Object-Oriented Analysis and Design

Session 5: Behavioral Modeling - Interactions

Outline

Introduction to System Behavior

Capturing the System Behavior

In an OO model – where there are only objects – behavior is reflected by CHANGE in object states

- Behavior can be described from two different view points:
  - Interaction between objects that causes the desired state changes
    - Sequence diagram
    - Communication diagram
  - The different states of an object that are caused by interactions
    - State machine diagram

Interaction between Objects

Interaction is captured by **processes**

-- or –

Processes are captured by **interaction**.

In OO modeling responsibility for processes is imposed on objects – instances of classes.

- A process is initiated by a **triggering event** – a **message** sent to an object that can be responsible for the process.

- A process is performed by initiating other processes – sending messages to other objects.

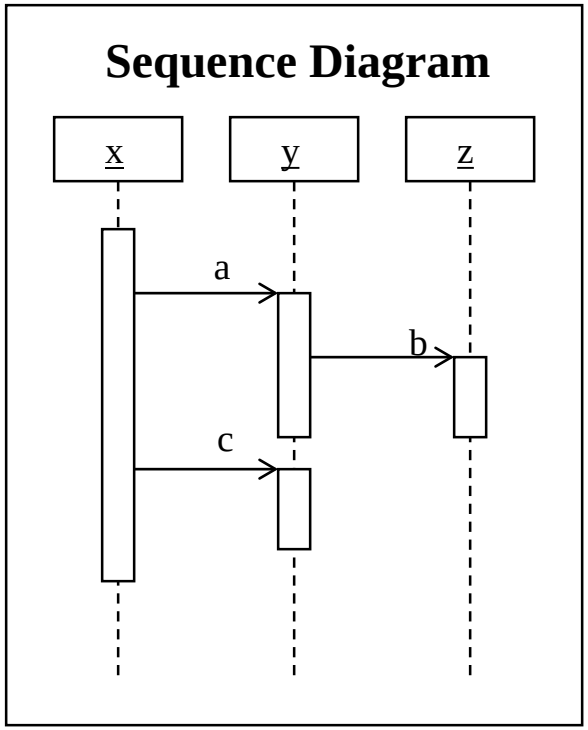
→

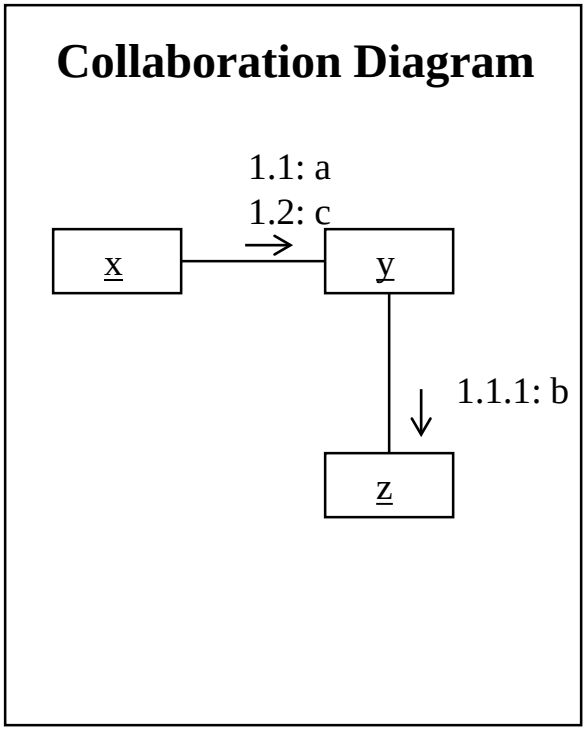
A process is performed by a group of objects that Collaborate (interact) to perform a task.

Interaction Diagrams

- Model the dynamic aspects of a system.
- Two types:
  - Sequence diagram - emphasizes the time ordering of messages.
  - Communication diagram - emphasizes the structural organization of the message sending objects.

Interaction Diagrams





Sequence Diagram – Syntax and Semantics

Interaction on Sequence Diagrams

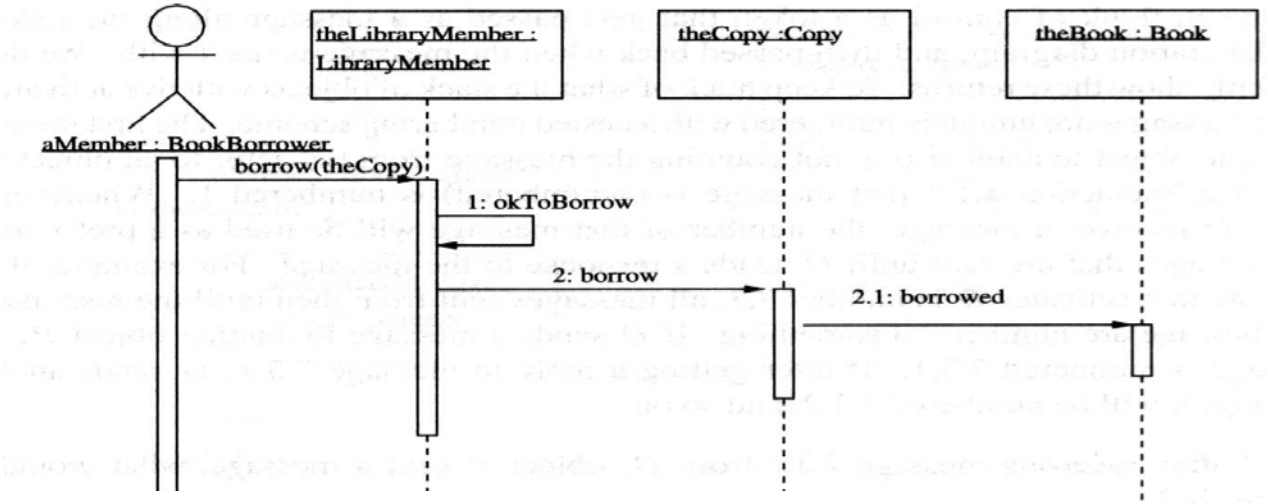
A sequence diagram shows:

- Object lifeline,
- messages between objects,
- Object live activation.

