Object Oriented Analysis & Design
面向对象分析与设计

Lecture_05 领域模型

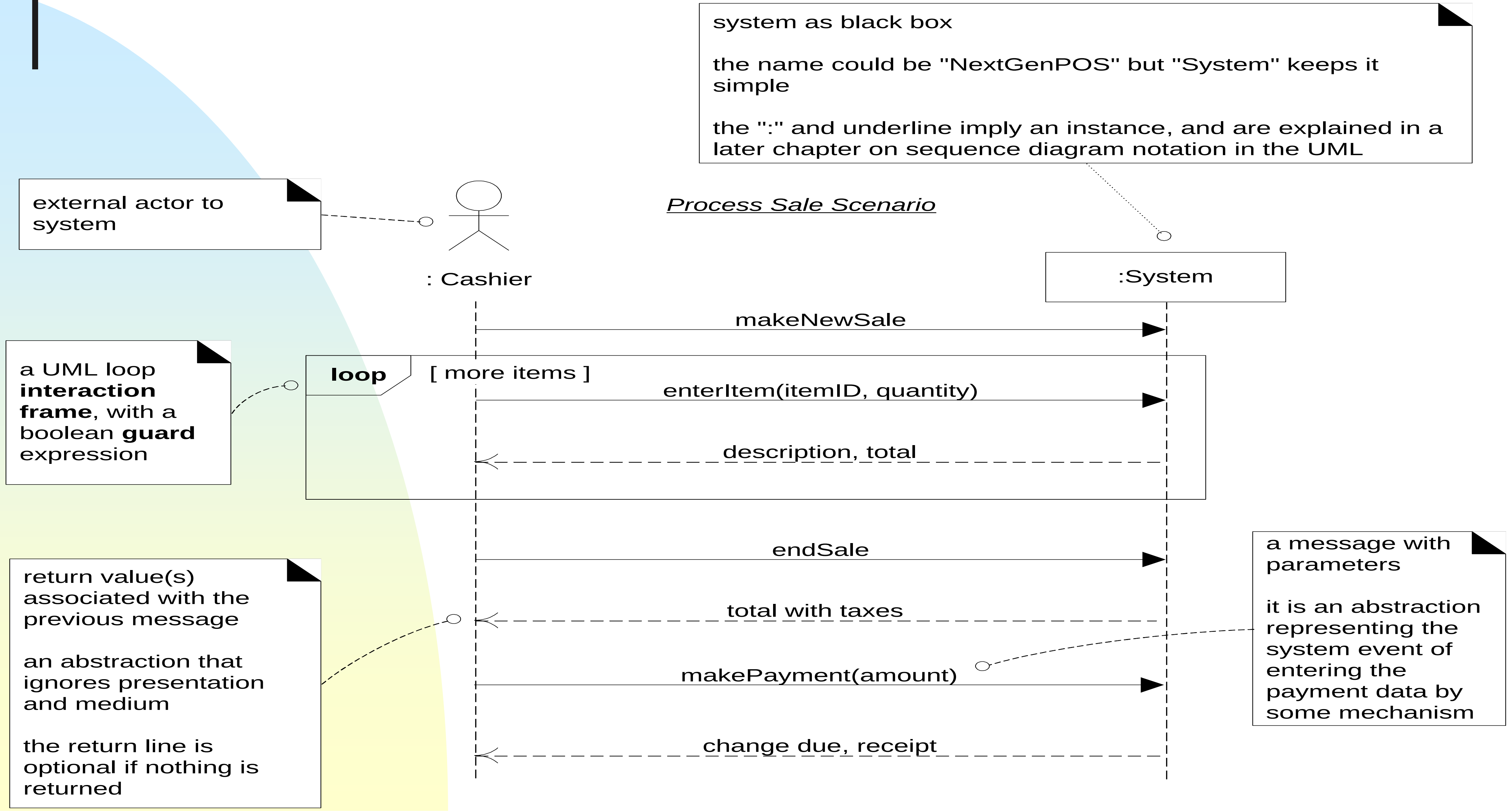
主讲: 姜宁康 博士

- 4、系统顺序图 System Sequence Diagram
  - 把待建系统看成一个**黑盒子**，研究参与者与系统边界的交互

4.1 System Sequence Diagram

- **SSD**
  - **System Sequence Diagram**
- **What is SSD**
  - **A SSD is a picture that shows, for one particular scenario of a use case, the events that external actors generate, inter-system events , and their order**
  - **All systems are treated as a black box**
  - **the emphasis of the diagram is events that cross the system boundary from actors to systems**
- **System Event**
  - **external input events**
    - **actor generates events to a system**
- **system operation**
  - **to handle the system event , for example**
  - **when a cashier enters an item's ID, the cashier is requesting the POS system to record that item's sale (the enterItem event). That event initiates an operation upon the system**

4.1 System Sequence Diagram


system as black box
the name could be "NextGenPOS" but "System" keeps it simple
the ":" and underline imply an instance, and are explained in a later chapter on sequence diagram notation in the UML
external actor to system
Process Sale Scenario
: Cashier
:System
makeNewSale
a UML loop **interaction frame**, with a boolean **guard** expression
loop
[more items]
enterItem(itemID, quantity)
description, total
endSale
a message with parameters
it is an abstraction representing the system event of entering the payment data by some mechanism
return value(s) associated with the previous message
an abstraction that ignores presentation and medium
the return line is optional if nothing is returned
total with taxes
makePayment(amount)
change due, receipt

