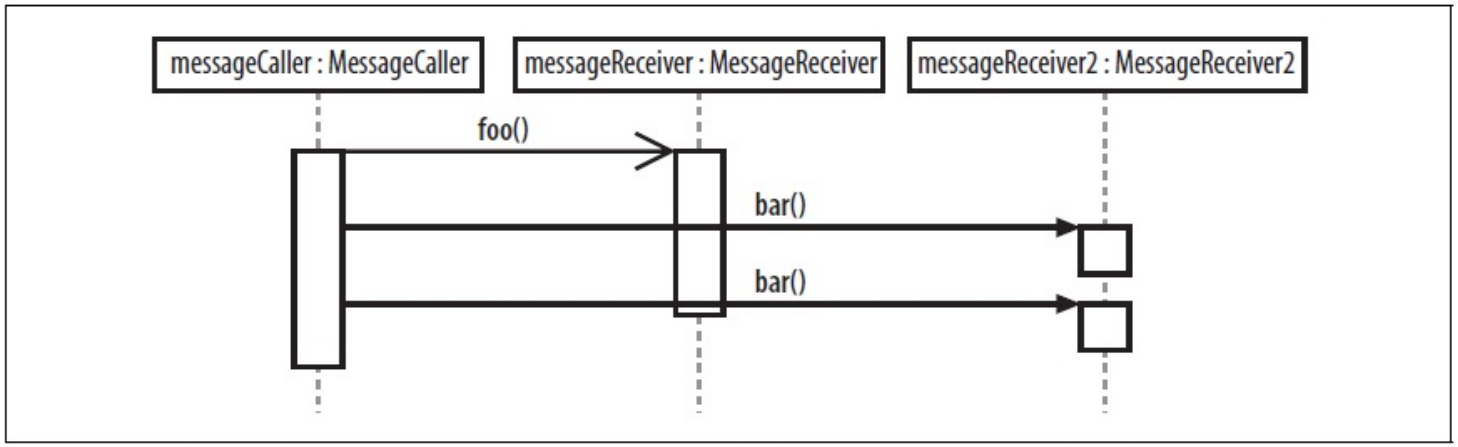
Nested Messages



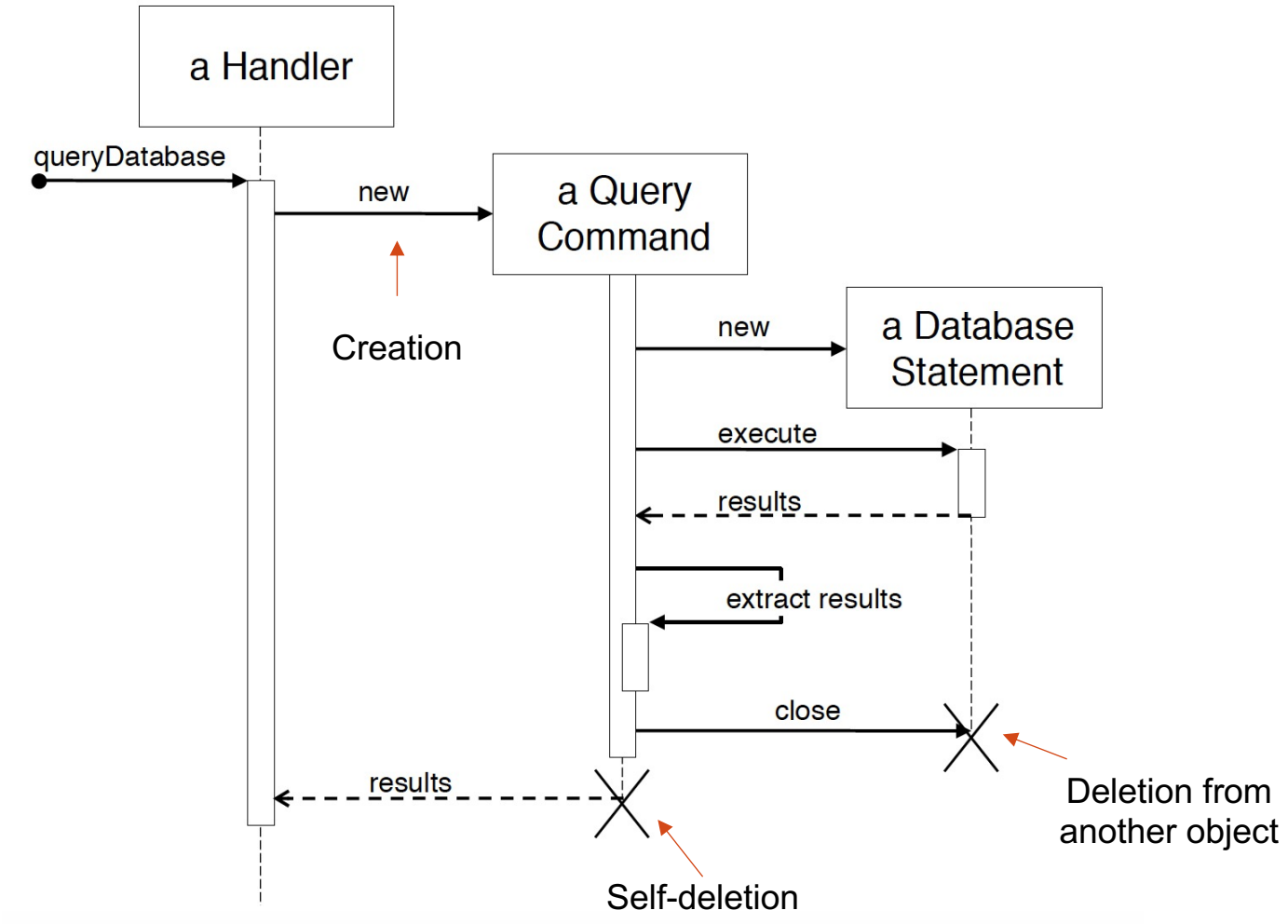
Synchronous Messages



Asynchronous Messages



Creation and Destruction Messages

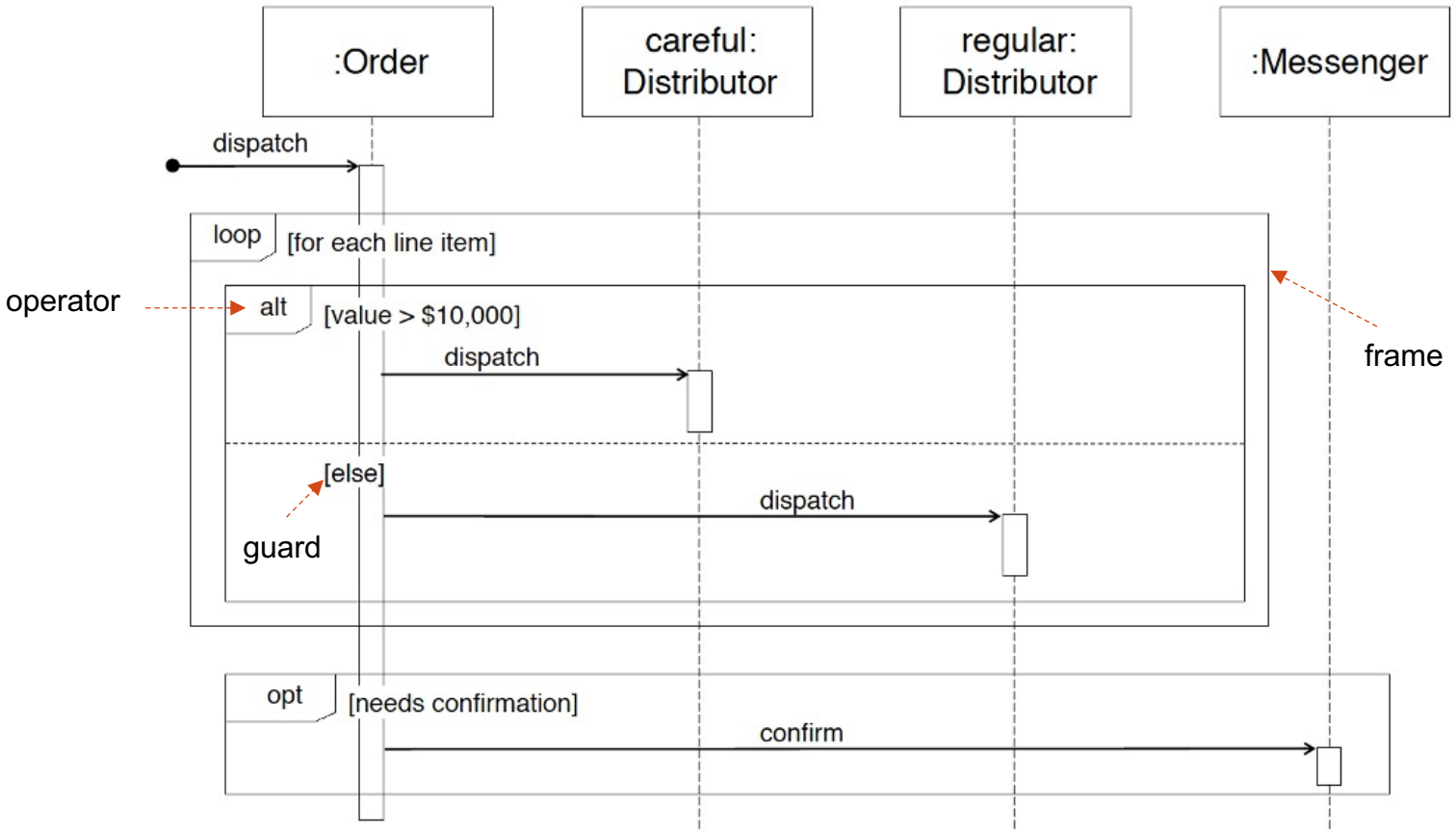


Loops, Conditionals and the Like

- How do we show looping and conditional behaviour?
- Example:

```
def dispatch():
      for lineItem in self.lineItems:
            if (product.value > $10K):
                  careful.dispatch()
            else:
                  regular.dispatch()
      if needsConfirmation:
            messenger.confirm()
```

Interaction Frames



Interaction Frames Operator

Operator	Meaning
alt	Alternative; only the frame whose guard is true will execute
opt	Optional; only executes if the guard is true