Time passage is captured by the top to bottom direction.

Object live activation is captured by the rectangles.

Borrow a book interaction:



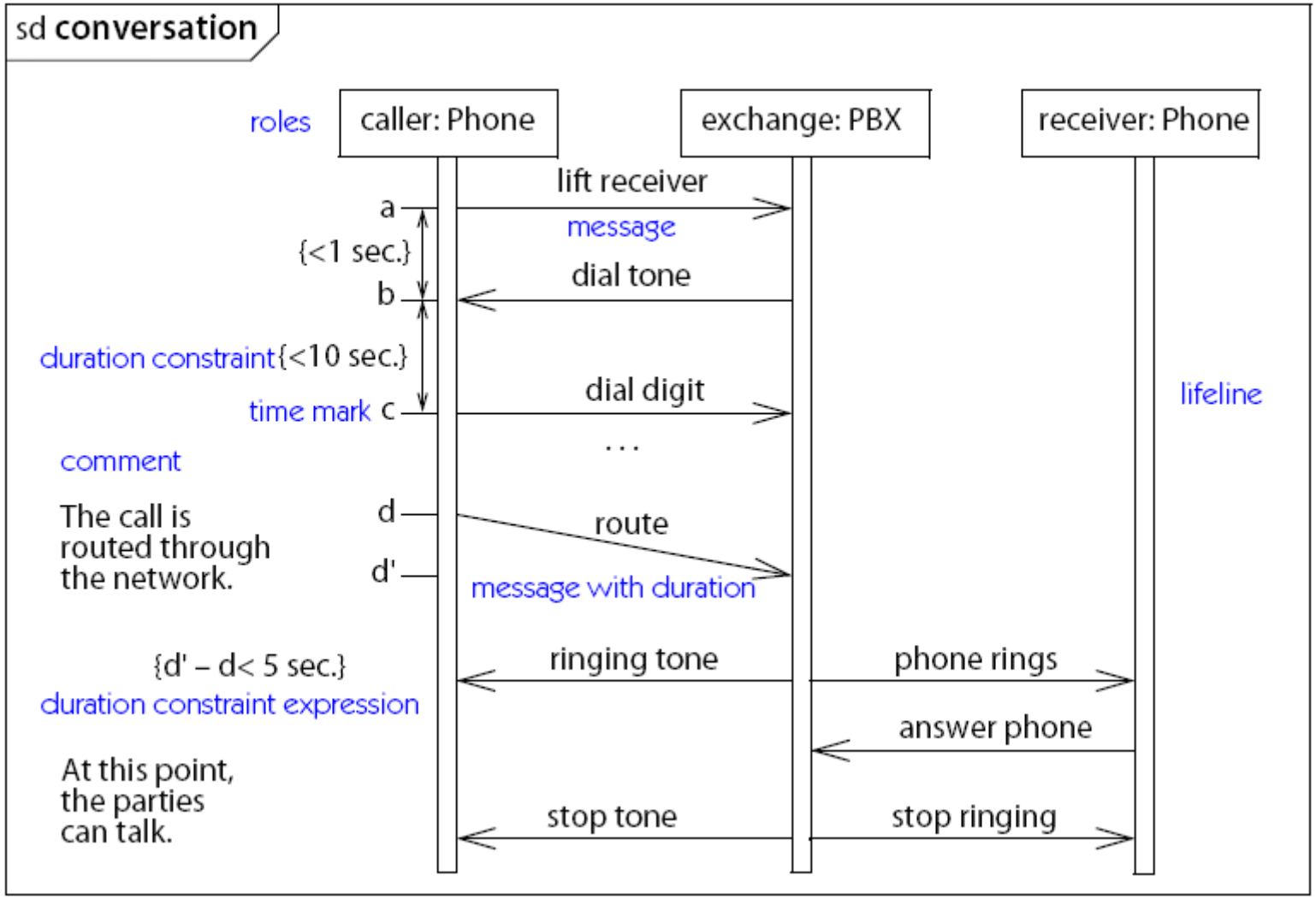
Sequence Diagrams (1)

- Emphasizes the <u>time ordering</u> of messages dispatched among objects.
- Objects that participate are placed in the top, along the X axis.
- Messages are placed along the Y axis.
- Object <u>lifeline</u> is a vertical dashed line that represents the existence of an object over a period of time.