4.1 System Sequence Diagram

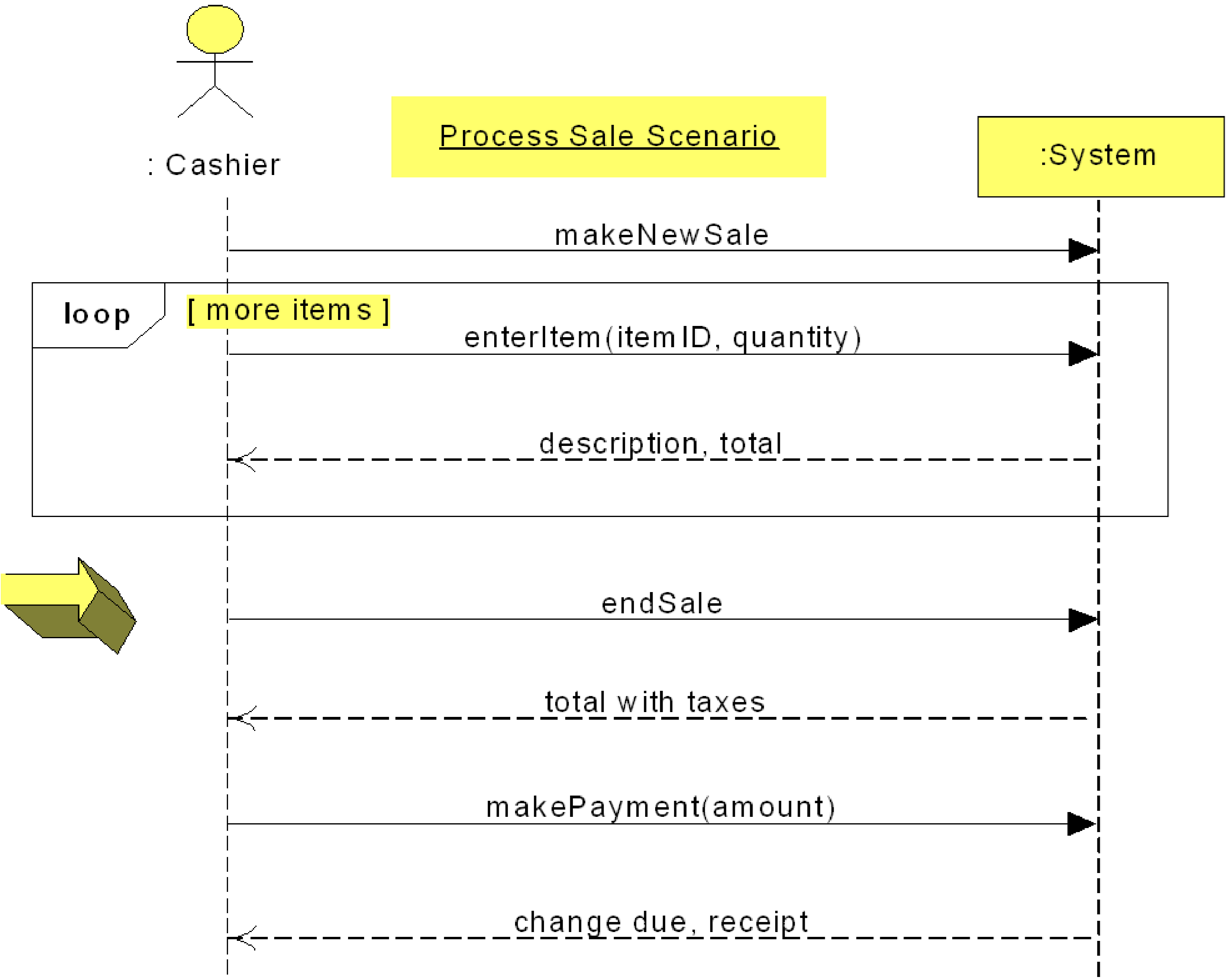
- 比较: 系统顺序图与顺序图 SSD & SD
  - SSD: to emphasize to treat systems as black boxes.
  - SD will be used to illustrate the design of interacting software objects to fulfill work
- 系统顺序图与用例 SSD and Use case
  - it is generated from inspection of a use case

