

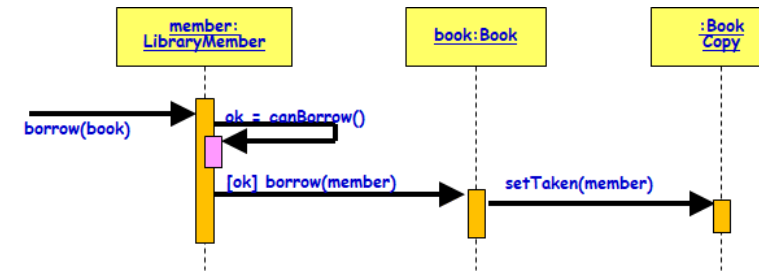
Interaction Diagrams

What are Interaction Diagrams?

- Can model the way a group of objects collaborate to realize some behaviour.
- How many interaction diagrams to draw for a development project?
 - Typically each interaction diagram realizes behaviour of a single use case
 - Draw one sequence diagram for each use case.

Interaction Diagrams

- Three kinds:
 - Interaction overview diagram
 - **Sequence** and **Collaboration** (now **communication** diagram in UML 2.0) diagrams.
- Sequence and Collaboration diagrams are actually equivalent:
 - But, portray different perspectives
- **As we shall see:**
 - These diagrams play a very important role in any design process.



Interaction Overview Diagram

- Focuses on the overview of the flow of control of the interactions.
- It is a variant of the Activity Diagram:
 - Nodes are sequence diagrams.
 - Describes the interactions where messages and lifelines are hidden...

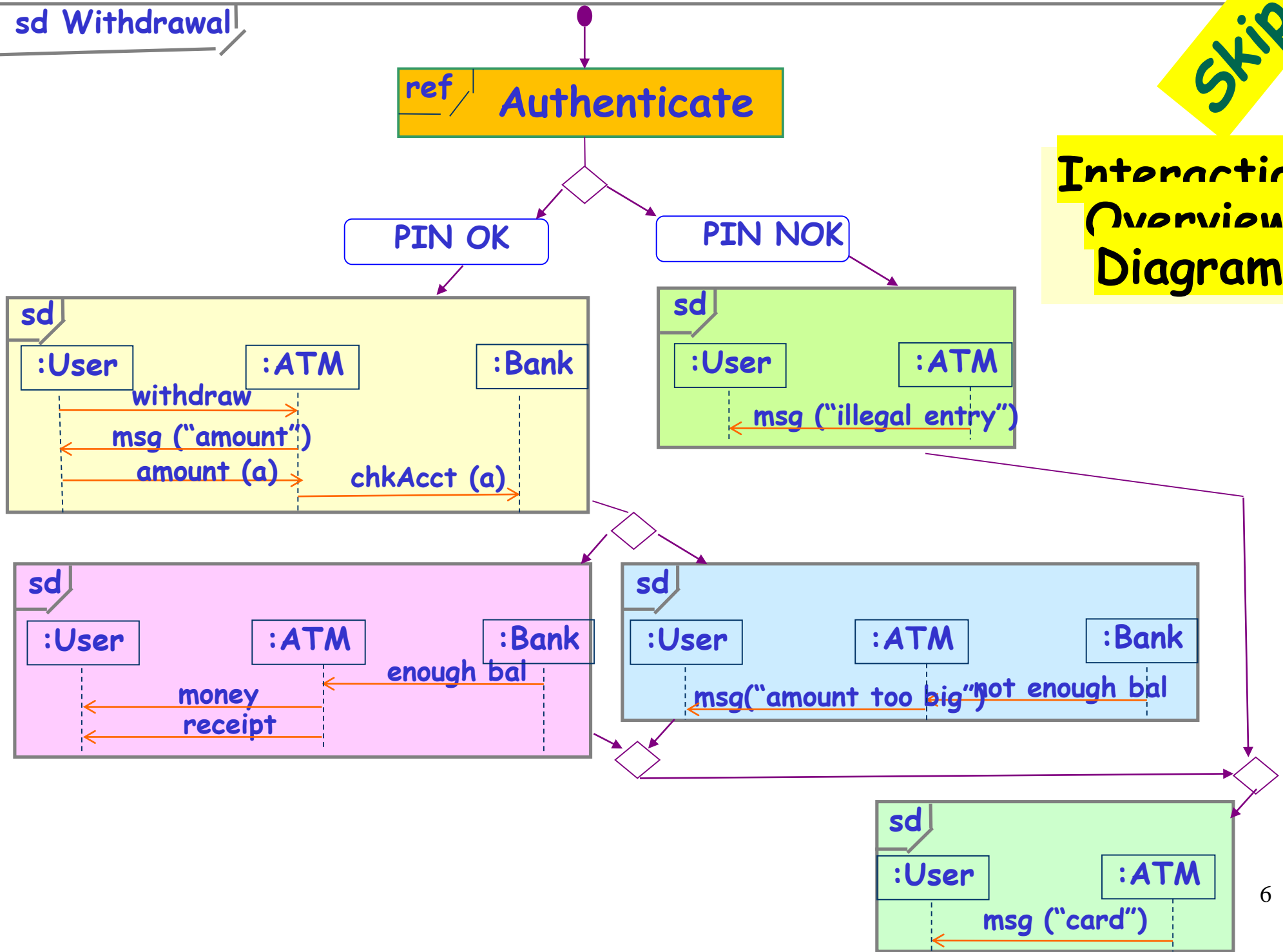
Interaction Overview Diagram

- Each individual activity is pictured as a frame:
 - Can contain nested interaction diagrams.
- Makes the interaction overview diagram useful to **deconstruct a complex scenario**:
 - Otherwise would require multiple if-then-else paths to be illustrated as a single sequence diagram"

sd Withdrawal

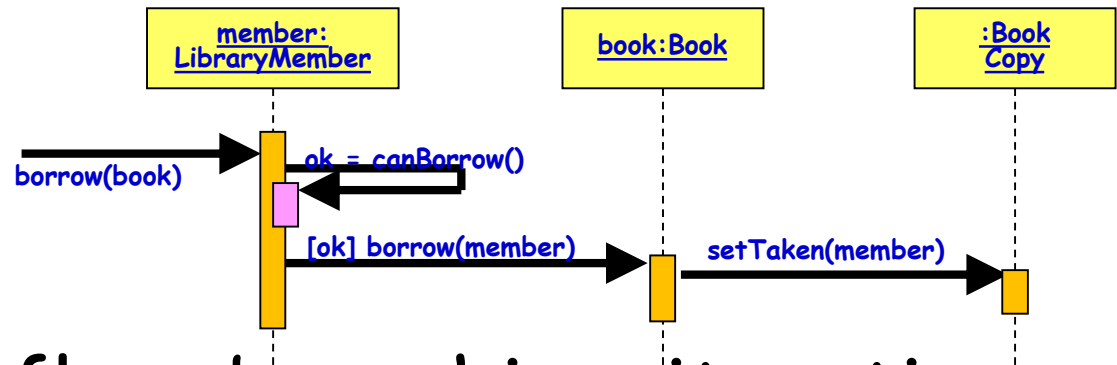
Skip

Interaction Overview Diagram

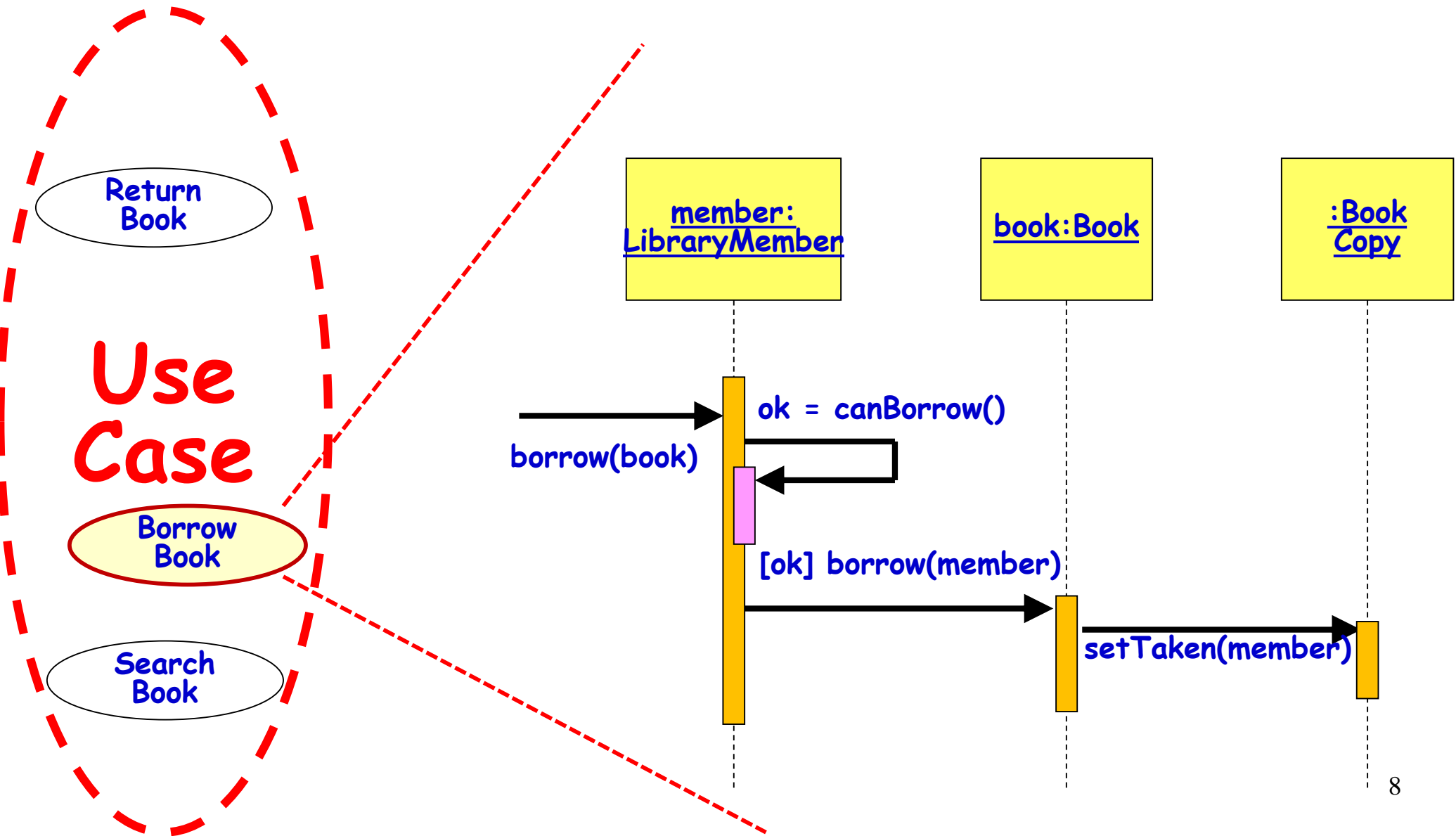


A First Look at Sequence Diagrams

- Captures how objects interact with each other:
 - To realize some behavior.
- Time ordering of messages.
- Can model:
 - Simple sequential flow, branching, iteration, recursion, and concurrency.

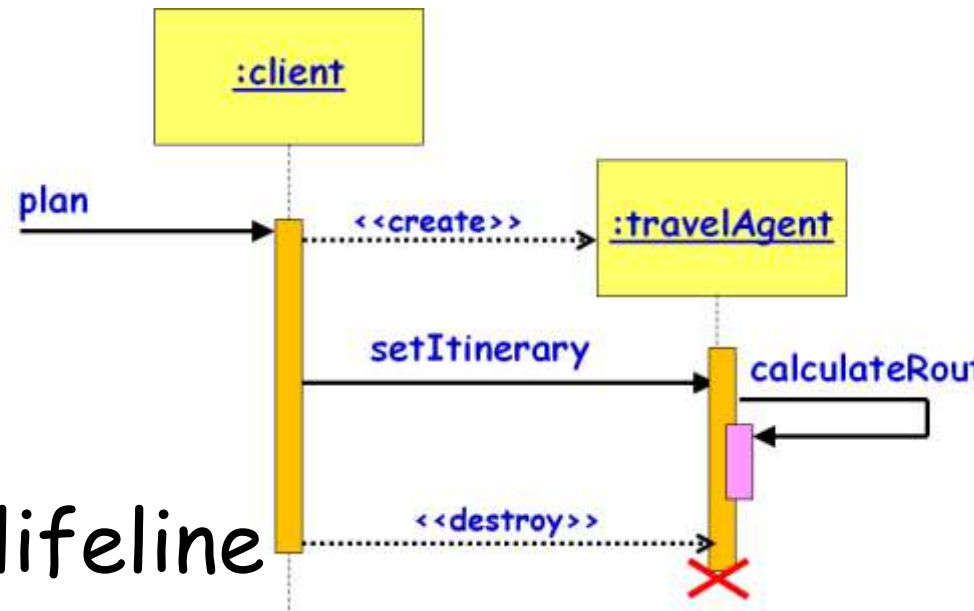


Simple Rule: Develop One Sequence diagram for every use case



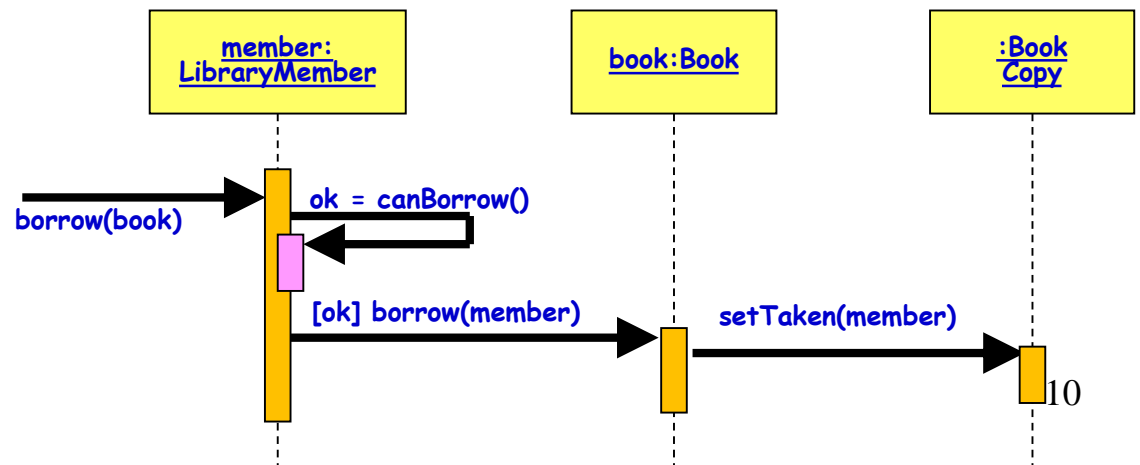
Sequence Diagram

- Shows interaction among objects as a two-dimensional chart
- **Objects** are shown as **boxes** at top
- If object created during execution:
 - Then shown at appropriate place in time line
- **Object existence** is shown as **dashed lines**
- **Object activeness**, shown as a **rectangle** on lifeline



Elements of A Sequence Diagram

- **Messages** are shown as **arrows**.
- Each message labelled with corresponding message name.
- Each message can be labelled with some **control information**.
- Two types of control information:
 - **condition** ([])
 - **iteration** (*)



Gist of Syntax

- iteration marker `*[for all objects]`

- [condition]**

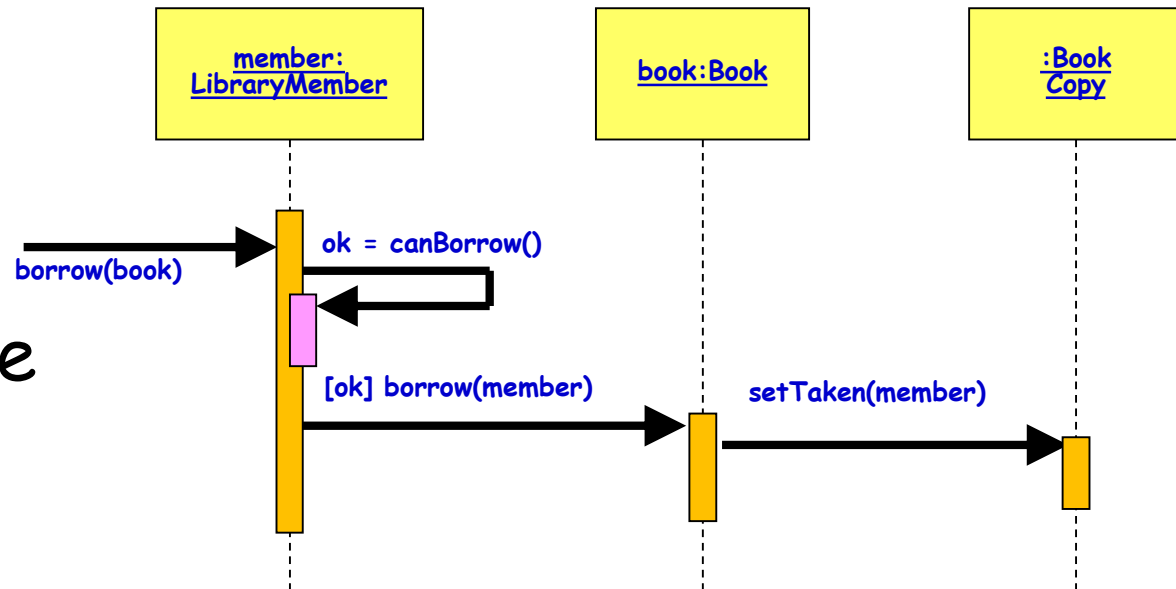
- Message is sent only if the condition is true