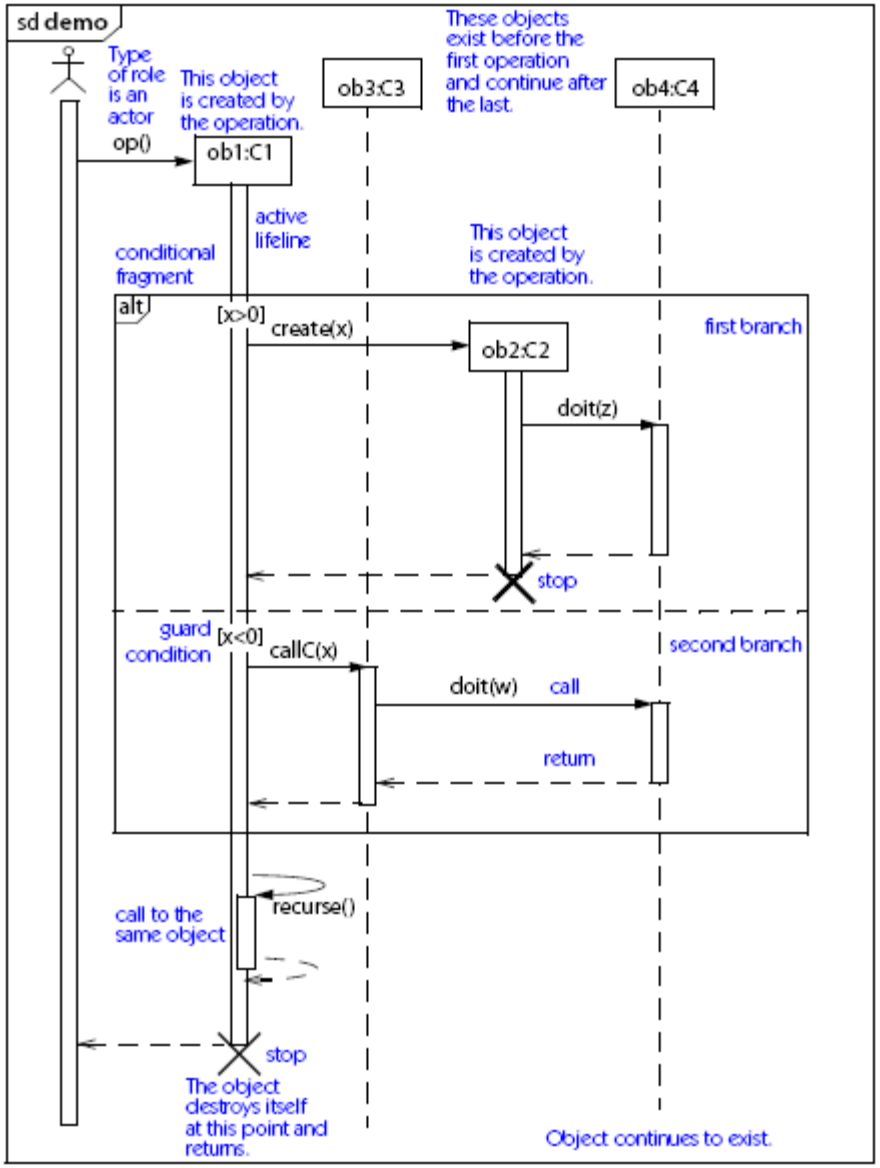
Sequence Diagrams (2)

- Objects may be created – their lifeline starts upon the receipt of the create message.
- Objects may be destroyed – their lifeline ends upon the receipt of the destroy message.
- The <u>focus of control</u> is a thin rectangle showing the period of time during which the object is performing actions.

Sequence Diagram – Notations (1)



Sequence Diagram – Notations (2)

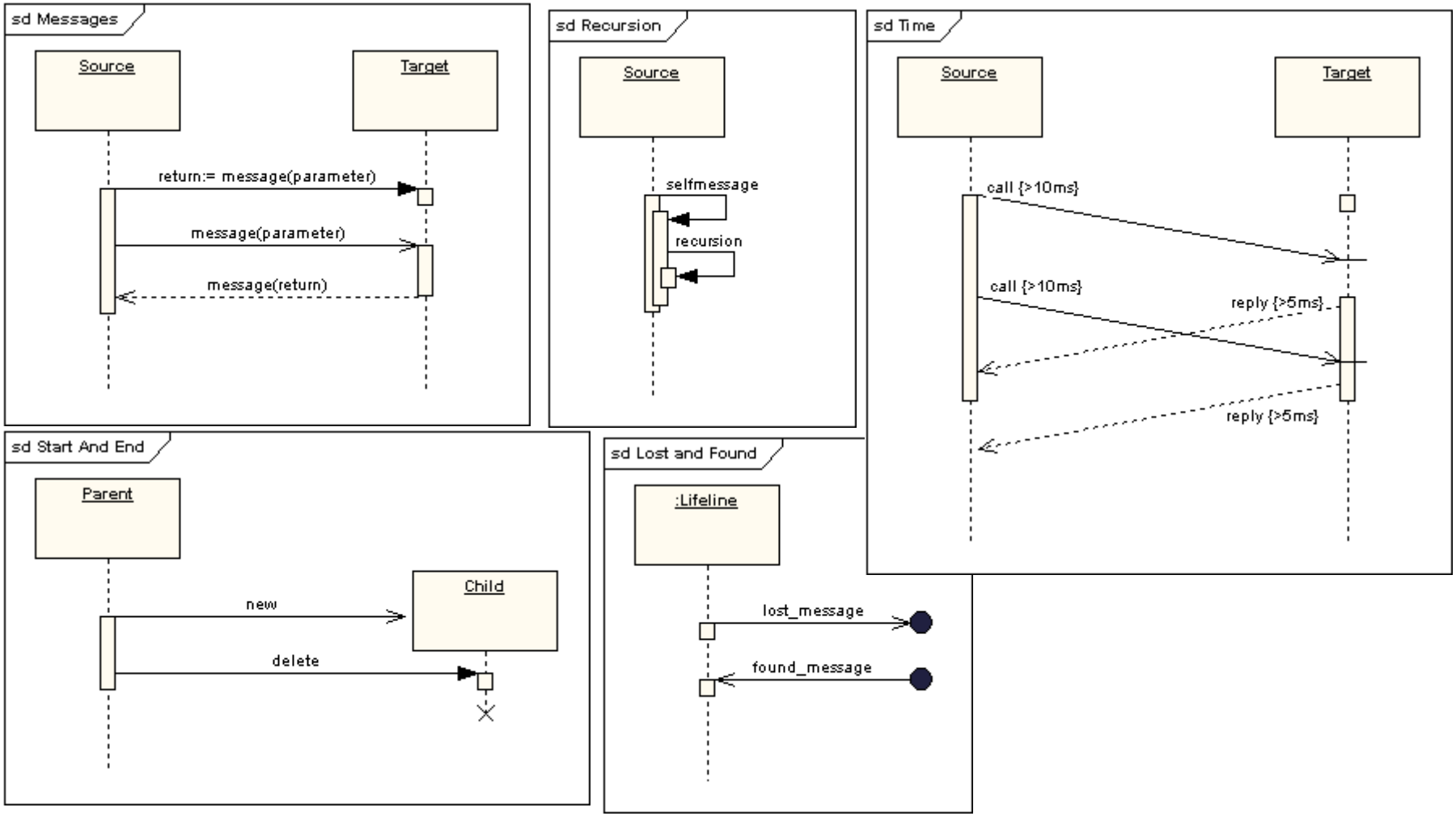


Messages (1)

- Messages involve the invocation of an operation, or sending of a signal, or creation or destruction of objects.
- A message is the specification of a communication among objects.
- Often a message results in an action.
- In UML one can model several kinds of actions:

  Call - invokes an operation
  Return - returns a value to the caller
  Send - sends a signal to an object
  Create - creates an object
  Destroy - destroys an object
  Lost - describes a message which its receiver is unknown
  Found - describes a message which its sender is unknown
- Message name and required parameters is marked above the middle of the arrow. e.g., connect (s).
- Message send-time name may be added to the left of the message name, e.g., a: route.