4.2 ProcessSale 的系统顺序图

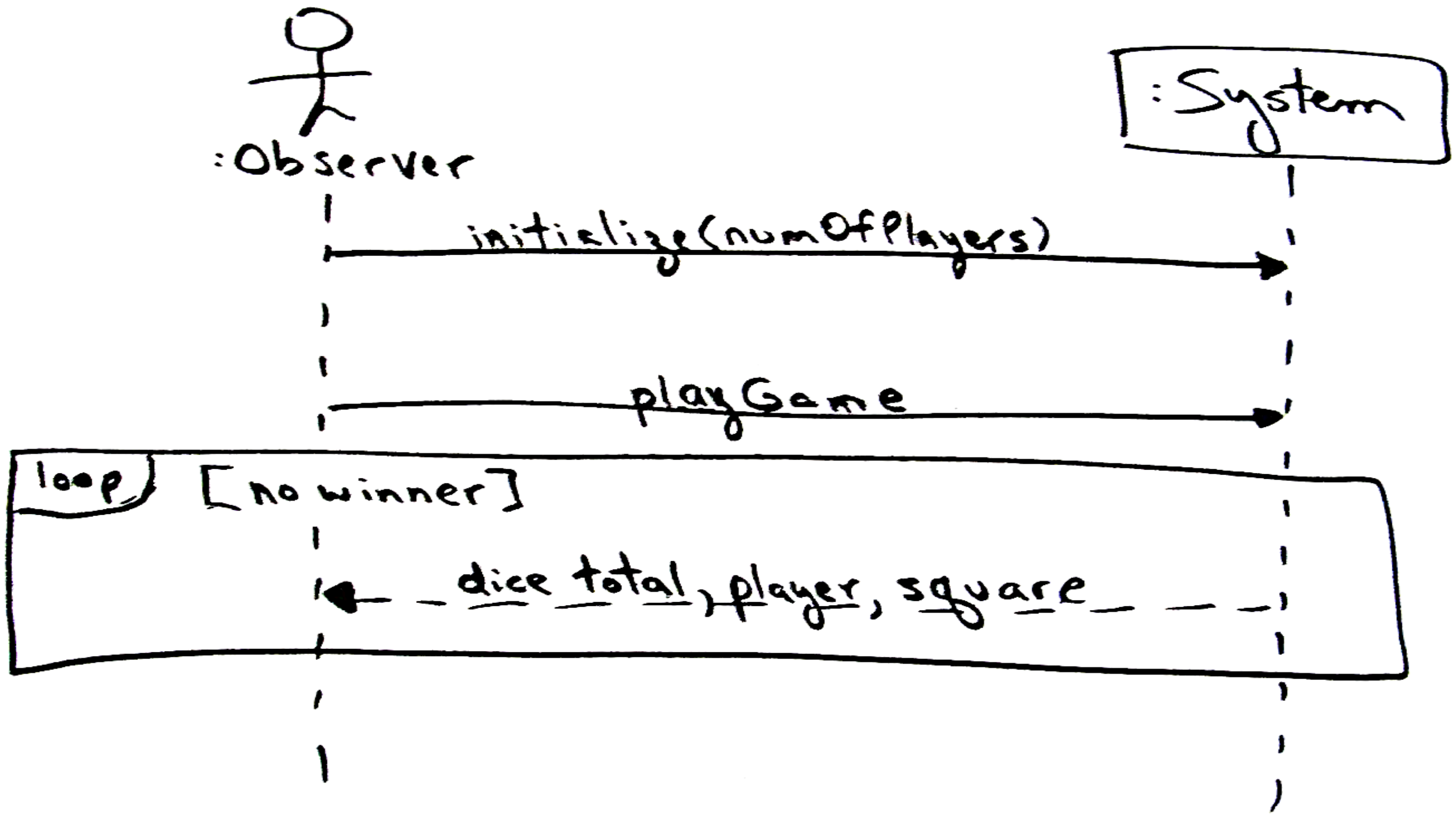
Simple cash-only Process Sale scenario:

1. Customer arrives at a POS checkout with goods and/or services to purchase.
2. Cashier starts a new sale.
3. Cashier enters item identifier.
4. System records sale line item and presents item description, price, and running total.
Cashier repeats steps 3-4 until indicates done.
5. System presents total with taxes calculated.
6. Cashier tells Customer the total, and asks for payment.
7. Customer pays and System handles payment.
...