- self-delegation** ↻

- a message that an object sends to itself

- Loops and conditionals:**

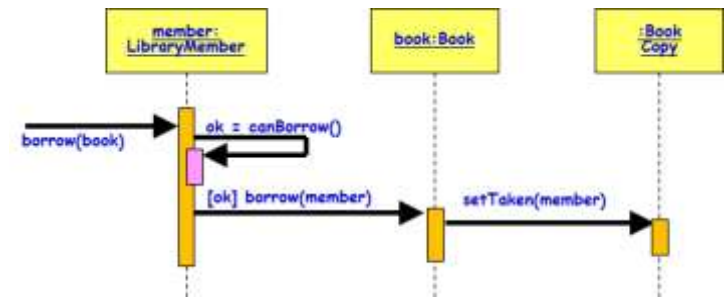
- UML2 uses a new notation called interaction frames to support these



Control logic in Interaction Diagrams

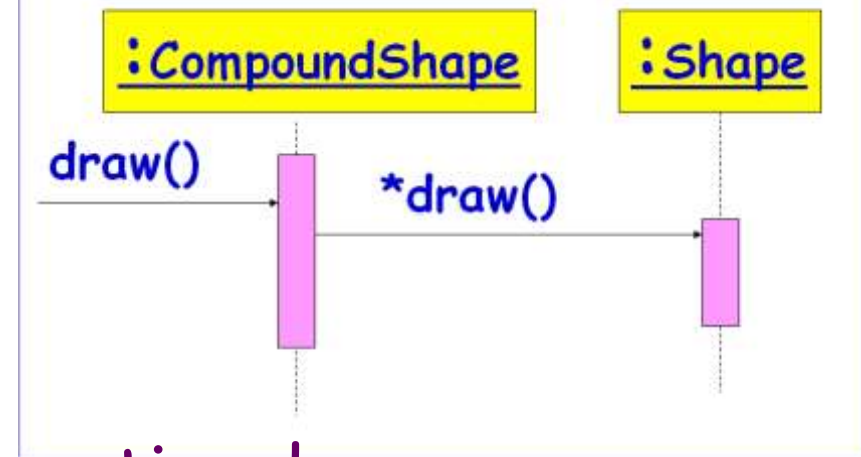
- **Conditional Message**

- [variable = value] message()
- Message is sent only if clause evaluates to *true*



- **Iteration (Looping)**

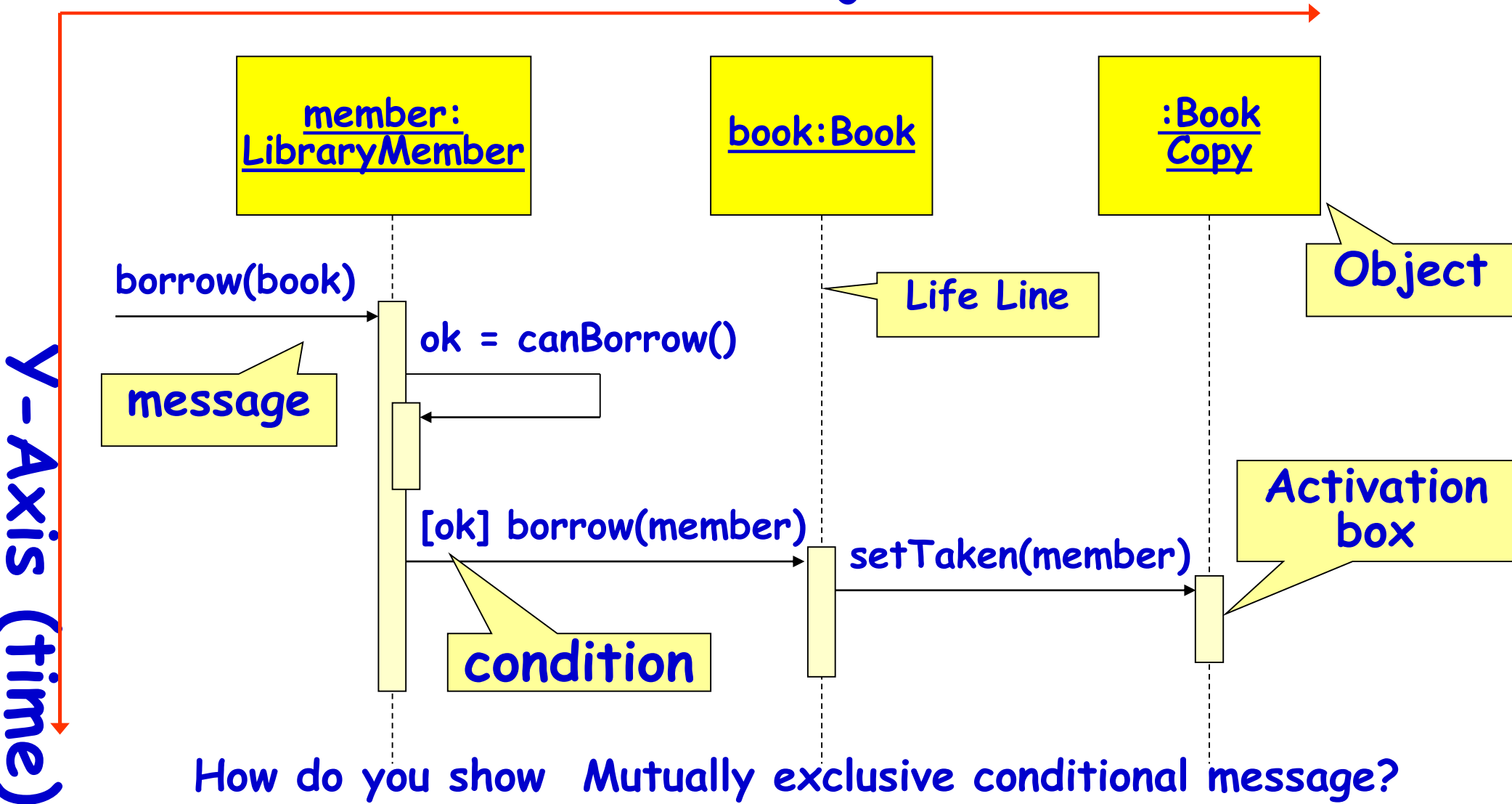
- * [i := 1..N] message()
- "*" is required; [...] clause is optional



- The message is sent many times to possibly multiple receiver objects.

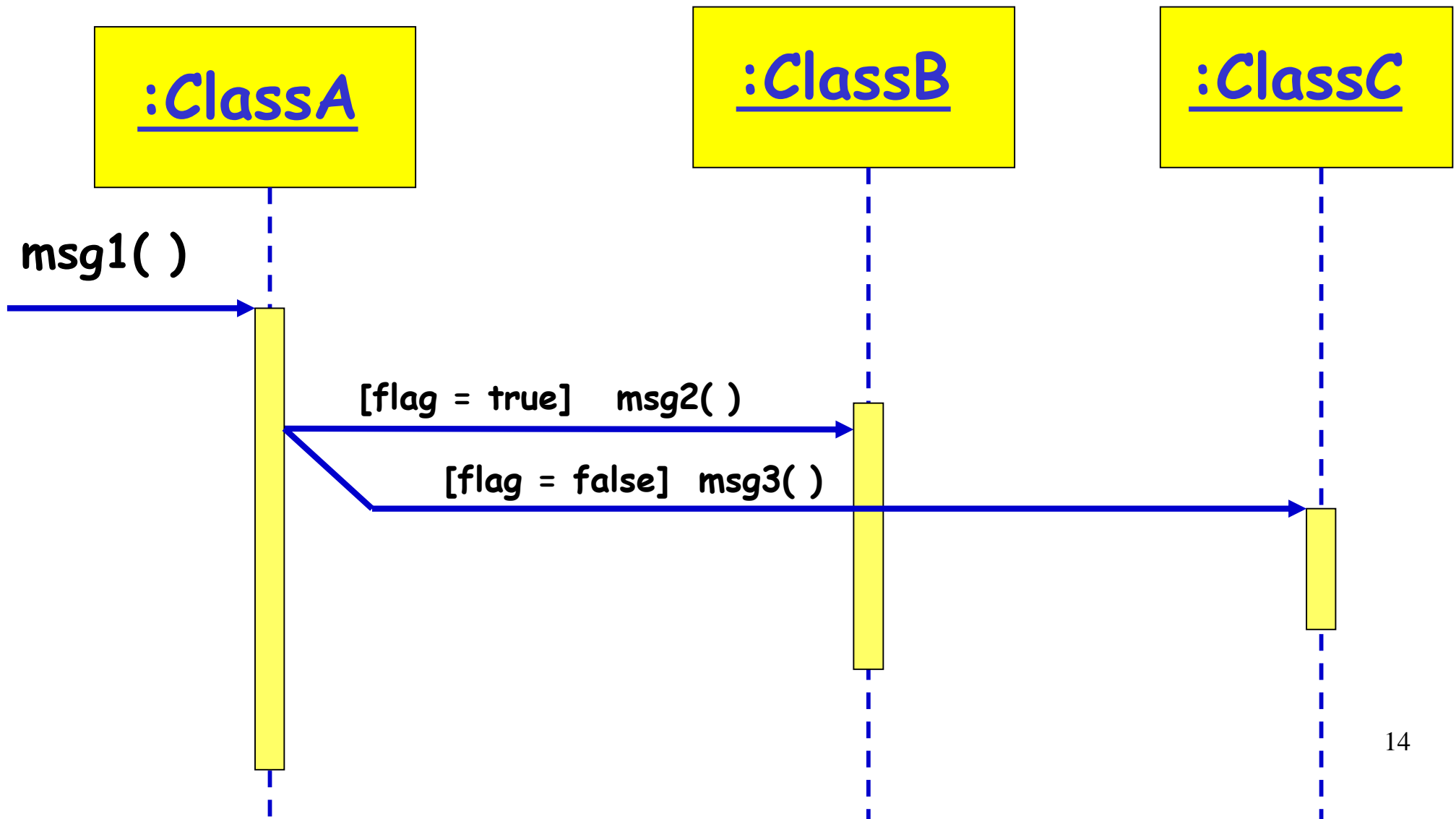
Elements of A Sequence Diagram

X-Axis (objects)



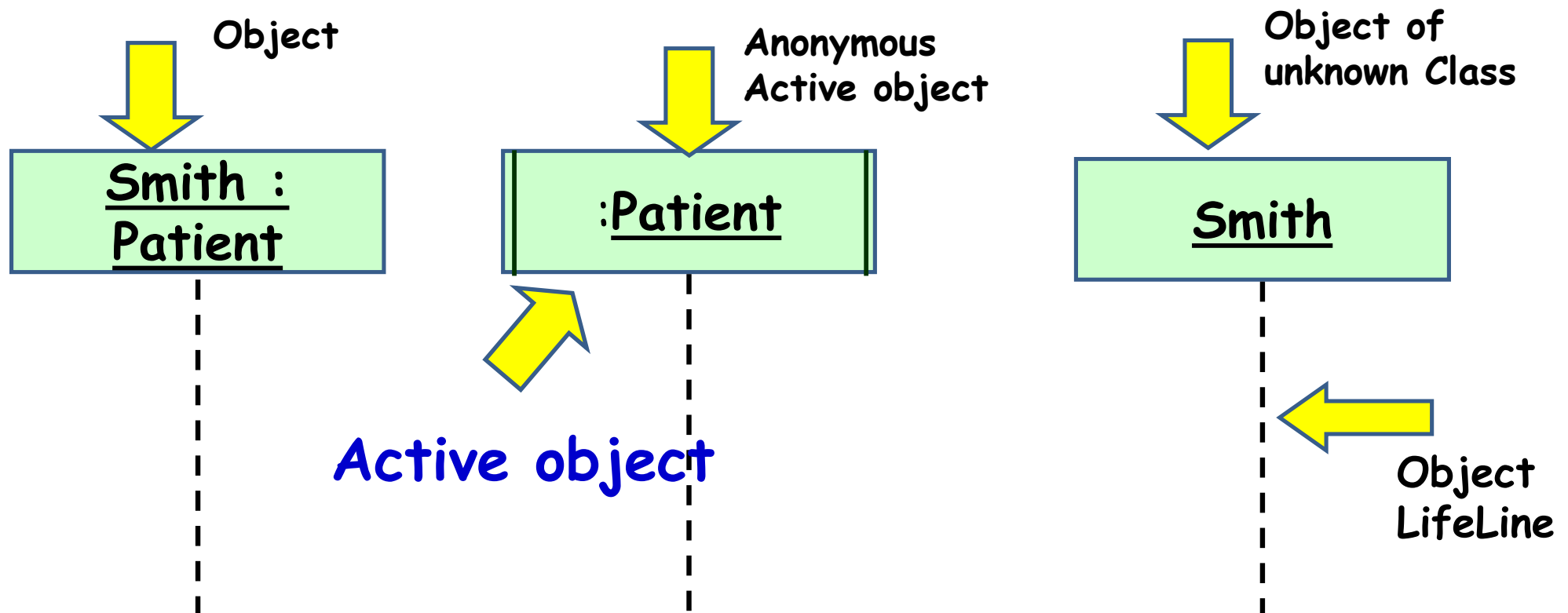
Sequence Diagrams

How to represent Mutually exclusive conditional messages? Eg. msg2 to ClassB or msg3 to classC,



Representing an object

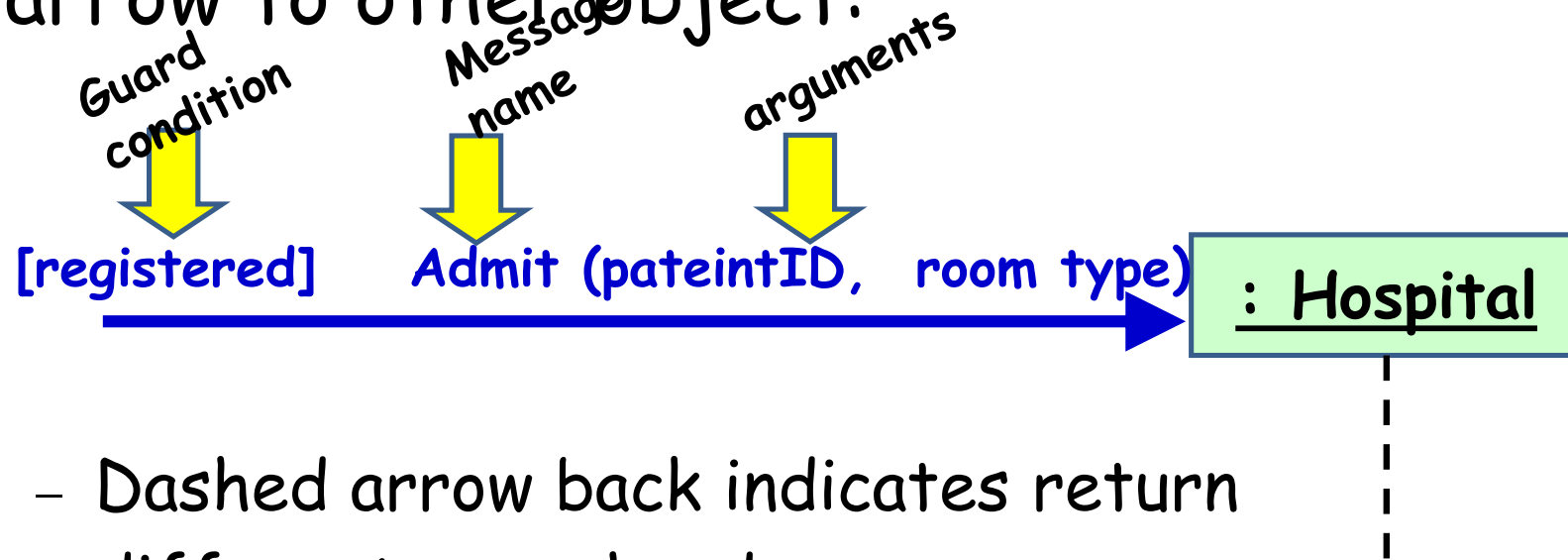
- Class name, preceded by object name and colon:
 - Write object's name if it clarifies the diagram
 - Object's "life line" represented by dashed vertical line