Messages (2)


sd Messages
Source
Target
return:= message(parameter)
message(parameter)
message(return)
sd Recursion
Source
selfmessage
recursion
sd Time
Source
Target
call {>10ms}
call {>10ms}
reply {>5ms}
reply {>5ms}
sd Start And End
Parent
Child
new
delete
sd Lost and Found
:Lifeline
lost_message
found_message

Combined Fragment (1)

- A combined fragment is one or more processing sequence enclosed in a frame and executed under specific circumstances
  - They allow for adding a degree of procedural logic to diagrams

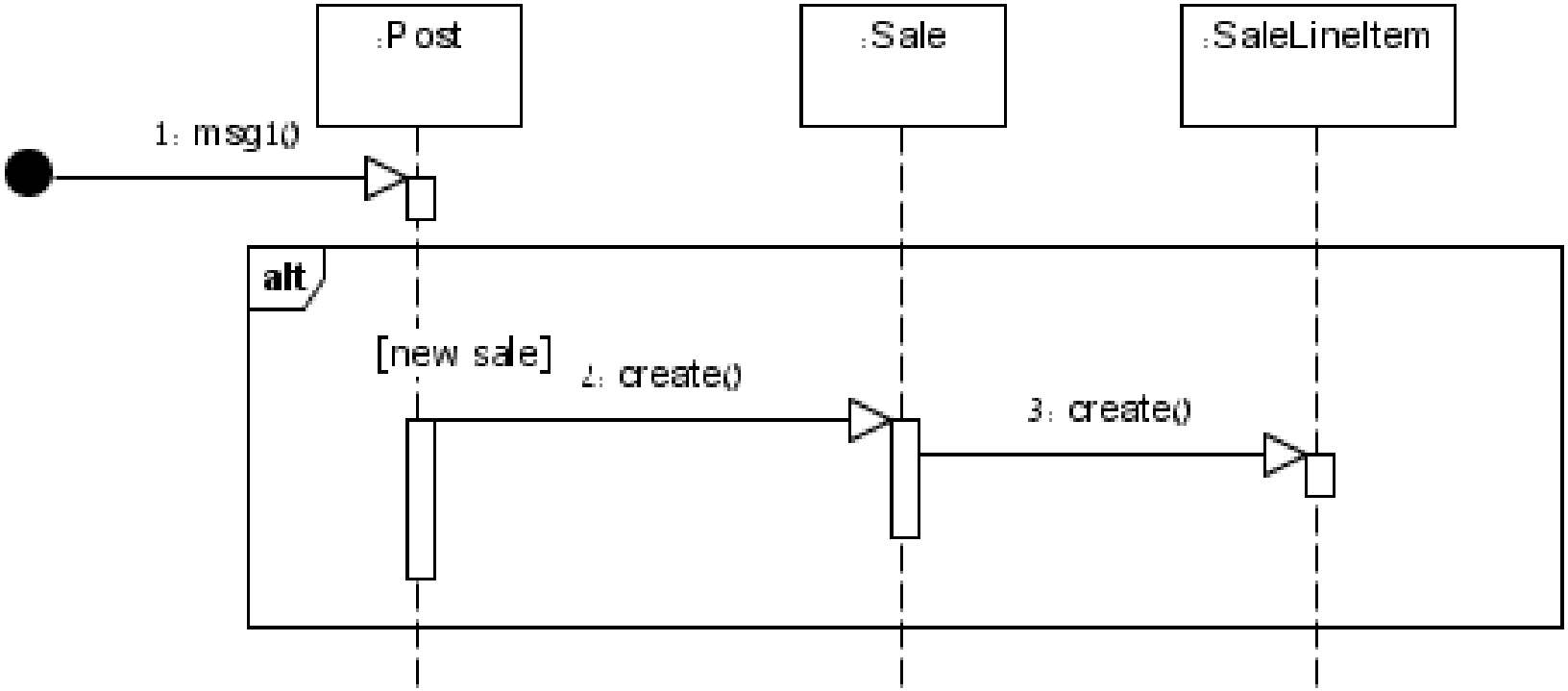
Combined Fragment (2)

Name	Symbol	Meaning
Alternative fragment	alt	if…then…else constructs
Option fragment	opt	switch constructs
Break fragment	break	an alternative sequence of events that is processed instead of the whole of the rest of the diagram
Parallel fragment	par	concurrent processing
Weak sequencing fragment	seq	number of messages that must be processed in a preceding segment before the following segment can start, but which does not impose any sequencing within a segment on messages that don't share a lifeline
Strict sequencing fragment	strict	a series of messages which must be processed in the given order

Combined Fragment (3)

