Software Design and Analysis
CS-3004
Lecture#09

Dr. Javaria Imtiaz,
Mr. Basharat Hussain,
Mr. Majid Hussain

Quiz 03

Use case: Loan a video
Actors: Assistant
Goal: To lend a video to a customer

Overview:
A customer chooses a video and gives their membership card and the video to the Assistant. The Assistant scans the customer's membership card and checks if they owe any money or have outstanding loans. The system searches for a specific video using the barcode scanned from the video they wish to borrow. The system locates the required video and displays the details on the screen. The Assistant checks that this is the video the customer wants to borrow and looks to see what the rental cost is for this video. The system then registers the loan transaction.

Typical course of events:

Actor action	System response
1 The customer chooses a video	
2 The Assistant scans in the membership card barcode	3 Displays customer details
4 The Assistant agrees the details	
5 The Assistant scans in the video barcode	6 Displays video details including hire cost
7 The Assistant agrees the cost and registers the loan	8 Stores the loan transaction
9 The customer pays for the loan	
10 The Assistant records the payment	11 Prints a receipt

Use case: Issue bike
Preconditions: 'Maintain bike list' must have been executed
Actors: Receptionist
Goal: To hire out a bike

Overview:
When a customer comes into the shop they choose a bike to hire. The Receptionist looks up the bike on the system and tells the customer how much it will cost to hire the bike for a specified period. The customer pays, is issued with a receipt, then leaves with the bike.

Cross-reference:
R3, R4, R5, R6, R7, R8, R9, R10

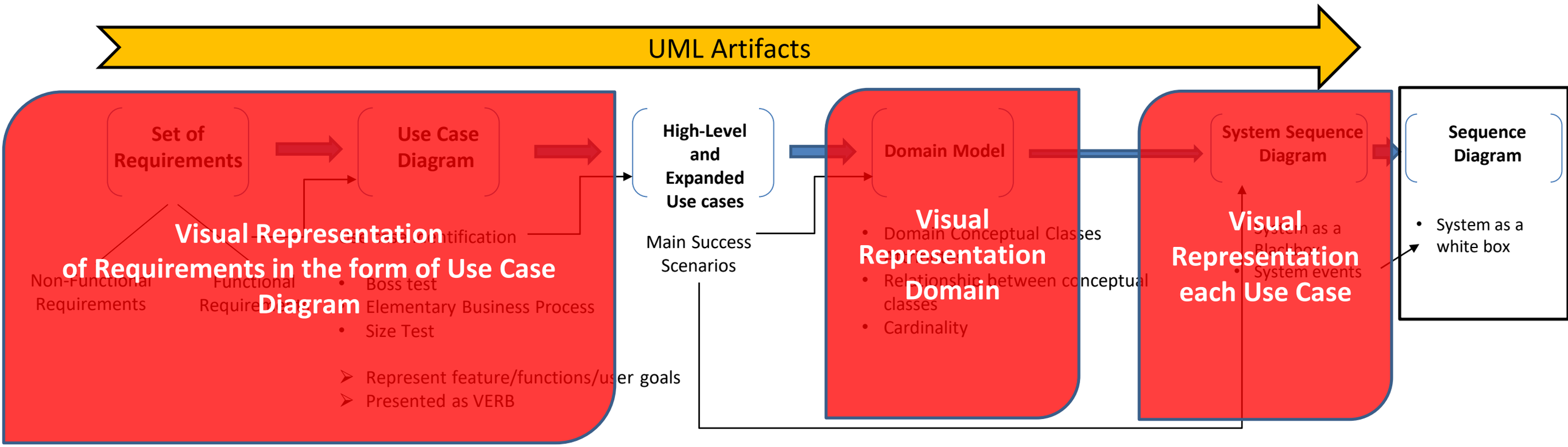
Typical course of events:

Actor action	System response
1 The customer chooses a bike	
2 The Receptionist keys in the bike number	3 Displays the bike details including the daily hire rate and deposit
4 Customer specifies length of hire	
5 Receptionist keys this in	6 Displays total hire cost
7 Customer agrees the price	
8 Receptionist keys in the customer details	9 Displays customer details
10 Customer pays the total cost	
11 Receptionist records amount paid	12 Prints a receipt

Alternative courses:

Steps 8 and 9	The customer details are already in the system so the Receptionist needs only to key in an identifier and the system will display the customer details.
Steps 7–12	The customer may not be happy with the price and may terminate the transaction

Revision up till now



UML Interaction Diagrams

- UML Sequence Diagrams

Sequence Diagram

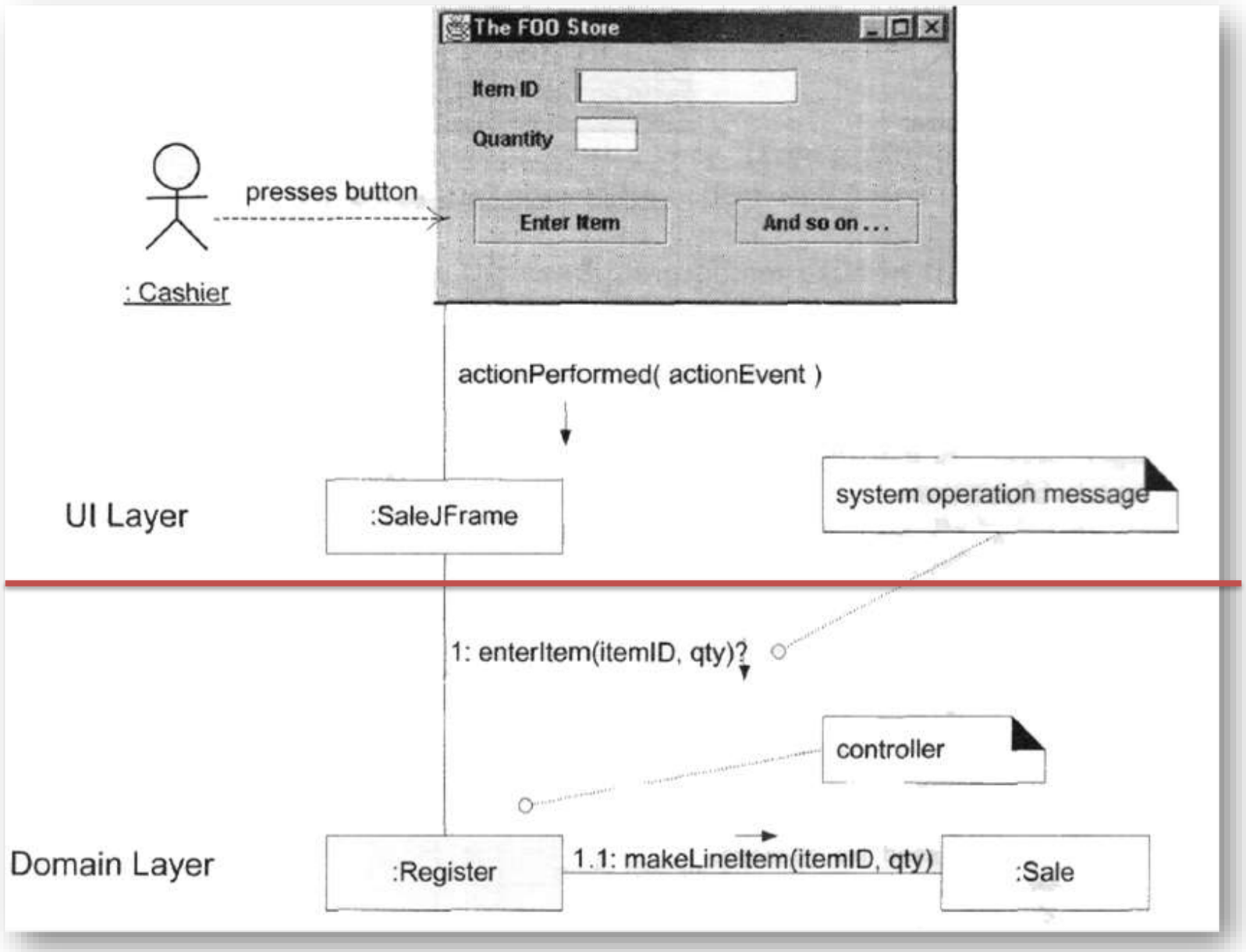
- Sequence diagram simply depicts *interaction between objects* in a sequential order (i.e. the order in which these interactions take place.)
- Sequence diagrams describe *how* and in what order the objects in a system function

