7. Behavioural Modelling using Interaction Diagrams

IT 3106– Object Oriented Analysis and Design

Level II - Semester 3

Overview

In this section students will

- develop scenarios to the system to describe how Use Cases are realized as interactions among societies of objects,
- describe a scenario using sequence diagrams and communication diagrams.

Intended Learning Outcomes

At the end of this lesson students will be able to

- understand the rules and style guidelines for sequence and communication diagrams,
- understand the processes used to create sequence and communication diagrams,
- understand the relationship between the behavioral models, structural models and functional models.

List of Subtopics

7. Behavioural Modelling using Interaction Diagrams (5 hours)

   7.1 Introduction to Behavioural Modeling using Interaction Diagrams [Ref 1: Pg. 203-204]

   7.2 Sequence Diagrams [Ref 1: Pg. 204-215]

      7.2.1 Elements of a Sequence Diagram

      7.2.2 Creating Sequence Diagrams

   7.3 Communication Diagrams [Ref 1: Pg. 216-221]

      7.3.1 Elements of a Communication Diagram

      7.3.2 Creating a Communication Diagram

Ref 1: Alan Dennis, Barbara Haley, David Tegarden, Systems analysis design, An Object-Oriented Approach with UML : an object-oriented approach, 5th edition, John Wiley & Sons, 2015, ISBN 978-1-118-80467-4

7.1 Introduction to Behavioural Modeling using Interaction Diagrams

- Behavioral models describe the internal dynamic aspects of an information system that supports the business processes in an organization.
- During analysis, behavioral models describe what the internal logic of the processes is without specifying how the processes are to be implemented.
- Later, in the design and implementation phases, the detailed design of the operations contained in the object is fully specified.

7.1 Introduction to Behavioural Modeling using Interaction Diagrams

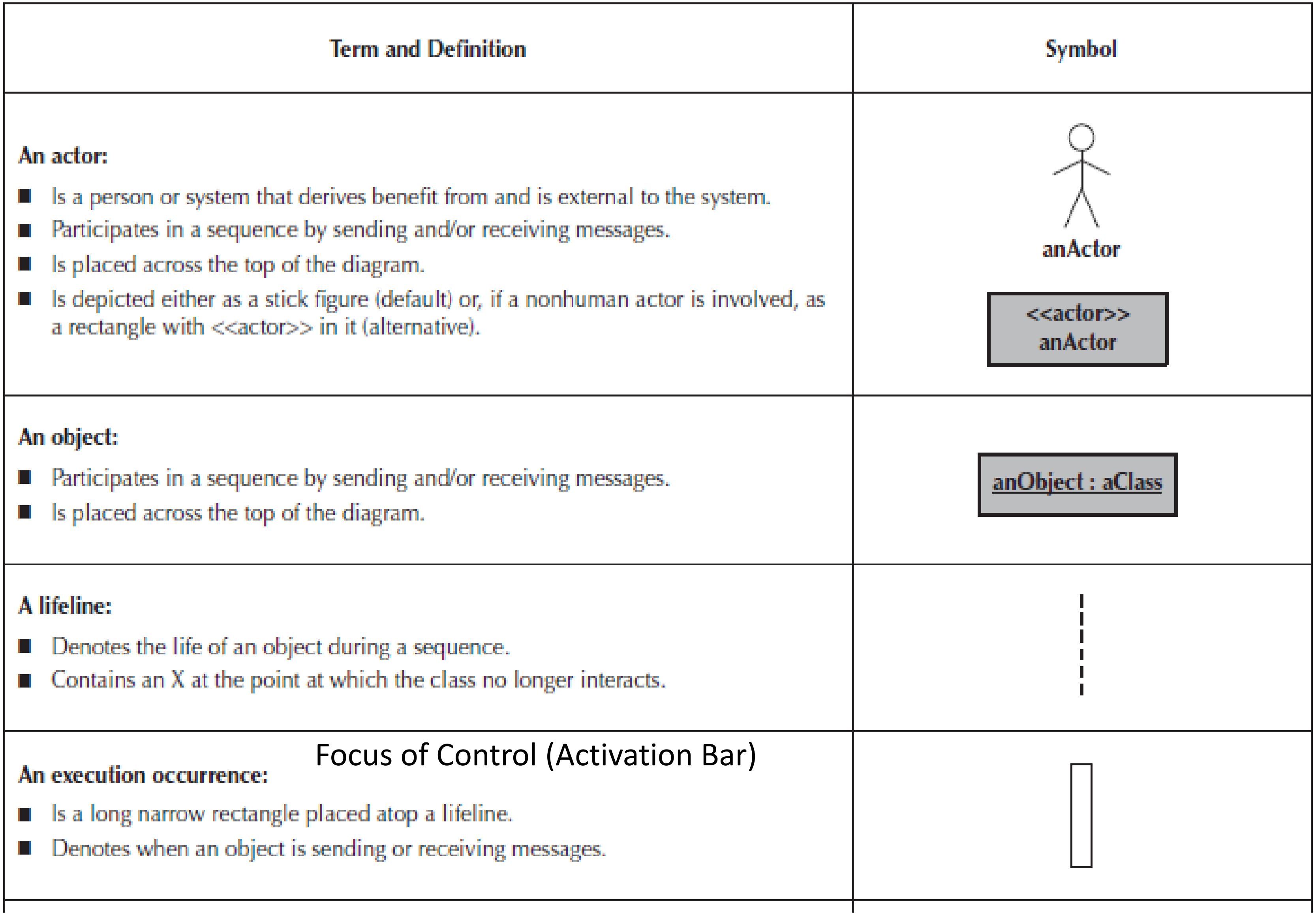
- The flow of events for a use case is captured in text, whereas scenarios are captured in Interaction diagrams.
- UML 1.x uses two types of Interaction Diagrams
  - **Sequence Diagrams,**
  - **Collaboration/Communication Diagrams**
- UML 2.0 introduces 2 more.
  - **Timing Diagrams,**
  - **Interaction overview Diagrams**
- Each Diagram is a graphical view of the scenario typically associated with Use Cases in the mode

7.2 Sequence Diagrams

Sequence diagrams shows the

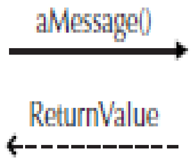
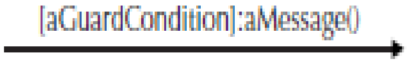
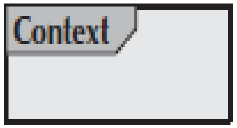
- object interactions arranged in time sequence.
- objects and classes involved in the scenario.
- sequence of messages exchanged between the objects needed to carry out the functionality of the scenario.

Elements of a Sequence Diagram



Term and Definition	Symbol
An actor: ■ Is a person or system that derives benefit from and is external to the system. ■ Participates in a sequence by sending and/or receiving messages. ■ Is placed across the top of the diagram. ■ Is depicted either as a stick figure (default) or, if a nonhuman actor is involved, as a rectangle with <<actor>> in it (alternative).	anActor <<actor>> anActor
An object: ■ Participates in a sequence by sending and/or receiving messages. ■ Is placed across the top of the diagram.	anObject : aClass
A lifeline: ■ Denotes the life of an object during a sequence. ■ Contains an X at the point at which the class no longer interacts.	
An execution occurrence: Focus of Control (Activation Bar) ■ Is a long narrow rectangle placed atop a lifeline. ■ Denotes when an object is sending or receiving messages.	

Elements of a Sequence Diagram cont…

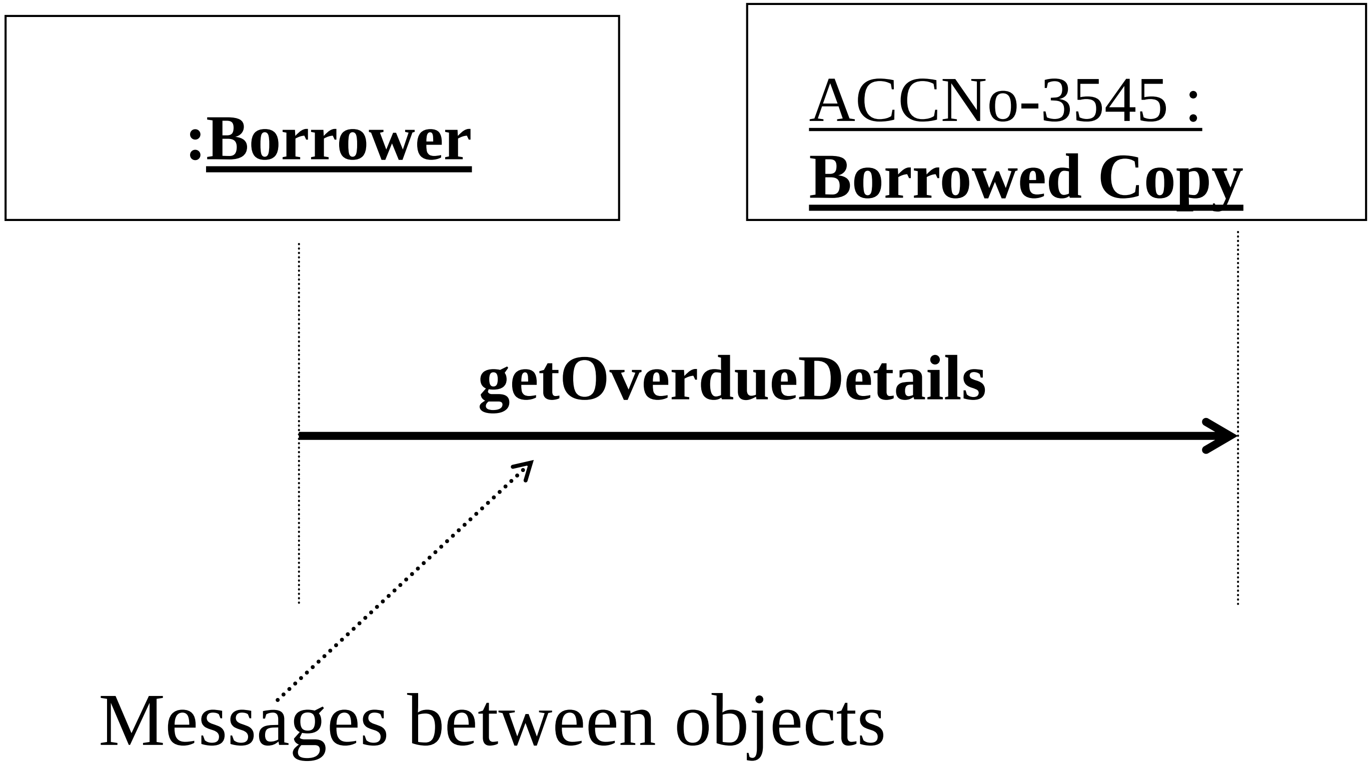
A message: ■ Conveys information from one object to another one. ■ A operation call is labeled with the message being sent and a solid arrow, whereas a return is labeled with the value being returned and shown as a dashed arrow.	
A guard condition: ■ Represents a test that must be met for the message to be sent.	
For object destruction: ■ An X is placed at the end of an object's lifeline to show that it is going out of existence.	X
A frame: ■ Indicates the context of the sequence diagram.	

Creating Sequence Diagrams

- In UML, an object in a sequence diagram is drawn as a rectangle, containing the name of the object, underlined.
- An object can be named in one of the three ways:
  - Object name,
  - Object name and its class,
  - Class name (anonymous object)

Creating Sequence Diagrams cont..

- UML Notation for objects and messages in a sequence diagrams is shown below:



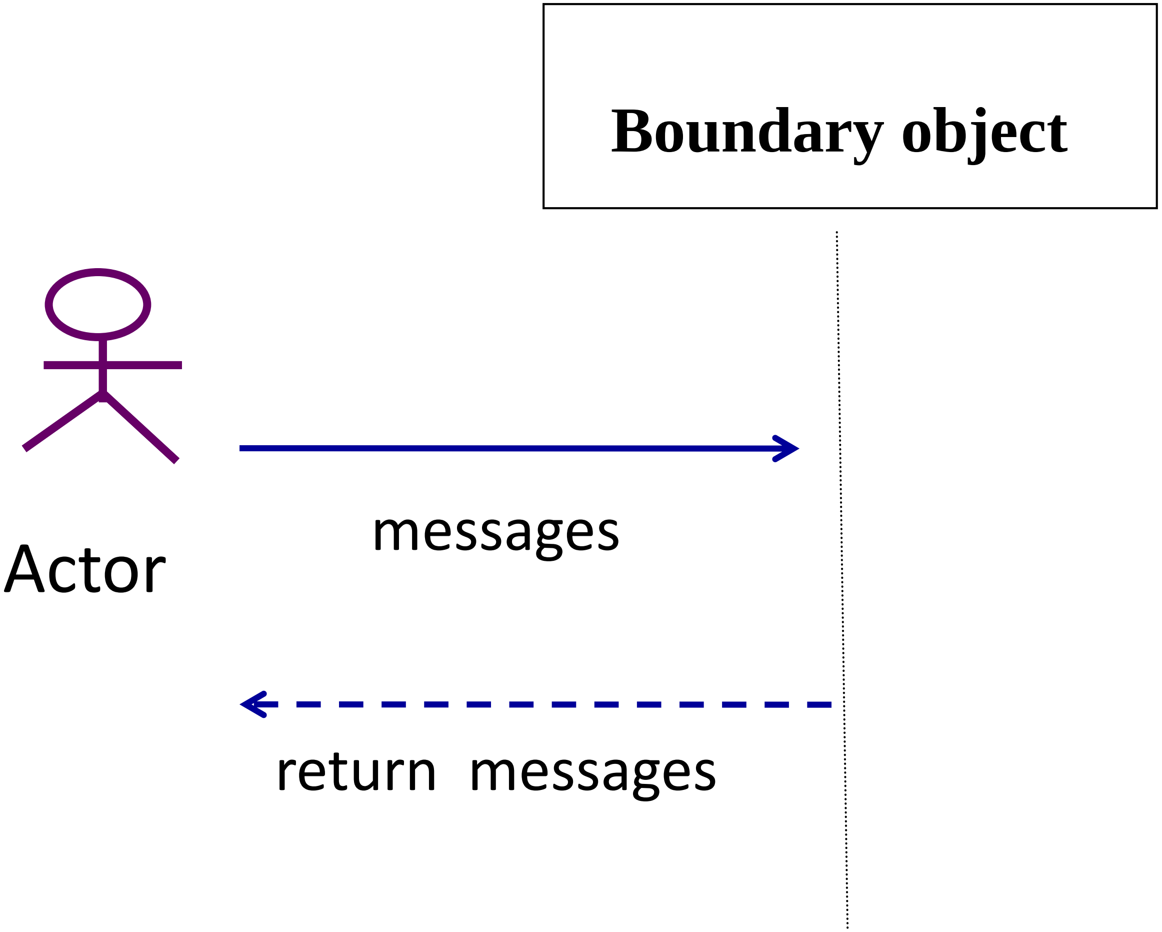
Sequence Diagrams and Boundary classes

- Boundary classes are added to sequence diagrams to show the interaction with the user or another system.
- During the early analysis phases, boundary classes are shown on a sequence diagram only to capture and document the interface requirements.
- Actual messages from the actor to boundary class with their sequencing information will depend on the application framework that will be selected later in development.

