

# Discovering Object Interactions

# Scenarios

- The functionality of the use case is captured in the use case narrative flow of events section.
- A scenario is an instance of a use case.
- It is one path through the flow of events for the use case.
- Scenarios document decisions about how the responsibilities specified in the use cases are distributed among the objects and classes in the system.

## Scenarios cont...

- Provides an excellent communication medium to discuss systems requirements with customers.
- Speaks the language of the end users and the domain experts.

# Documenting Use Cases

## Use Case Narratives

- The **Use Case Narratives** for a use case is captured in text.
- Whereas scenarios are captured in interaction diagrams.
- A **Use Case Narrative** document is created for each use case
  - Written from an actor point of view
- The **Use Case Narrative** for a use case is a description of the events needed to accomplish the required behavior of the use case.

## Use Case Narratives cont..

- **Use Case Narratives :**

### **Typical contents**

- When and how the use case starts and ends
- What interaction the use case has with the actors
- What data is needed by the use case
- The normal sequence of events for the use case.
- The description of any alternate or exceptional flows. etc.

# Use Case Narratives cont..

## Template

- Priority
  - Preconditions
  - Primary and other participating actors
  - Trigger
  - Typical Flow of Events
  - Alternate Courses
  - Post conditions
- are included.**

The flow of events documentation typically is created in the *Elaboration Phase* (a phase in IBM Rational Unified Process) in an iterative manner

## Eg. 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

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



## Documenting Scenarios using Interaction Diagrams

- The flow of events for a use case is captured in text,
- Where as 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, interaction overview**
- Each Diagram is a graphical view of the scenario
- Typically associated with Use Cases in the model

# Sequence Diagrams

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

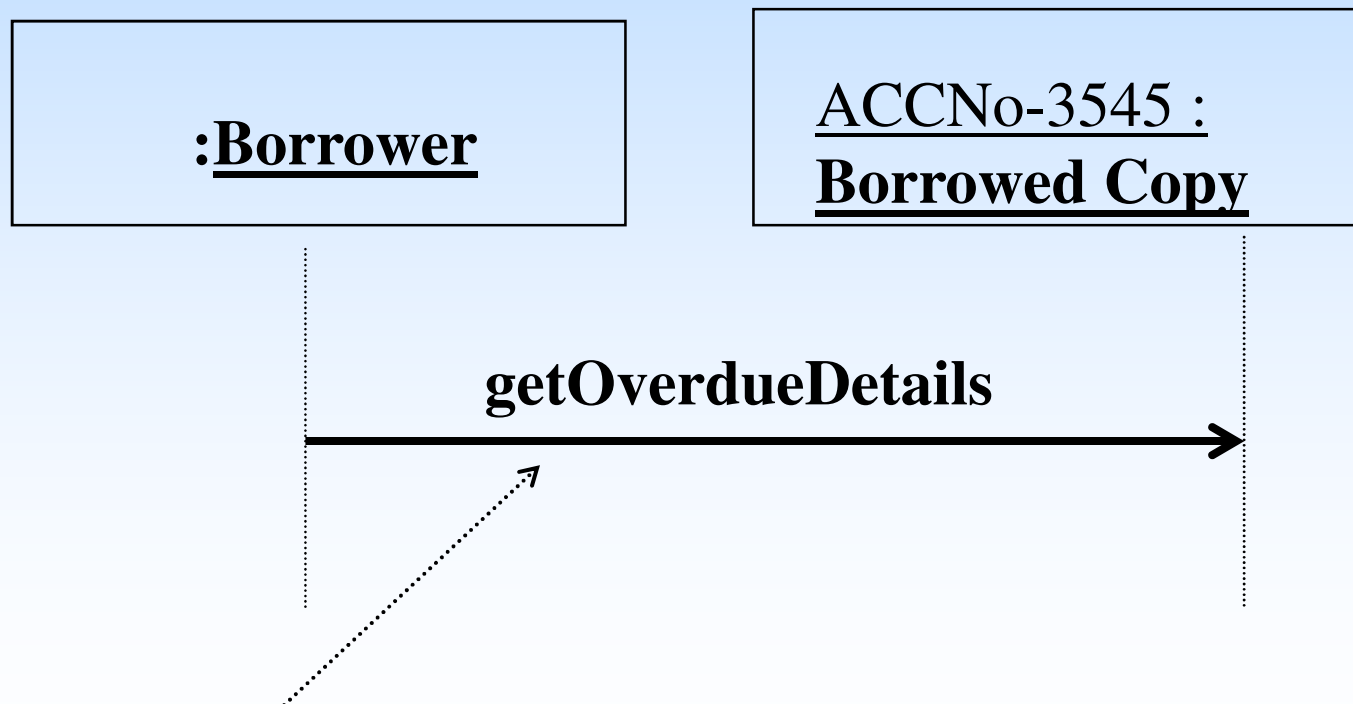
# Sequence Diagrams

cont..

- 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)

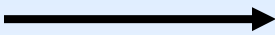


## Sequence Diagrams cont..

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



Messages between objects

# Messages

- UML represents a message as an arrow that starts at one lifeline and ends at another.
- An object can also send a message to itself.
- In UML 2.0 following message types are available.
  - Call or synchronous  **(Sender waits for the receiver to carry out the operation)**
  - Asynchronous  **(Sender transfers control to the receiver and doesn't wait for the operation to complete.)**
  - Return message  **(Modelers often omit this symbol)**

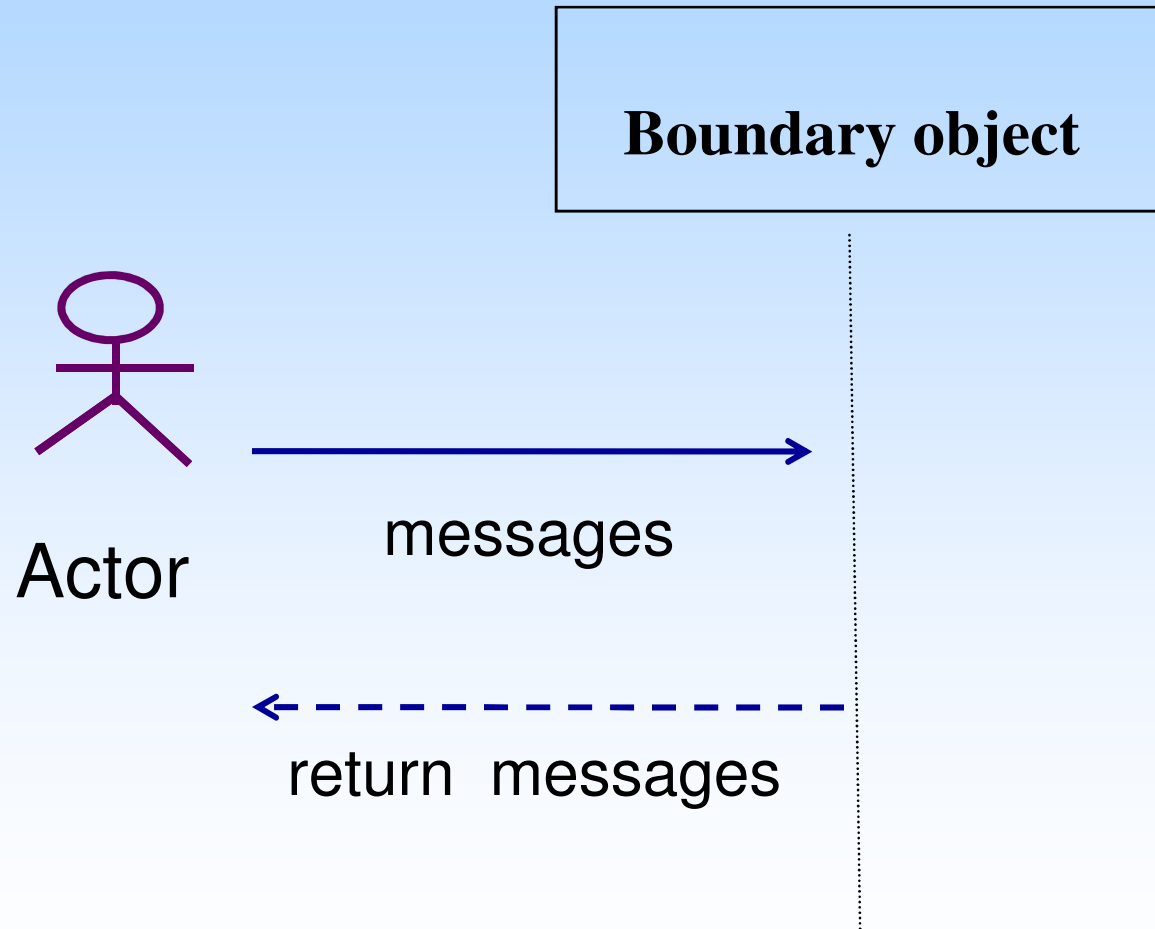
## Sequence Diagrams and Boundary classes