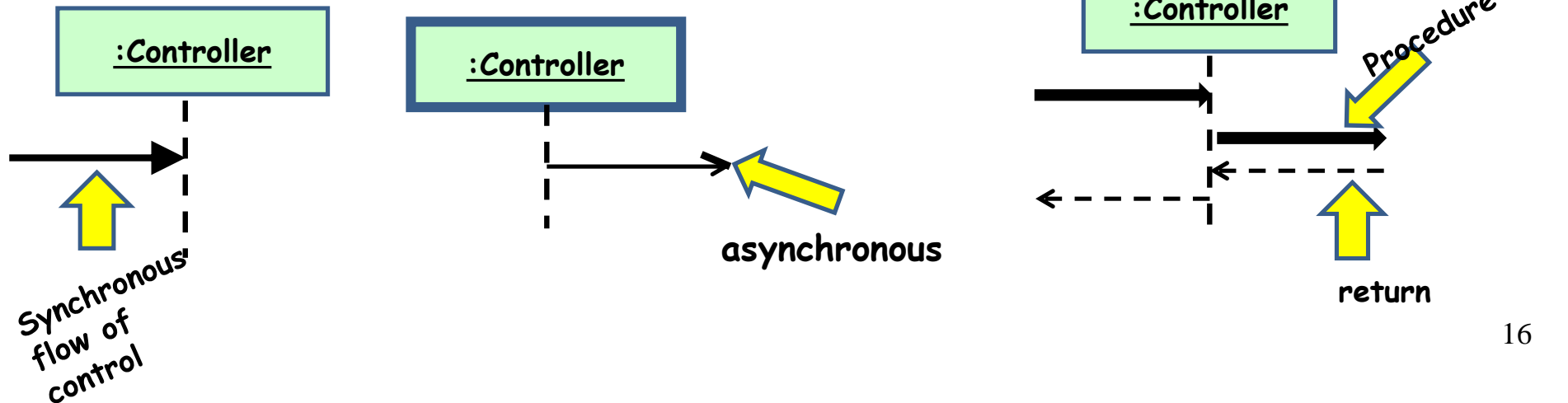
Name syntax: <object name>:<class name>

Messages between objects

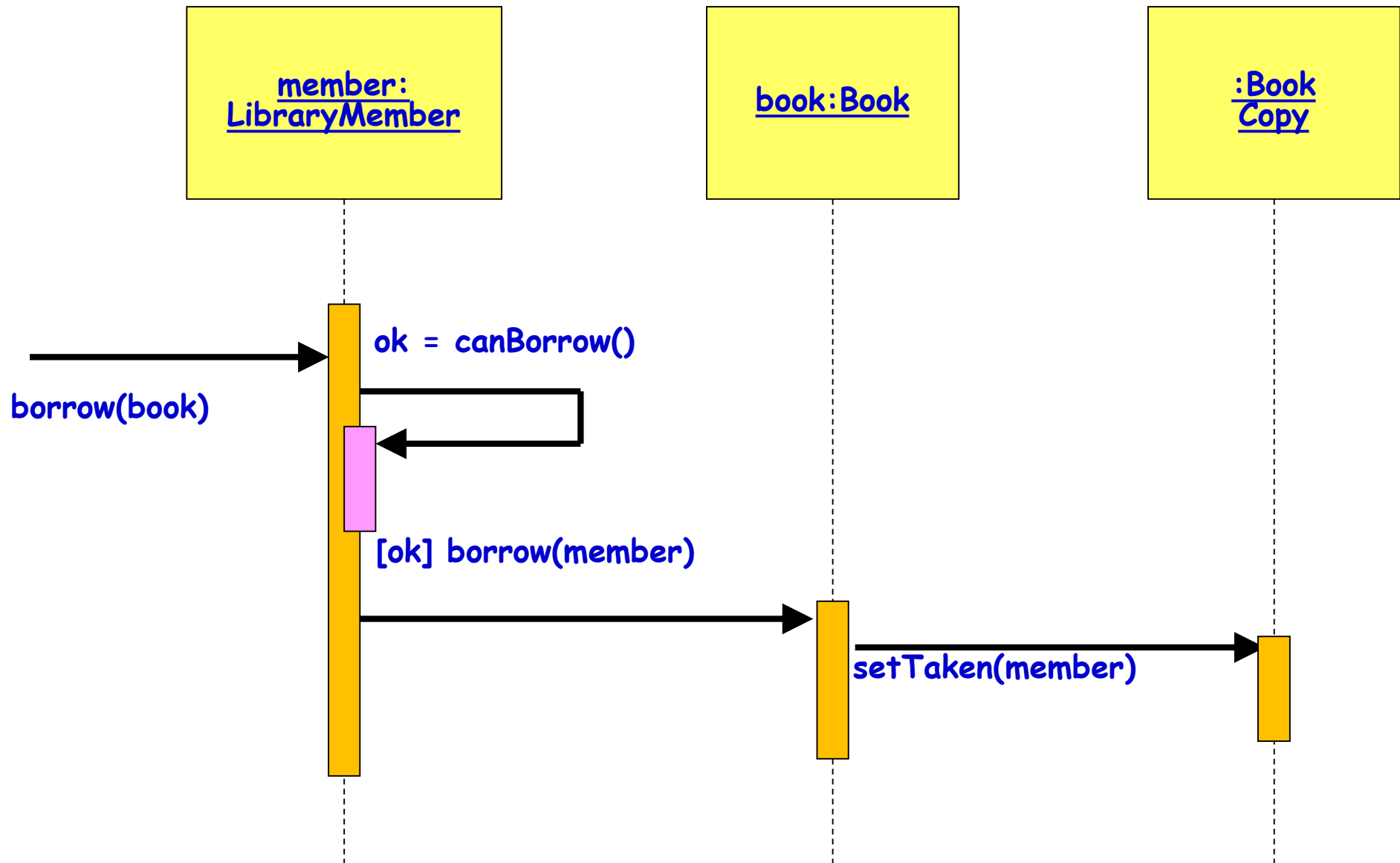
- Message (method call) indicated by a horizontal arrow to other object:



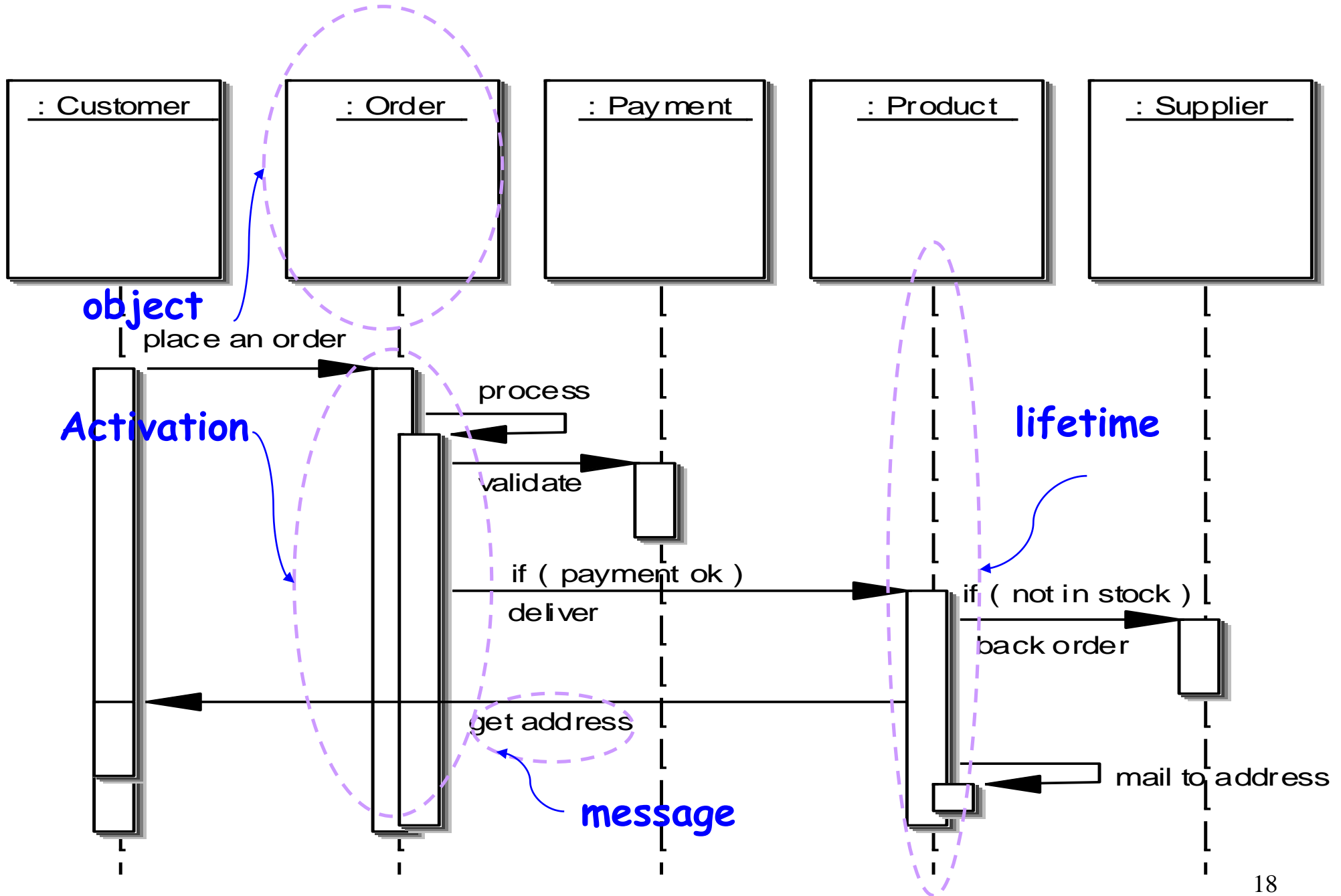
- Dashed arrow back indicates return
- different arrowheads
Sync/Async calls



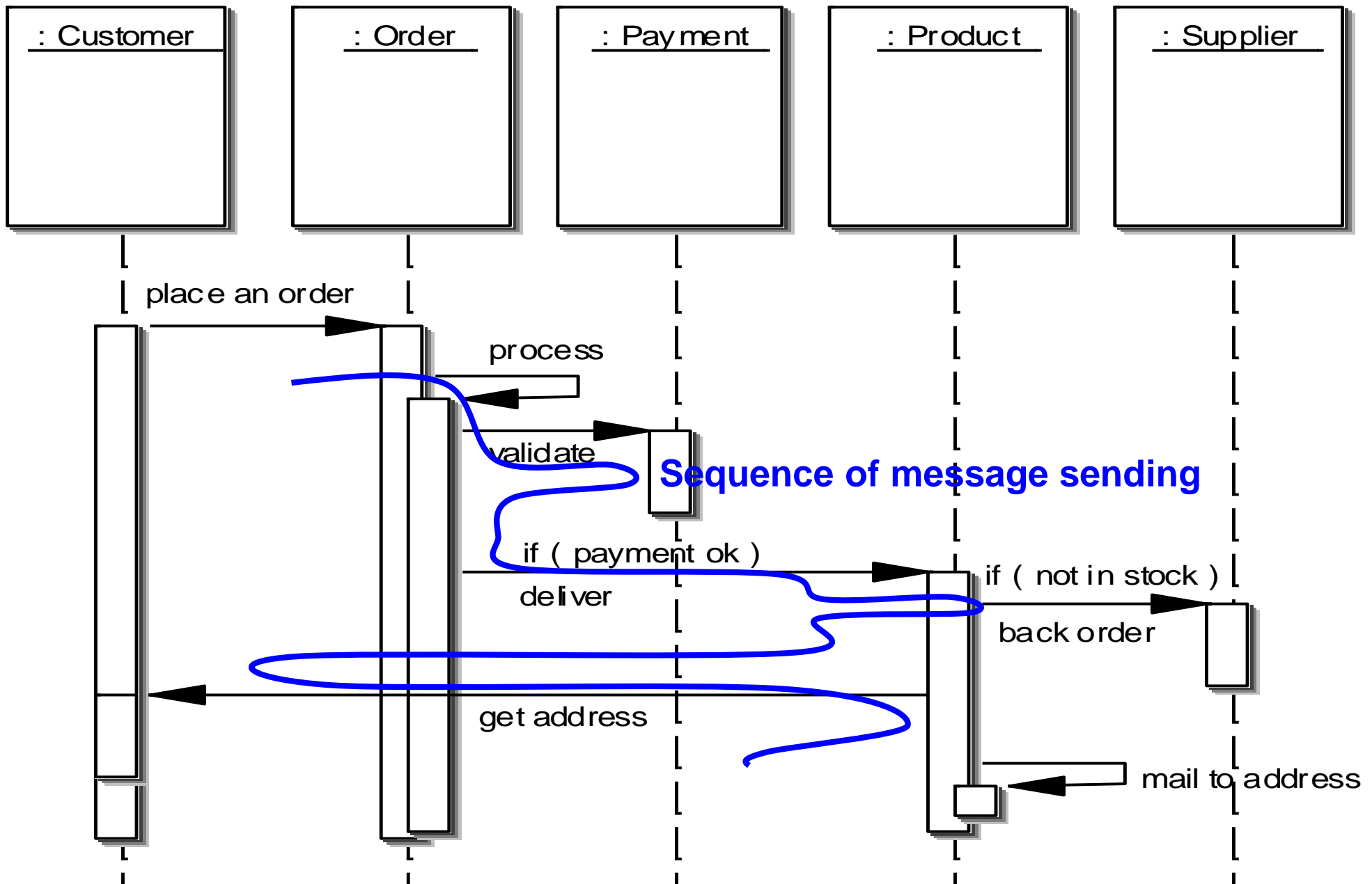
Example Sequence Diagram



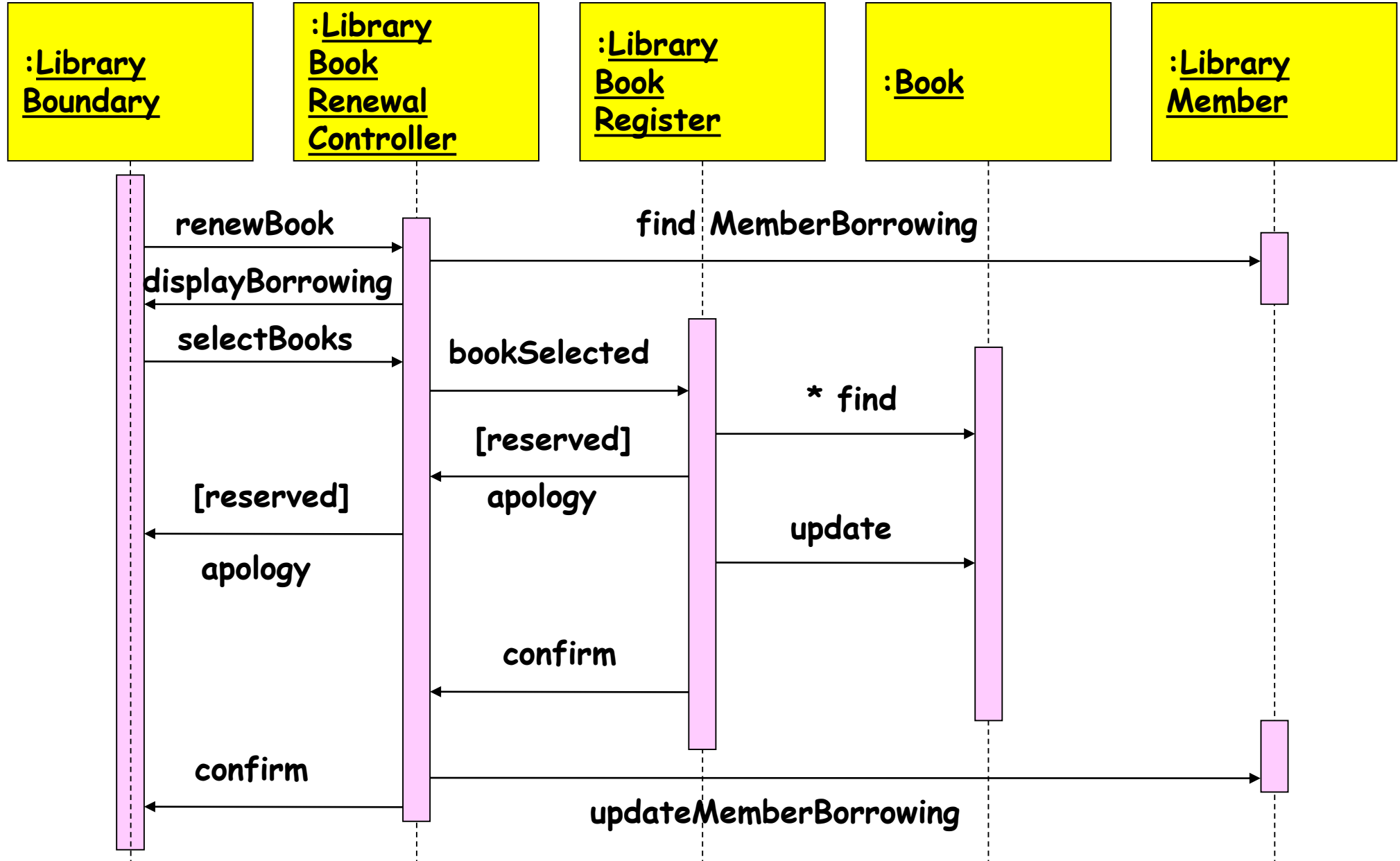
Elements of a Sequence Diagram



Example Cont...