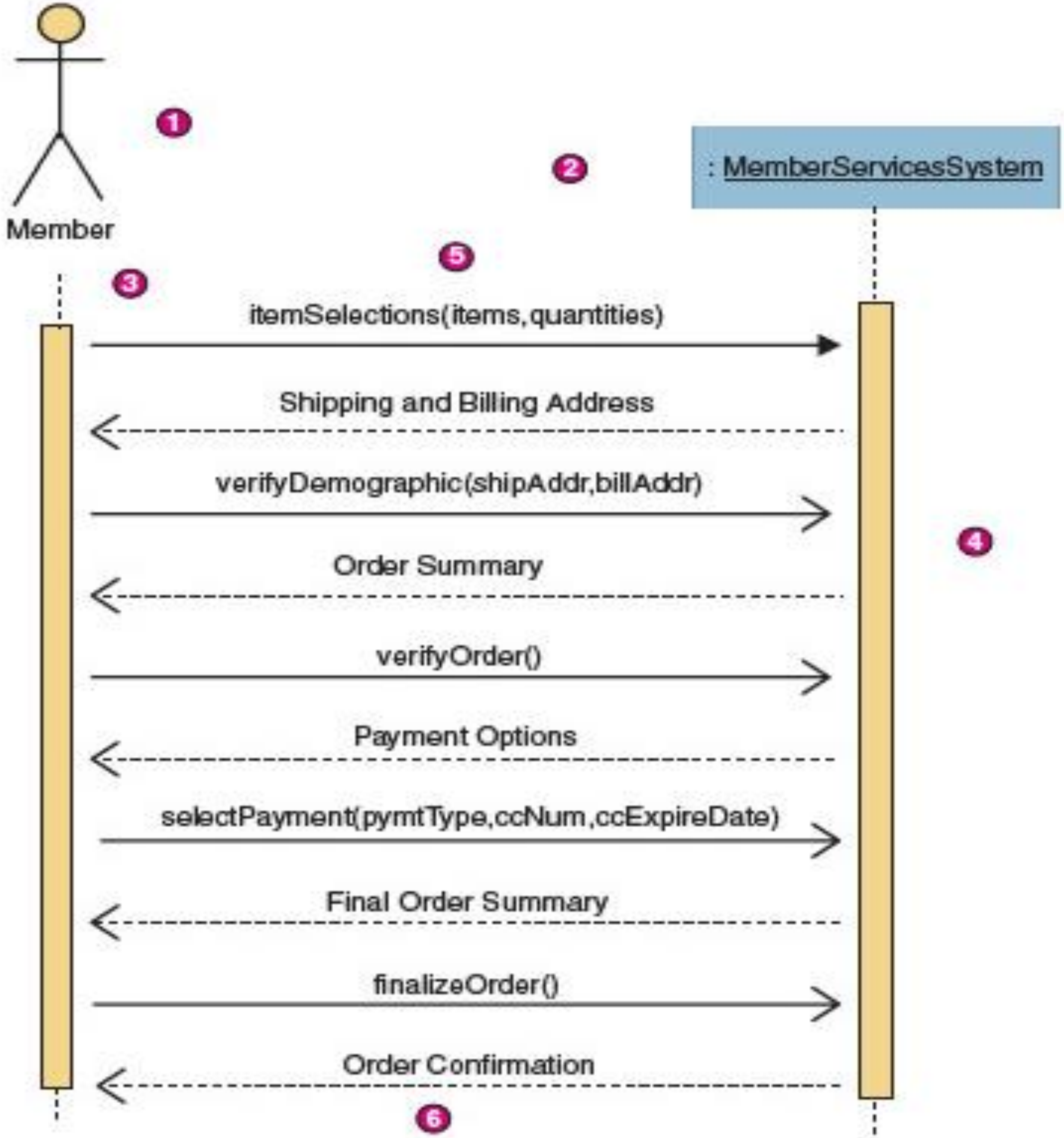
System Sequence Diagrams

- The system sequence diagram is a tool used by some Analysts in logical design phase.
- The system sequence diagram helps to identify the high-level messages that enter and exit the system.
    - Later these messages will become the responsibility of individual objects.
    - These individual objects will fulfill those responsibilities by communicating with other objects.

System Sequence Diagrams



e.g.


Member
1
2
: MemberServicesSystem
5
3
itemSelections(items,quantities)
Shipping and Billing Address
verifyDemographic(shipAddr,billAddr)
4
Order Summary
verifyOrder()
Payment Options
selectPayment(pymtType,ccNum,ccExpireDate)
Final Order Summary
finalizeOrder()
Order Confirmation
6

Use Case Narratives for Borrowing Scenario: Library System

e.g. Borrowing Scenario Flow of Events

Main Flow

Librarian enters the borrower id.

System checks whether borrower id exist.

If not exist (E-1) end use case.

else **Process**

Check for Overdue Books.

If yes (E-2) end use case

Check Over limit (E-3)

If yes (E-3) end use case

Use Case Narratives cont..

Enter copy id

Check Borrow able (E-4)

  If No (E-4) end use case

Librarian Confirm Borrowing (C-5)

Update Borrowed Copy details

Alternate Flows

E-1 : Borrower id exist

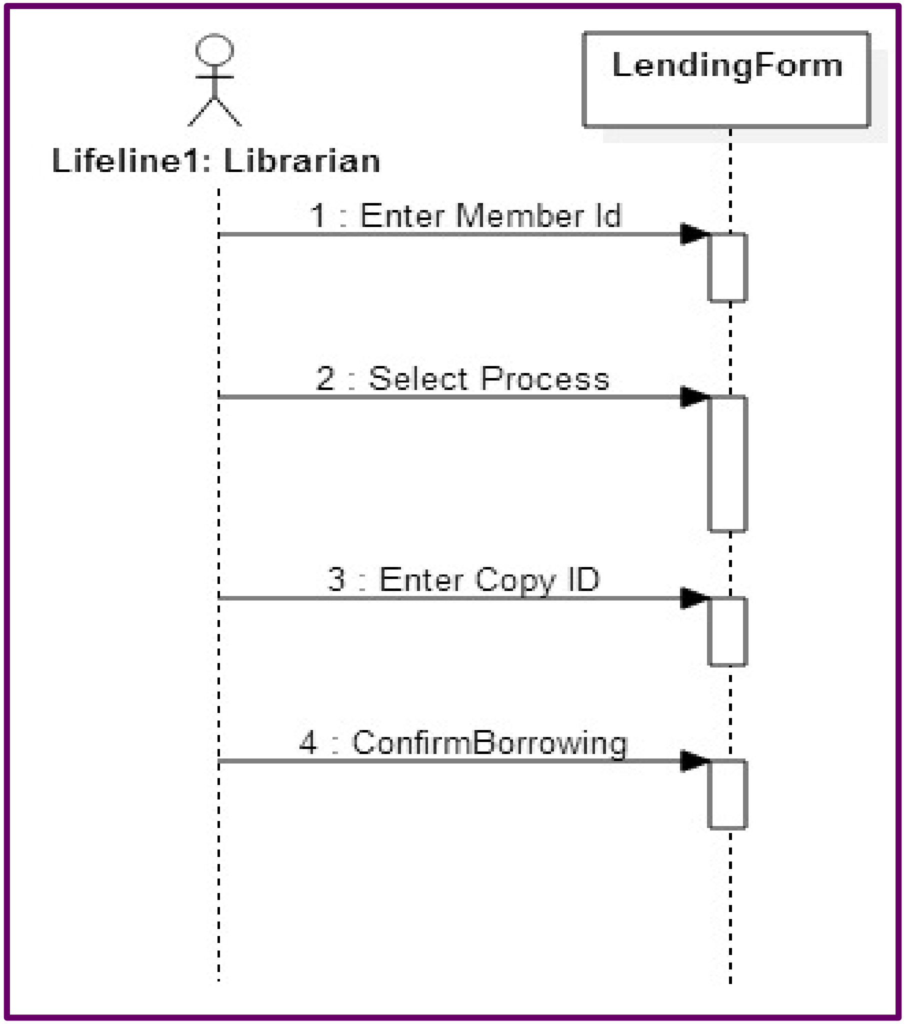
E-2 : There are overdue books

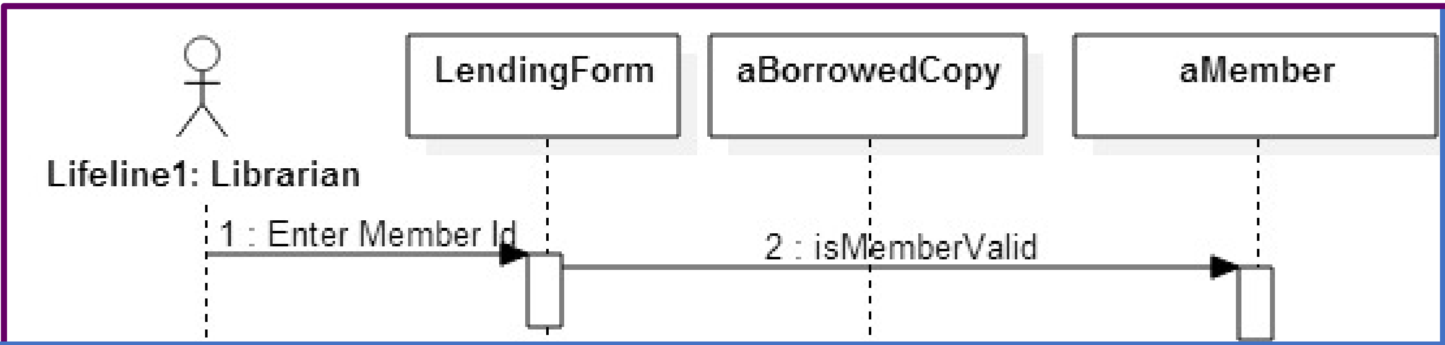
E-3 : Borrower has already borrowed 5 books (max)