par	Parallel; frames execute in parallel
loop	Frame executes multiple times, guard indicates how many
region	Critical region; only one thread can execute this frame at a time
neg	Negative; frame shows an invalid interaction
ref	Reference; refers to a sequence shown on another diagram
sd	Sequence Diagram; used to surround the whole diagram (optional)

When to use sequence Diagram

- Comparing Design Options
  - Shows how objects collaborate to carry out a single use case
  - Graphical form shows alternative behaviours
- Assessing Bottlenecks
  - e.g., an object through which many messages pass
- Elaborating Use Cases
  - Shows how the user expects to interact with the system

Style Guide for Sequence Diagrams

- Spatial Layout
  - Strive for left-to-right ordering of messages
  - Put proactive participants on the left
  - Put reactive participants on the right
- Readability
  - Keep diagrams simple
  - Don’t show obvious return values
  - Don’t show object destruction
- Consistency
  - Class names must be consistent with class diagram
  - Message routes must be consistent with (navigable) class associations

Summary

Summary

- Sequence diagrams model the interactions between objects in a single use case.
- They illustrate how the different parts of a system interact with each other to carry out a function, and the order in which the interactions occur when a particular use case is executed.
- Elements:
  - Participants or objects
  - Lifeline
  - Activation bar
  - Messages
  - Interaction frames