Another Example of A Sequence Diagram



Sequence Diagram for the renew book use case

Message Arrows for Communication

- Message arrows represent the type of communications between two objects in a sequence diagram:
 - **Synchronous** message :
 - Sending object suspends action and waits for the response to the message  (filled head)
 - **Asynchronous** message:
 - Sending object continues with its operations without waiting for the response  (open head)
 - A **return of control** from the synchronous message 
 - A **creation** of a new entity 

Asynchronous Message Example

```
timer = new Timer(ONE_SECOND, new TimerListener());  
  
timer.start();  
  
class TimerListener implements ActionListener {  
    public void actionPerformed(ActionEvent evt) {  
        progressBar.setValue(task.getCurrent());  
        if (task.done()) {  
            Toolkit.getDefaultToolkit().beep();  
            timer.stop();  
            startButton.setEnabled(true);  
        }  
    }  
}
```

- An example is Timer handling in Java:
 - An object that implements **ActionListener** may register itself as a listener for a Timer object
 - The Timer object will store a handle to the object,
 - Later, when the Timer expires, an event will be issued to the object's 'actionPerformed' method
- If the object had to wait for the timer to expire:
 - **It could have wasted a long period of potential computation time.**

```
timer = new Timer(ONE_SECOND, new TimerListener());
```

```
. . .
```

```
timer.start();
```

Example

```
class TimerListener implements ActionListener {
```

```
    public void actionPerformed(ActionEvent evt) {
```

```
        progressBar.setValue(task.getCurrent());
```

```
        if (task.done()) {
```

```
            Toolkit.getDefaultToolkit().beep();
```

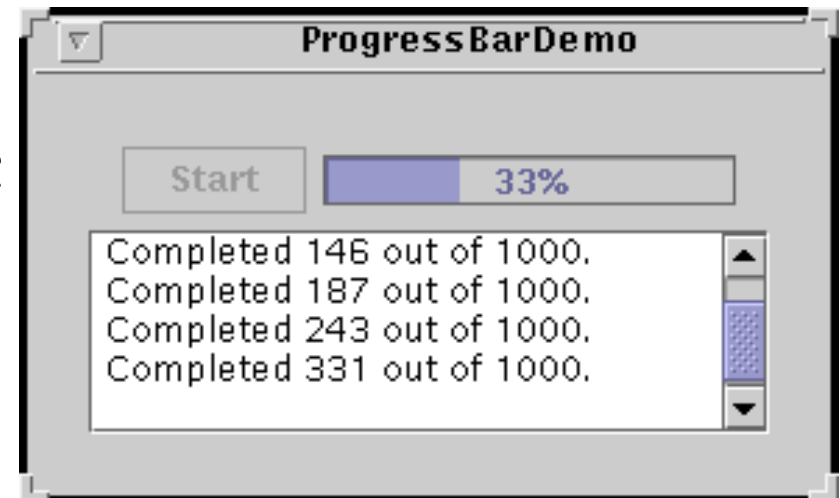
```
            timer.stop();
```

```
            startButton.setEnabled(true);
```

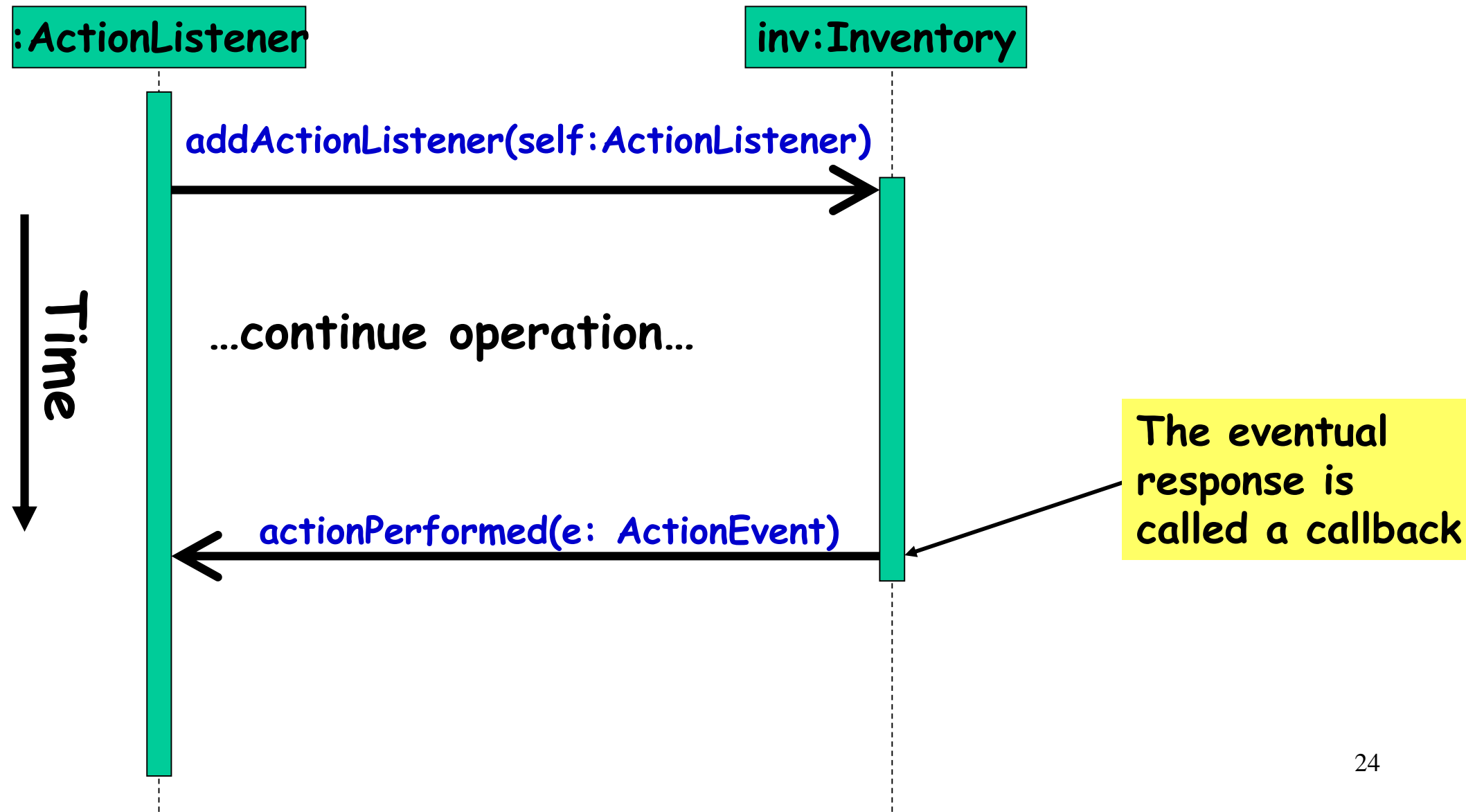
```
        }
```

```
    }
```

```
}
```



Another Asynchronous Message Example



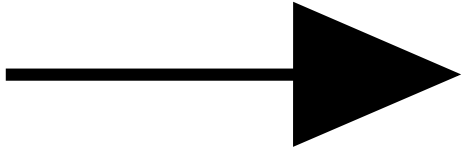
Return Values

- Optional --- indicated using a dashed arrow:
 - Label indicates the return value.
 - Don't need when it is obvious what is being returned, e.g. `getTotal()`



- **Model a return value only when you need to refer to it elsewhere:**
 - **Example:** A parameter passed to another message.

Summary of Kinds of Arrows



Procedure call or other kind of nested flow of control
(The sender loses control until the receiver finishes handling the message)



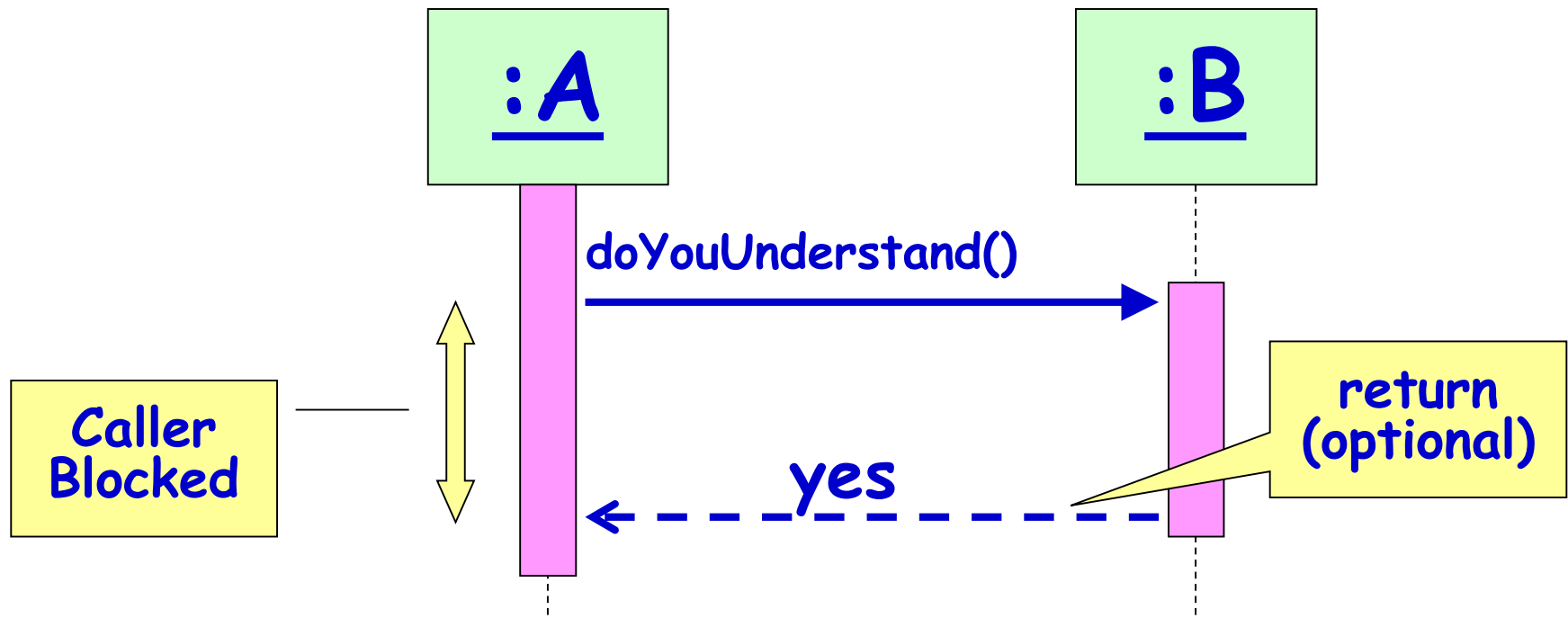
Flat flow of control (The sender does not expect a reply)



Return
(Unblocks a synchronous send)

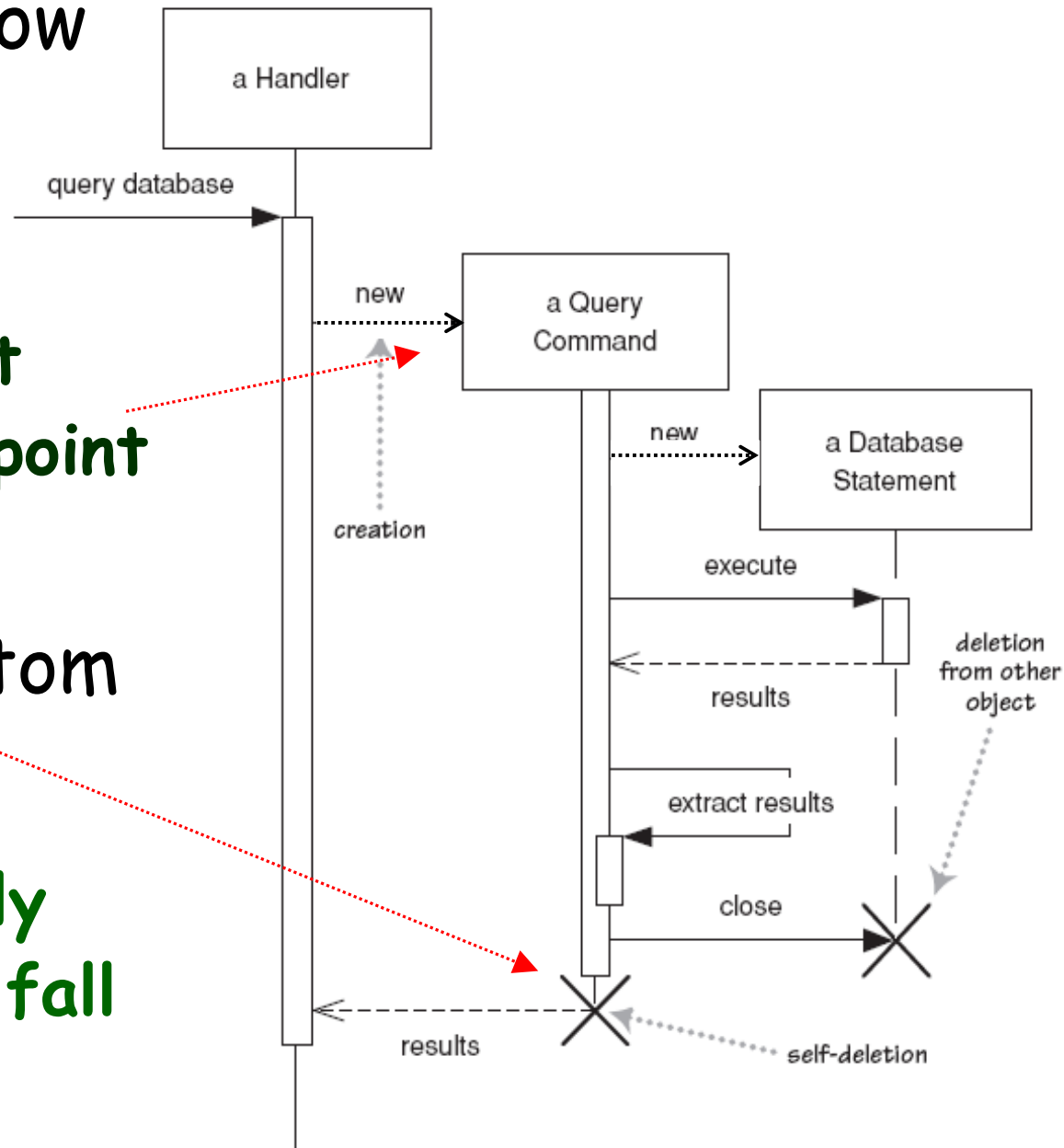
Synchronous Messages

- Flow of control, typically implemented as a method call.
 - The routine that handles the message is completed before the caller resumes execution.