E-4 : Copy is not borrow able

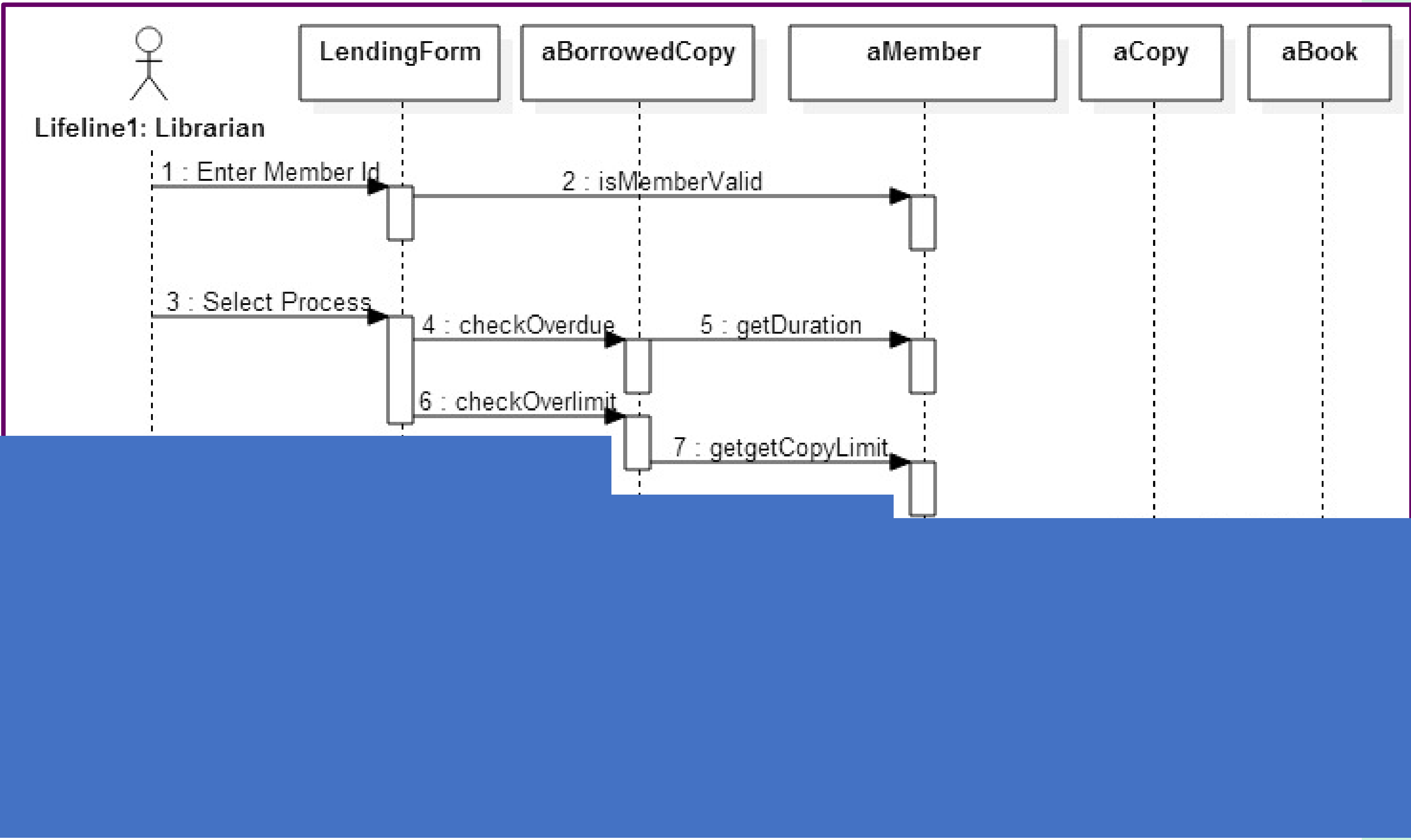
C-1 : Confirm borrowing Message box

Initial Steps in Drawing Sequence Diagram for Borrowing Scenario

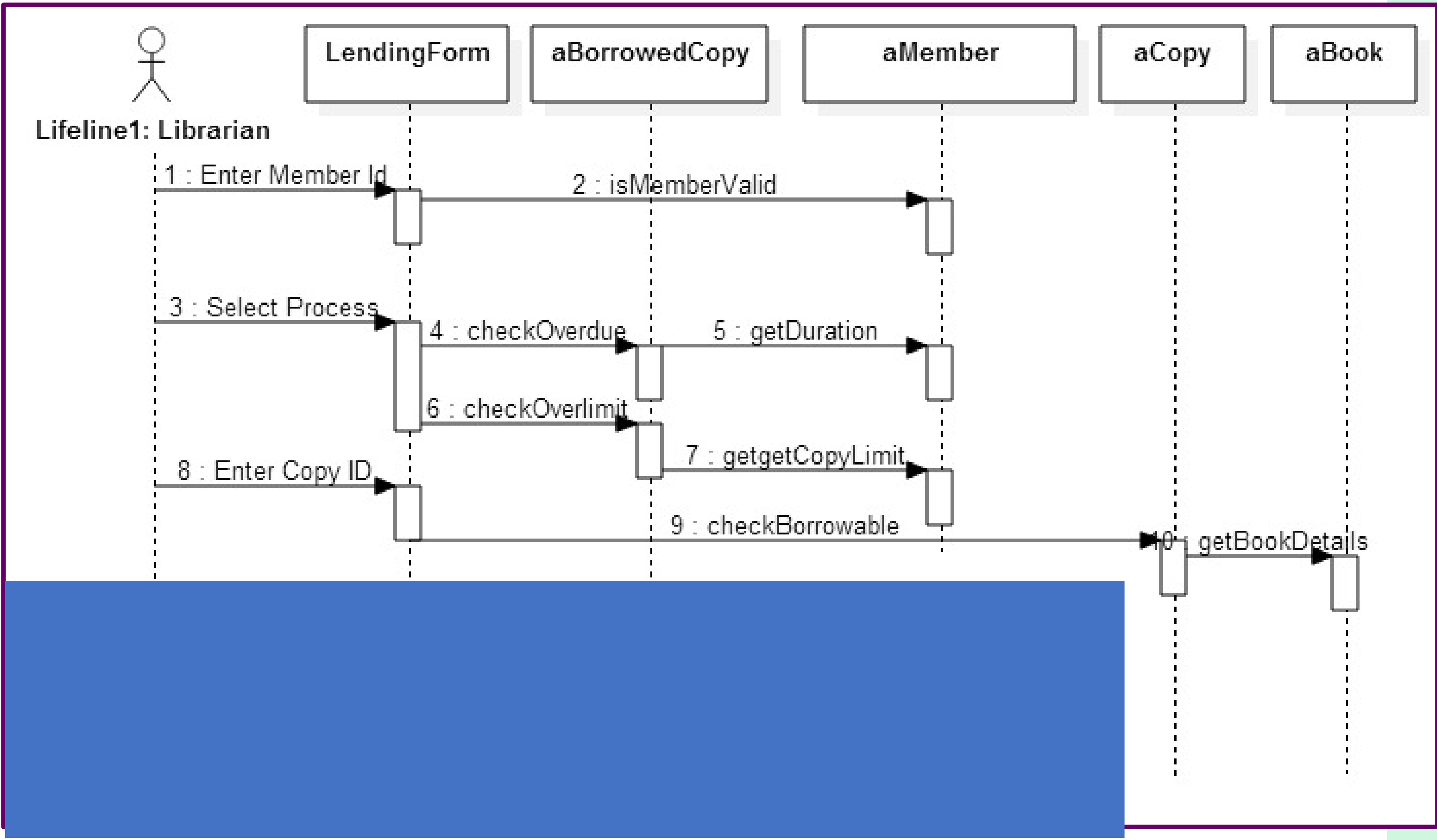




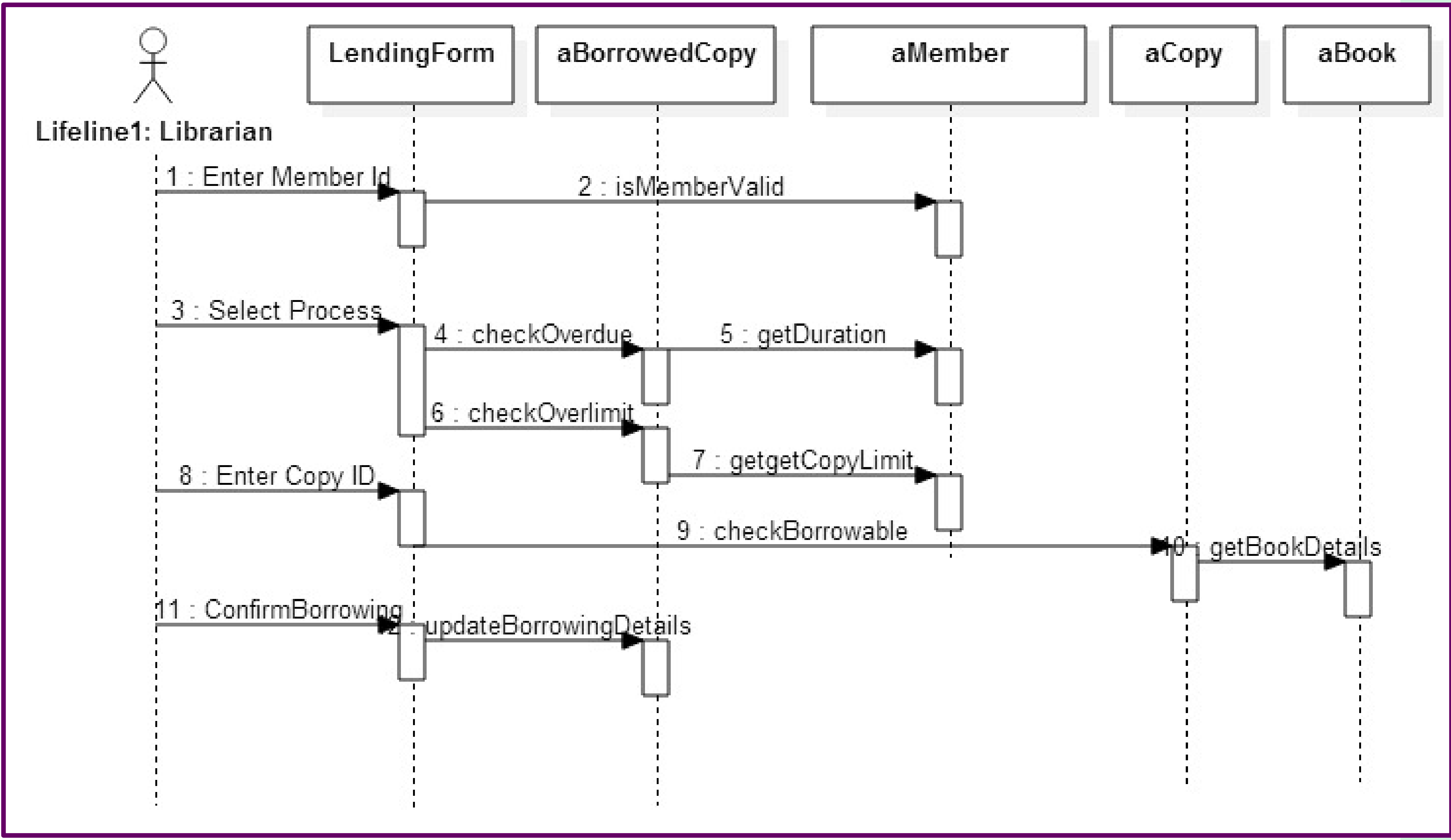
Borrowing Sequence Diagram **with objects and messages**



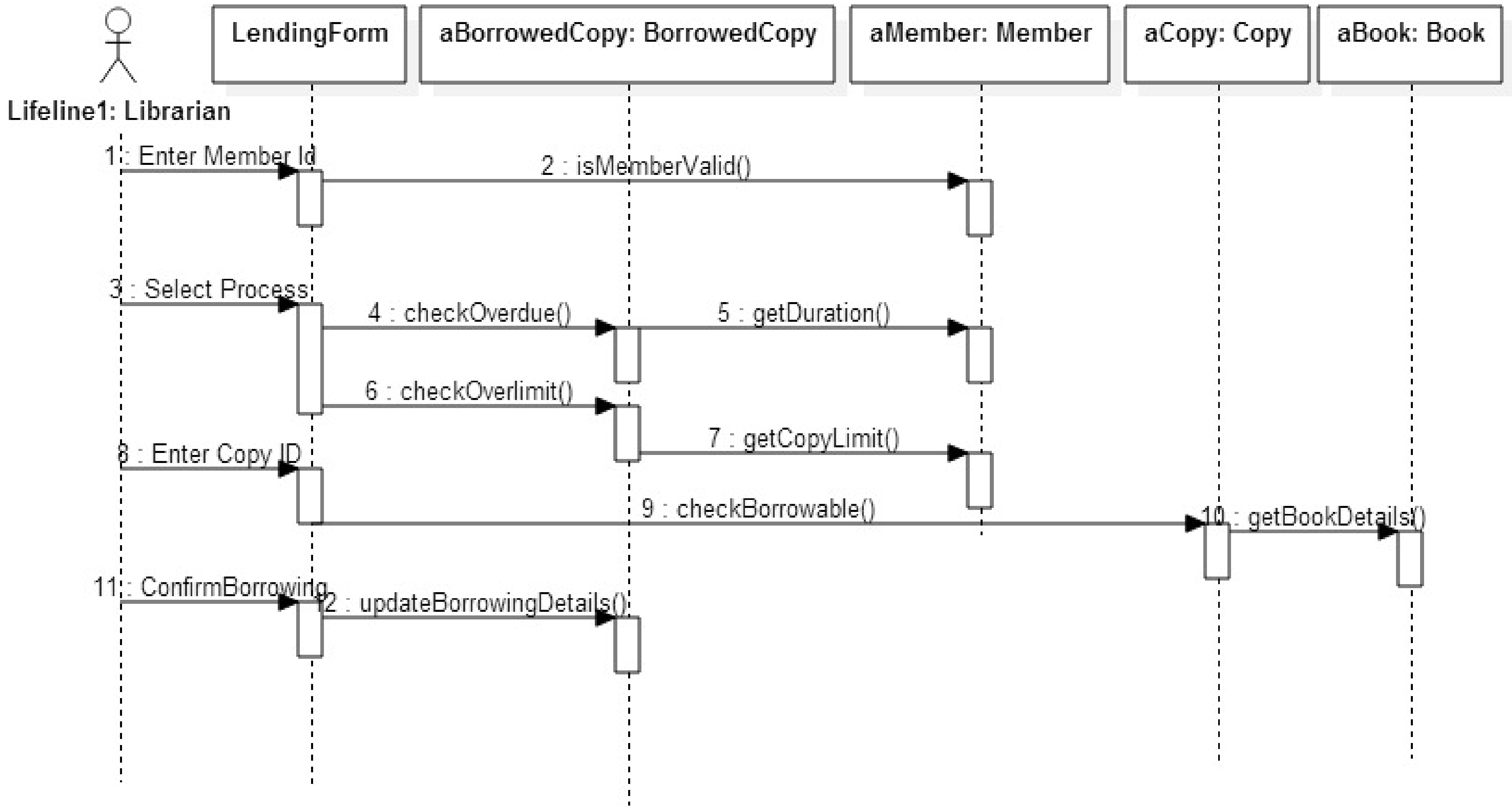
Borrowing Sequence Diagram with objects and messages



Borrowing Sequence Diagram with objects and messages



Borrowing Sequence Diagram with objects and messages



Borrowing Sequence Diagram

After mapping objects to classes and messages to methods

Sequence Diagrams cont..

- Sequence diagrams should be kept as simple as possible then
- it is easy to see:
  - the objects,
  - object interactions,
  - the messages between the objects, and
  - the functionality captured by the scenario.

Sequence Diagrams cont..

How to handle conditional logic in a sequence diagram? (*if, then*, *else* logic that exists in the real world)

- If the logic is simple, involving only a few messages:
  - add the logic to one diagram and use notes to communicate the choices to be made.
- If the logic is complex, involves complicated messages,
  - draw a separate diagram- one for the *if* case, one for the *then* case, and one for the *else* case.

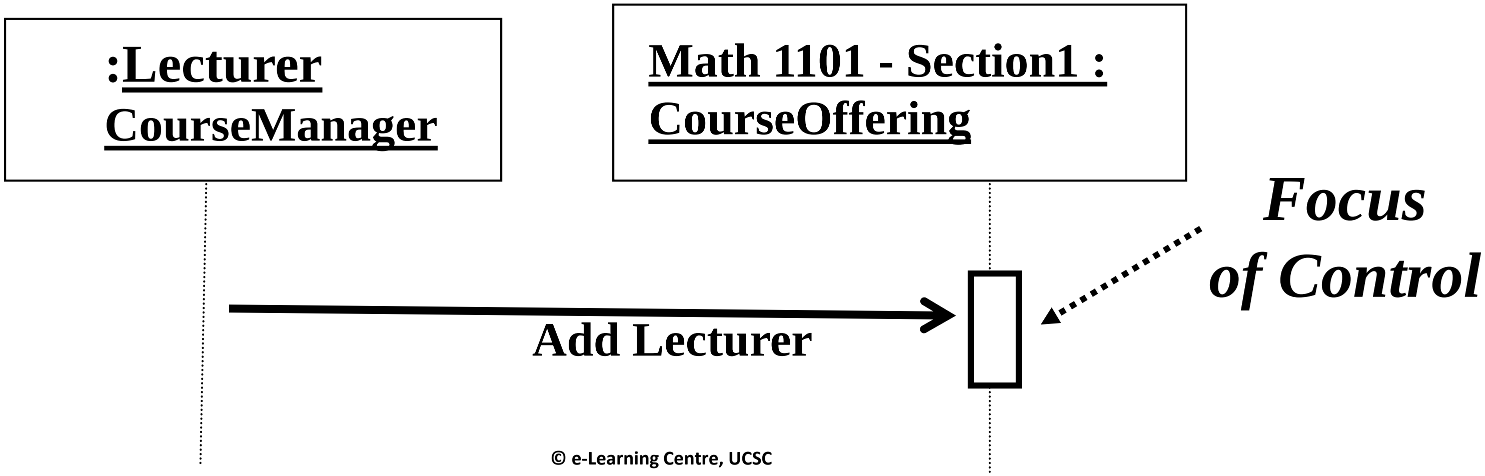
This is done to keep the diagrams simple as possible.

Sequence Diagrams cont..

- In most I-CASE tools, diagrams may be linked to one another.
- This allows the user to navigate through a set of diagrams.

Focus of Control (Activation Bar)

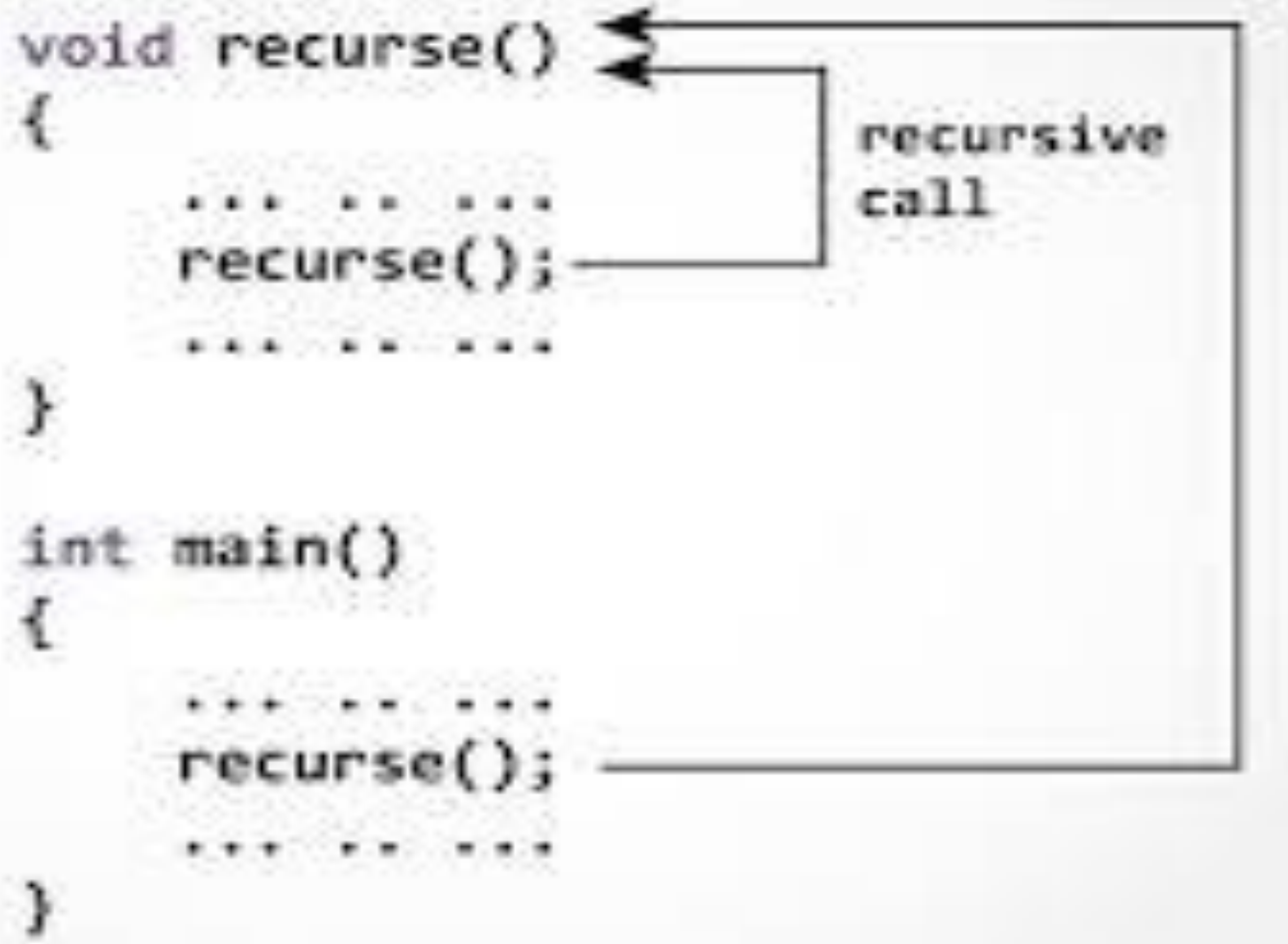
- The focus of control is a small rectangle, that will let you know which object has control at a particular point in time.
- This is one of the differences between Sequence and a Communication diagram.
- The focus of control is shown only on a sequence diagram. (optional)