- 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

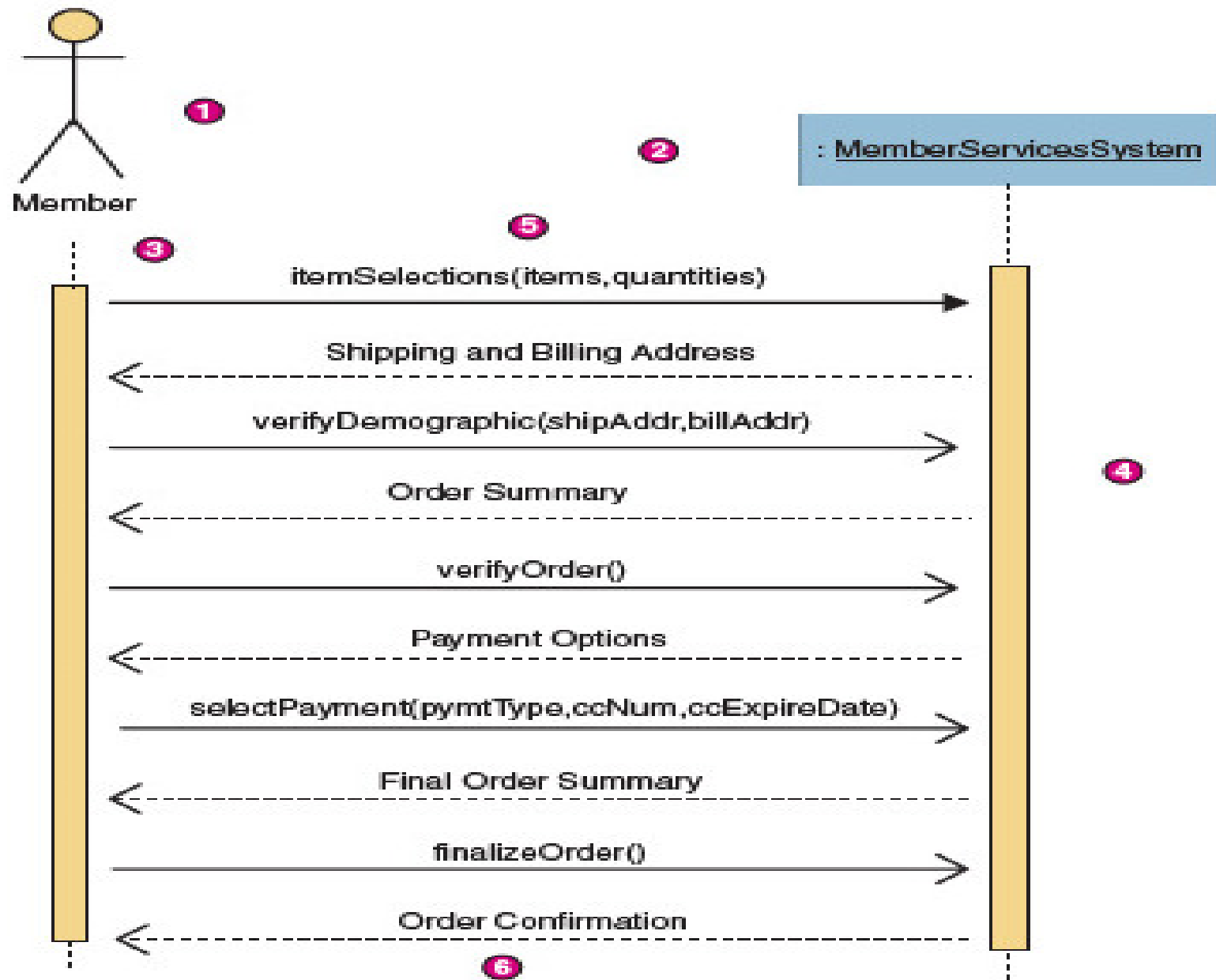
- A tool used by some Analysts in logical design phase is the system sequence diagram.
- It helps us to identify the high level messages that enter and exit the system.
- Later these messages will become the responsibility of individual objects.
- These individual objects will fulfill those responsibilities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