Lifetime of objects

- **creation:** Dotted arrow with 'new' or create written above it
 - Notice that an object created, appears at point of call.
- **deletion:** an X at bottom of object's lifeline
 - Java doesn't explicitly delete objects; they fall out of scope and are garbage-collected



Method calls

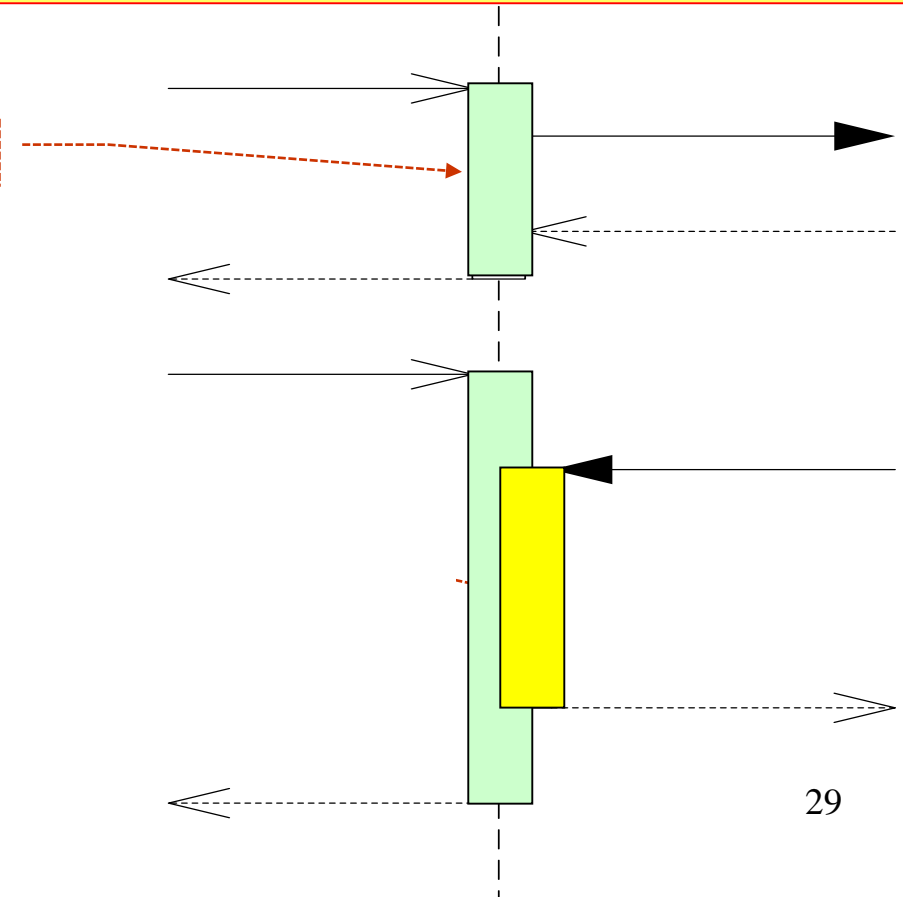
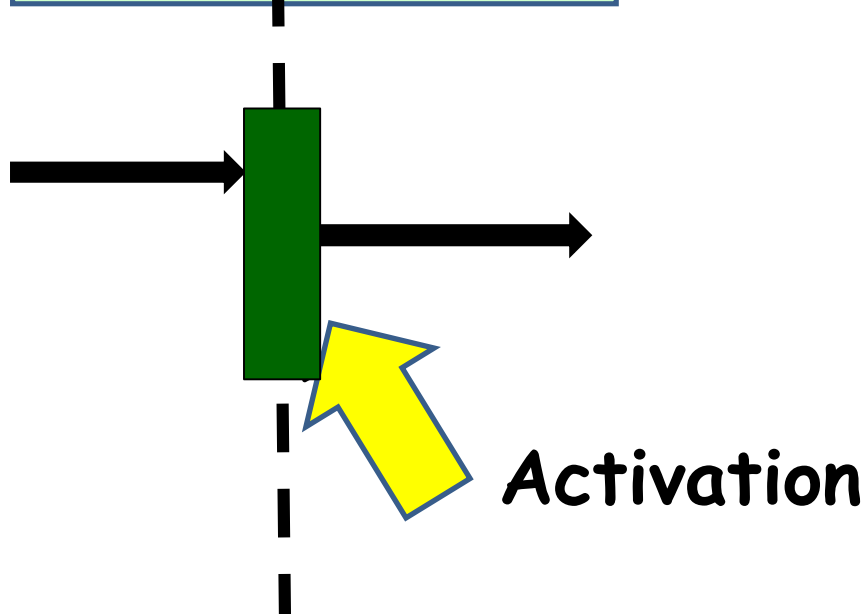
- **Activation:** thick box over object's life line; drawn when object's method is already on the stack

– Either that object is running its code, or it is on the stack waiting for another object's method to finish

– Nest to indicate recursion

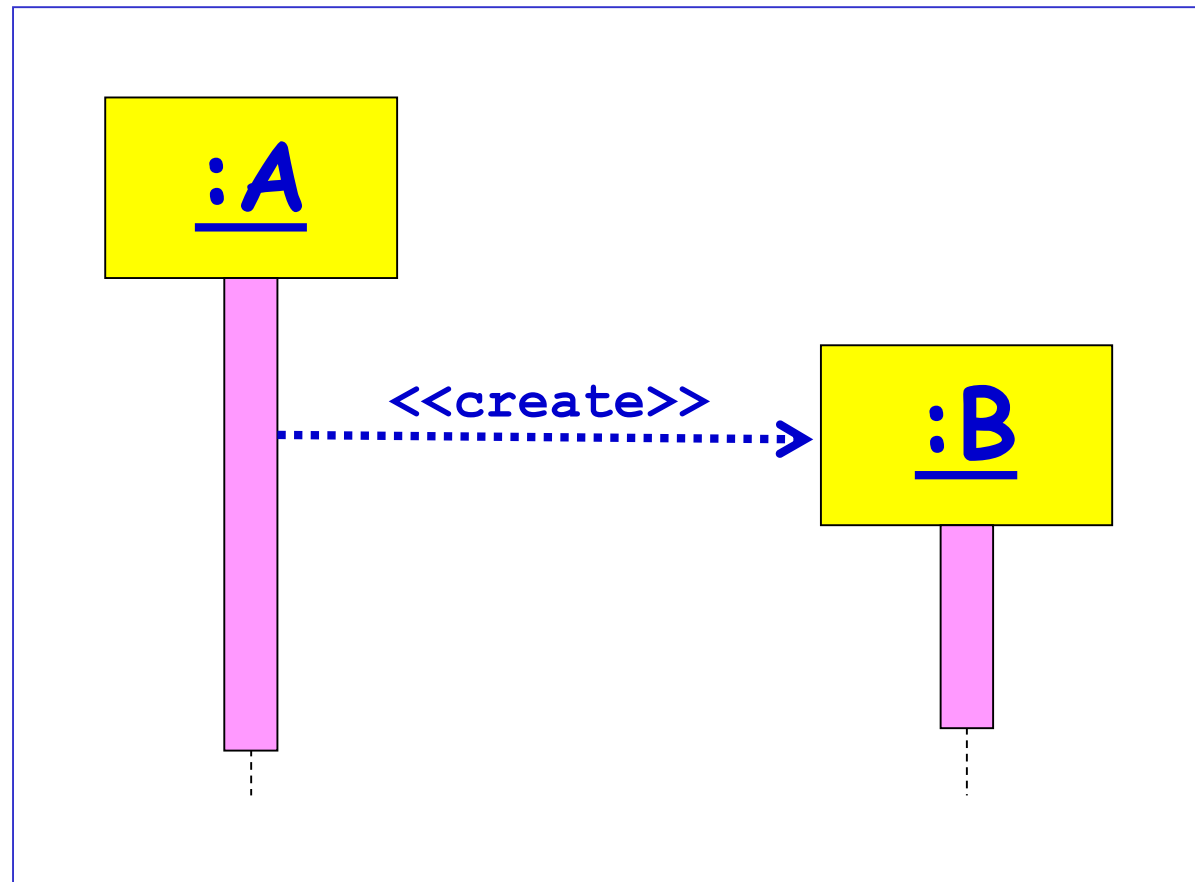
Activation

:Controller



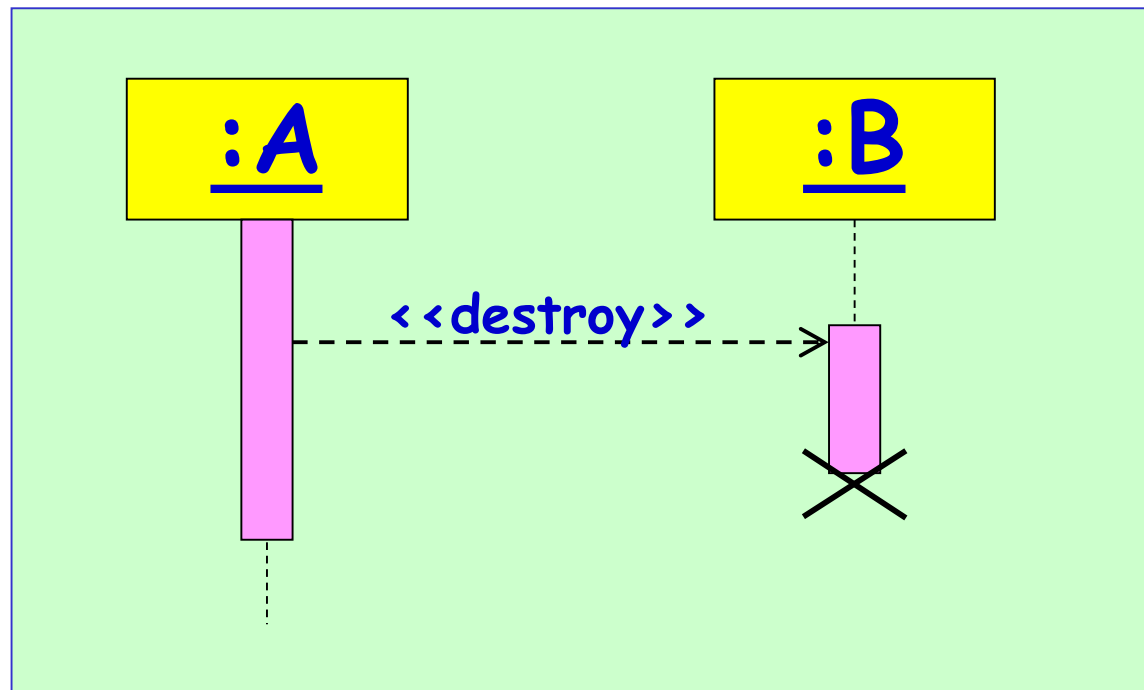
Object Creation

- An object may create another object via a `<<create>>` message.



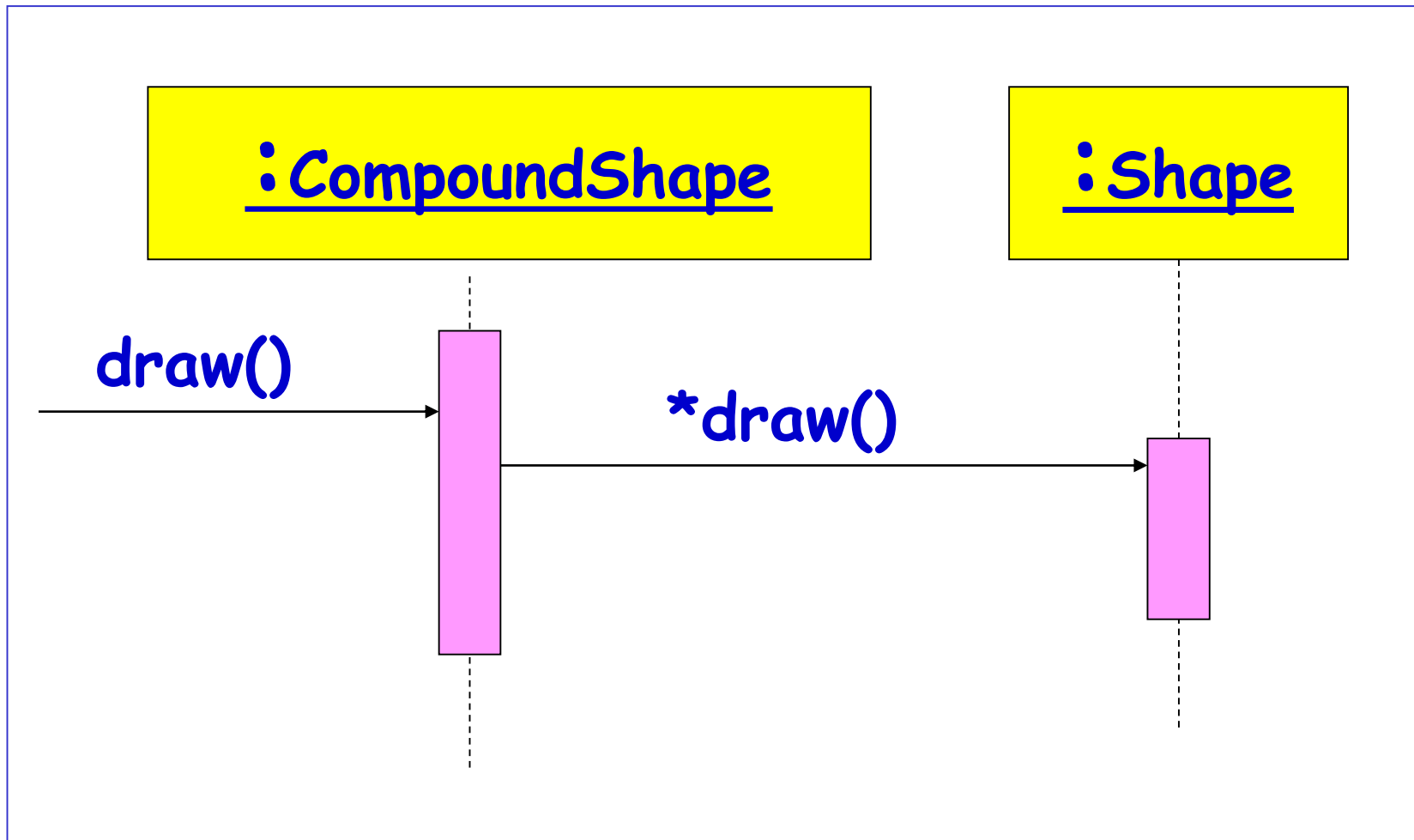
Object Destruction

- An object may destroy another object via a `<<destroy>>` message.
 - An object may also destroy itself.
- But, how do you destroy an object in Java?
 - Avoid modeling object destruction unless memory management is critical.