Self Message(non-recursive) vs Self Recursive Message

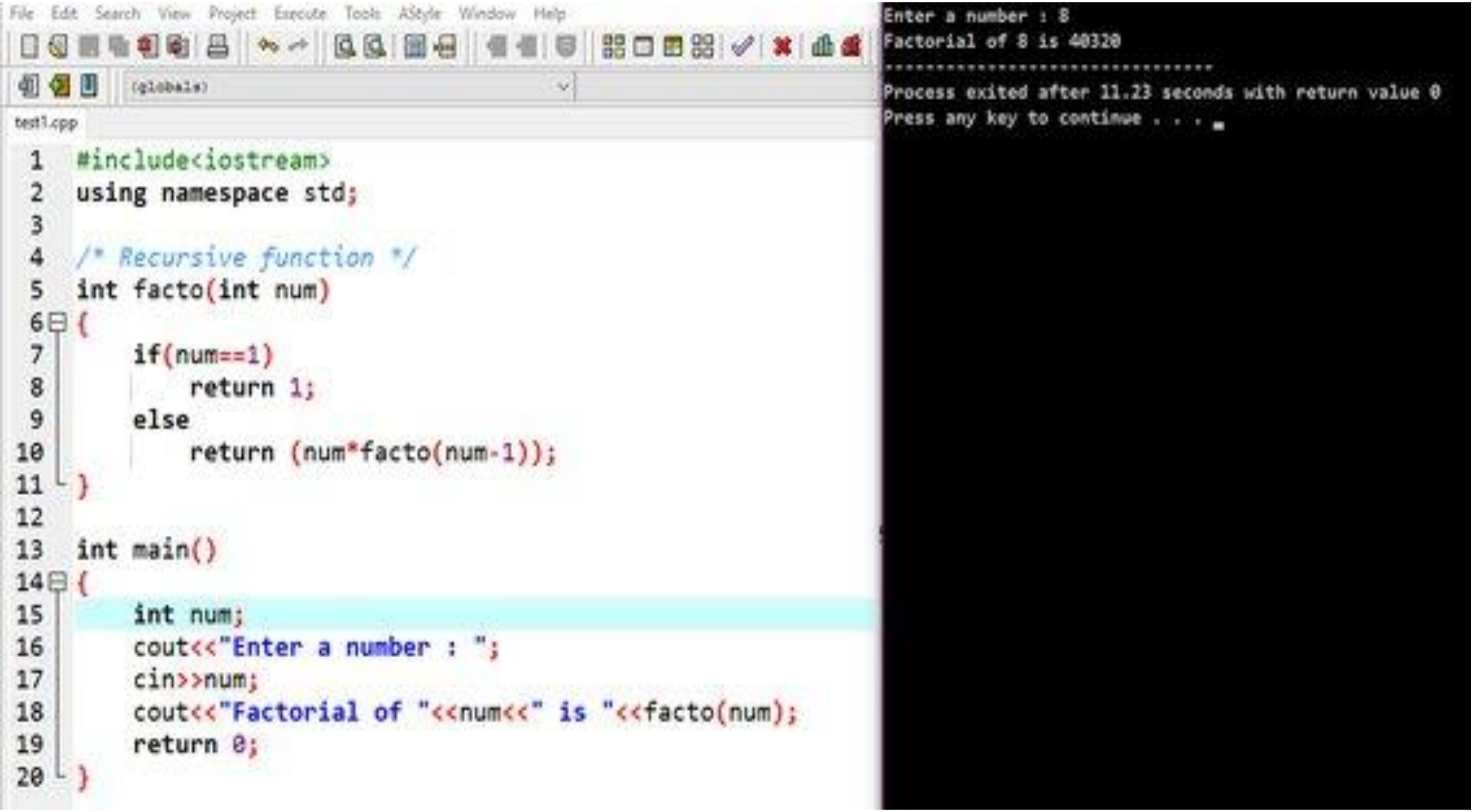
- A self message is a message that an object sends to itself.
- It is a message that represents the invocation of message of the same lifeline.
- A self message can represent
  - **a recursive call of an operation**, or
  - **one method calling another method belonging to the same object**.

How does Recursion work?



```
void recurse()
{
    ... .. ...
    recurse();
    ... .. ...
}

int main()
{
    ... .. ...
    recurse();
    ... .. ...
}
```

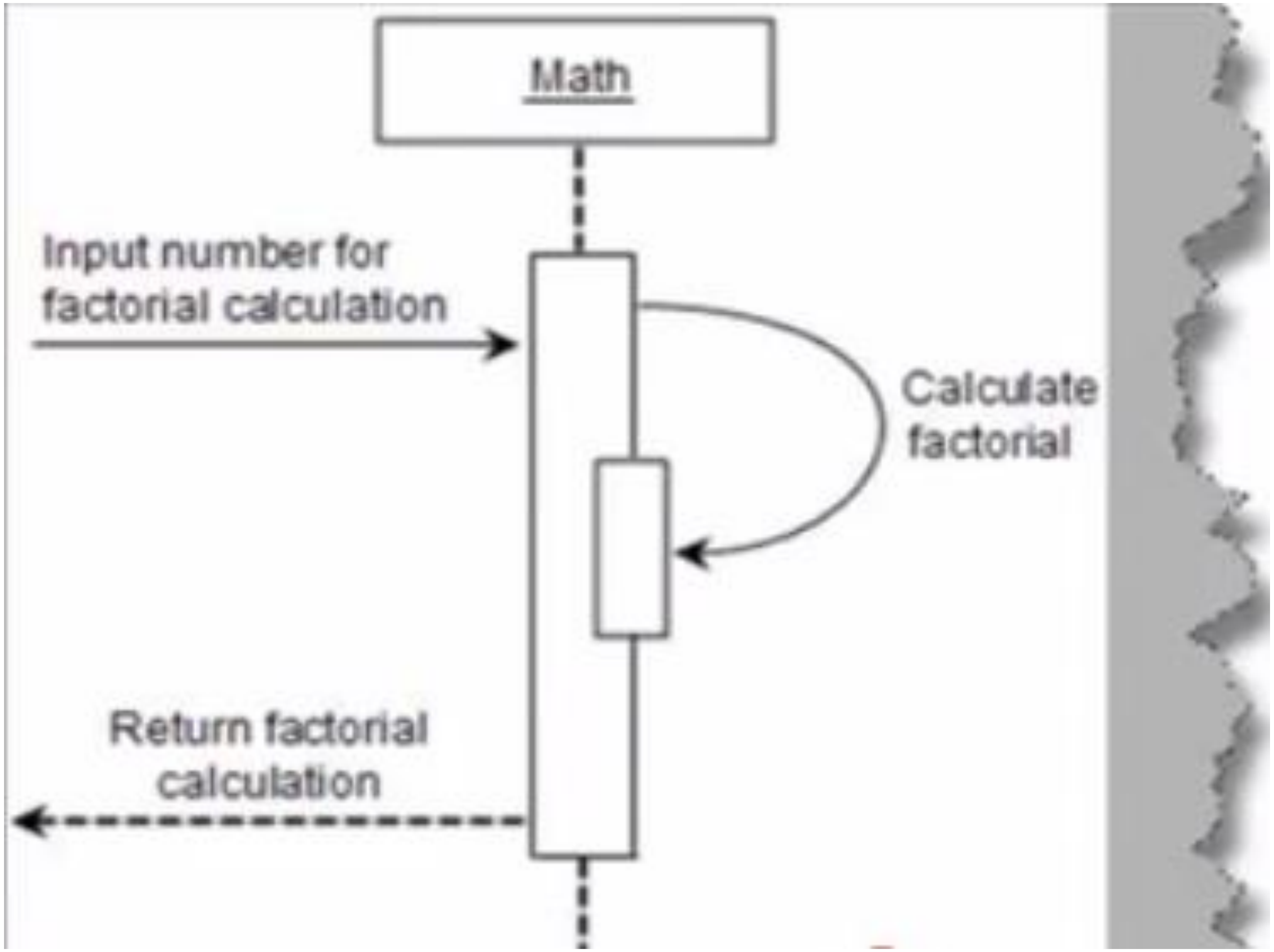
Self Message (Recursive) C++ example


File Edit Search View Project Execute Tools AStyle Window Help
(globals)
test1.cpp

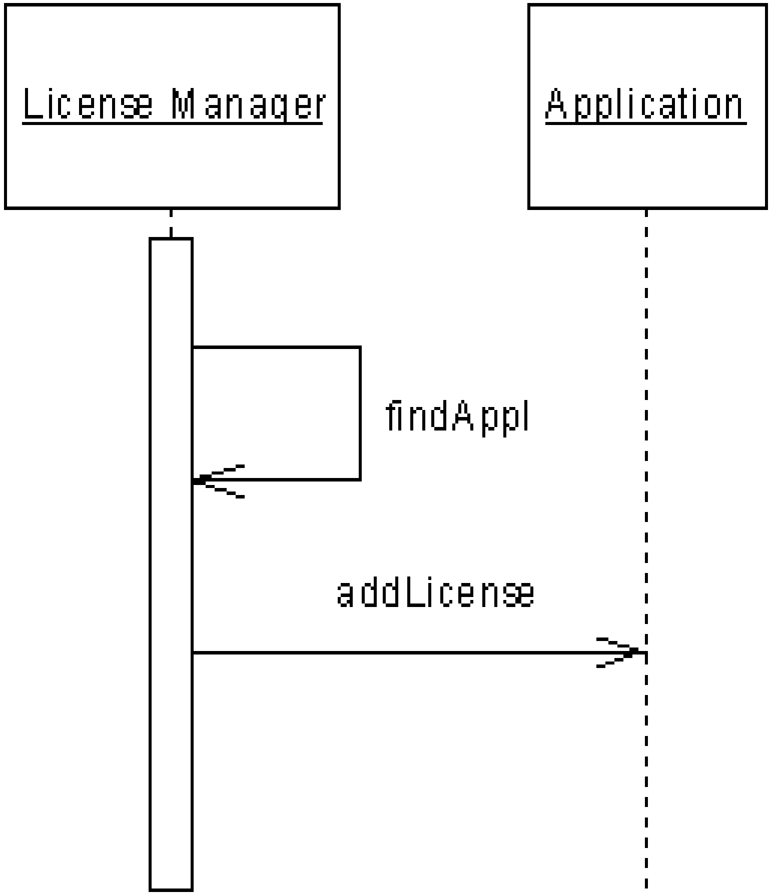
```cpp
#include<iostream>
using namespace std;

/* Recursive function */
int facto(int num)
{
    if(num==1)
        return 1;
    else
        return (num*facto(num-1));
}

int main()
{
    int num;
    cout<<"Enter a number : ";
    cin>>num;
    cout<<"Factorial of "<<num<<" is "<<facto(num);
    return 0;
}
```

```
Enter a number : 8
Factorial of 8 is 40320
--------------------------------
Process exited after 11.23 seconds with return value 0
Press any key to continue . . .
```

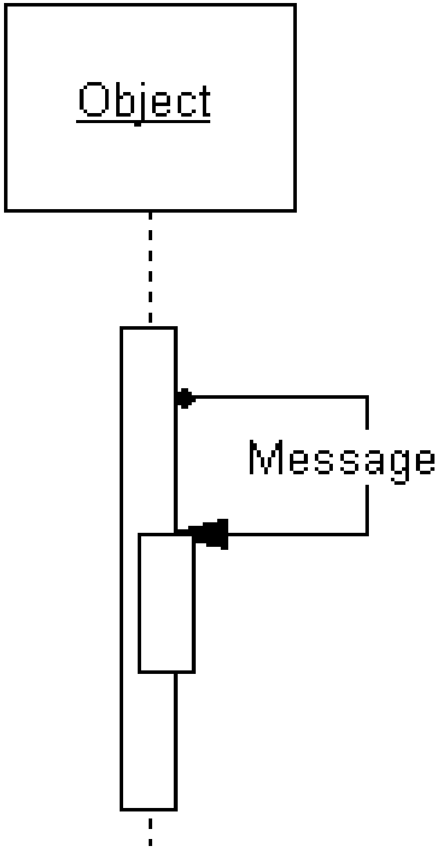
Self Message (Recursive)


Math
Input number for factorial calculation
Calculate factorial
Return factorial calculation