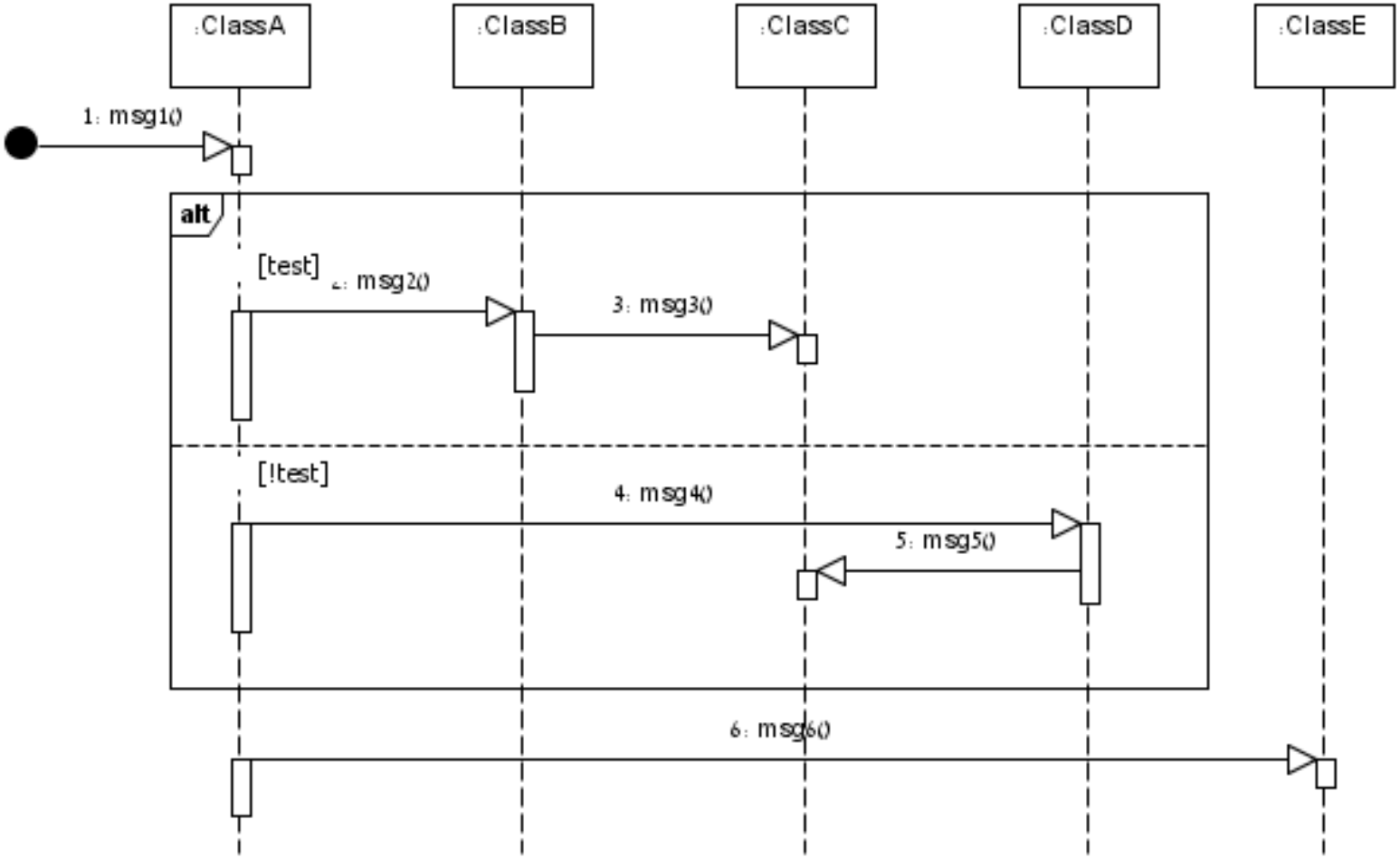
Name	Symbol	Meaning
Negative fragment	neg	an invalid series of messages
Critical fragment	critical	encloses a critical section
Ignore fragment	ignore	a message or message to be of no interest if it appears in the current context
Consider fragment	consider	the opposite of the ignore fragment: any message not included in the consider fragment should be ignored
Assertion fragment	assert	any sequence not shown as an operand of the assertion is invalid
Loop fragment	loop	a series of messages which are repeated

Combined Fragment (4)



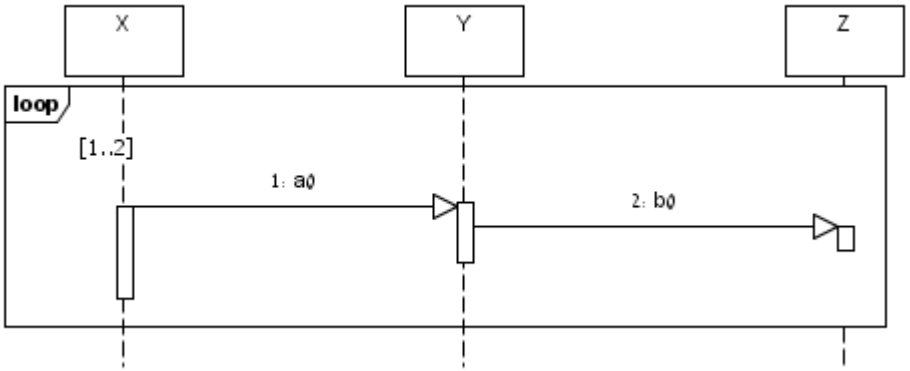
Combined Fragment (5)