# System Sequence Diagrams





## Eg. System Sequence Diagram Ref 1. pg395



# General activities in performing OOA

- Modeling the functions of the system

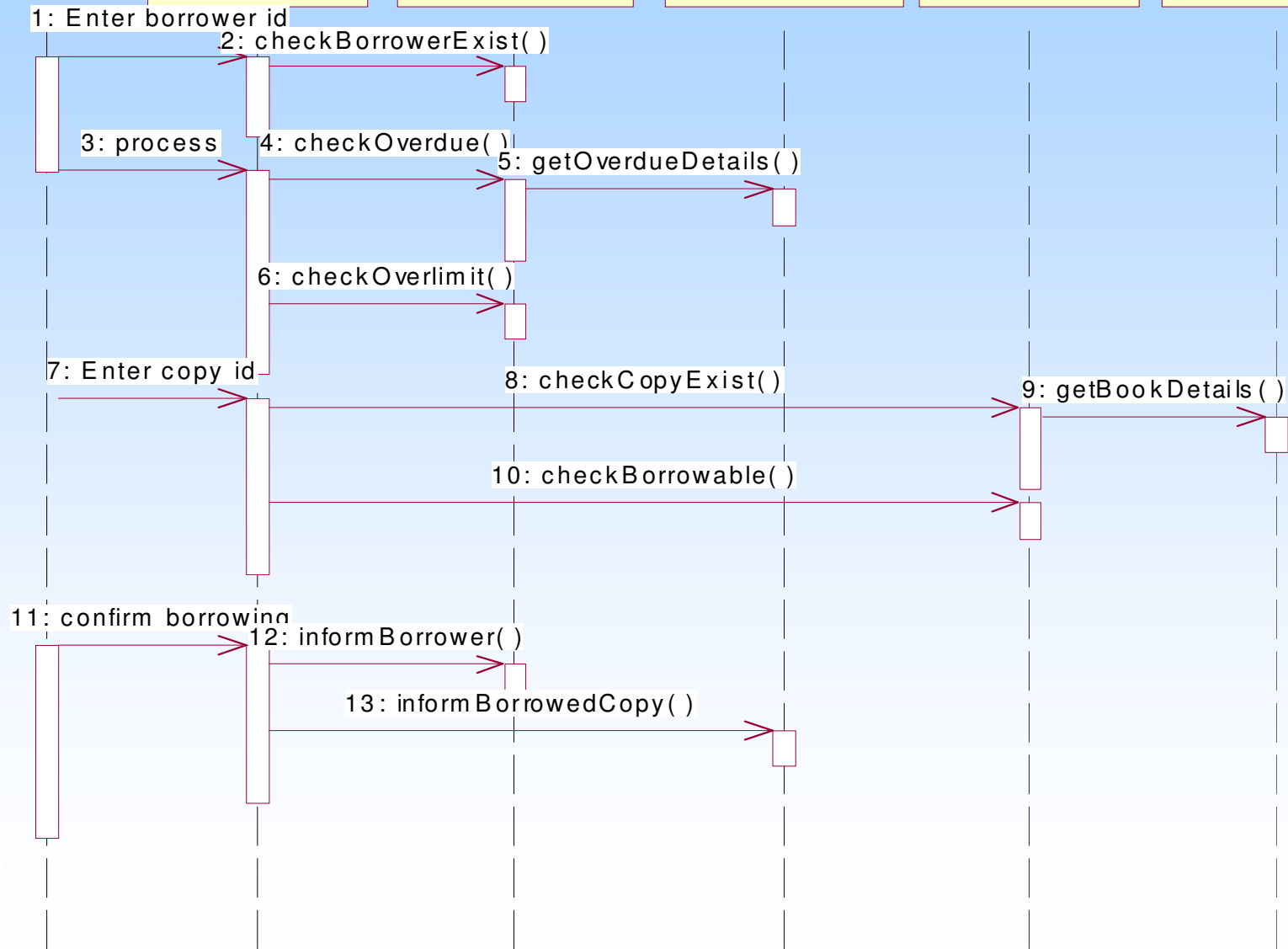
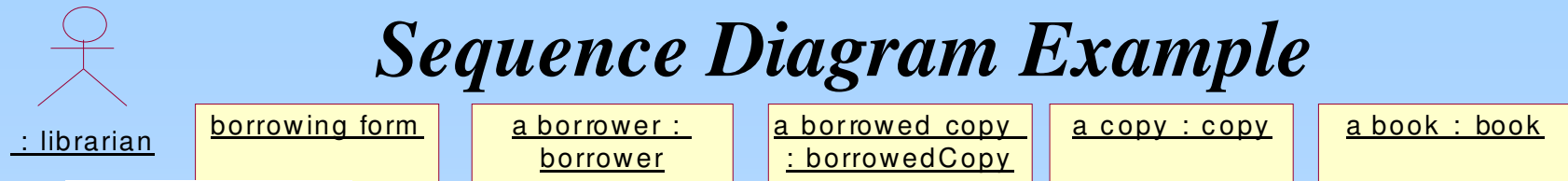
Refine the requirements use case model and use case narratives, Model the use case activities (draw activity diagrams), Draw **system sequence diagram** ,

- Finding and Identifying the Business Objects

Noun analysis, CRC analysis

- Organizing the Objects and Identifying the Relationships

Identify types of relationships between the objects. Class diagram with problem domain classes.



## Sequence Diagrams cont..

- How Complex can a sequence diagram be?
    - Keep them simple
- 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?  
( *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.

## Sequence Diagrams cont..

- In tools such as Rational *Rose*, diagrams may be linked to one another.
- This allows the user to navigate through a set of diagrams

## Let us look at the Order Processing example: Purchase Items

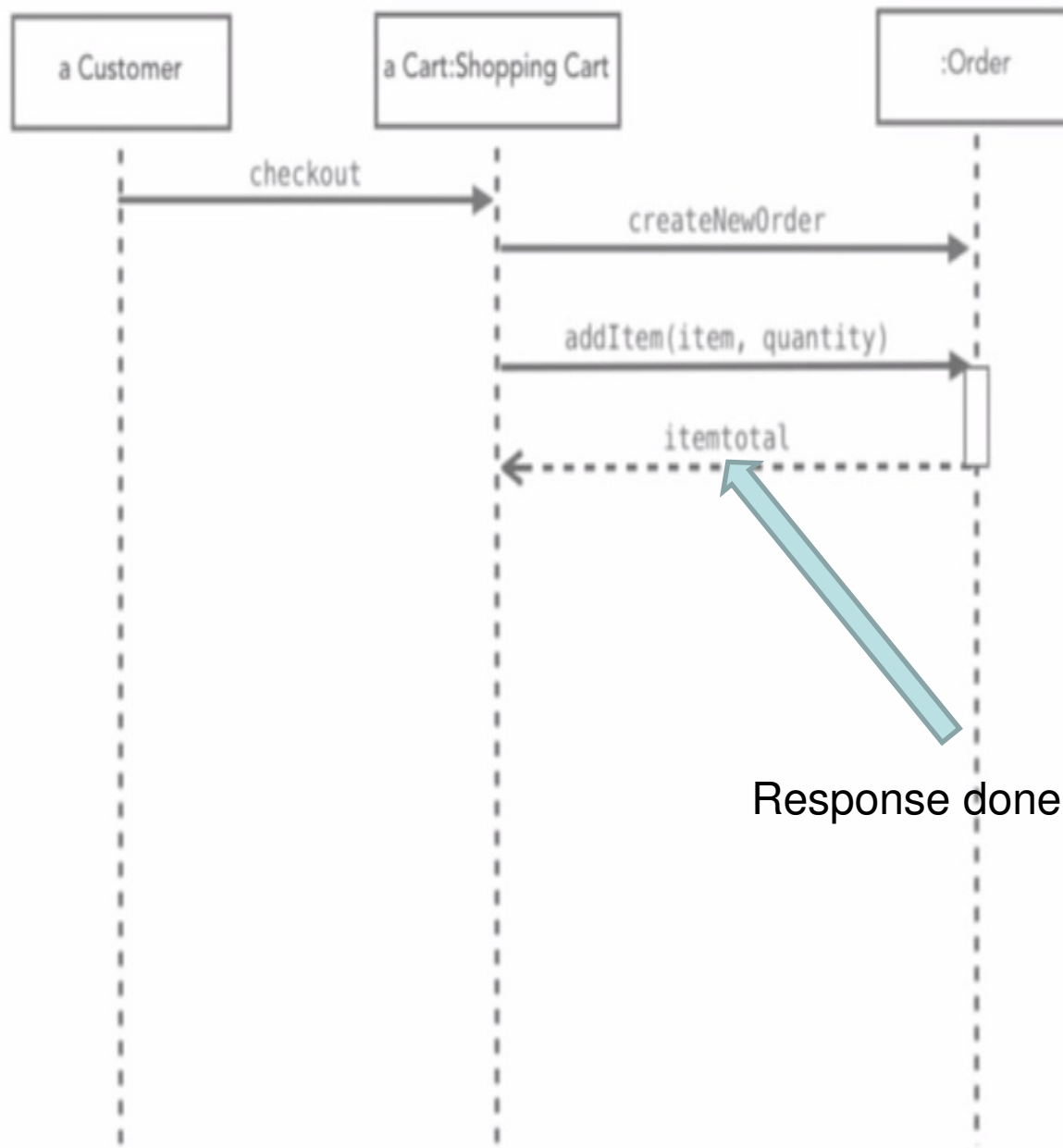
- Customer confirms items in the Shopping Cart and creates a new order

- Add different items to the order.