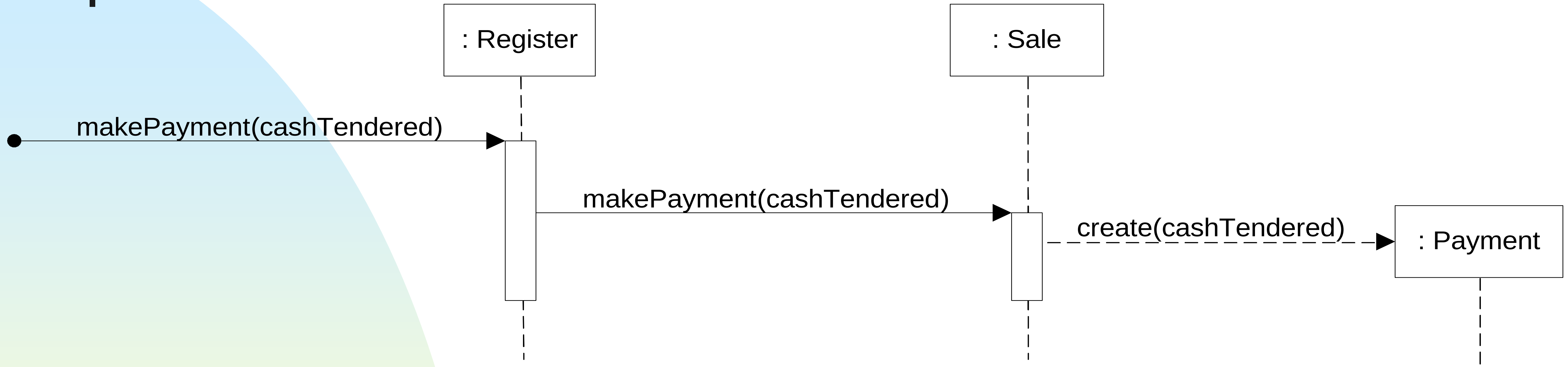
Fig 10-3



4.3 MonoPlayGame 的系统顺序图



4.4 比较：Sale的顺序图（注意：不是系统顺序图）

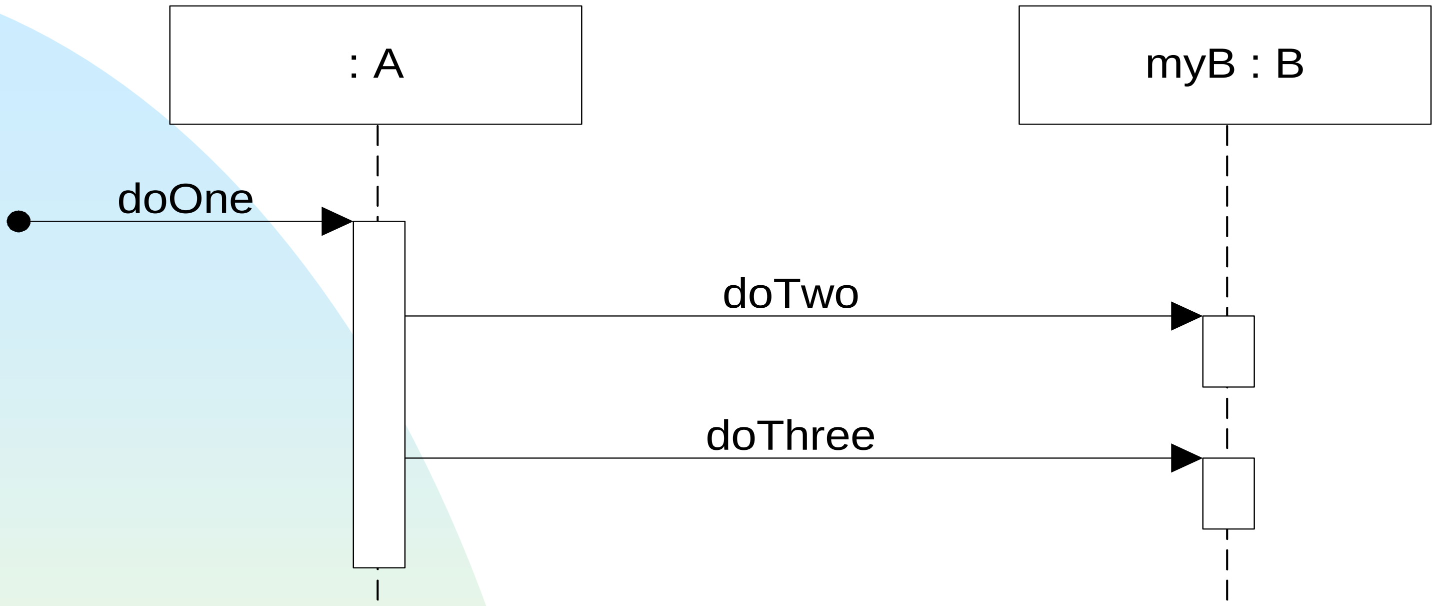


Above sequence diagram is read as follows:

- **The message makePayment is sent to an instance of a Register. The sender is not identified**
- **The Register instance sends the makePayment message to a Sale instance**
- **The Sale instance creates an instance of a Payment**