Mutually exclusive messages

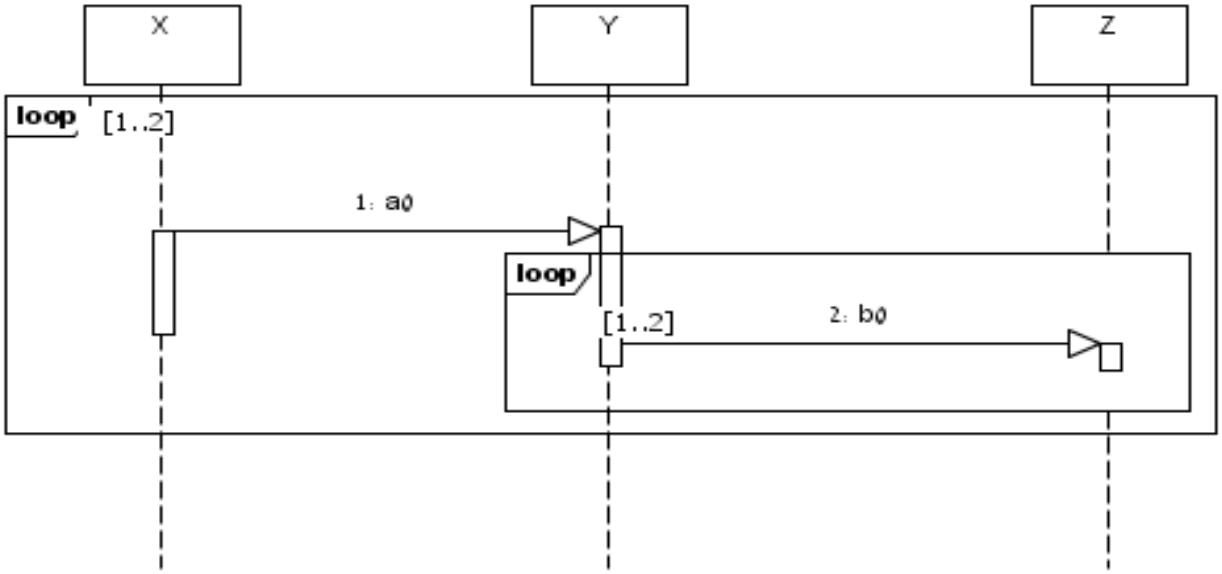


Combined Fragment (6)

Messages abab:



Messages abbabb:



Communication Diagram – Syntax and Semantics

Interaction on Communication Diagrams

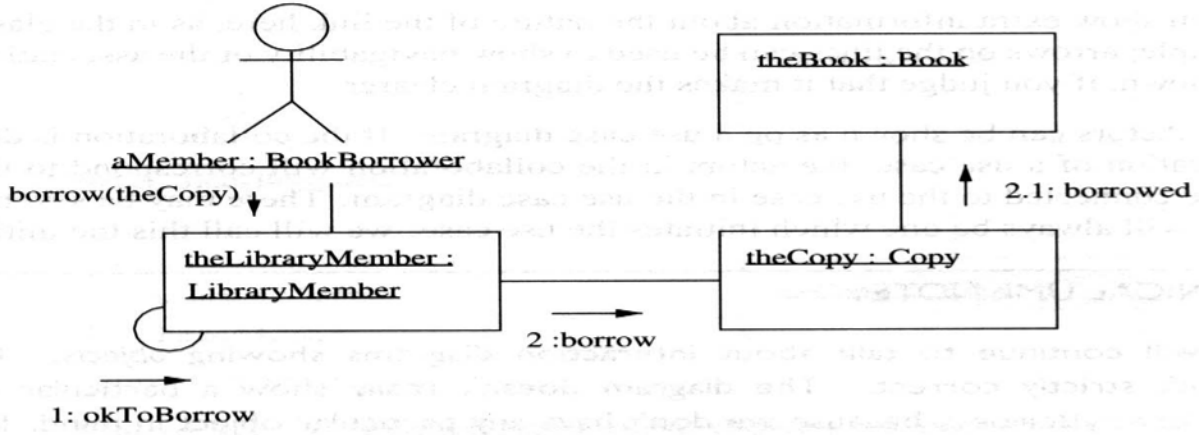
A communication diagram shows:

- Objects,
- links between objects,
- messages between objects,
- temporal ordering and nesting of these messages.

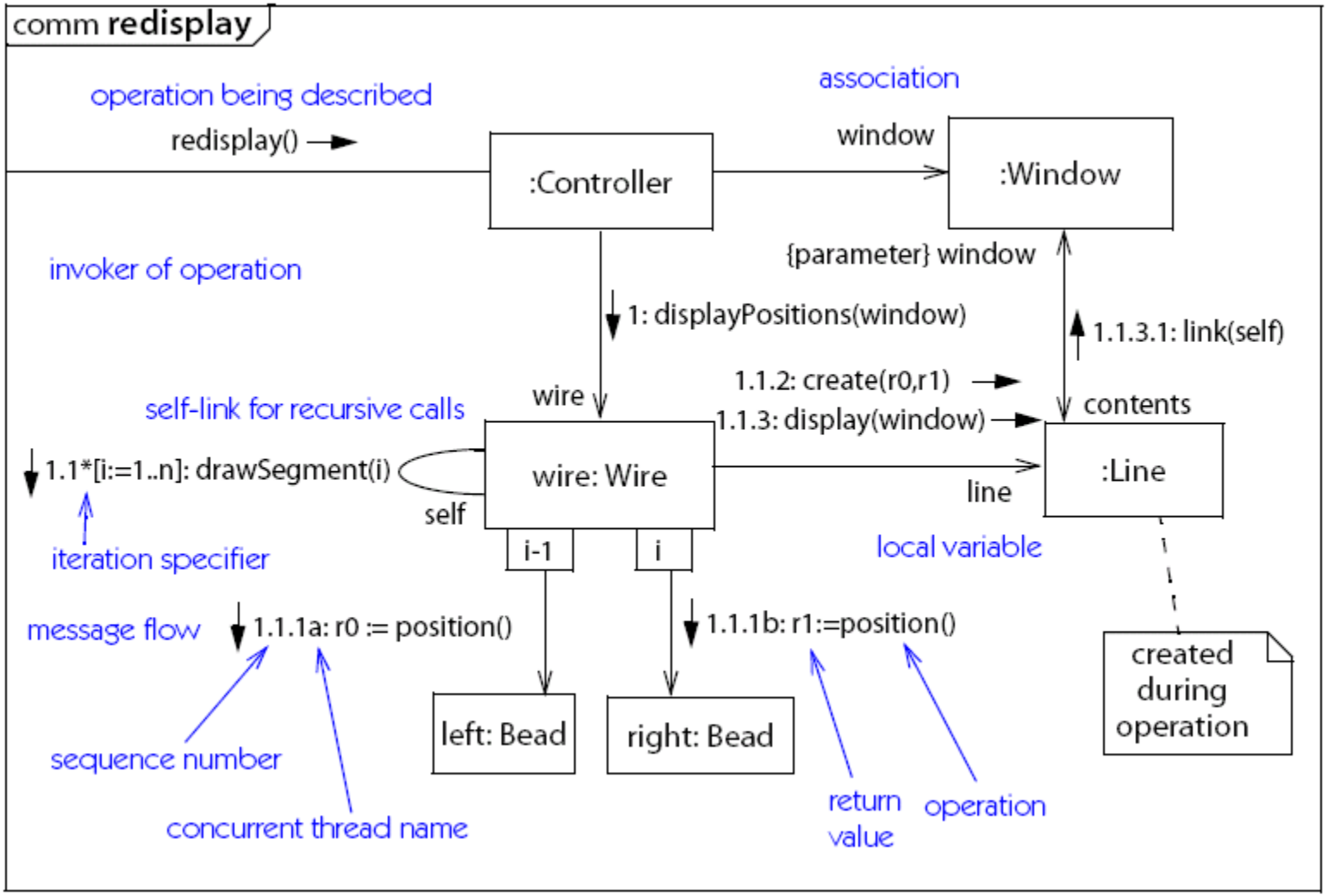
Time passage is captured by the ordered numbering.