Customer provides payment and *address* to process sales

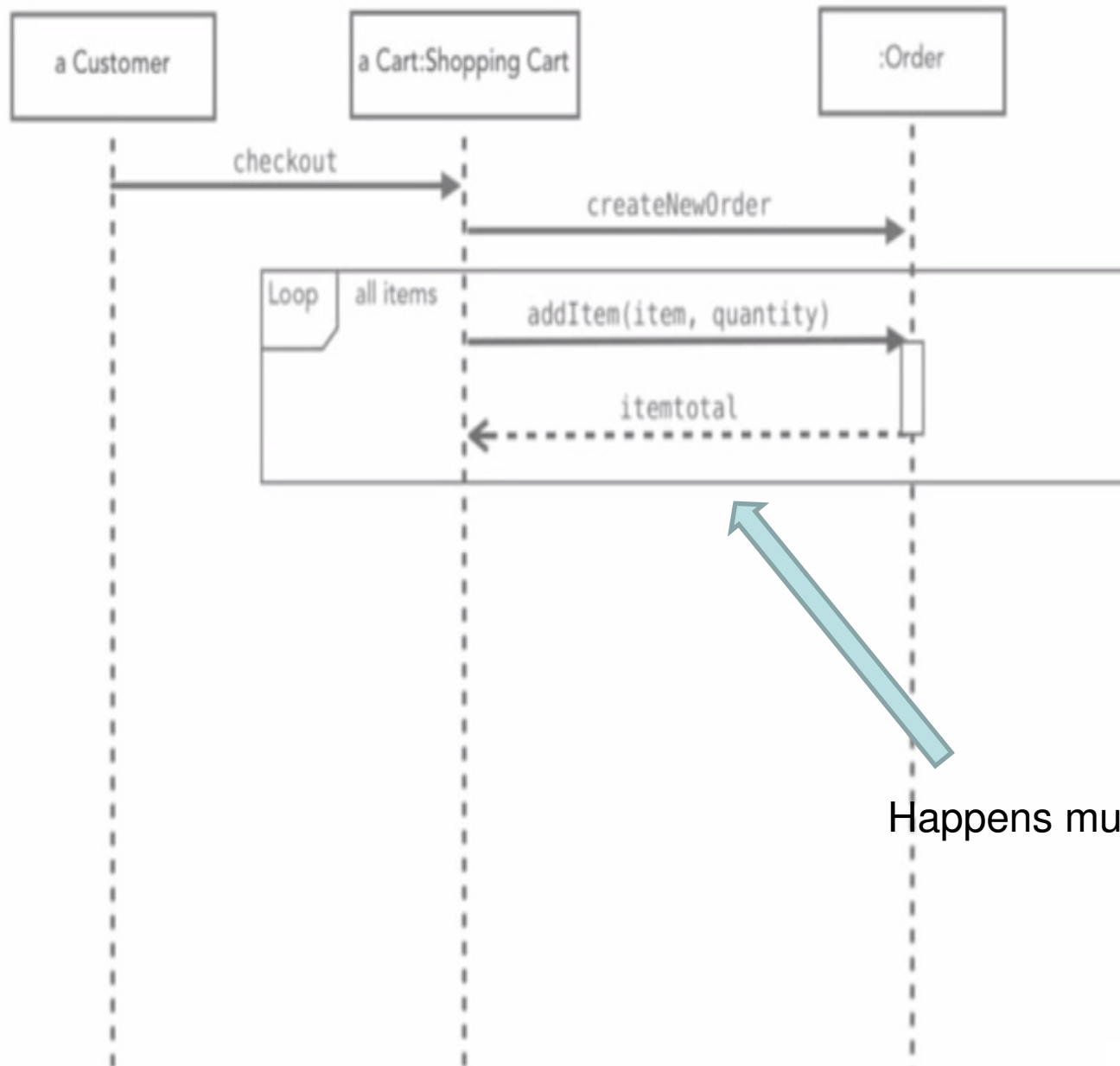
- System validate payment Information and respond with confirmation of order and provides order number that customer can use to check on order status.

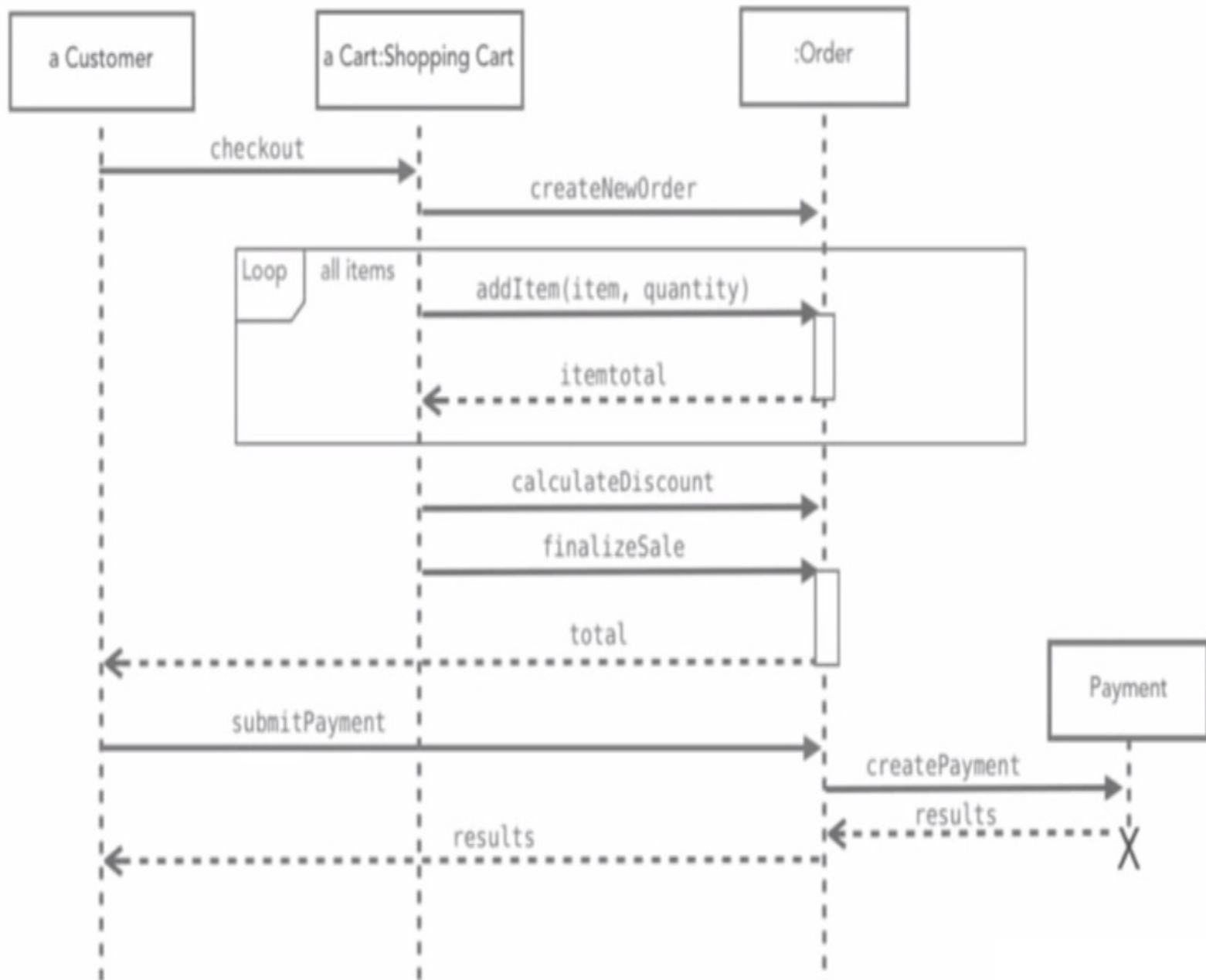
- System will send confirmation of order details to customer in an email.