- **初学建模者对交互图往往没有给予足够的重视！**

4.5 复习顺序图与代码



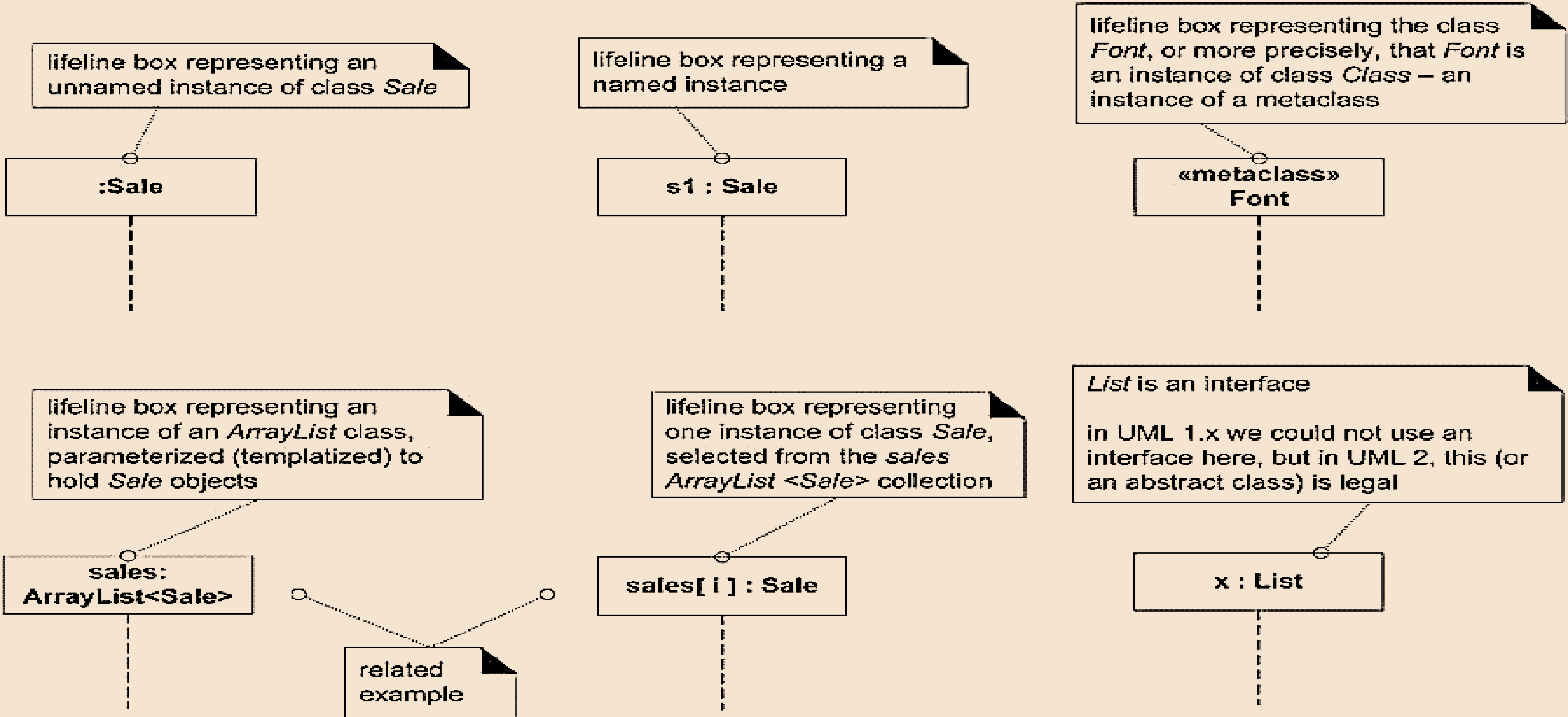
What class A&B looks like?
(write pseudocode)

```
public class A {
  private B myB = new B();
  public void doOne() {
     myB.doTwo();
     myB.doThree();
   }
   // ...
 }
```

```
public class B {
  ....
  public void doTwo() {
   ....
  }
  public  void doThree(){
  ......;
   }
   // ...
 }
```