Self Message(non recursive) vs Self Recursive Message

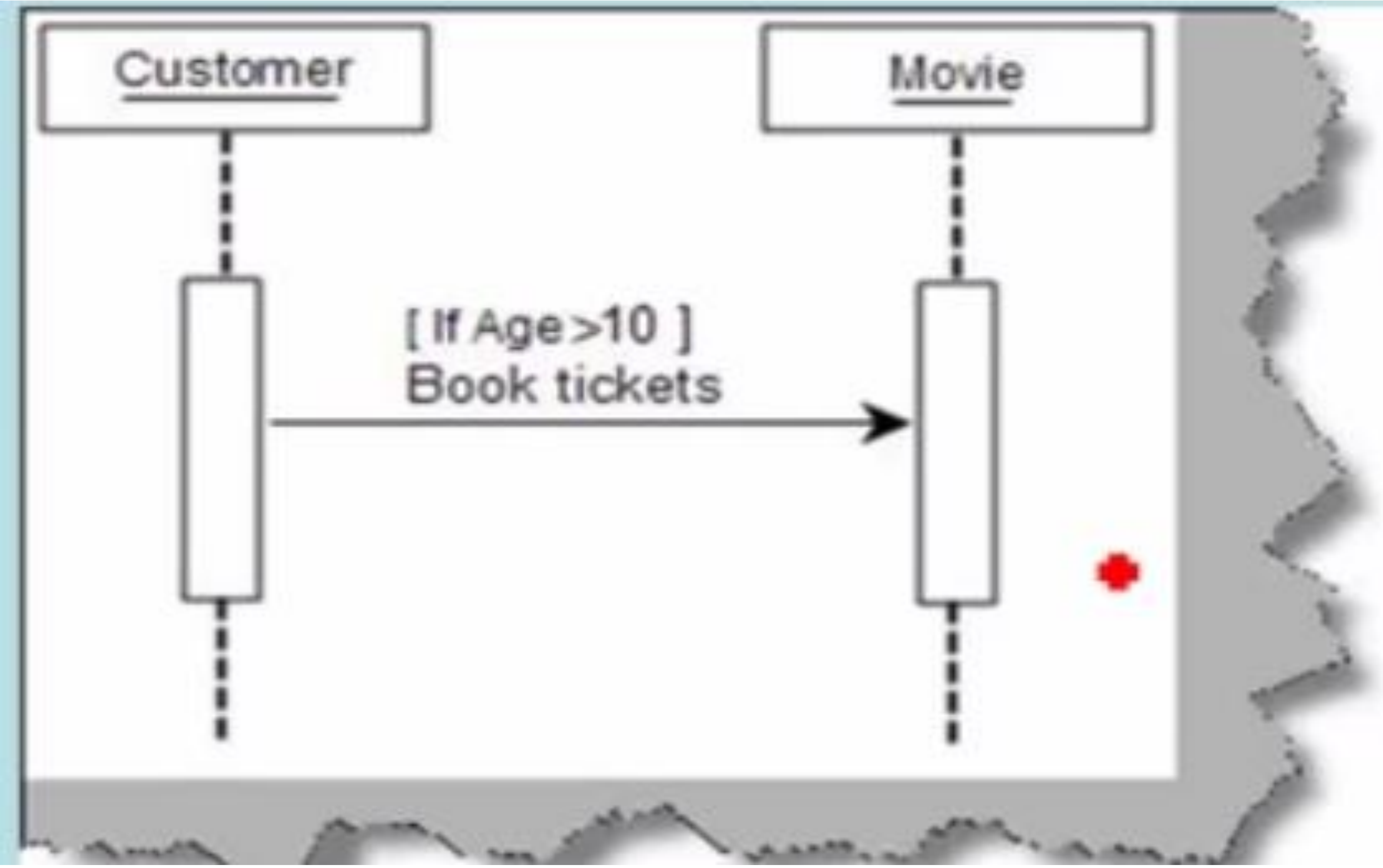


Non-Recursive



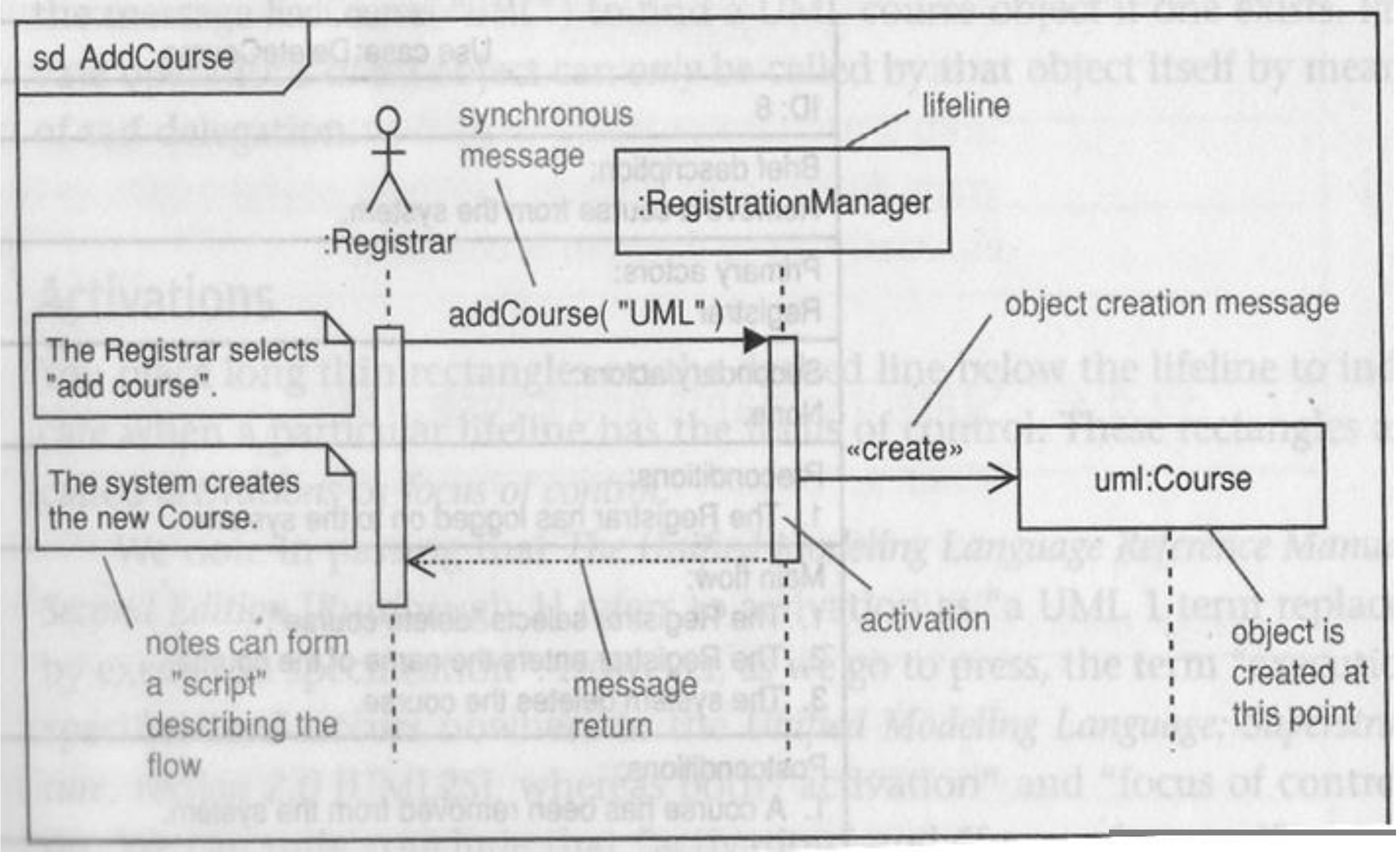
Recursive

Message Constraints



Message interactions happen only if the condition is true

E.g. Sequence Diagram: Add Details of a new Course



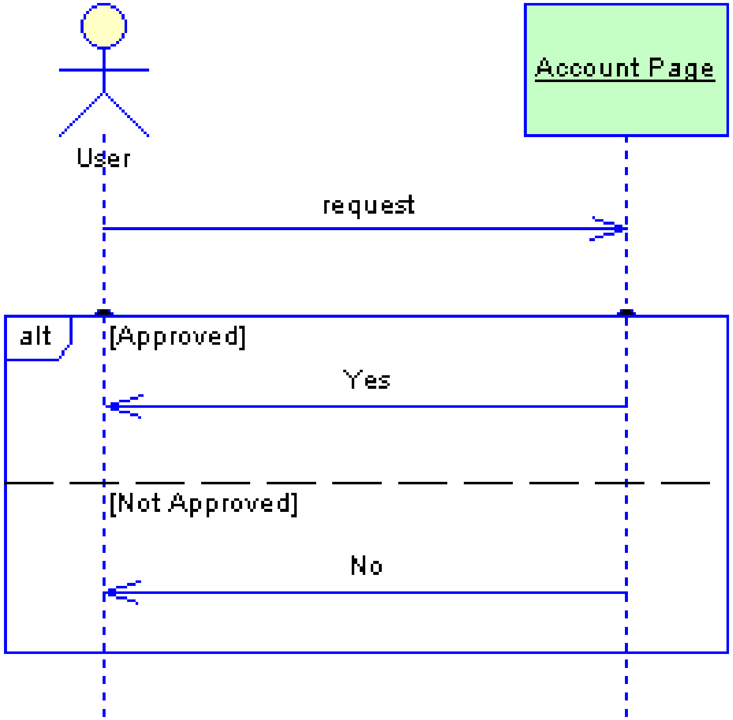
Framing in Sequence Diagrams

- Frame is a UML 2.* feature.
- One can frame a sequence diagram by surrounding it with a border and adding a compartment in the upper left corner.
- The compartment contains information that identifies the diagram.
- These interaction fragments can be combined.
- Gives you a quick and easy way to reuse part of one sequence diagram in another.

Interaction Fragment

Interaction Fragment

- An interaction fragment allows you to group related messages in a sequence diagram.



The User sends a request to the Account Page.
The **two alternative responses** and the **conditions** on which they depend, are contained within an interaction fragment.

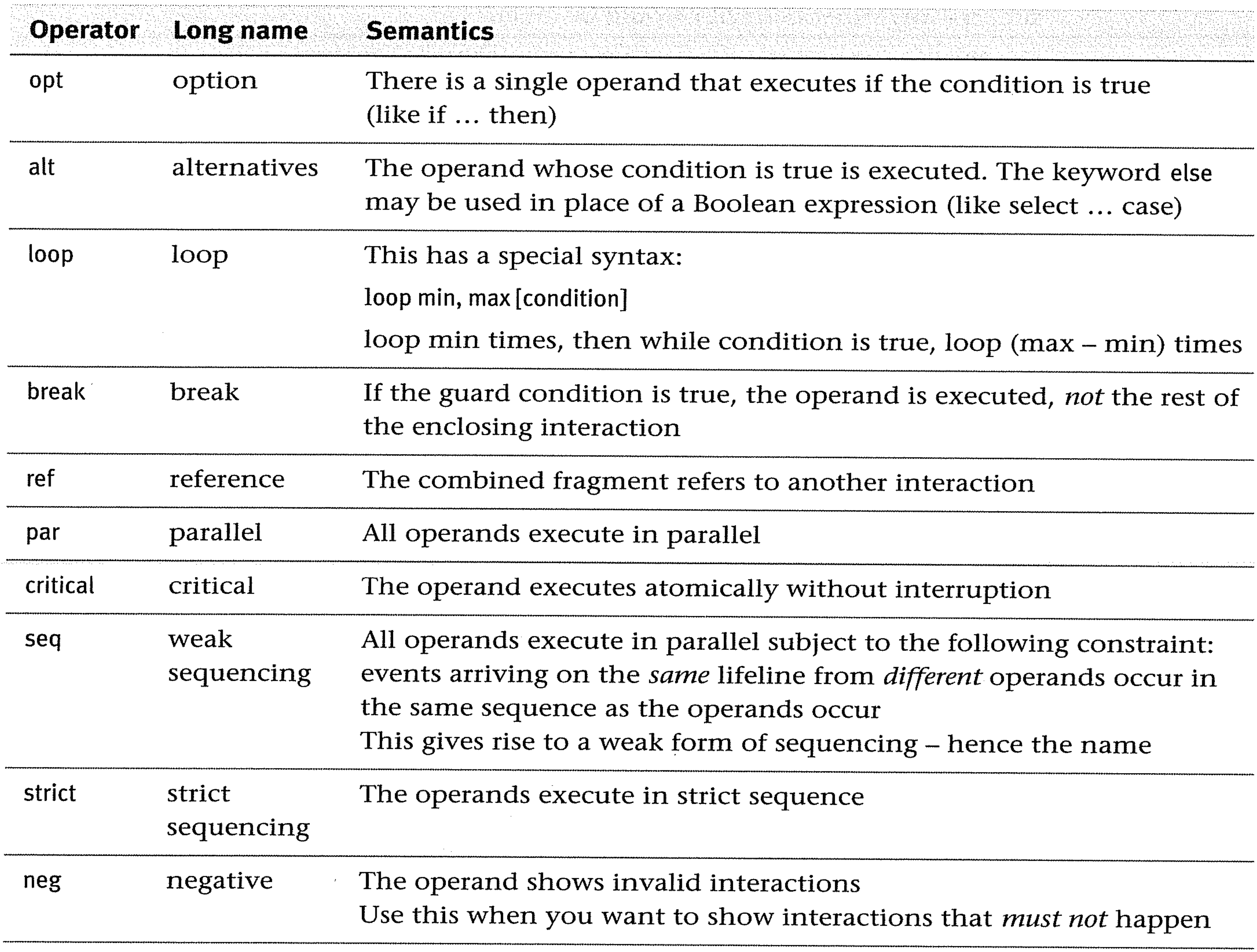
Combined Fragment

Combined Fragment

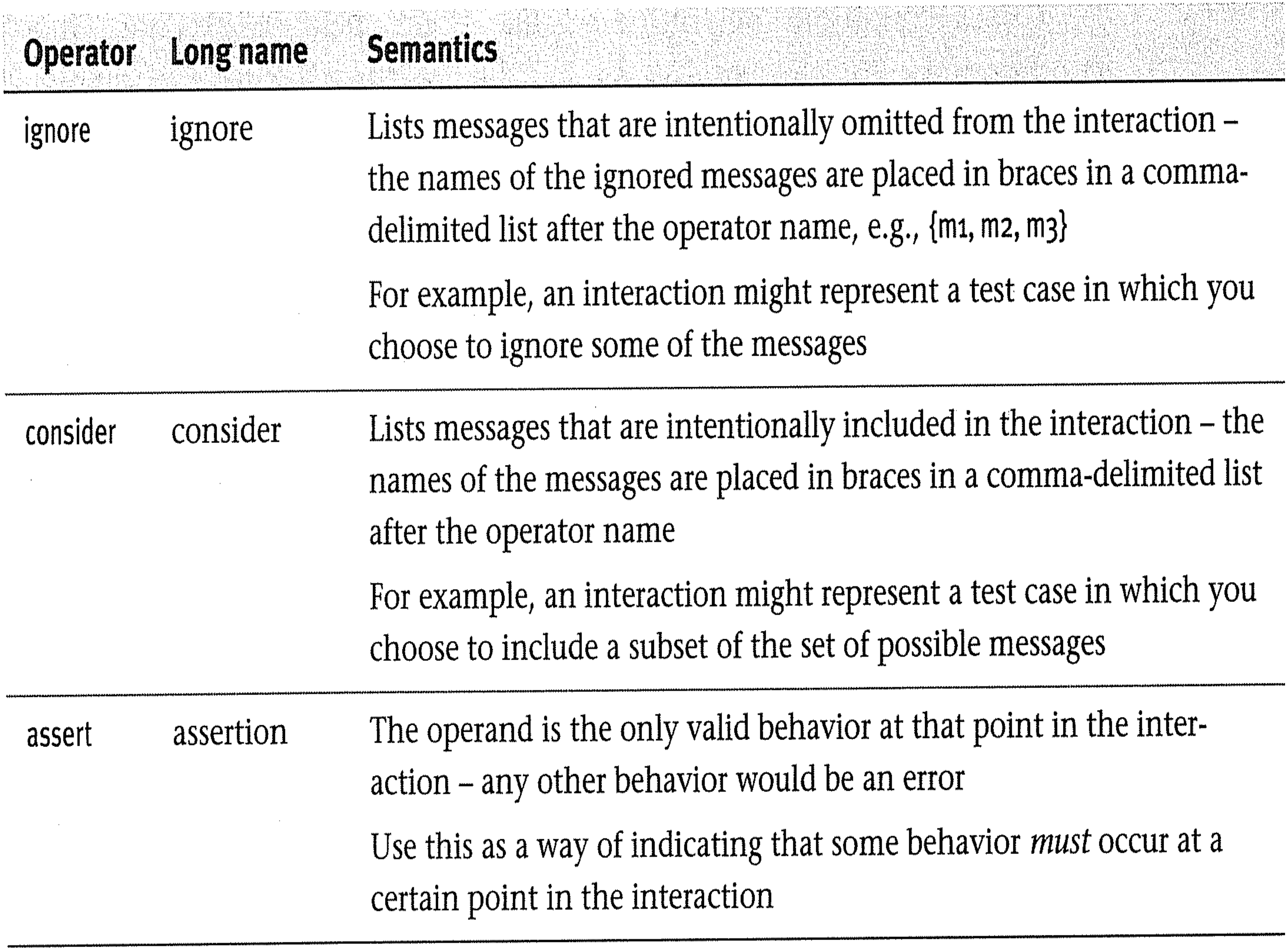
- is a subtype of interaction fragment.
- defines an expression of interaction fragments.
- defines by an **interaction operator** and corresponding **interaction operands.**

Combined Fragment: Interaction Operators

- A combined fragment defines an expression of interaction fragments
- The following operators are commonly used in a combined fragment expression
  - Alt
  - Opt
  - Par
  - Loop



Operator	Long name	Semantics
opt	option	There is a single operand that executes if the condition is true (like if … then)
alt	alternatives	The operand whose condition is true is executed. The keyword else may be used in place of a Boolean expression (like select … case)
loop	loop	This has a special syntax: loop min, max [condition] loop min times, then while condition is true, loop (max – min) times
break	break	If the guard condition is true, the operand is executed, *not* the rest of the enclosing interaction
ref	reference	The combined fragment refers to another interaction
par	parallel	All operands execute in parallel
critical	critical	The operand executes atomically without interruption
seq	weak sequencing	All operands execute in parallel subject to the following constraint: events arriving on the *same* lifeline from *different* operands occur in the same sequence as the operands occur This gives rise to a weak form of sequencing – hence the name
strict	strict sequencing	The operands execute in strict sequence
neg	negative	The operand shows invalid interactions Use this when you want to show interactions that *must not* happen



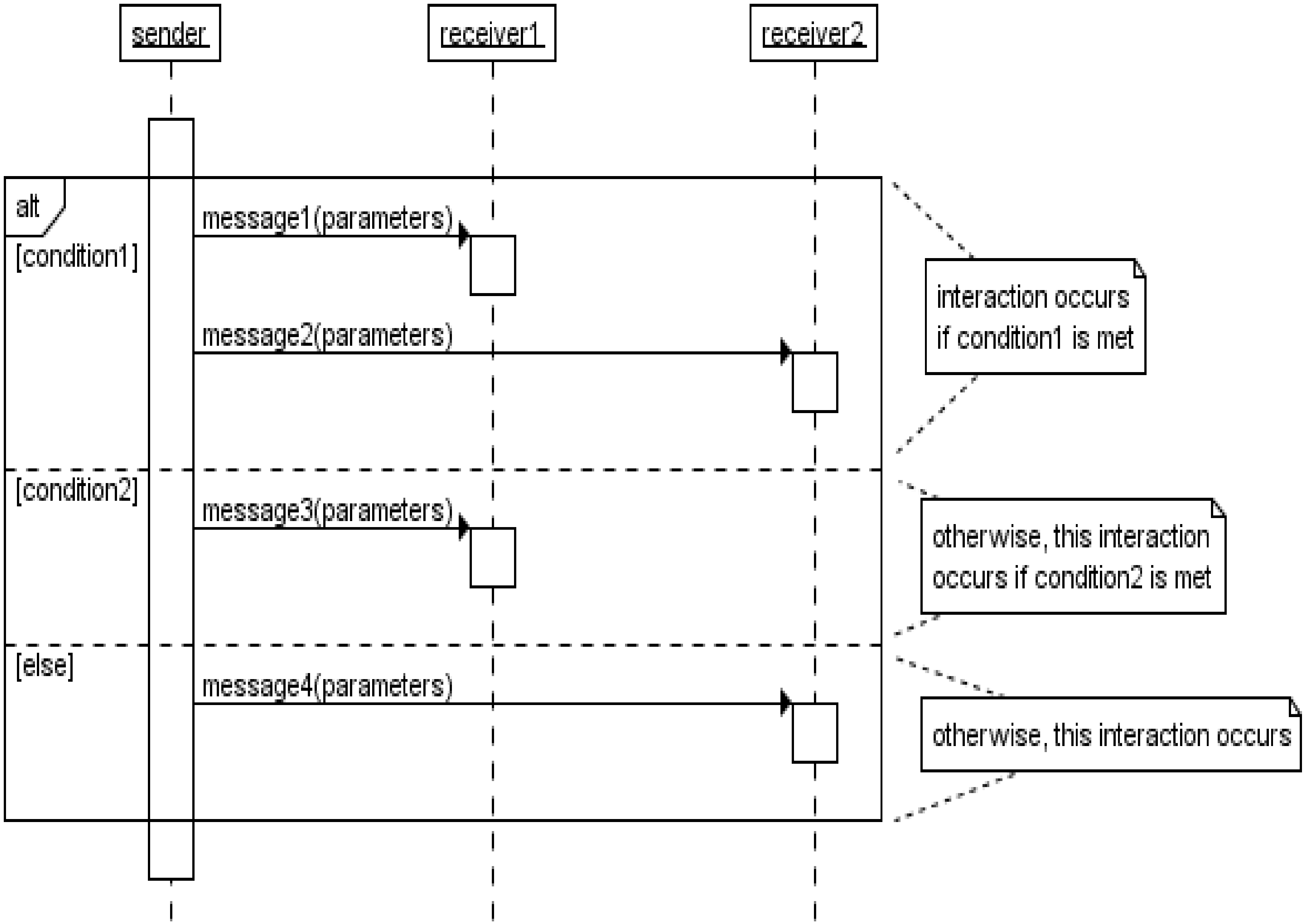
Operator	Long name	Semantics
ignore	ignore	Lists messages that are intentionally omitted from the interaction – the names of the ignored messages are placed in braces in a comma-delimited list after the operator name, e.g., {m1, m2, m3} For example, an interaction might represent a test case in which you choose to ignore some of the messages
consider	consider	Lists messages that are intentionally included in the interaction – the names of the messages are placed in braces in a comma-delimited list after the operator name For example, an interaction might represent a test case in which you choose to include a subset of the set of possible messages
assert	assertion	The operand is the only valid behavior at that point in the interaction – any other behavior would be an error Use this as a way of indicating that some behavior *must* occur at a certain point in the interaction