Object live activation is captured by the nested numbering.

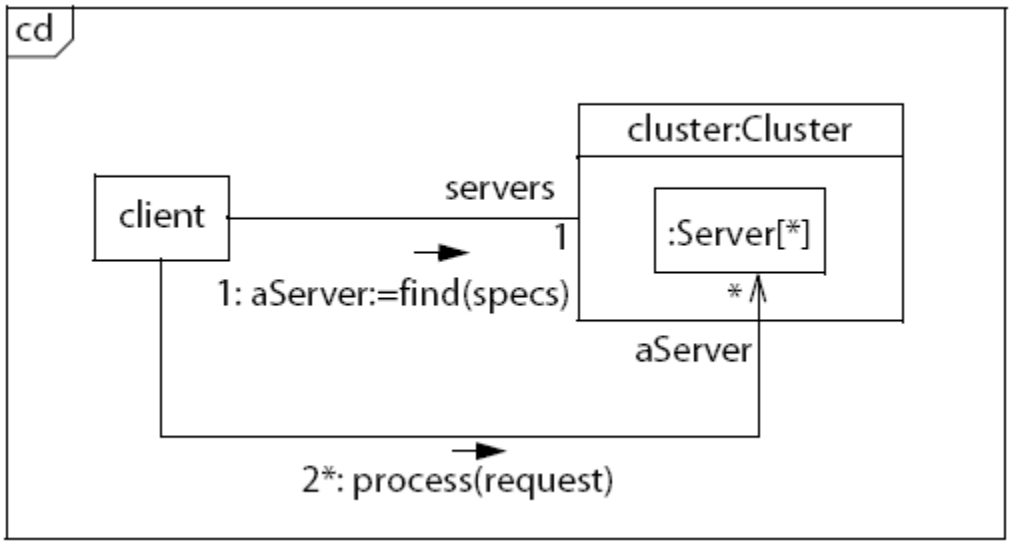
Borrow a book interaction:



Communication Diagram



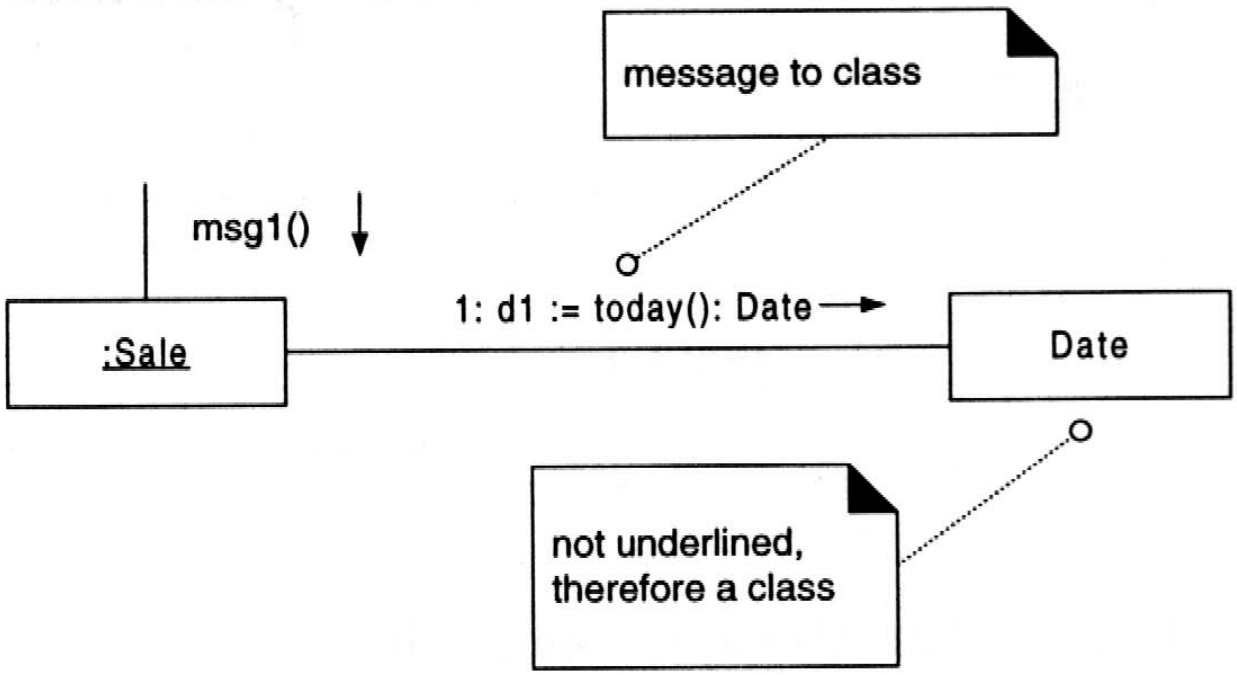
Messages to collection objects



Results from “many” multiplicity constraint in the static diagram.