Response done using Dash Arrow

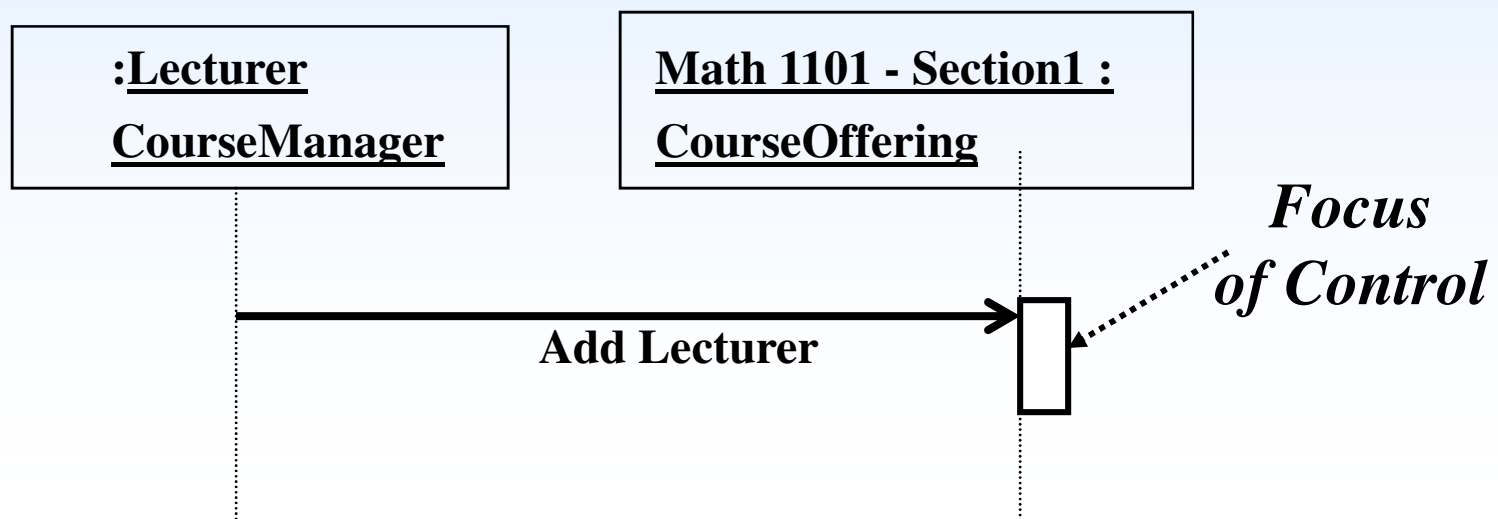






## Focus of Control (Activation Bar)

- 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 Collaboration diagram.
- The focus of control is shown only on a sequence diagram. (optional)

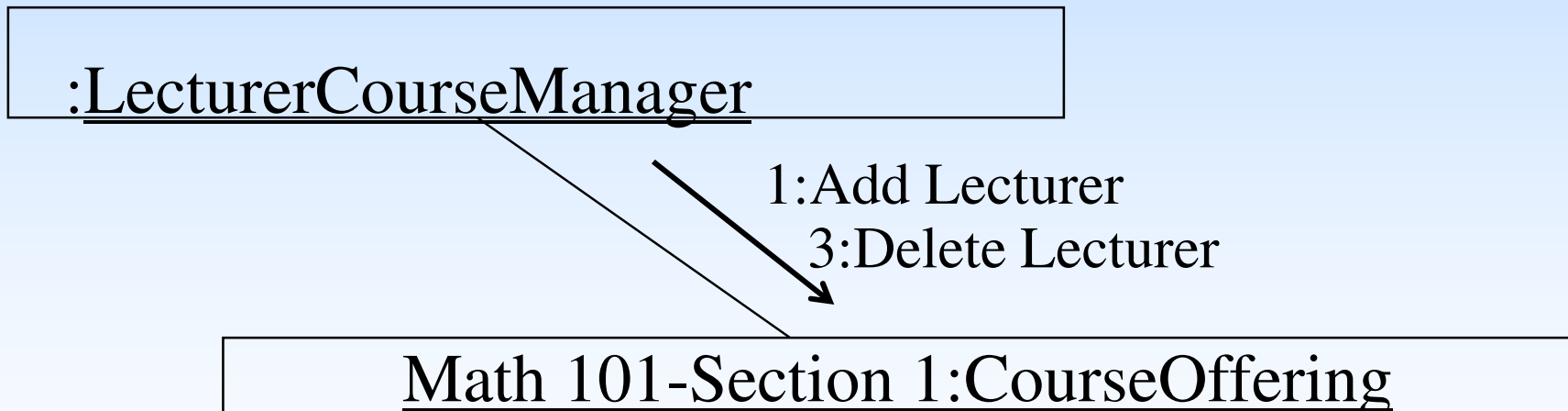


## Collaboration Diagrams/Communication Diagrams

- An alternative way to show a scenario.
- Shows Object interactions organized around the objects and their links to each other.
- A collaboration 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.

## Collaboration Diagrams /Communication Diagrams cont..

UML Notation for objects, links and Messages in a collaboration diagram.



## Collaboration Diagrams /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.

## Collaboration Diagrams /Communication Diagrams cont..

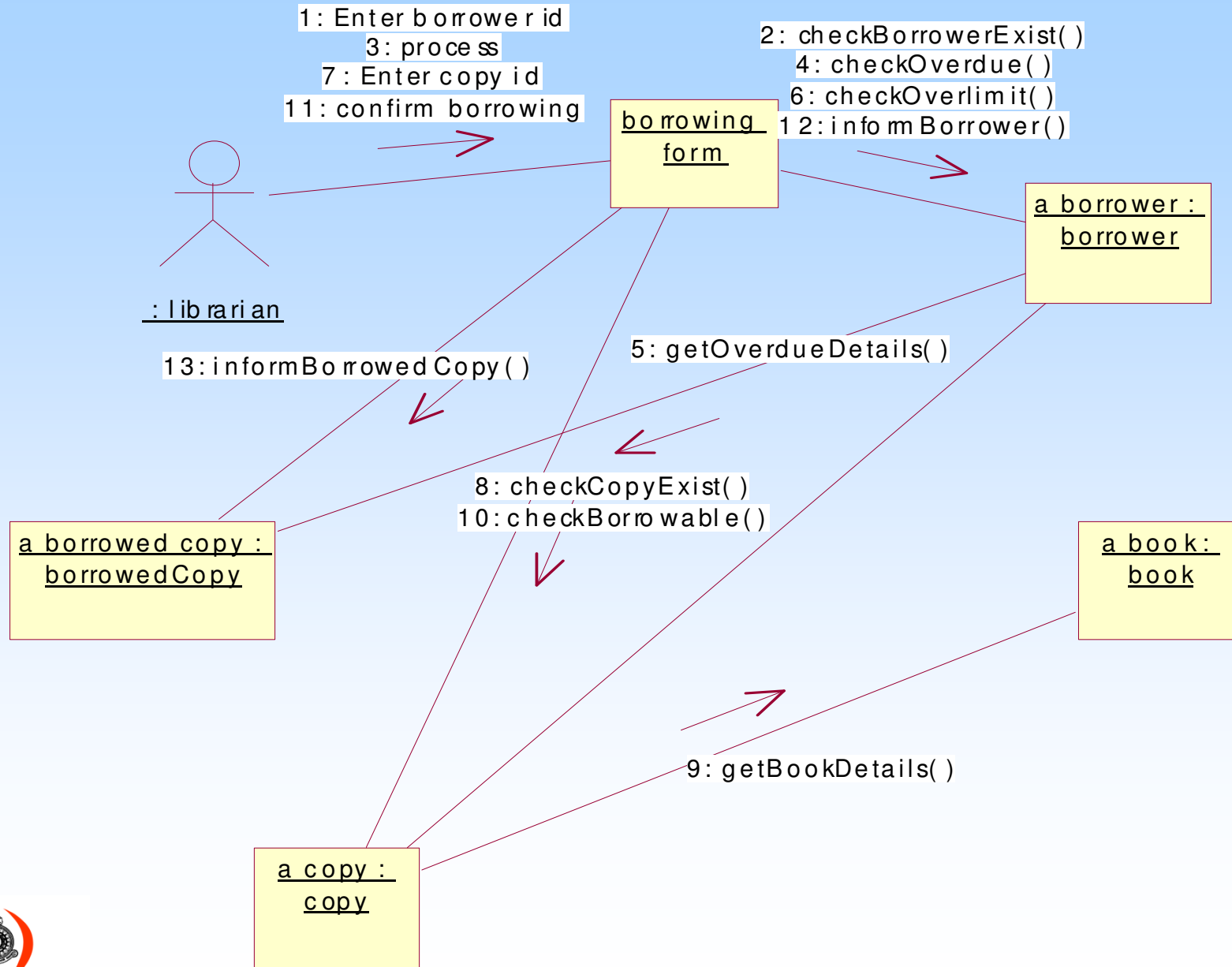
### *Collaboration Diagrams/Communication Diagrams:*