Sequence Diagram

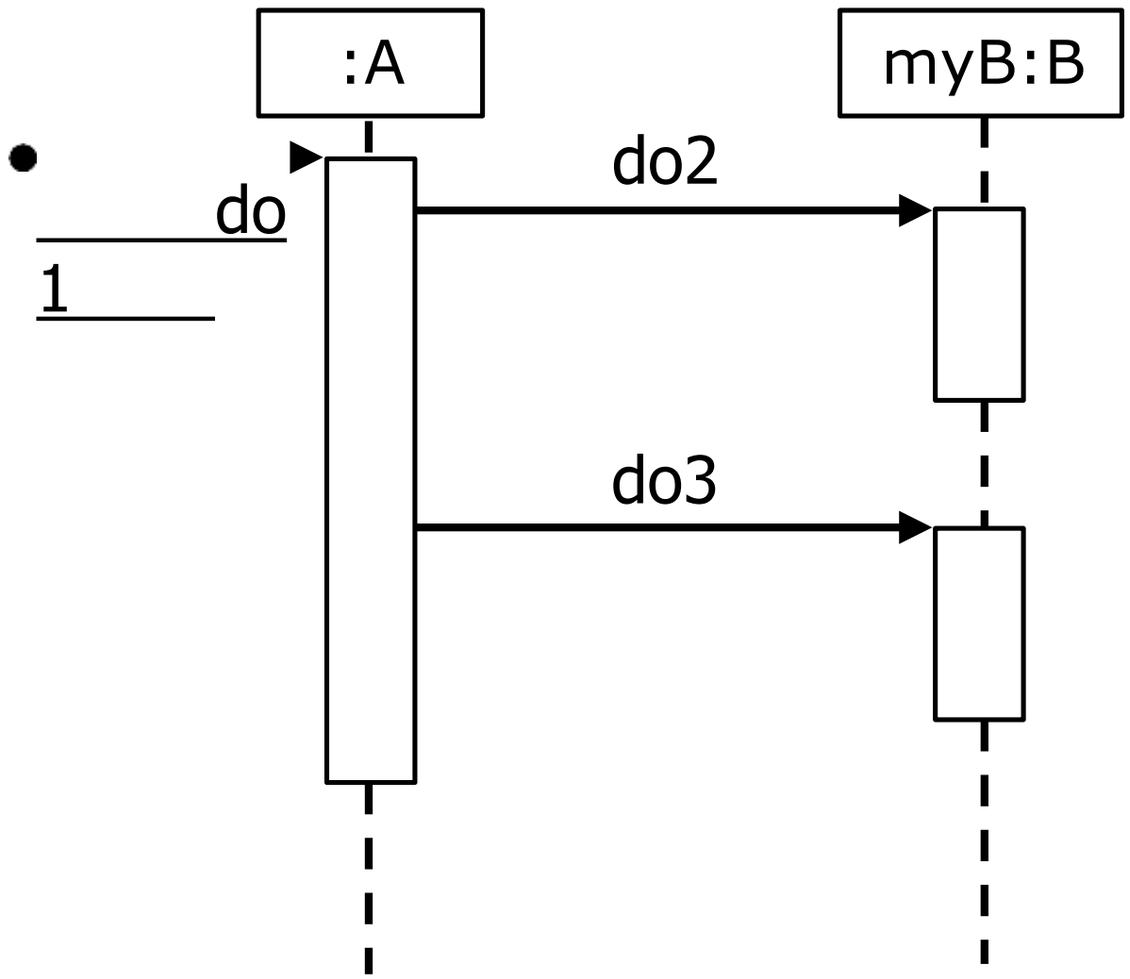
They illustrate how the different parts of a system interact with each other to carry out a function, and the order in which the interactions occur when a particular use case is executed

Interaction diagrams are used to visualize the interaction via messages between objects; they are used for *dynamic object modeling*.