Messages to a class object

Static method invocation:



Object Visibility

Object Visibility (1)

- An object can *send a message* only to objects that are **visible** to it.
- An object *O* that receives a message *m* can send a message only to the following objects that are in its *visibility region*:
  1. *Self visibility:* Itself.
  2. *Attribute visibility:* Objects which are directly accessible from *O* – through directed links (reference attributes).
  3. *Parameter visibility:* Objects that are sent as arguments in the message *m*.
  4. *Creation visibility:* Objects that *O* creates as part of its reaction to m.

Object Visibility (2)

- *Permanent visibility:*

  Self visibility.

  Attribute visibility.

  Example: Library – book.

- *Temporary visibility:*

  Parameter visibility.

  Creation visibility.

  Example: Automatic authentication in a building entrance.

Object Visibility (3)

- Every message in an interaction diagram that is justified by an attribute visibility (**directed link**) from one object to another should be enabled by an **association** in the class diagram.
- When writing the interaction diagrams the class diagram should be observed and corrected.
- The messages in the interaction diagrams show **attribute navigability** (visibility) on the class diagram.

Concluding Example

Ordering System

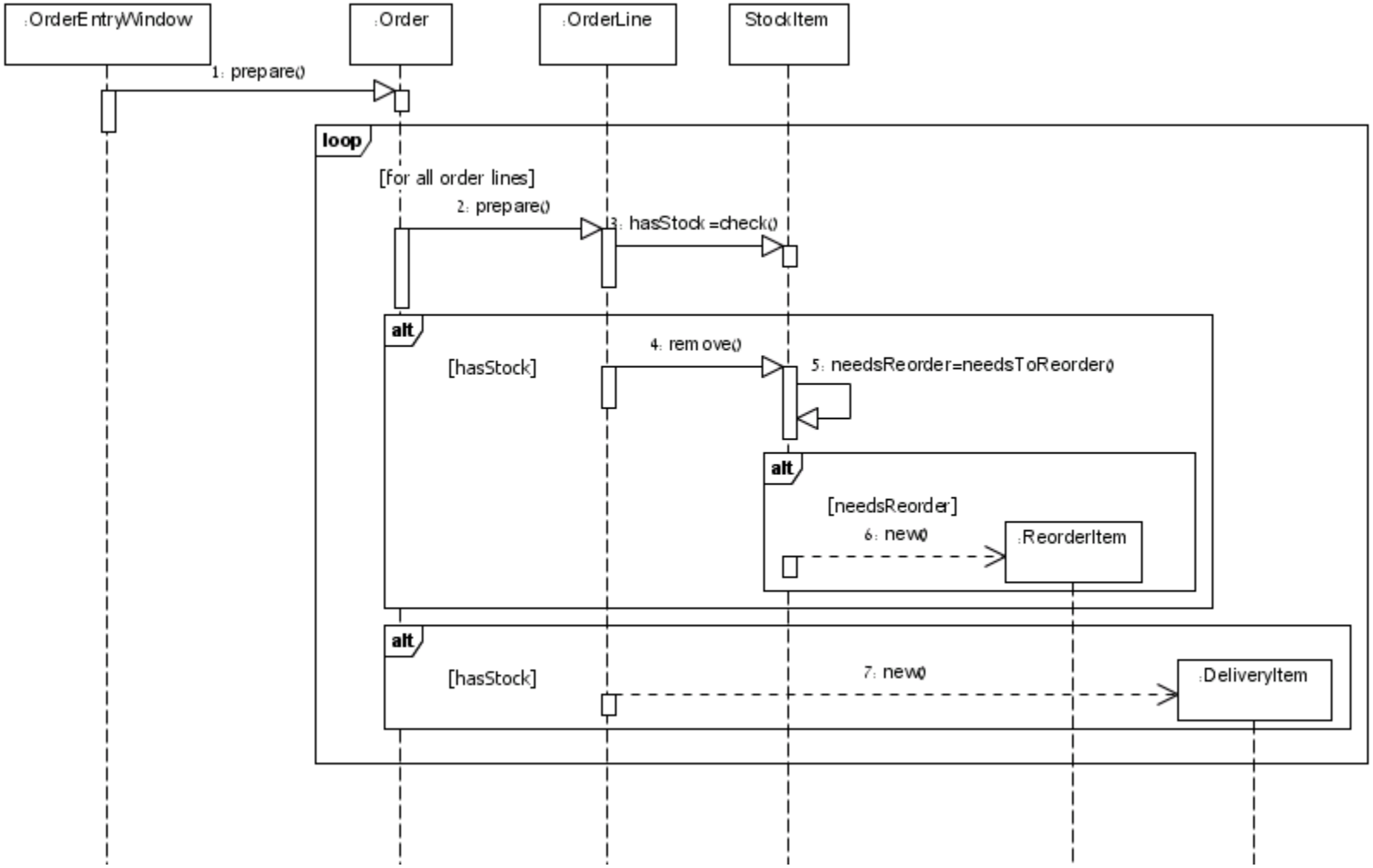
A class diagram with classes:

OrderEntryWindow (a singleton), Order, OrderLine, StockItem, ReorderItem, DeliveryItem.

An Order object is associated with multiple OrderLine objects (one for each kind of StockItem)

Operation: The OrderEntryObject receives a message for preparing an order. It sends the message to “its” Order object. The Order object iterates the message over all “its” OrderLine Objects. The OrderLine object passes the message to “its” StockItem, and creates a DeliveryItem.

Ordering System: A Sequence Diagram



Intermediate Summary

- Set the context for the interaction.
- Include only those features of the instances that are relevant.
- Express the flow from left to right and from top to bottom.
- Put active instances to the left/top and passive ones to the right/bottom.
- Use sequence diagrams
  - to show the explicit ordering between the stimuli
  - when modeling real-time
- Use communication diagrams
  - when structure is important
  - to concentrate on the effects on the instances
- Update class diagram