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

# Collaboration Diagram Example

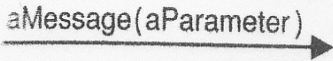
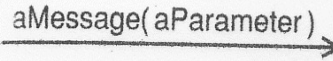

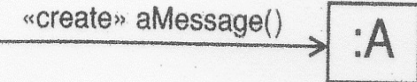
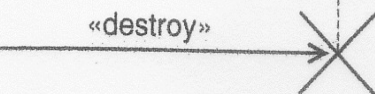
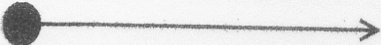





## Message Numbering in a Collaboration Diagram

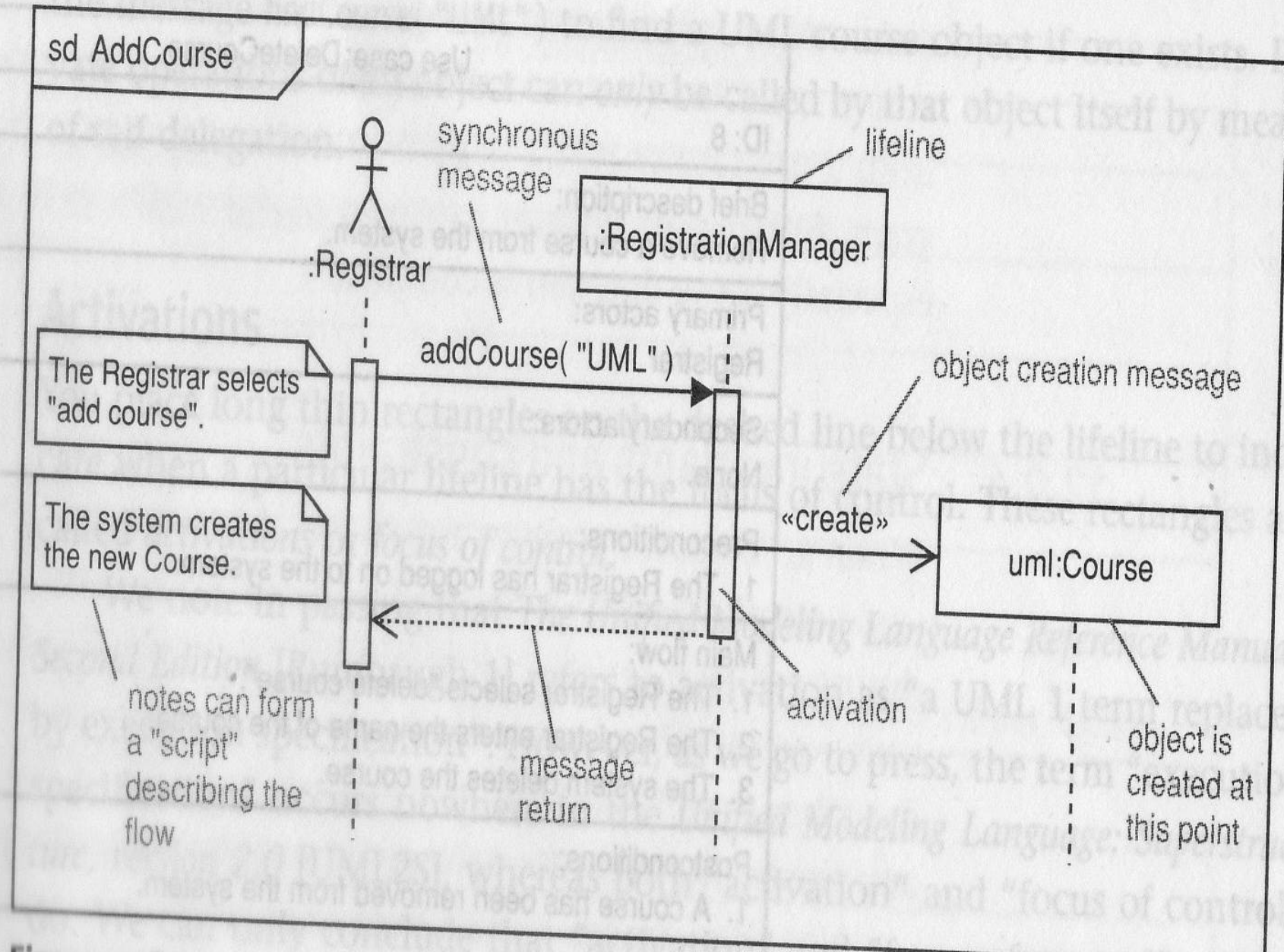
- Sequence diagram is read from top to bottom.
- So Message numbering is not necessary.
- A Collaboration diagram, however loses its sequencing information, if you do not have message numbering.
- Message numbering can be turn on/off in Rational Rose.

# Types of message in UML 2.\*

Syntax	Name	Semantics
	Synchronous message	The sender waits for the receiver to return from executing the message
	Asynchronous message	The sender sends the message and continues executing – it does <i>not</i> 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
	Object creation	The sender creates an instance of the classifier specified by the receiver
	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



## Eq. Sequence Diagram: Add Details of a new Course



## Framing in Sequence Diagrams

- 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.
- Ref.  
<http://www.youtube.com/watch?v=4WDbte6cPa8>

# Interaction Fragment

- Interaction Fragment
  - Is a piece of an **interaction**
  - Acts like an interaction itself
- Combined Fragment
  - Is a subtype of interaction fragment
  - defines an expression of interaction fragments
  - defined by an interaction operator and corresponding interaction operands

# Interaction Operators

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



**Ref:**

[http://www.sparxsystems.com/enterprise\\_architect\\_user\\_guide/10/standard\\_uml\\_models/interactionoperators.html](http://www.sparxsystems.com/enterprise_architect_user_guide/10/standard_uml_models/interactionoperators.html)

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, <i>not</i> 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 <i>same</i> lifeline from <i>different</i> 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 <i>must not</i> happen

Operator	Long name	Semantics
ignore	ignore	<p>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}</p> <p>For example, an interaction might represent a test case in which you choose to ignore some of the messages</p>
consider	consider	<p>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</p> <p>For example, an interaction might represent a test case in which you choose to include a subset of the set of possible messages</p>
assert	assertion	<p>The operand is the only valid behavior at that point in the interaction – any other behavior would be an error</p> <p>Use this as a way of indicating that some behavior <i>must</i> occur at a certain point in the interaction</p>