Alt and Else Operators

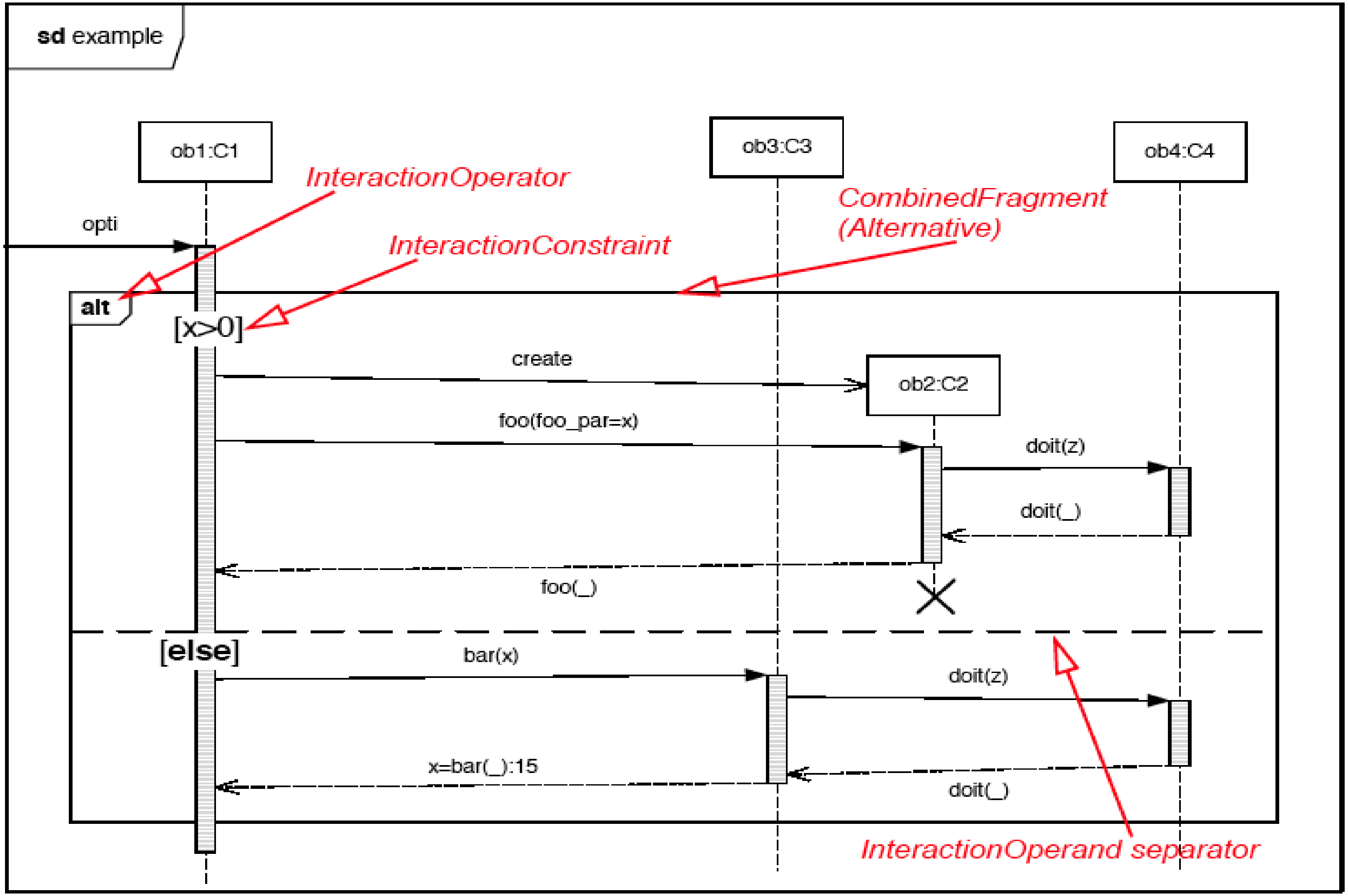
- The interaction operator alt designates that the combined fragment represents a choice of behavior.
    - At most one of the operands will be chosen.
    - The chosen operand must have an explicit or implicit guard expression that evaluates to true at this point in the interaction.
    - An implicit true guard is implied if the operand has no guard.
    - The set of traces that defines a choice is the union of the (guarded) traces of the operands.

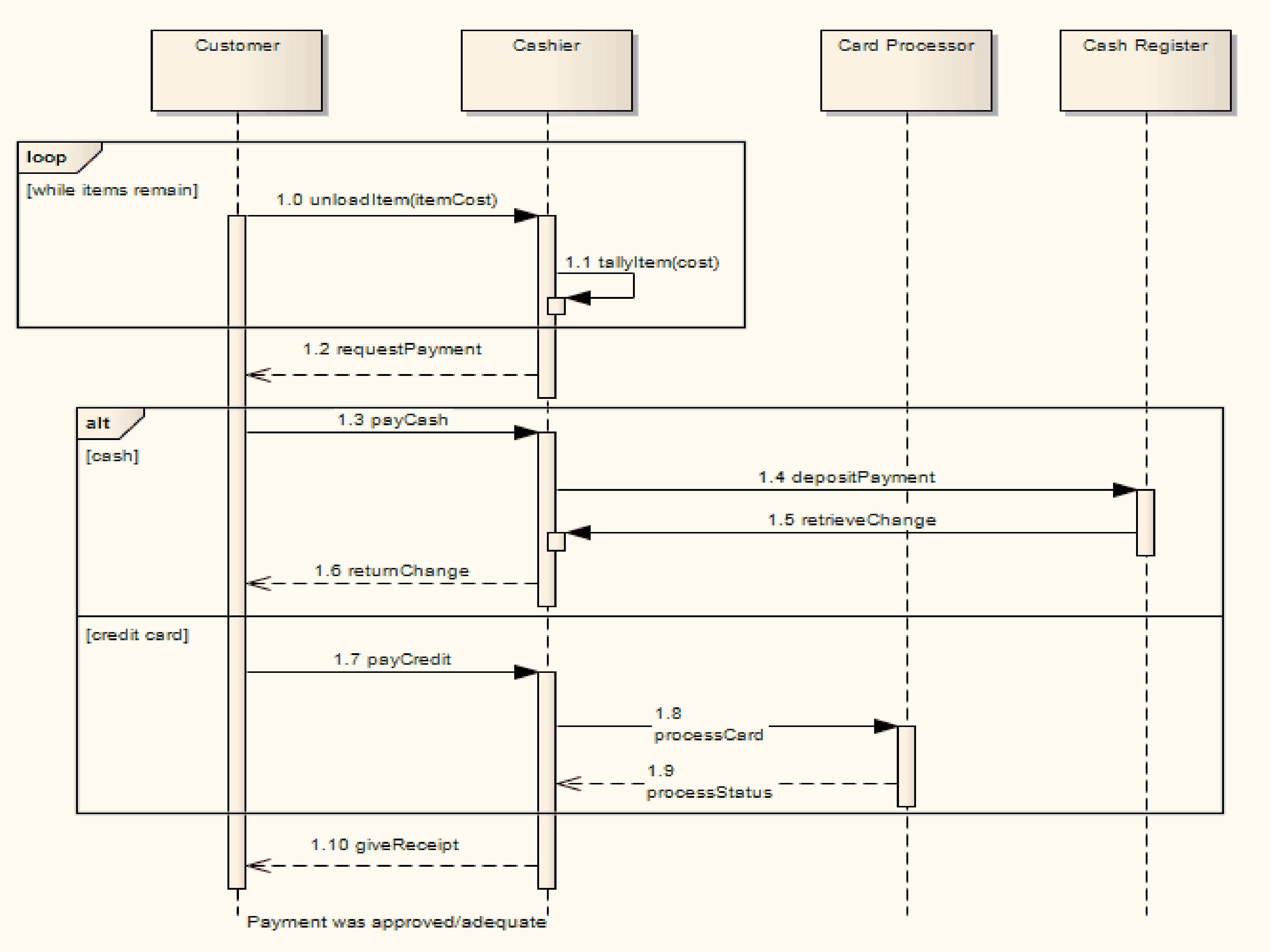
Alt and Else Operators

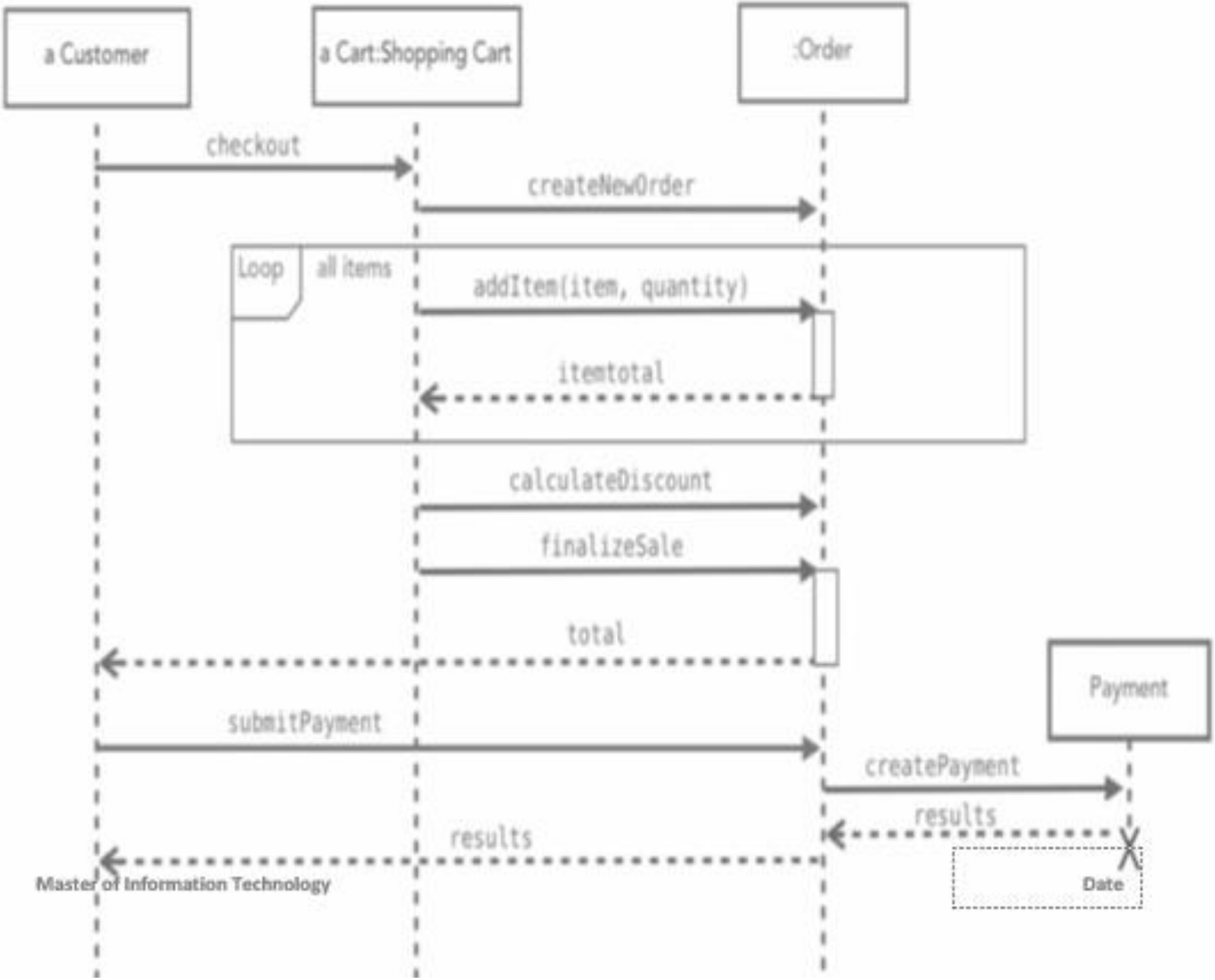
- An operand guarded by else designates a guard that is the negation of the disjunction of all other guards in the enclosing combined fragment.
  - If none of the operands has a guard that evaluates to true, none of the operands are executed and
  - the remainder of the enclosing interaction fragment is executed.


sender
receiver1
receiver2
alt
[condition1]
message1(parameters)
message2(parameters)
[condition2]
message3(parameters)
[else]
message4(parameters)
interaction occurs if condition1 is met
otherwise, this interaction occurs if condition2 is met
otherwise, this interaction occurs

Example of a Combined Fragment




Customer
Cashier
Card Processor
Cash Register
loop
[while items remain]
1.0 unloadItem(itemCost)
1.1 tallyItem(cost)
1.2 requestPayment
alt
[cash]
1.3 payCash
1.4 depositPayment
1.5 retrieveChange
1.6 returnChange
[credit card]
1.7 payCredit
1.8 processCard
1.9 processStatus
1.10 giveReceipt
Payment was approved/adequate