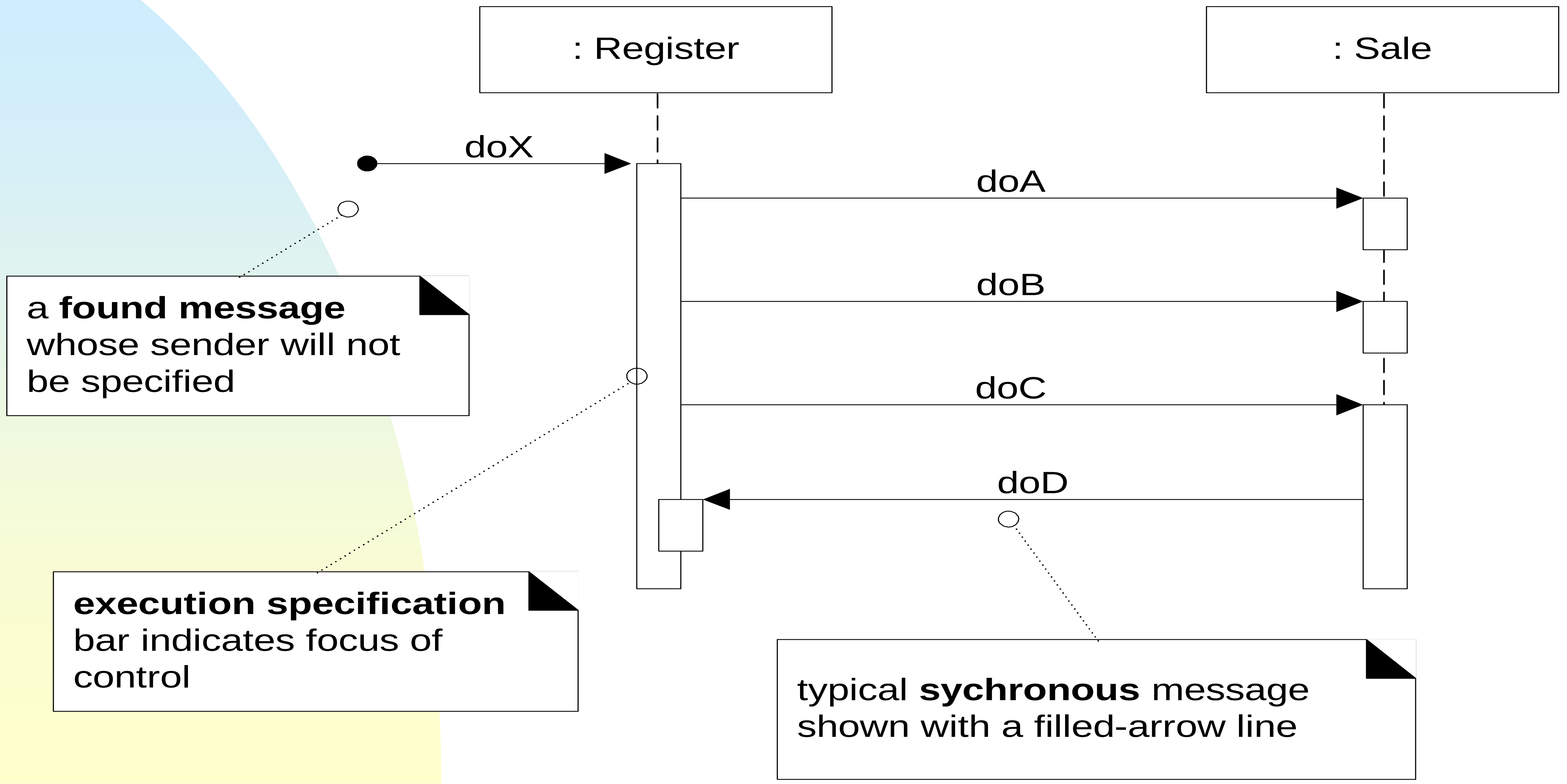
4.5 复习顺序图

- Fig 15-5 Lifeline boxes to show participants in interactions


lifeline box representing an unnamed instance of class *Sale*
lifeline box representing a named instance
lifeline box representing the class *Font*, or more precisely, that *Font* is an instance of class *Class* – an instance of a metaclass
:Sale
s1 : Sale
«metaclass» Font
lifeline box representing an instance of an *ArrayList* class, parameterized (templatized) to hold *Sale* objects
lifeline box representing one instance of class *Sale*, selected from the *sales* *ArrayList <Sale>* collection
List is an interface
in UML 1.x we could not use an interface here, but in UML 2, this (or an abstract class) is legal
sales: ArrayList<Sale>
related example
sales[i] : Sale
x : List

4.5 复习顺序图

- **Messages and focus of control with execution specification bar**


: Register
: Sale
doX
doA
doB
doC
doD
a **found message** whose sender will not be specified
execution specification bar indicates focus of control
typical **sychronous** message shown with a filled-arrow line