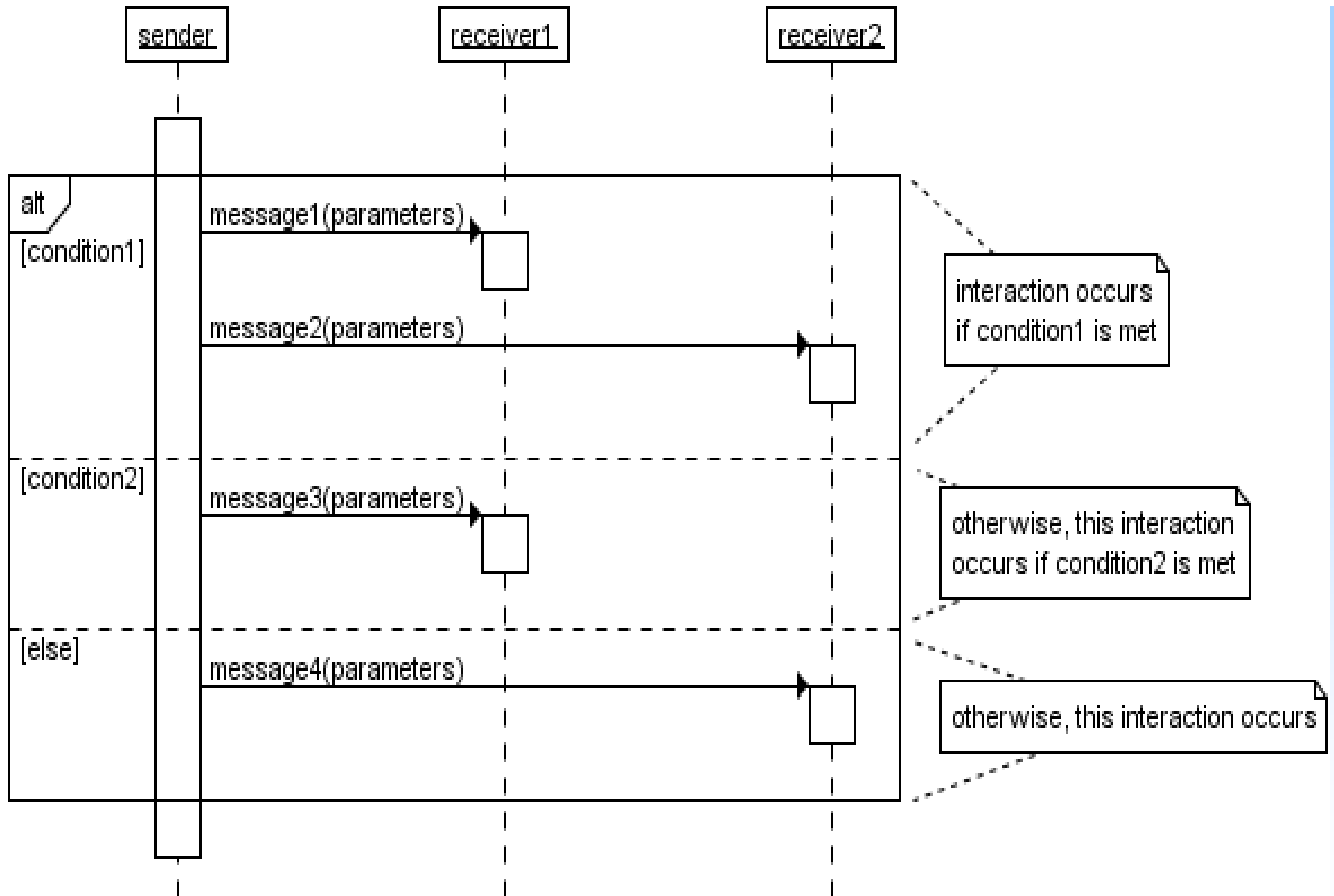


# Alt and Else Operators

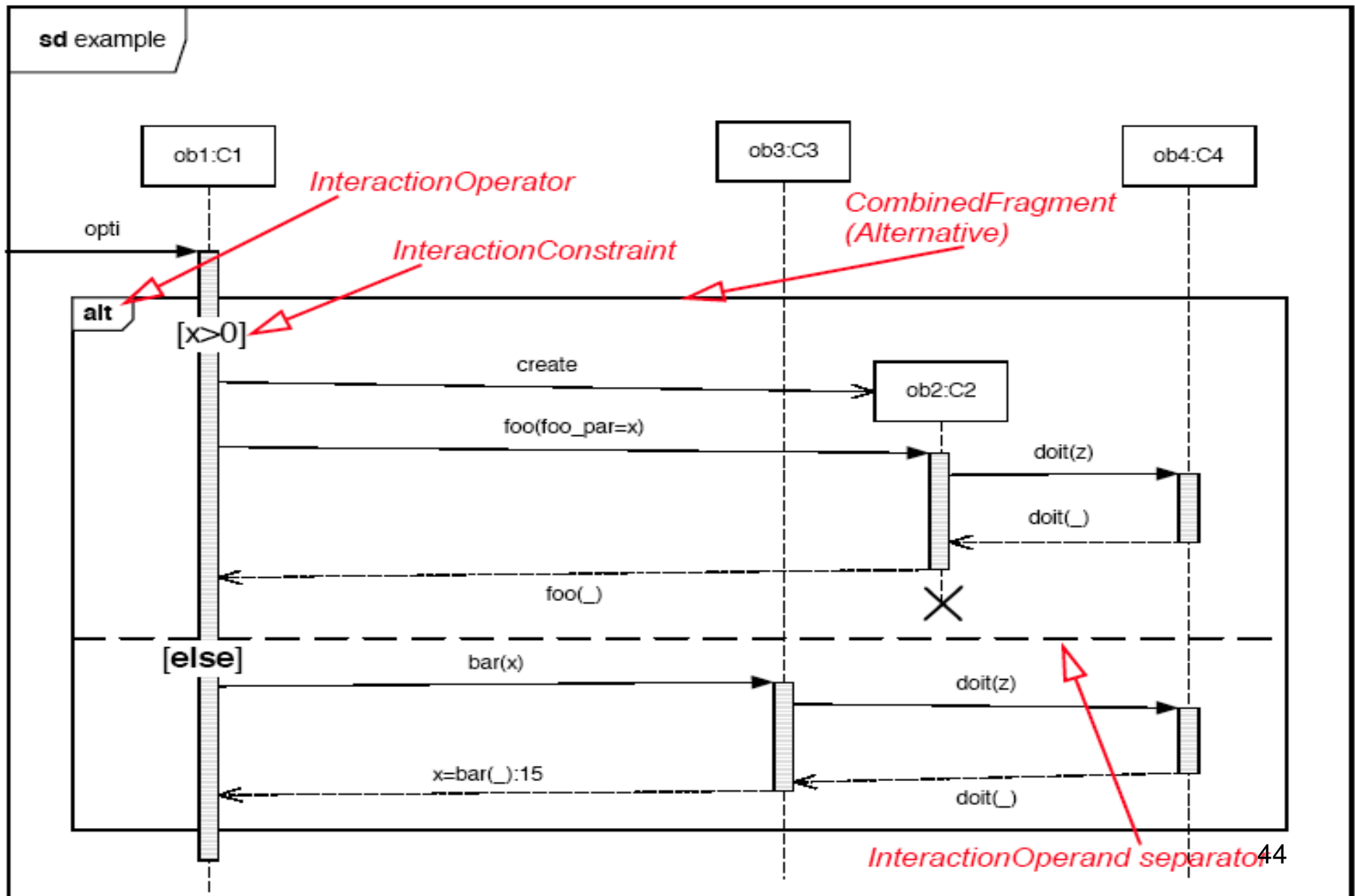
- The interaction operator **alt** designates that the combined fragment represents a choice of behavior.
  - At most one of the operands will be chosen. The chosen operand must have an explicit or implicit guard expression that evaluates to true at this point in the interaction. An implicit true guard 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.