Looping

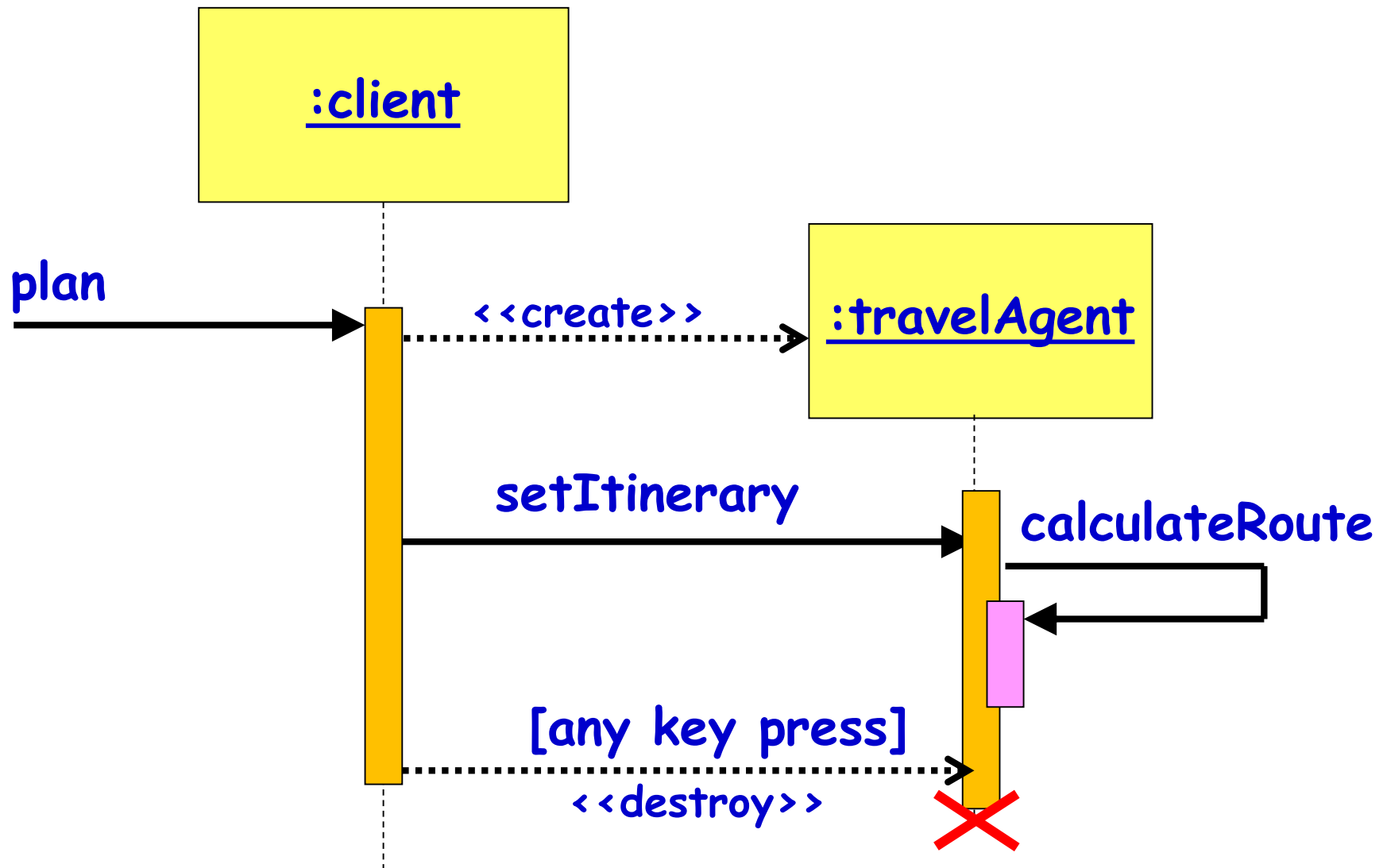
- Iteration example UML 1.x:



Exercise 0

- A user can use a travel portal to plan a travel
- When the user presses the plan button, a travel agent applet appears in his window
- Once the user enters the source and destination,
 - The travel agent applet computes the route and displays the itinerary.
 - Then travel agent applet gets destroyed once the user presses any key

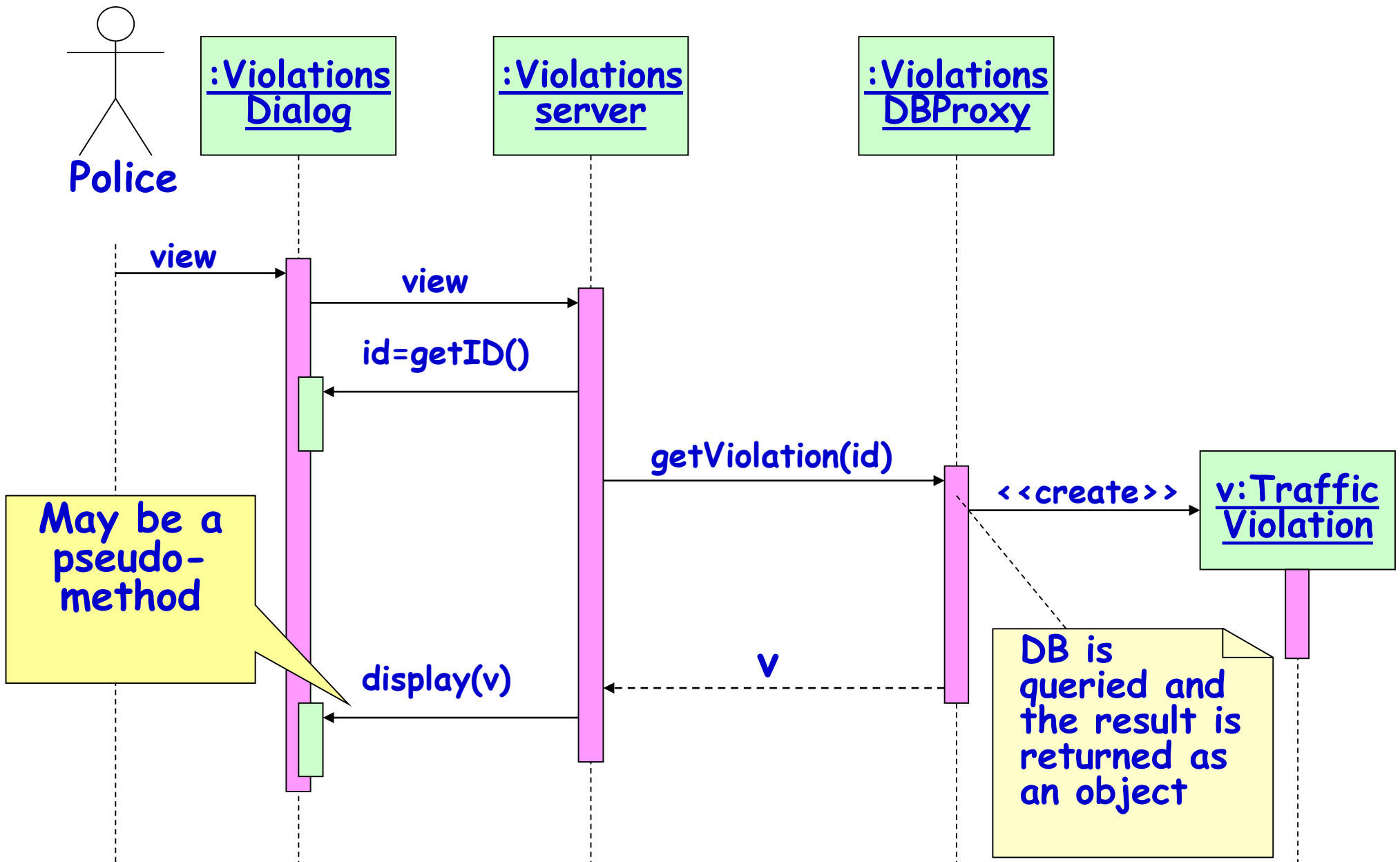
Exercise 0: Solution



Exercise 1

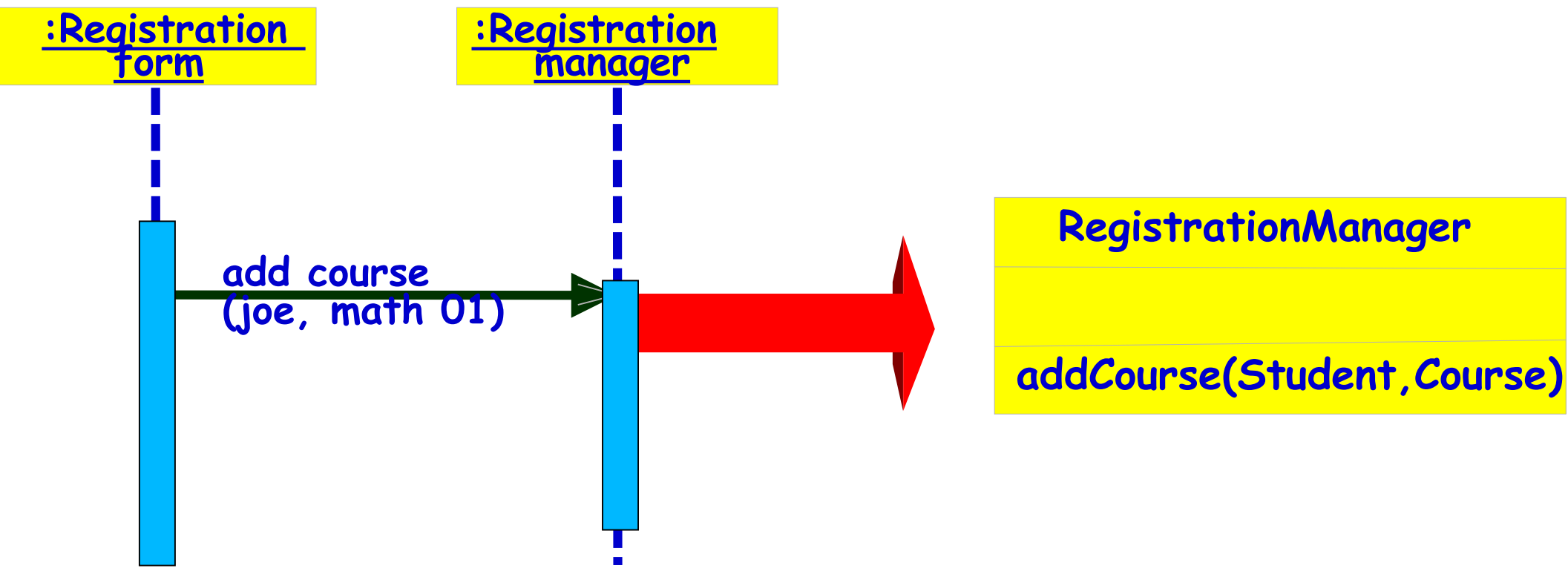
- A traffic police uses a traffic violation app to find the details of any traffic violation by entering the violation id. Details of the interactions are the following:
- The police first presses the view button in the Violation dialog box of the app.
 - This request is conveyed to the server object
 - The server object asks for the violation id
- Once the police enters the violation id, the server queries the DBproxy object using the id
 - Based on the response received from the DBProxy, the server creates a violation object and passes it to the dialog box for display.

Exercise 1: Solution

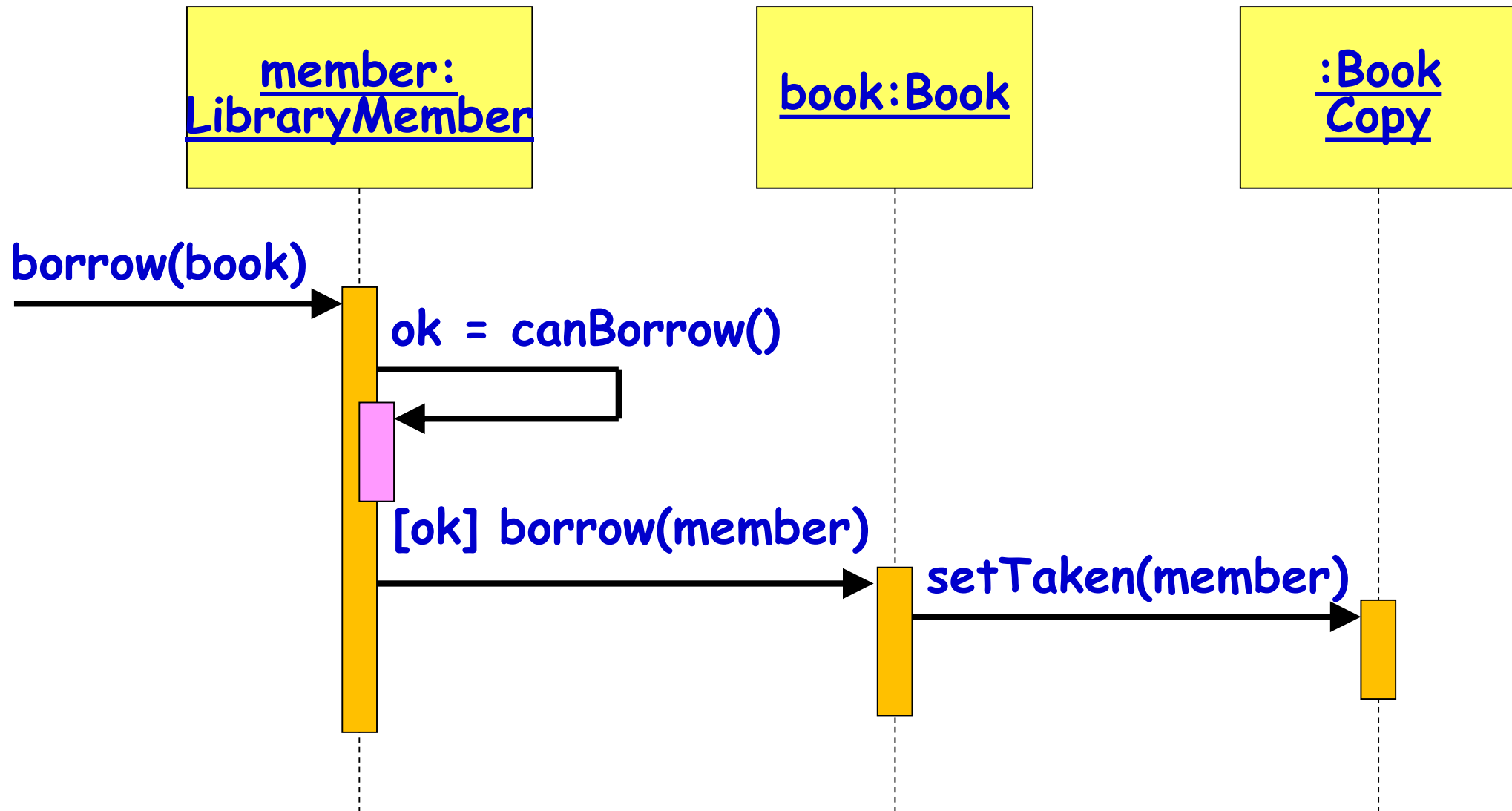


Method Population in Classes

- Which methods should a class have?
 - Methods of a class are determined from the interaction diagrams...



Example Sequence Diagram: Borrow Book Use Case



Sequence Diagram: Frames

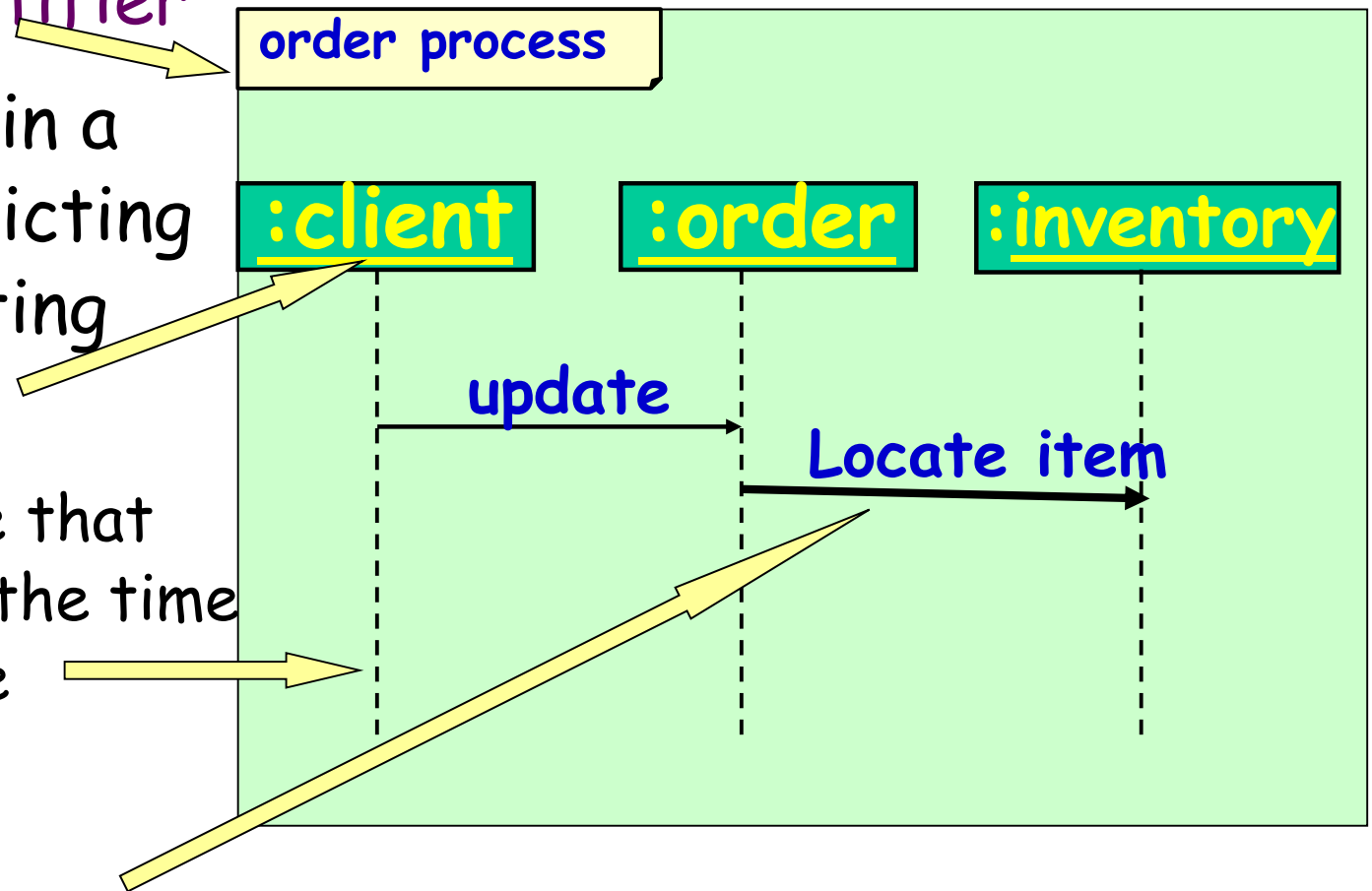
- A **frame** consists of:

1) Diagram **identifier**

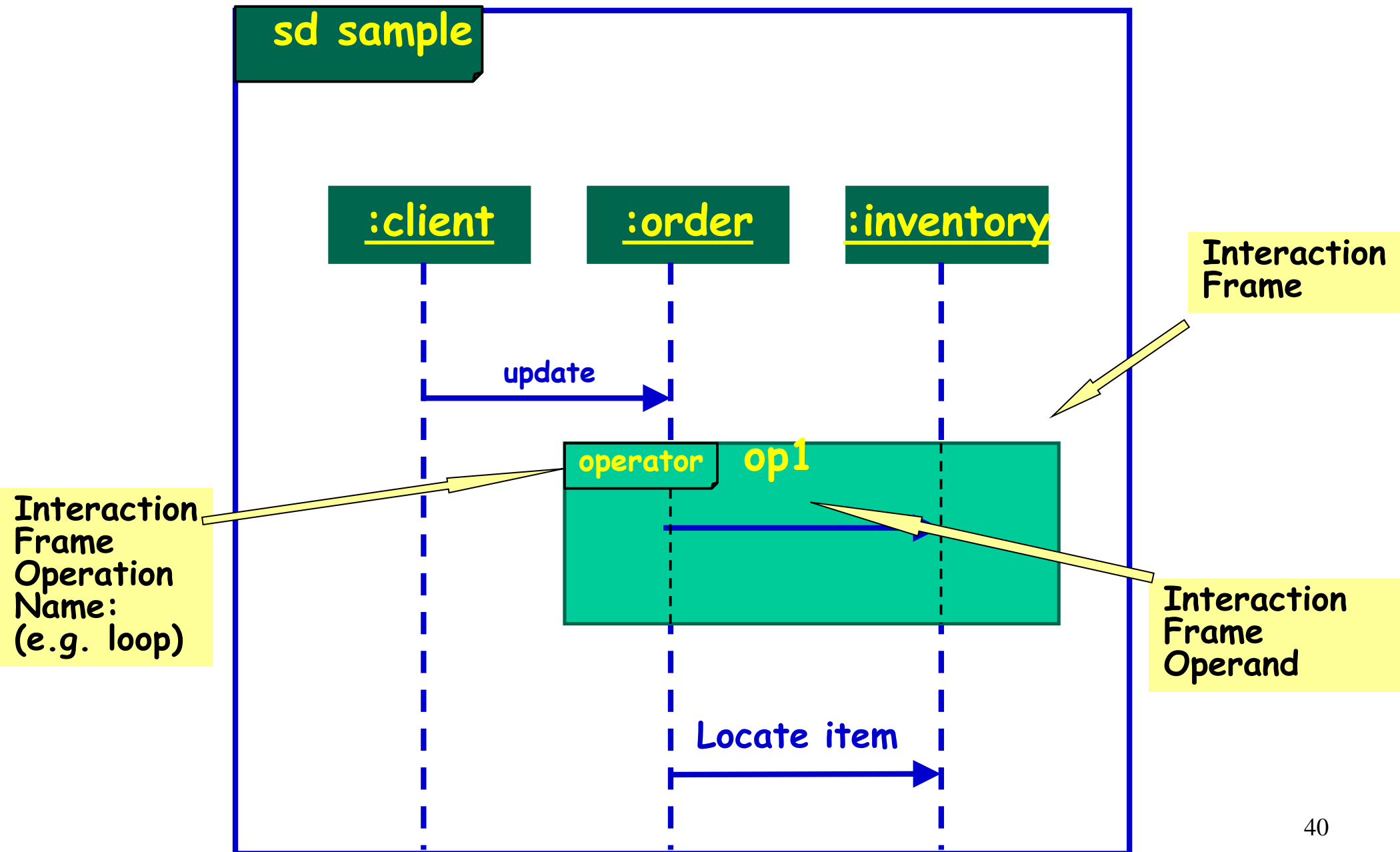
2) Participants in a rectangle depicting the participating objects:

- A dotted line that extends for the time period of the interaction

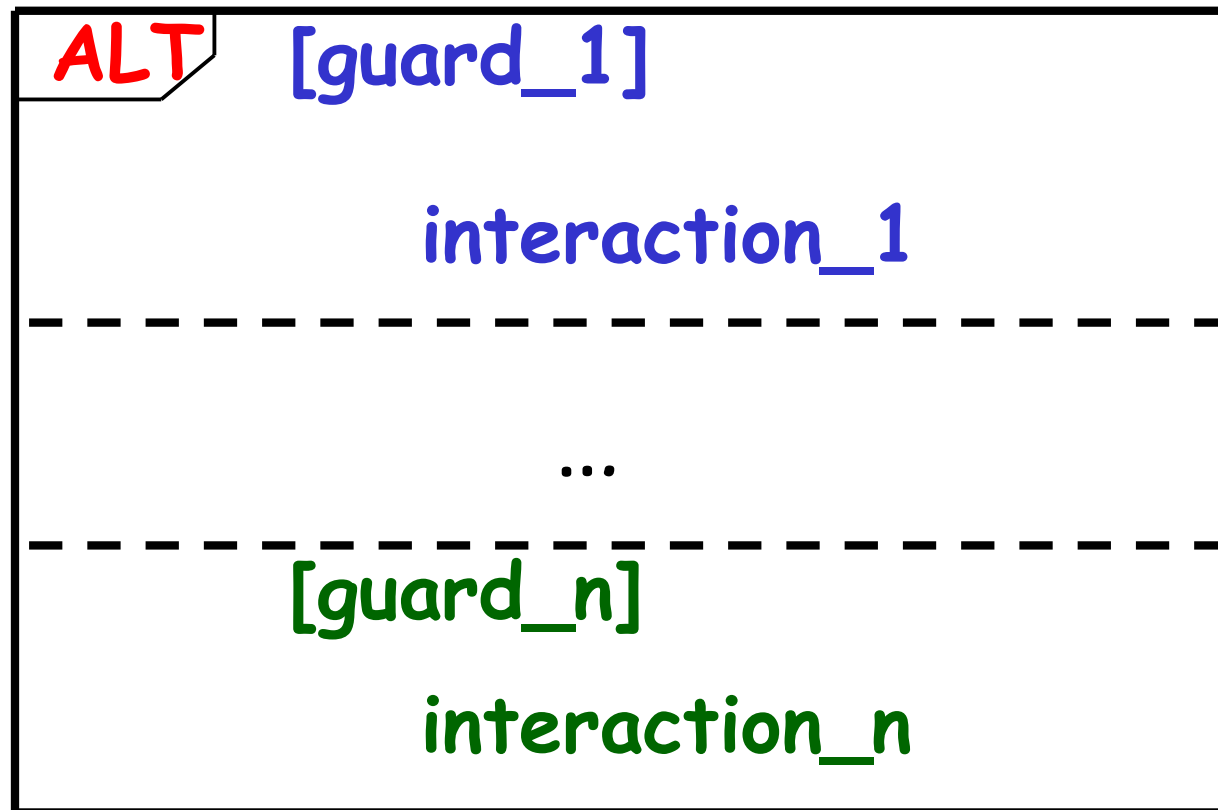
3) **Messages** to communicate among the participants



Depicting a Frame Graphically



Frame Operator



- Divided into a set of interactions by dotted lines
- **interaction_i** is executed if **guard_i** is true

Frame Operators

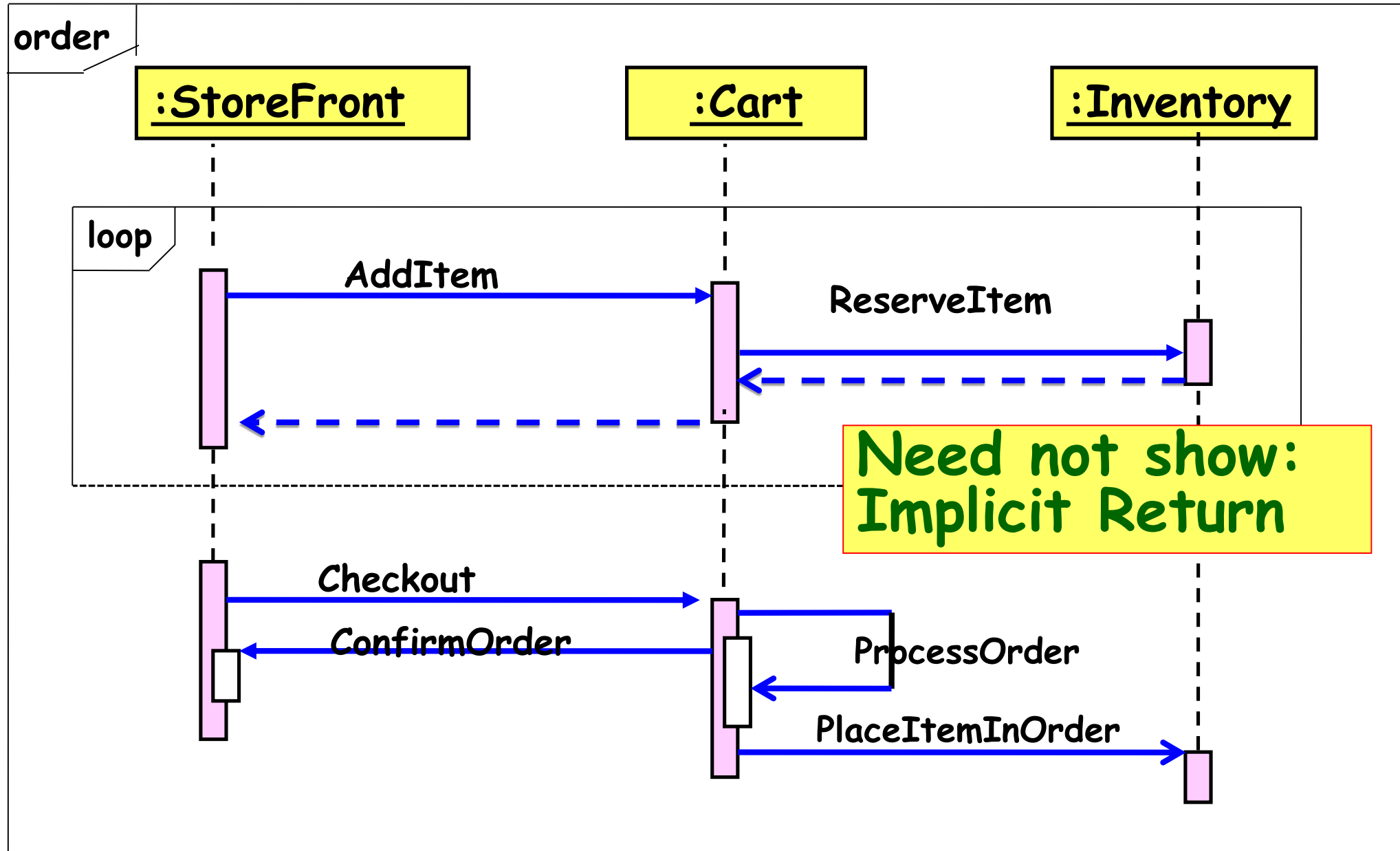
Frame Operator	Meaning
Alt	Alternative fragment for conditional logic expressed in the guards
Loop	Loop fragment while guard is true. Can also write loop(n) to indicate looping n times.
Opt	Optional fragment that executes if guard is true
Par	Parallel fragments that execute in parallel
Region	Critical region within which only one thread can run

Fragment Operators: Full List

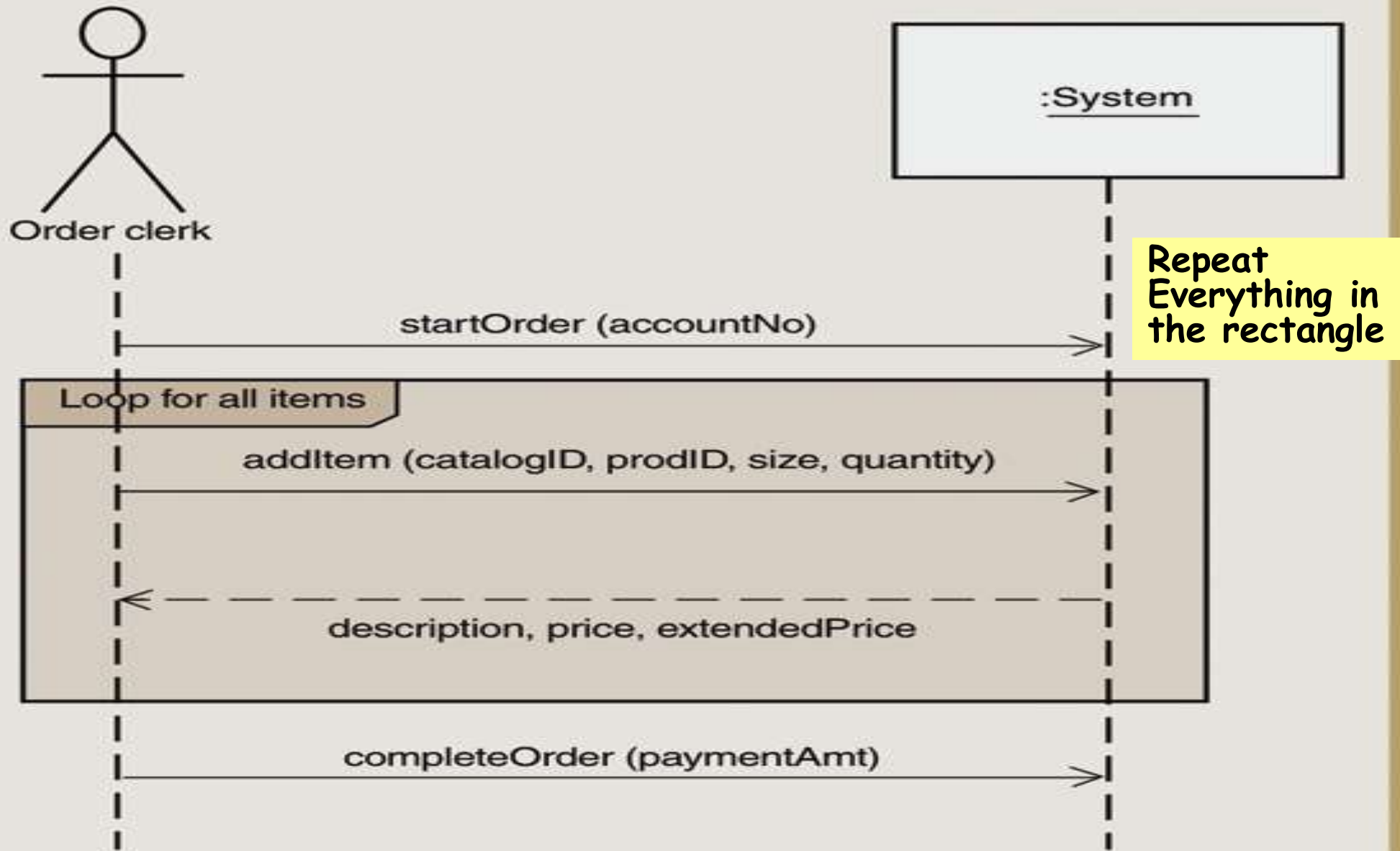


- **Conditional:** operator "opt"
- **Loop:** operator "loop"
- **Branching:** operator "alt"
- **Reference:** operator "ref"
- **Parallel:** operator "par"
- **Critical Region:** operator "critical"
- **Interrupt:** operator "break"
- **Assertion:** operator "assert"