4.5 复习顺序图

Two ways to show a return result from a message

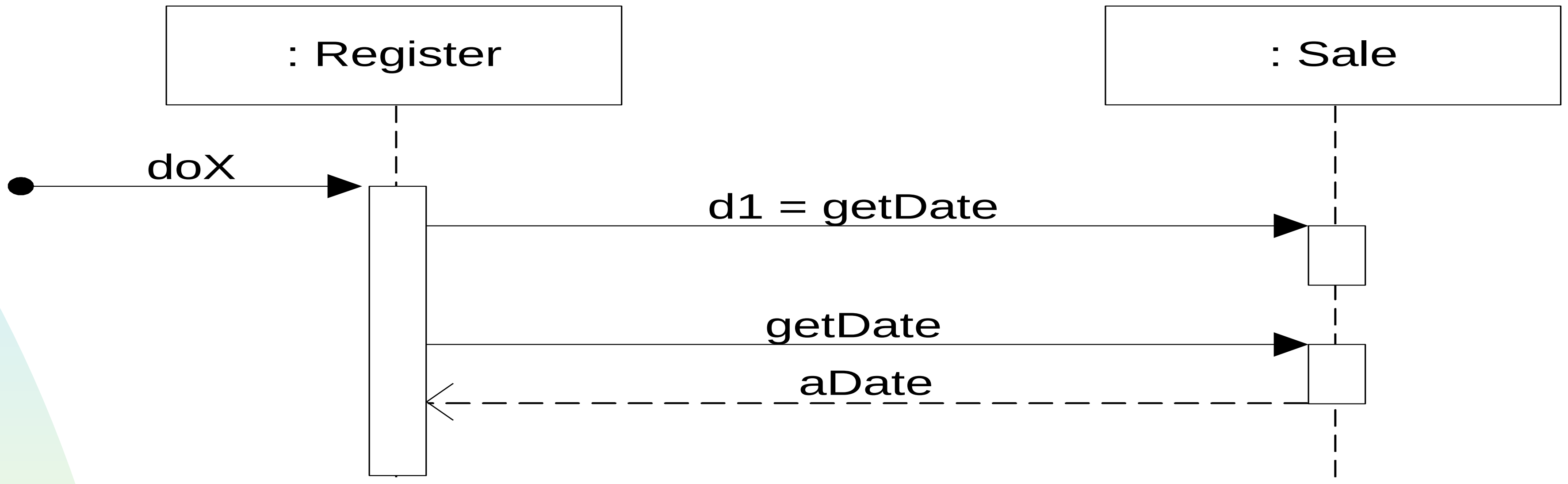
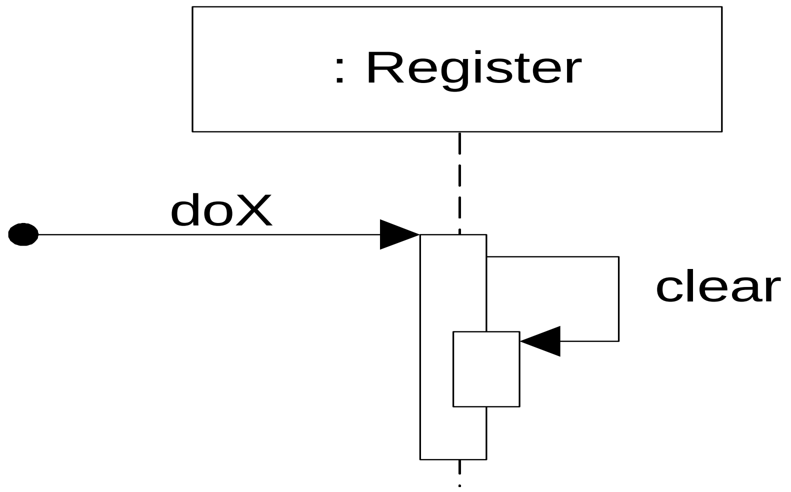
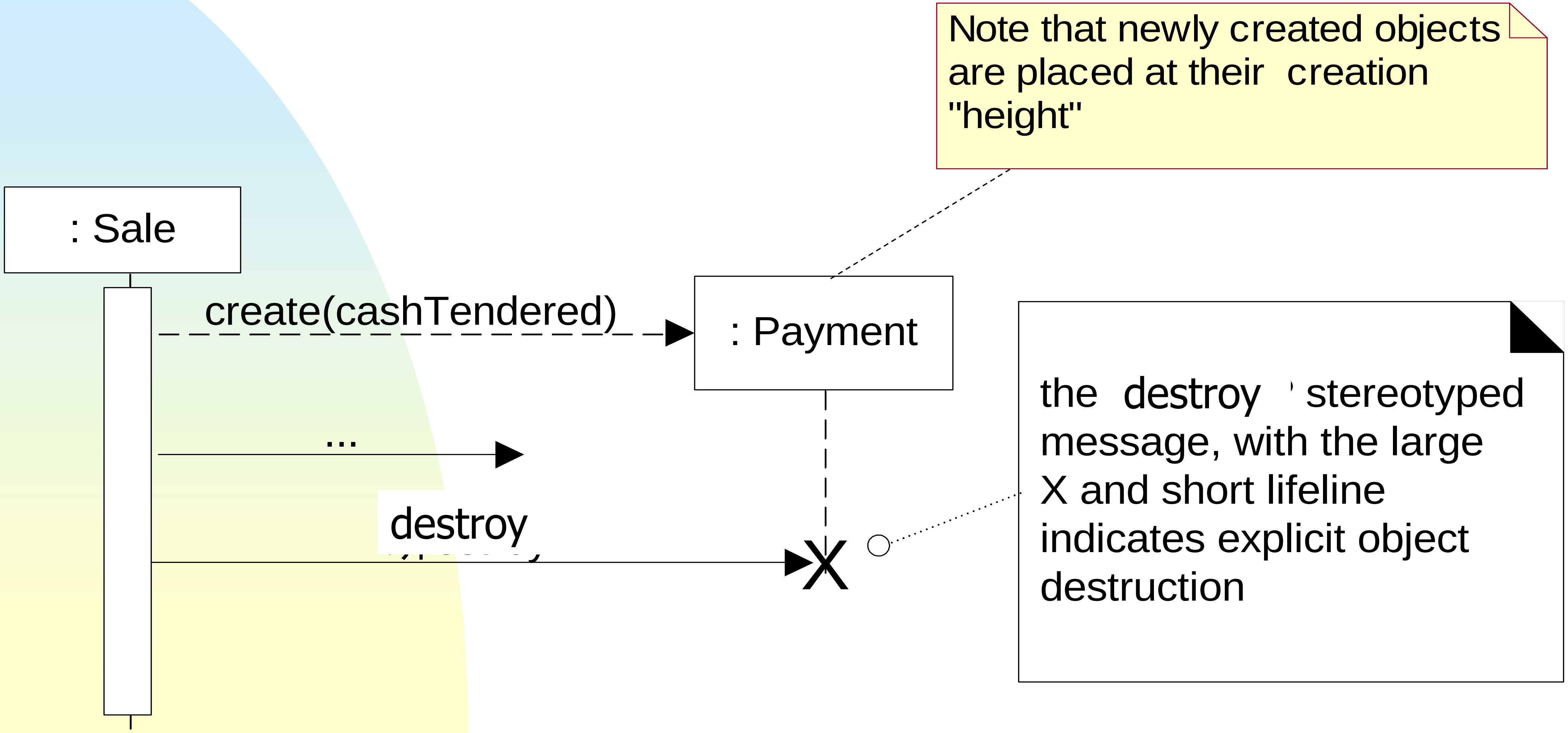


Fig 15-9 Messages to "this"