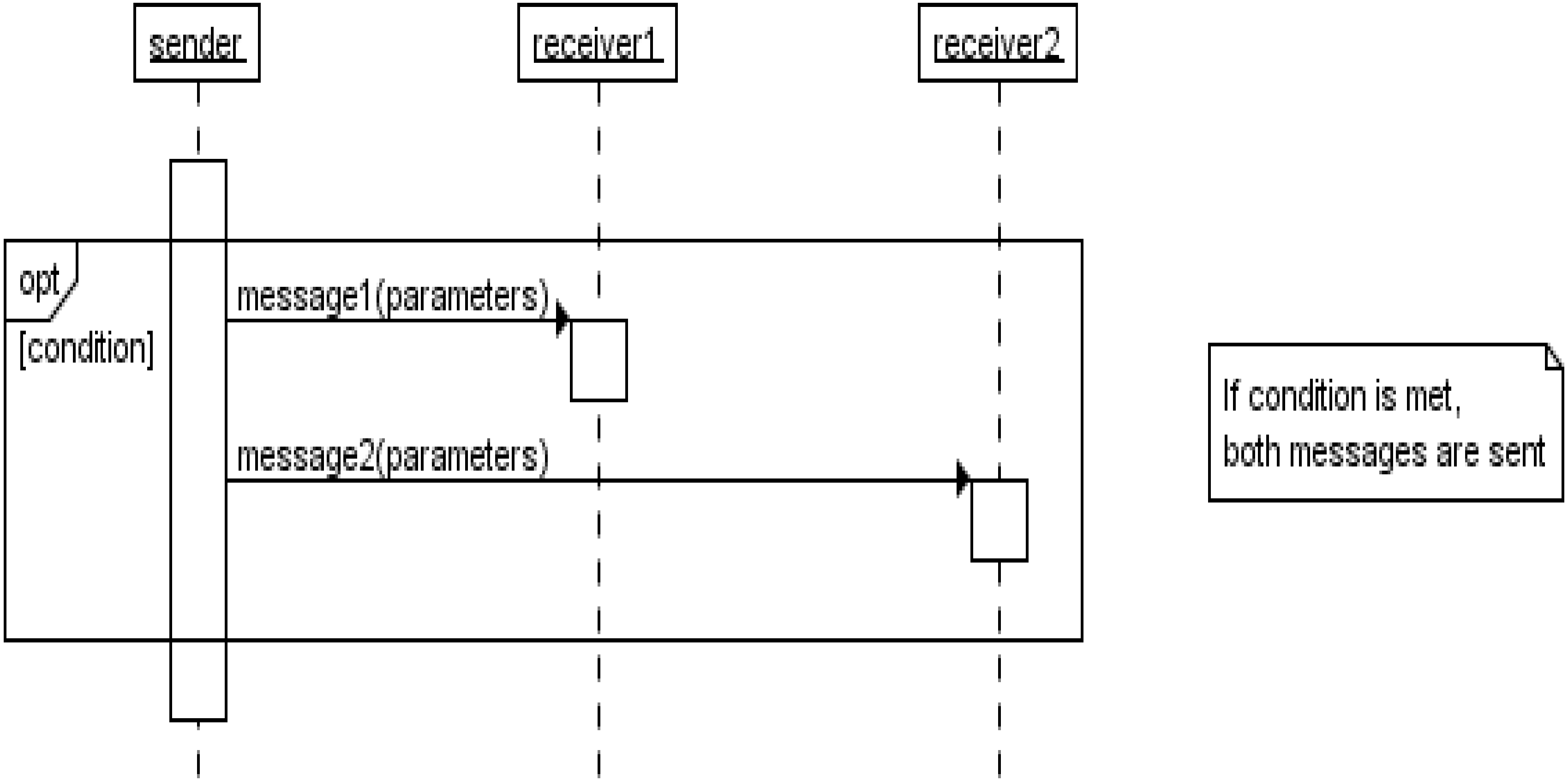


Master of Information Technology

Opt and Break Operators

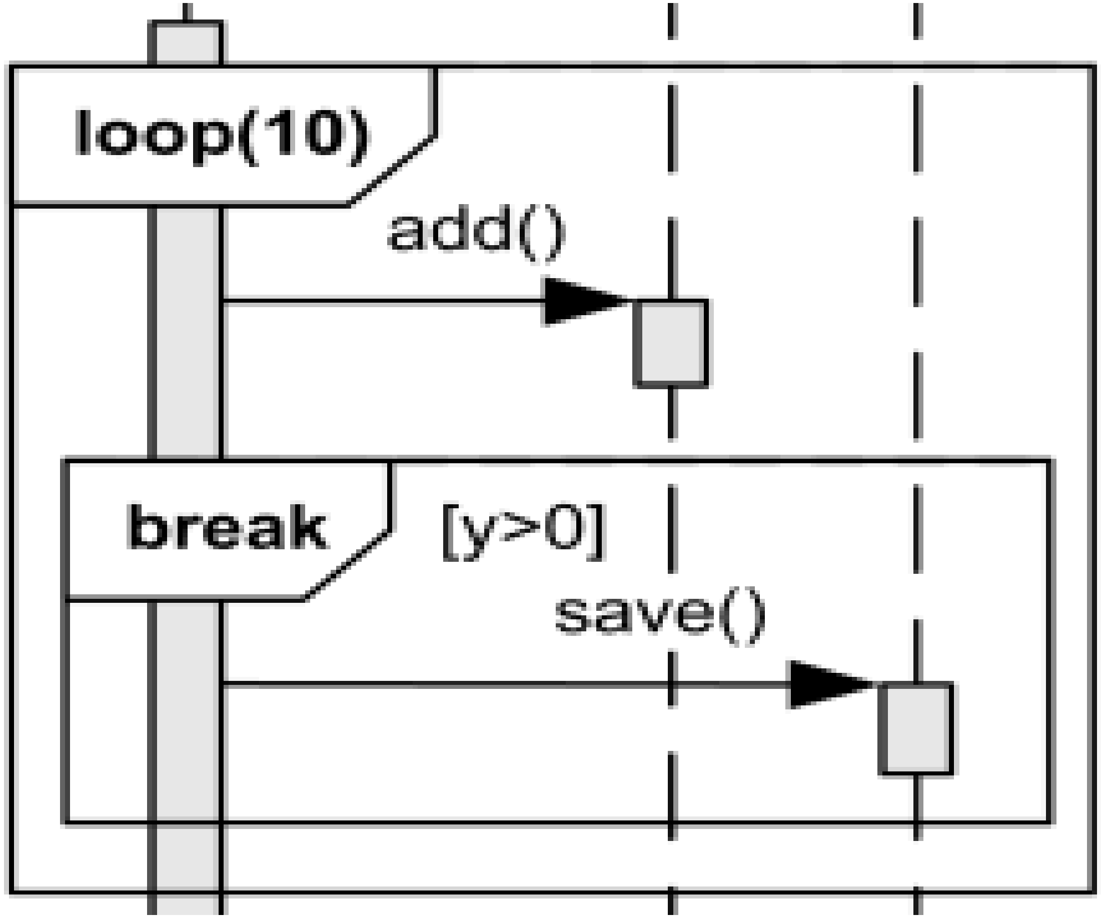
- Option: The interaction operator **opt** designates a choice of behavior where either the (sole) operand happens or nothing happens.
- Break: The interaction operator **break** represents a breaking scenario: The operand is a scenario that is performed instead of the remainder of the enclosing interaction fragment.
  - A break operator with a guard is chosen when the guard is true
  - The break operand is ignored, and the rest of the enclosing interaction fragment is chosen when the guard of the break operand is false.

Opt Operator



If condition is met, both messages are sent

Loop and Break Operators



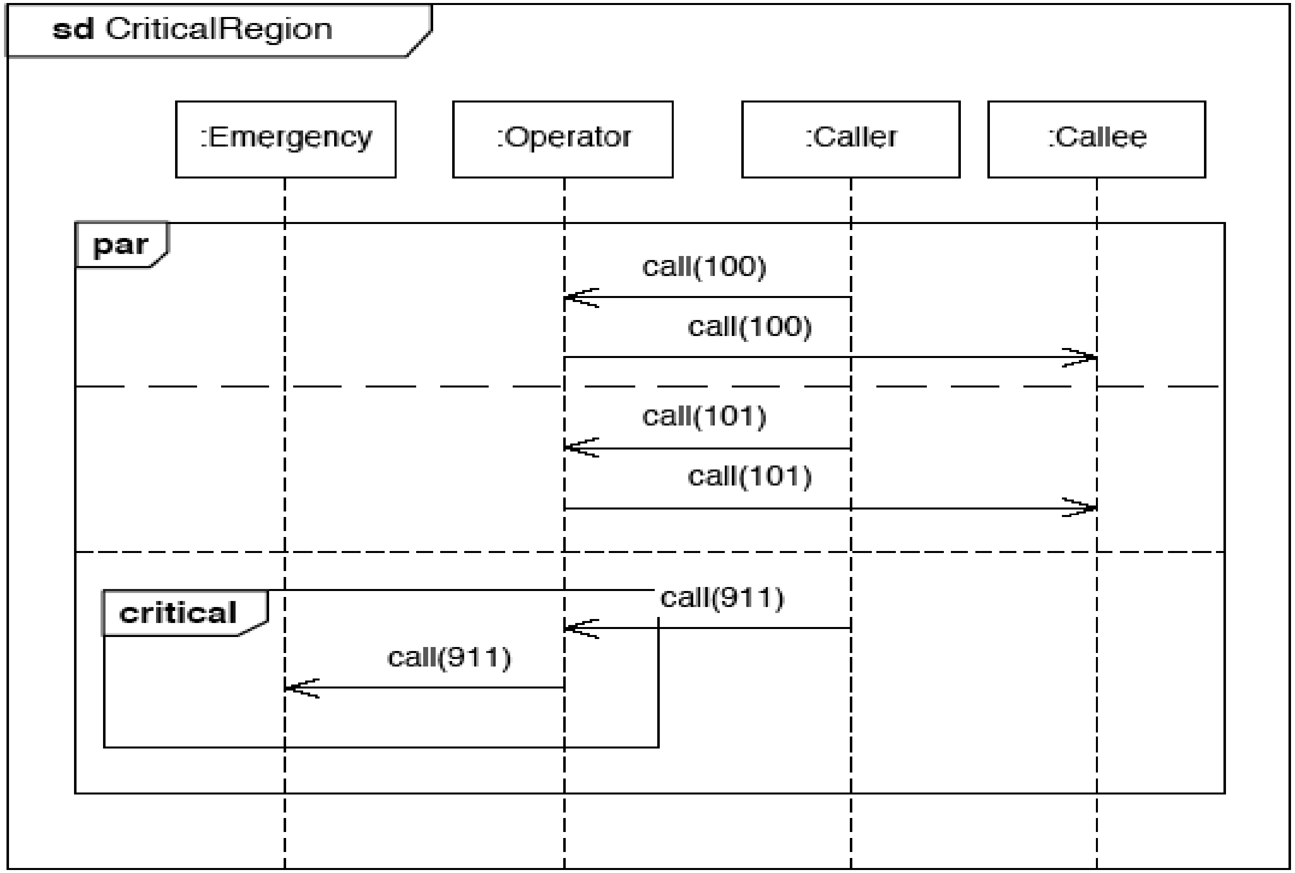
Parallel Operator

- Parallel: The interaction operator **par** designates a parallel merge between the behaviors of the operands of a combined fragment.
  - The event occurrences of the different operands can be interleaved in any way as long as the ordering imposed by each operand is preserved.
  - A parallel merge defines a set of traces that describes all the ways that event occurrences of the operands may be interleaved.

Critical Operator

- Critical: The interaction operator **critical** designates that the combined fragment represents a critical region.
  - The traces of the region cannot be interleaved by other event occurrences (on the Lifelines covered by the region). This means that the region is treated atomically by the enclosing fragment

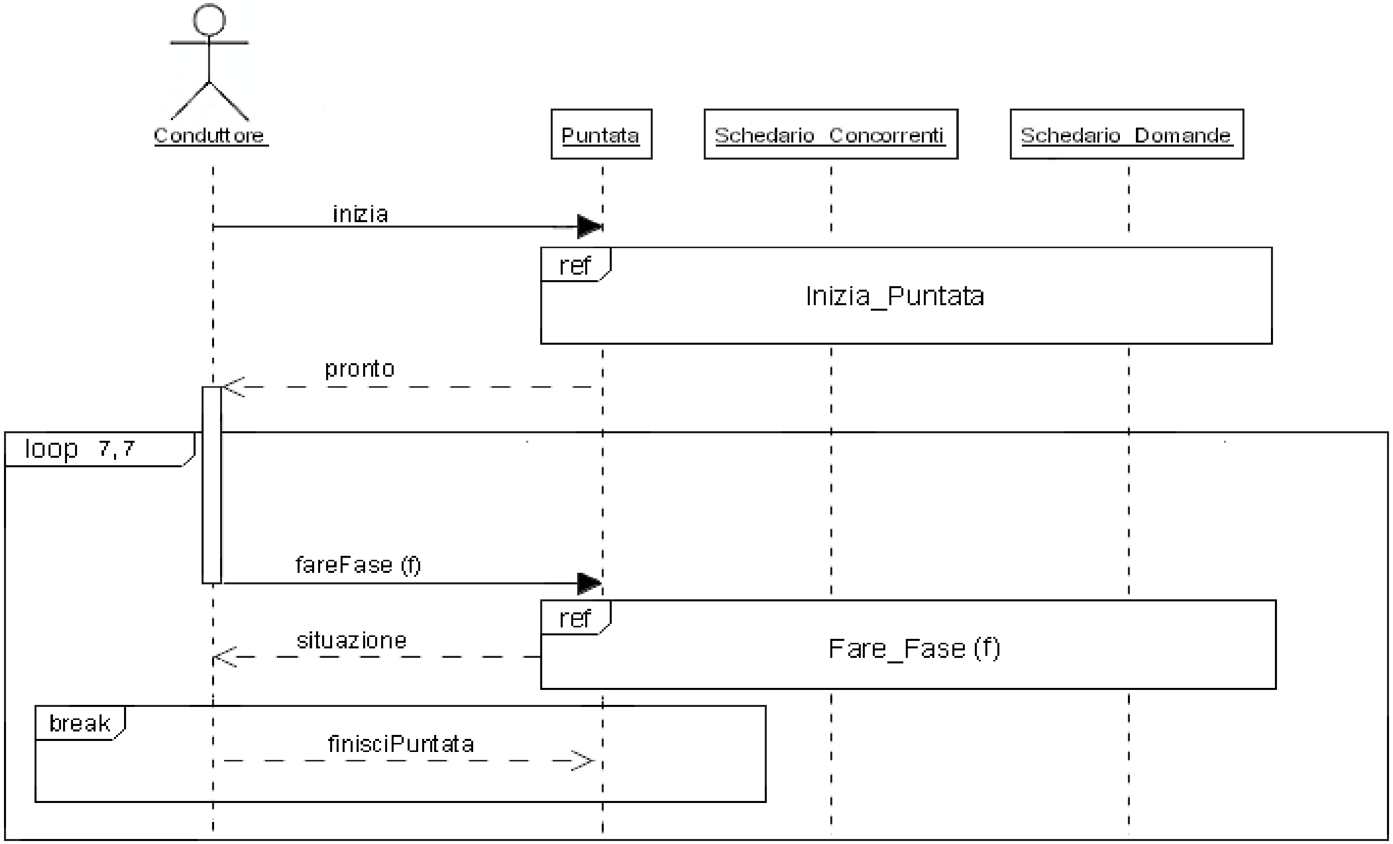
Example of a Critical Region



The Operator must make sure to forward a 911-call to the Emergency object before doing anything else. Normal calls can be freely interleaved.

Ref Operator

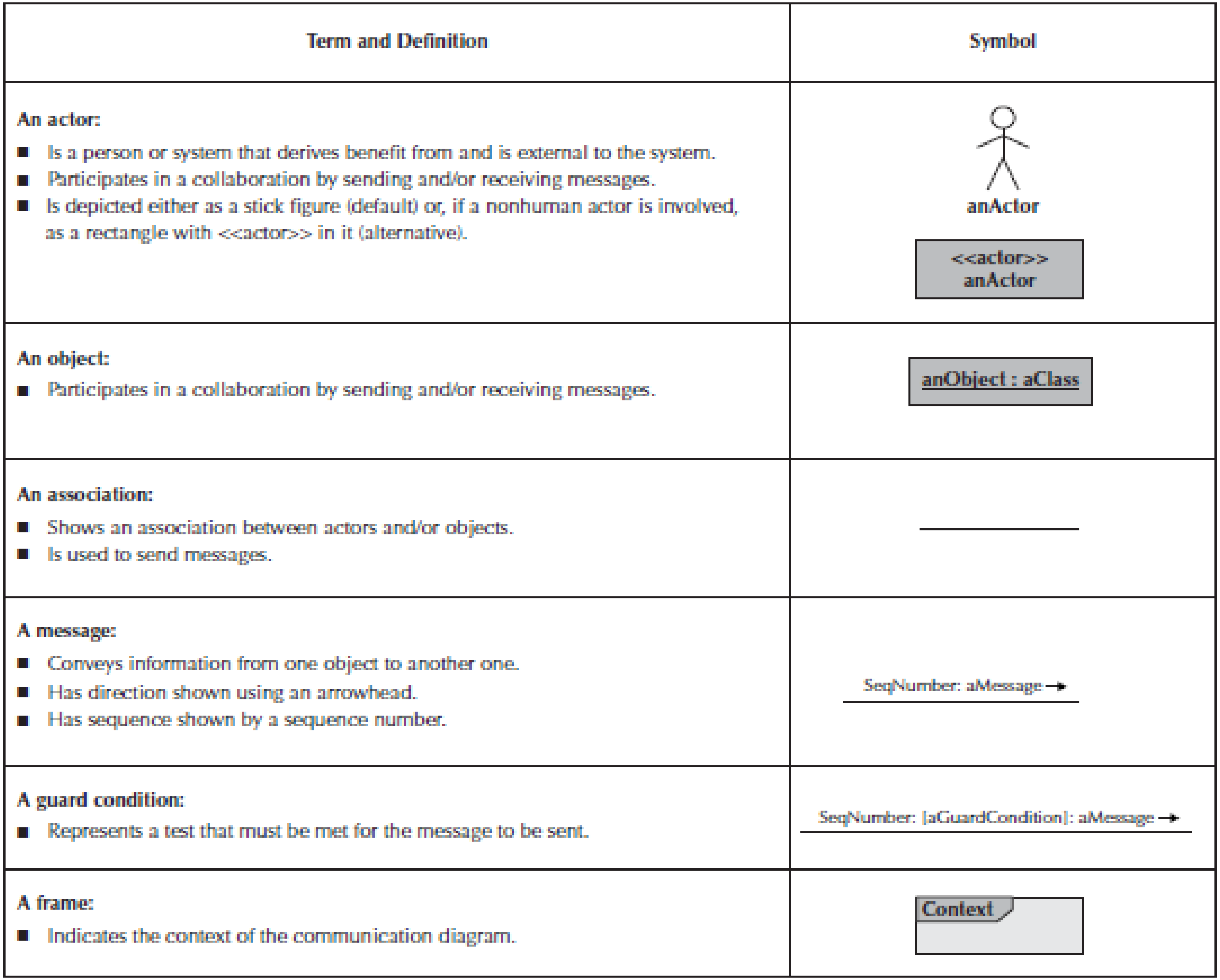
- **Ref**: Ref operator refers to a diagram defined elsewhere with the **sd** operator



7.3 Communication Diagrams

- Communication diagrams, like sequence diagrams, essentially provide a view of the dynamic aspects of an object-oriented system. They can show how the members of a set of objects collaborate to implement a use case or a use-case scenario.
- Communication diagrams are equivalent to sequence diagrams, but they emphasize the flow of messages through a set of objects, whereas the sequence diagrams focus on the time ordering of the messages being passed.
- Therefore, to understand the flow of control over a set of collaborating objects or to understand which objects collaborate to support business processes, a communication diagram can be used.