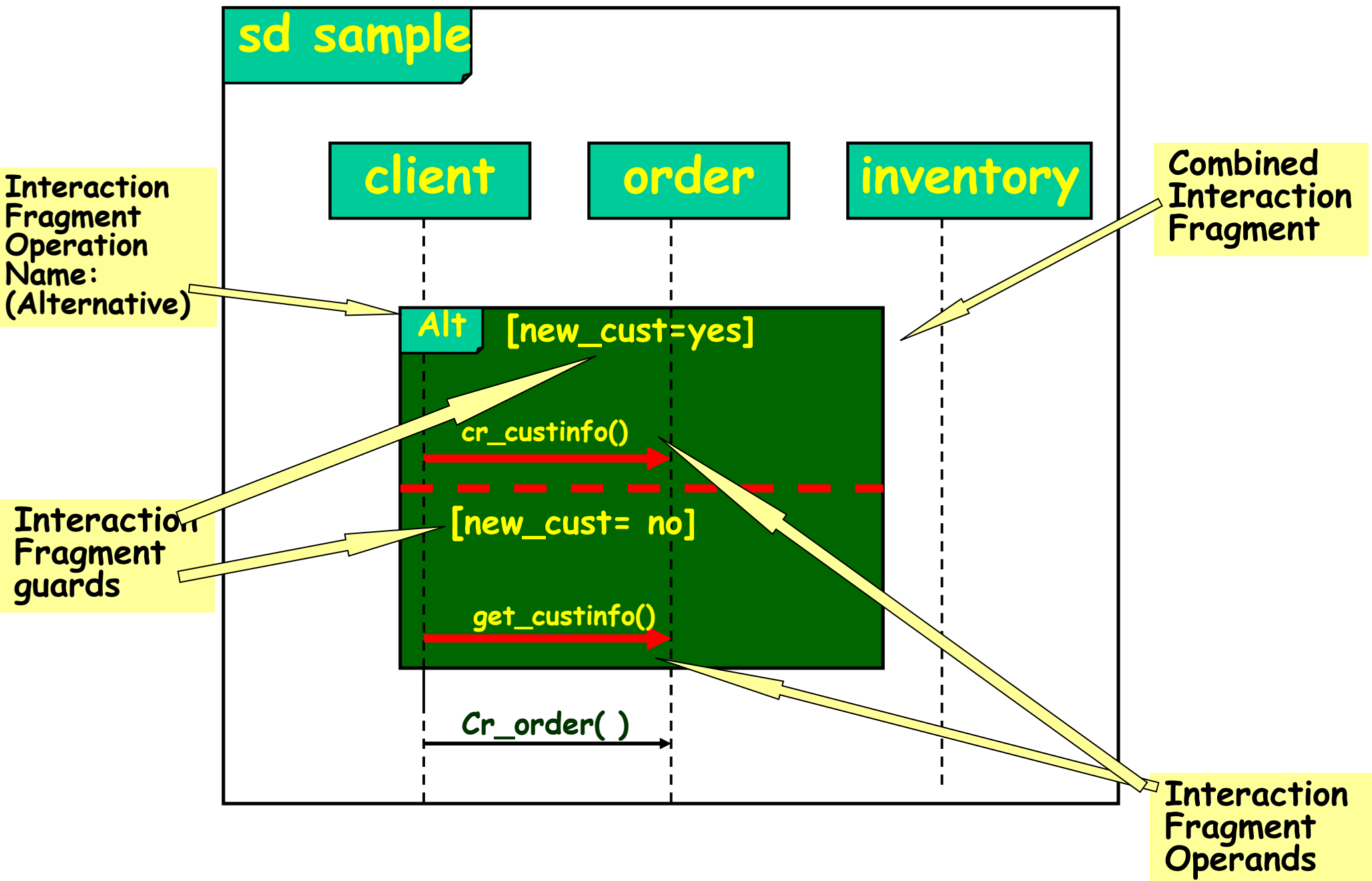
Example Sequence Diagram



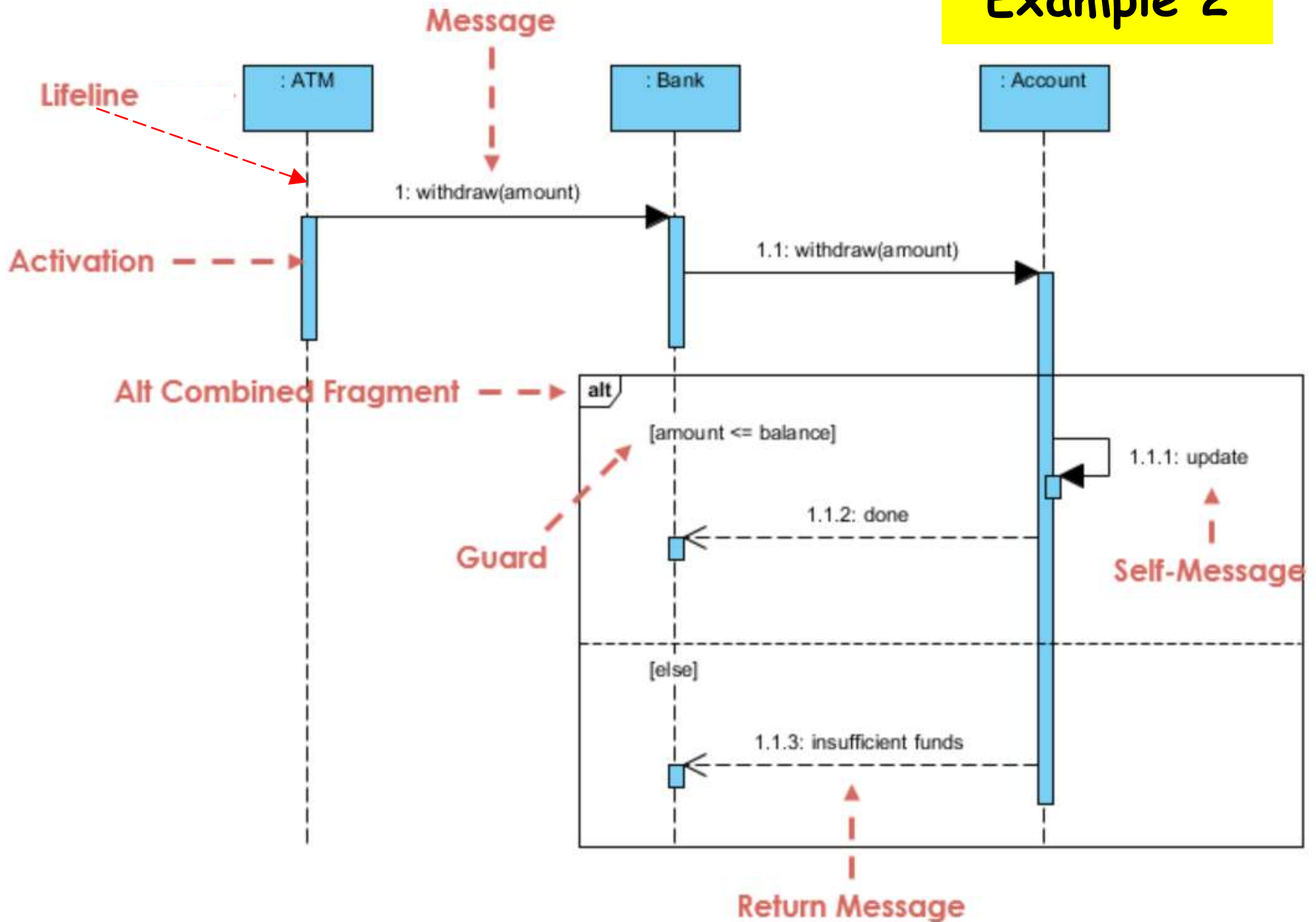
Sequence Diagram for Telephone Order use case



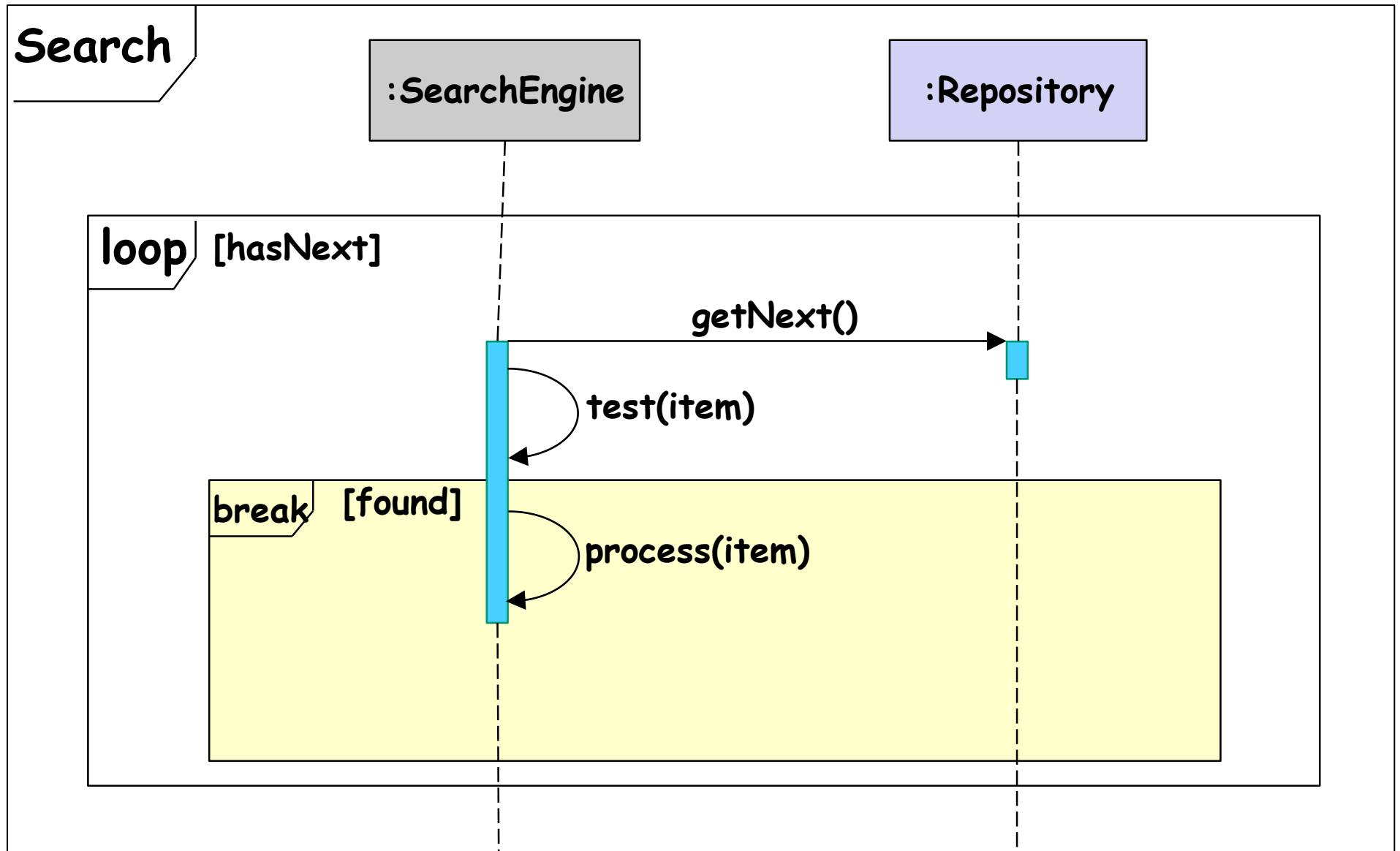
Alternative Fragment



Example 2

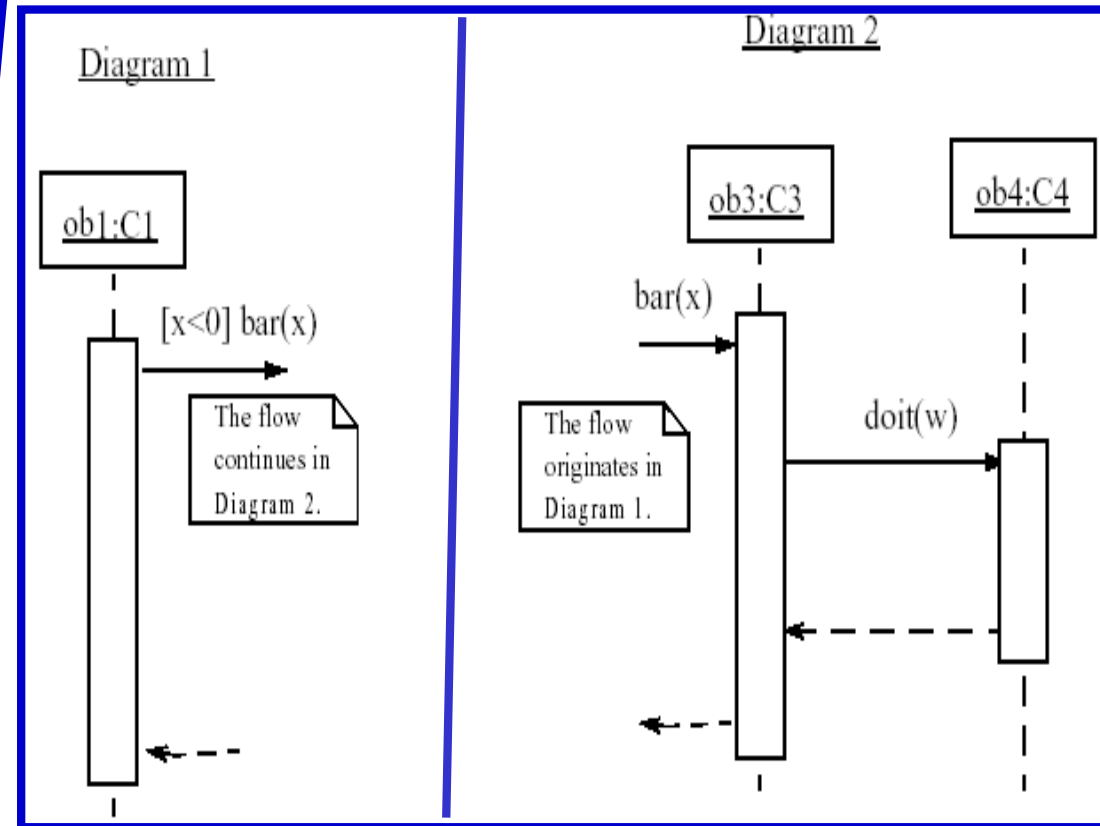
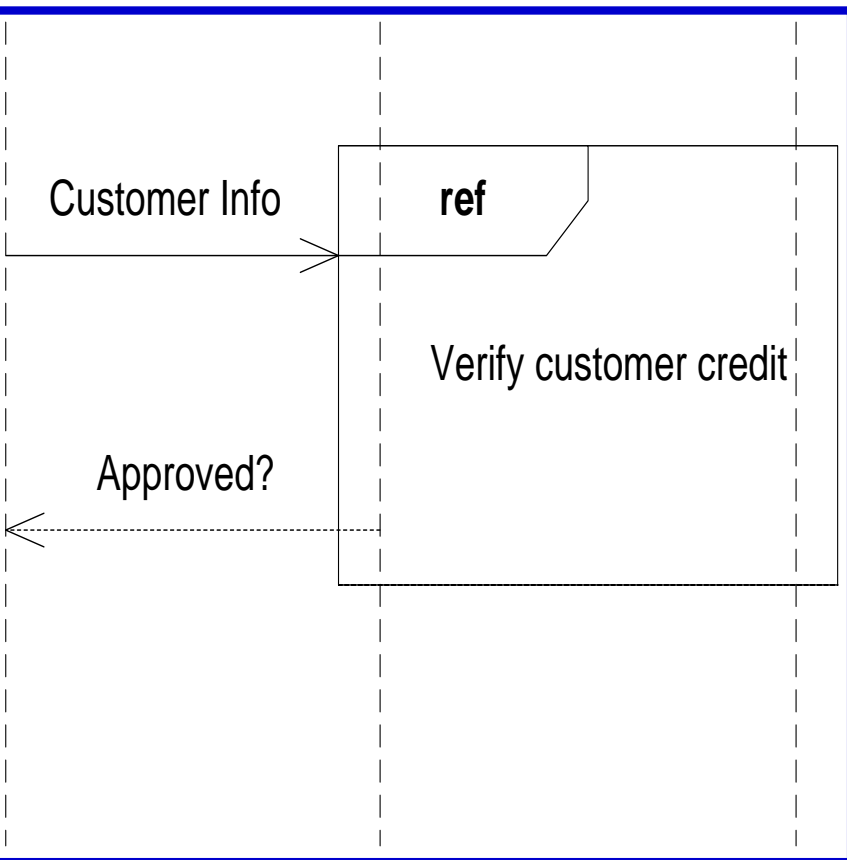


Loop Fragment : Example



Linking Sequence Diagrams

- If one sequence diagram is too large or refers to another diagram, indicate it with either:
 - an unfinished arrow and comment
 - a "ref" frame that names the other diagram



Nesting of Frames