4.5 复习顺序图

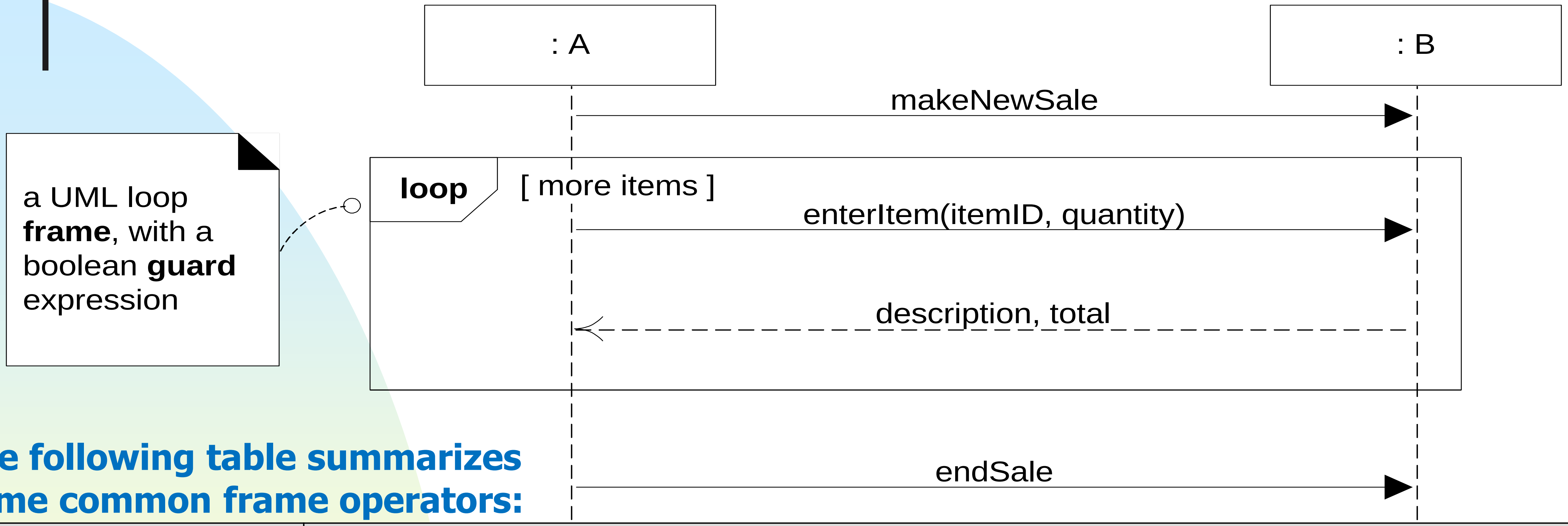
Fig 15-11 Object destruction



Note that newly created objects are placed at their creation "height"

the destroy ' stereotyped message, with the large X and short lifeline indicates explicit object destruction

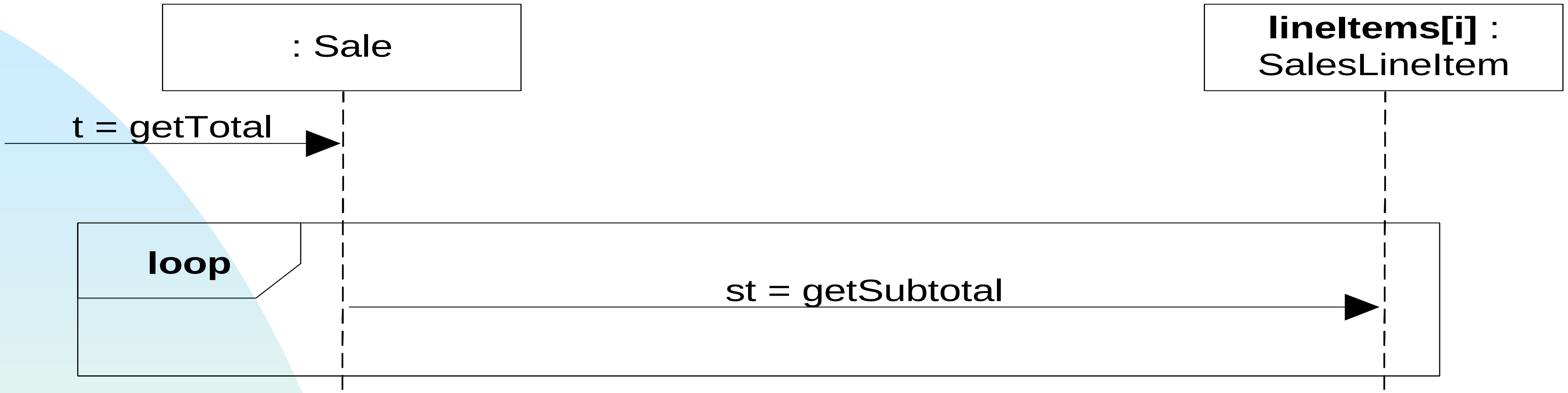
4.5 复习顺序图 :Diagram Frames in UML


: A
: B
makeNewSale
a UML loop **frame**, with a boolean **guard** expression
loop
[more items]
enterItem(itemID, quantity)
description, total
endSale

The following table summarizes some common frame operators:

Frame Operator	Meaning
alt	Alternative fragment for mutual exclusion conditional logic expressed in the guards.
loop	Loop fragment while guard is true. Can also write loop(n) to indicate looping n times. There is discussion that the specification will be enhanced to define a FOR loop, such as loop(i, 1, 10)
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

4.5 复习顺序图



```
public class Sale {
        private List<SalesLineItem> lineItems = new ArrayList<SalesLineItem>();
        public Money getTotal() {
                Money total = new Money();
                Money subtotal = null;
                for ( SalesLineItem lineItem : lineItems ) {
                        subtotal = lineItem.getSubtotal();
                        total.add( subtotal );
                }
                 return total;
        }
         // ...
}
```

请同学们课后练习：
如何从顺序图写出代码

- 本讲结束