Elements of a Communication Diagram



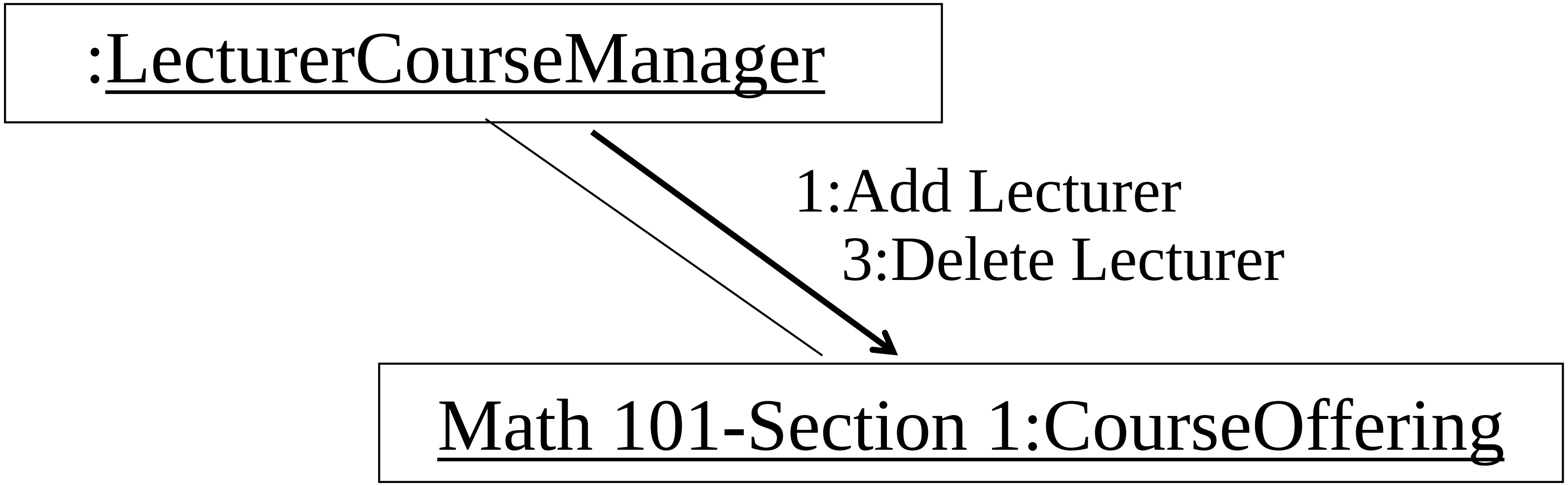
Term and Definition	Symbol
An actor: ■ Is a person or system that derives benefit from and is external to the system. ■ Participates in a collaboration by sending and/or receiving messages. ■ Is depicted either as a stick figure (default) or, if a nonhuman actor is involved, as a rectangle with <<actor>> in it (alternative).	anActor <<actor>> anActor
An object: ■ Participates in a collaboration by sending and/or receiving messages.	anObject : aClass
An association: ■ Shows an association between actors and/or objects. ■ Is used to send messages.	———
A message: ■ Conveys information from one object to another one. ■ Has direction shown using an arrowhead. ■ Has sequence shown by a sequence number.	SeqNumber: aMessage →
A guard condition: ■ Represents a test that must be met for the message to be sent.	SeqNumber: [aGuardCondition]: aMessage →
A frame: ■ Indicates the context of the communication diagram.	Context

Creating a Communication Diagrams

- An alternative way to show a scenario.
- Shows Object interactions organized around the objects and their links to each other.
- A communication diagram contains:
    - Objects drawn as rectangles.
    - Links between objects shown as lines connecting the linked objects.
    - Messages shown as text and an arrow that points from the client to the supplier.

Communication Diagrams cont..

UML Notation for objects, links and messages in a communication diagram.



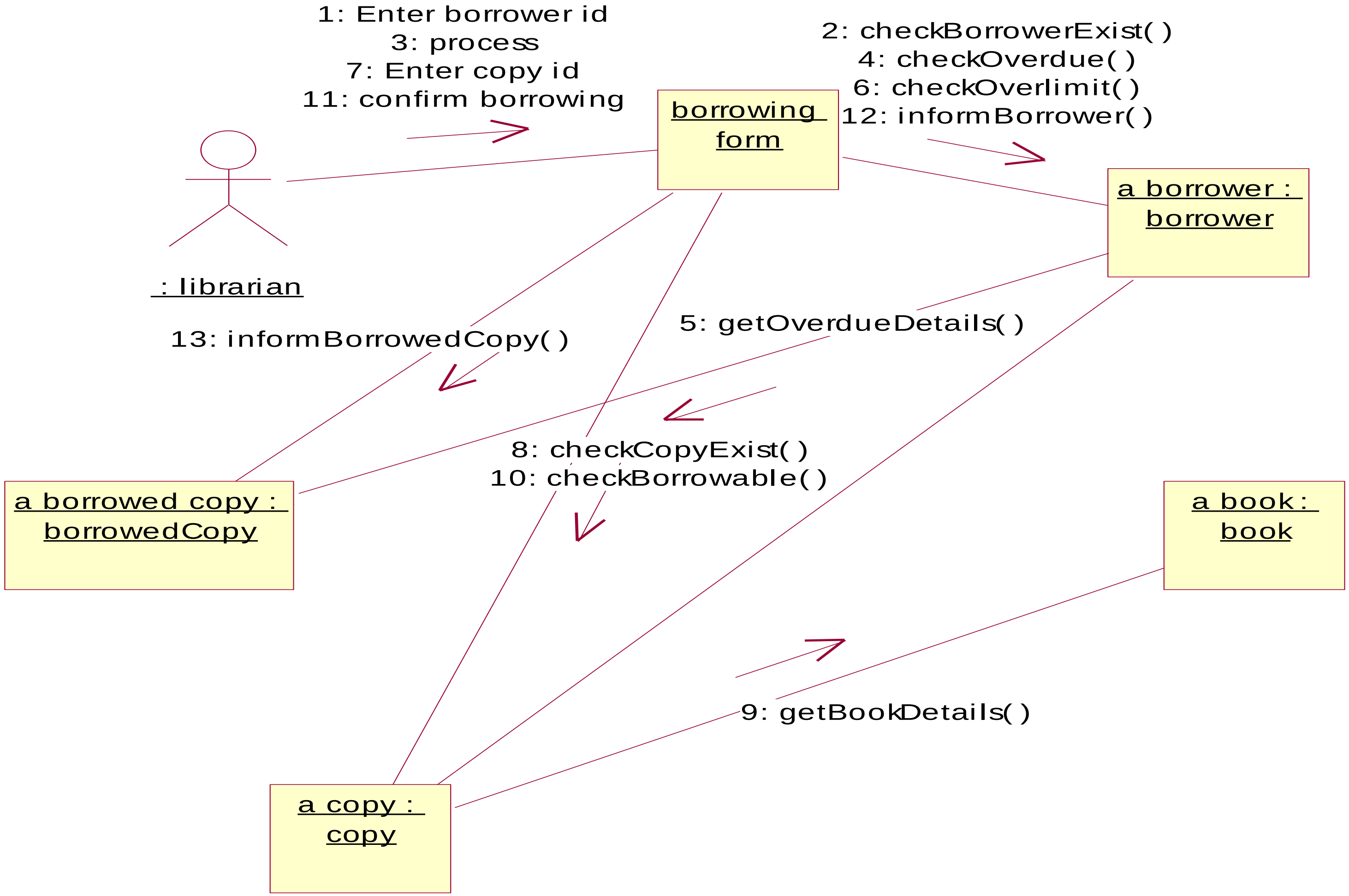
© e-Learning Centre, UCSC

Communication Diagrams cont..

Why do you need two different diagrams?

- *Sequence diagrams* :
  - Show a scenario in a time-based order: what happens first and what happens next.
  - Customers can easily read and understand sequence diagrams.
  - Useful in early analysis phases.
- *Communication diagrams* :
  - Tend to provide the big picture for a scenario.
  - Organized around the object links to one another.
  - Used more in the design phase of development

Communication Diagrams cont..

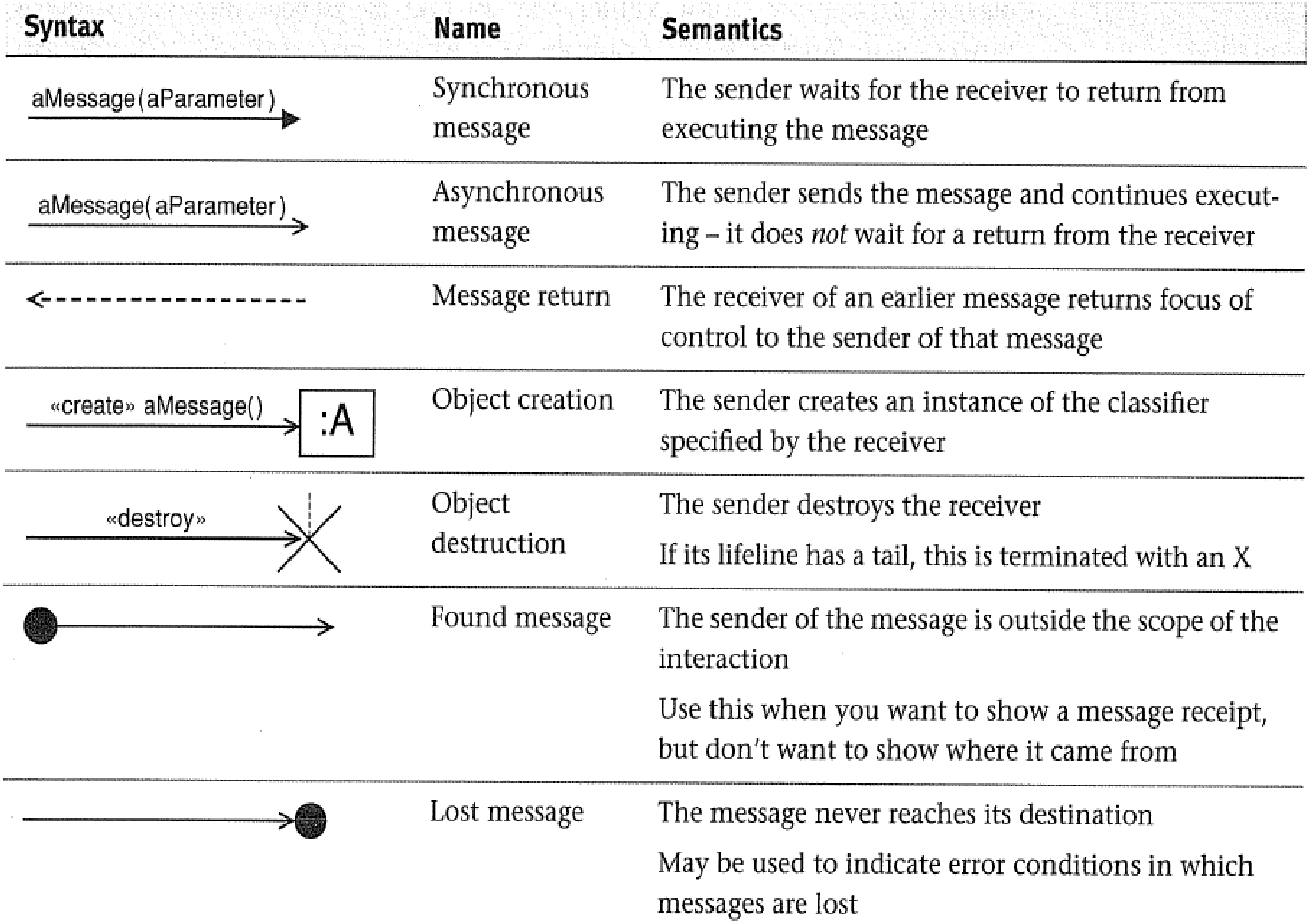


© e-Learning Centre, UCSC

Message Numbering in a Communication Diagram

- A Sequence diagram is read from top to bottom.
  - So, Message numbering is not necessary.
- A Collaboration diagram, however, losses its sequencing information, if you do not have message numbering.
- Message numbering can be turn on/off in Rational Rose.

Types of Messages in UML 2.*



Syntax	Name	Semantics
aMessage(aParameter)	Synchronous message	The sender waits for the receiver to return from executing the message
aMessage(aParameter)	Asynchronous message	The sender sends the message and continues executing – it does *not* wait for a return from the receiver
	Message return	The receiver of an earlier message returns focus of control to the sender of that message
«create» aMessage() :A	Object creation	The sender creates an instance of the classifier specified by the receiver
«destroy»	Object destruction	The sender destroys the receiver If its lifeline has a tail, this is terminated with an X
	Found message	The sender of the message is outside the scope of the interaction Use this when you want to show a message receipt, but don't want to show where it came from
	Lost message	The message never reaches its destination May be used to indicate error conditions in which messages are lost

Summary

- Behavioral models describe the internal dynamic aspects of an information system that supports the business processes in an organization.
- The flow of events for a use case is captured in text, whereas scenarios are captured in Interaction diagrams.
- There are four interaction diagrams in UML 2.0.

  Sequence Diagrams, Communication Diagrams, Timing diagrams and Interaction overview diagrams.
- Each Diagram is a graphical view of the scenario typically associated with Use Cases in the model.
- Sequence diagrams show object interactions arranged in time sequence. It shows the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario.

Summary cont..

- Boundary classes are added to sequence diagrams to show the interaction with the user or another system.
- Actual messages from the actor to boundary class with their sequencing information will depend on the application framework that will be selected later in development.
- Keep sequence diagrams simple so that it is easy to see the objects, object interactions, the messages between the objects, and the functionality captured by the scenario.
- The focus of control is a small rectangle, that will let you know which object has control at a particular point in time.
- Sequence diagrams show a scenario in a time-based order (what happens first and what happens next).
- Customers can easily read and understand sequence diagrams. They are useful in early analysis phases.
- An interaction fragment allows you to group related messages in a sequence diagram.
- Combine fragment is a subtype of interaction fragment. It defines an expression of interaction fragments defined by an interaction operator and corresponding interaction operands

Summary cont..

- Communication diagrams, like sequence diagrams, essentially provide a view of the dynamic aspects of an object-oriented system. They can show how the members of a set of objects collaborate to implement a use case or a use-case scenario.
- Communication diagrams are equivalent to sequence diagrams, but they emphasize the flow of messages through a set of objects, whereas the sequence diagrams focus on the time ordering of the messages being passed.
- Communication Diagrams tend to provide the big picture for a scenario organized around the object links to one another. They are used more in the design phase of development

*****