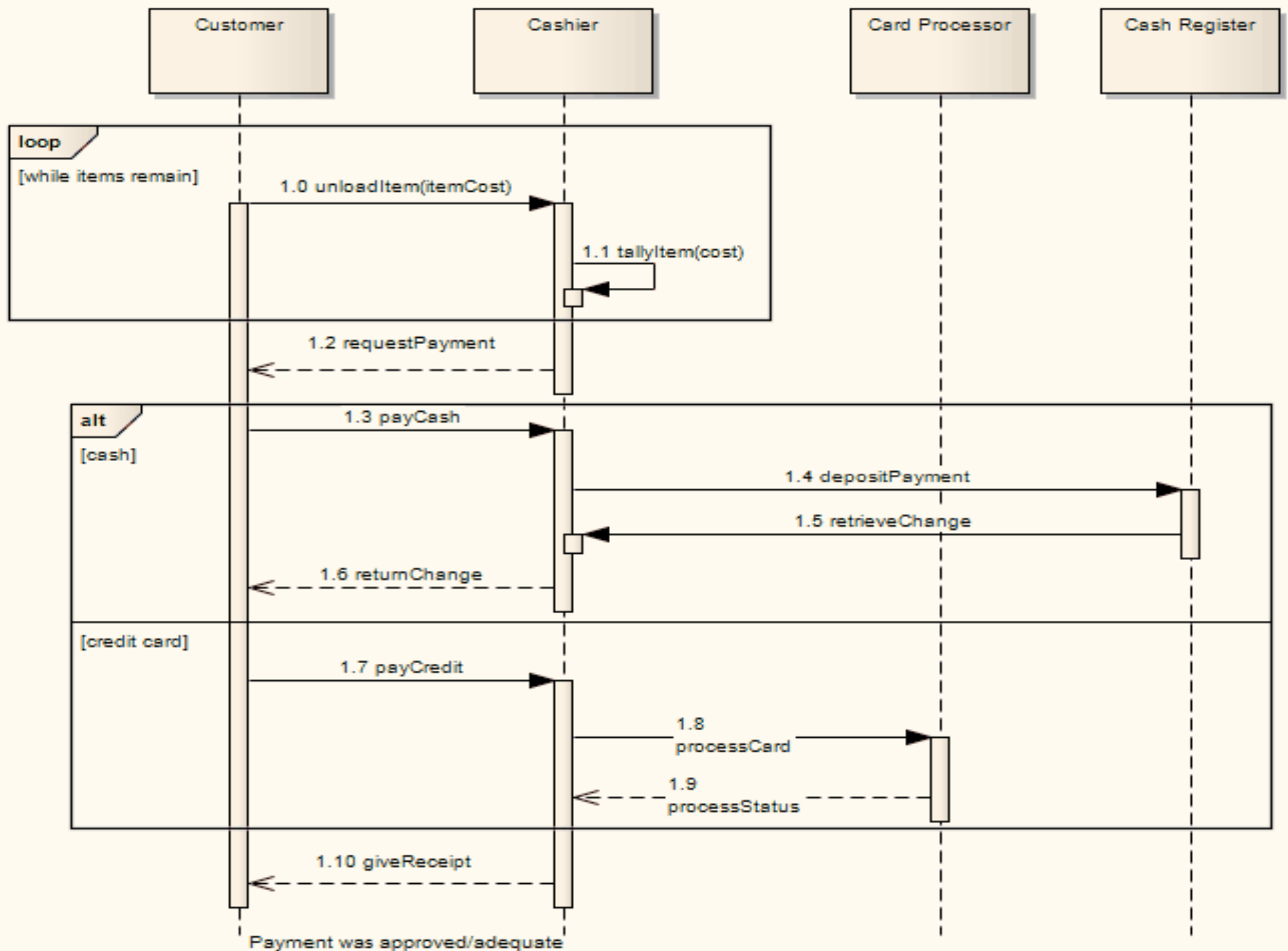
# Alt and Else Operators

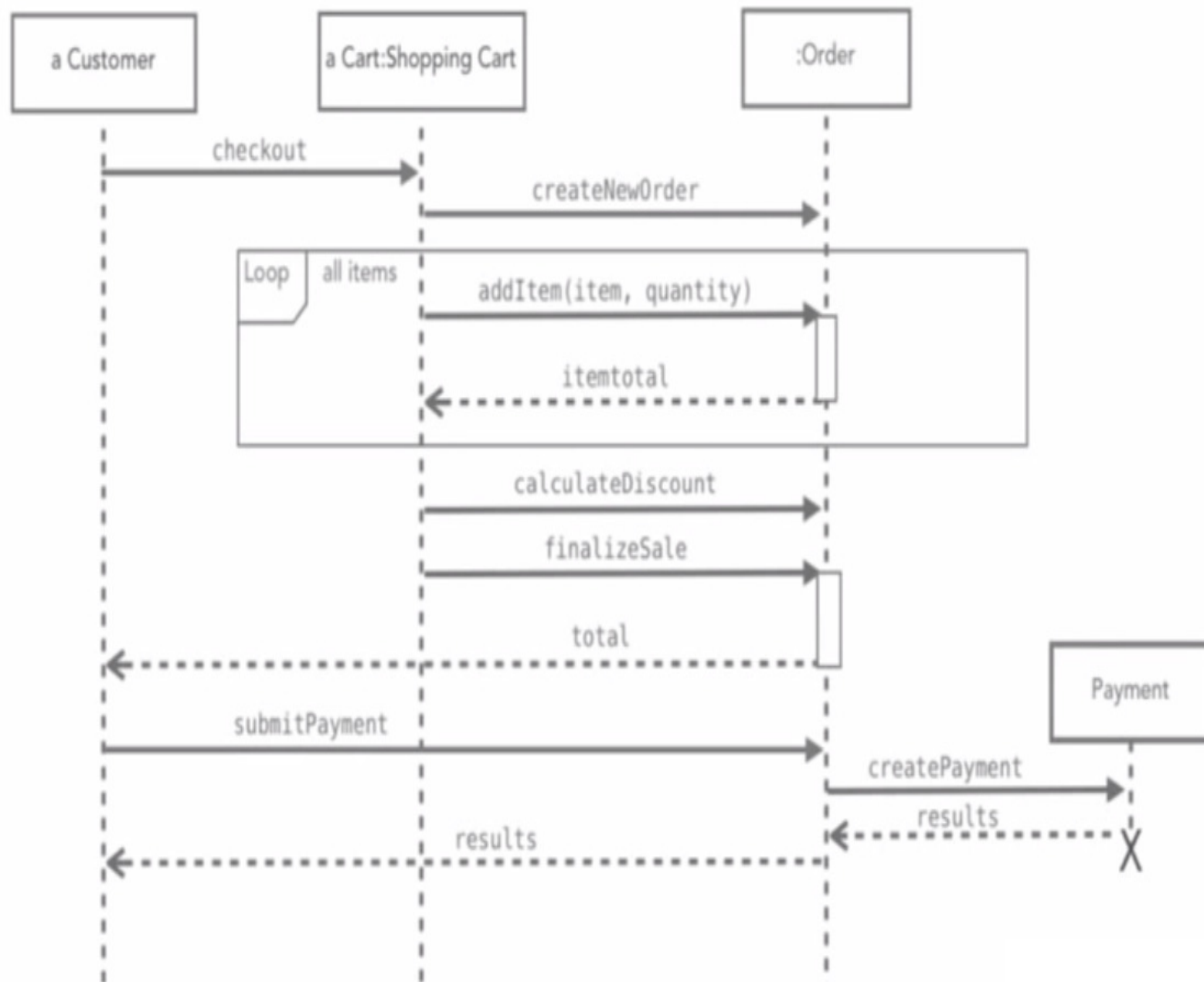
- An operand guarded by **else** designates a guard that is the negation of the disjunction of all other guards in the enclosing combined fragment.
  - If none of the operands has a guard that evaluates to true, none of the operands are executed and the remainder of the enclosing interaction fragment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



# Example of a Combined Fragment







# 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*
  - *When the guard of the break operand is false, the break operand is ignored and the rest of the enclosing interaction fragment is chosen. .*

# Opt Operator