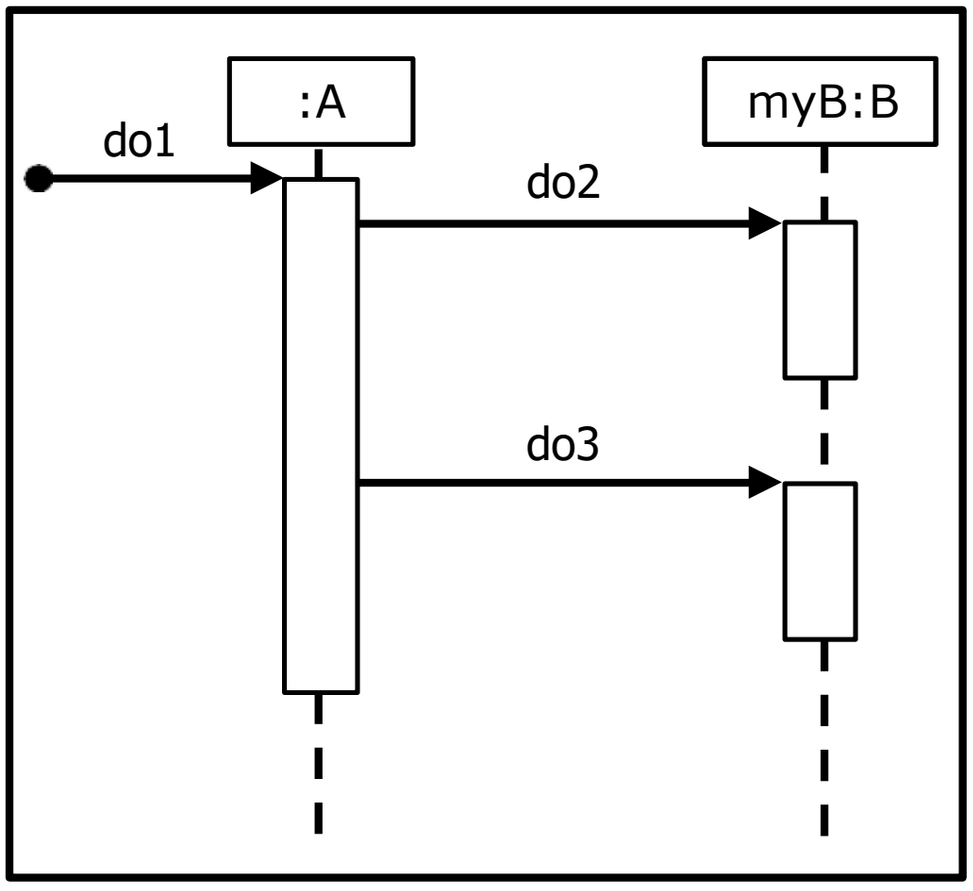
Example


The FOO Store
Item ID
Quantity
Enter Item
And so on . . .
presses button
: Cashier
actionPerformed(actionEvent)
UI Layer
:SaleJFrame
system operation message
1: enterItem(itemID, qty)?
controller
Domain Layer
:Register
1.1: makeLineItem(itemID, qty)
:Sale

Example of Sequence Diagram



Java Code for Interaction Diagrams



Sequence Diagram

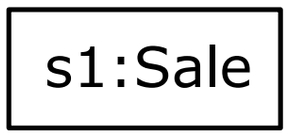
Example

```
public class A
  { private B myB
  = ...;

  public void
      do1() {
    myB.do2();
    myB.do3();
  }
}
```

Common Notations for UML Interaction Diagrams

Lifeline box representing an unnamed instance of class Sale.

Common Notations for UML Interaction Diagrams



Java Code:

```
Sale s1 = …;
```

Lifeline box representing a named instance (s1) of Sale.

Common Notations for UML Interaction Diagrams

Java Code:

```
ArrayList<Sale> sales = …;
```

Lifeline box representing an instance of an ArrayList class, parameterized to hold Sale objects.

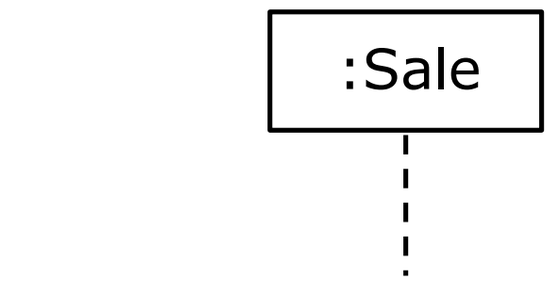
Common Notations for UML Interaction Diagrams

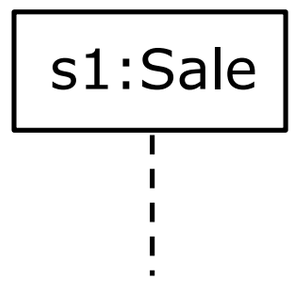
Java Code:

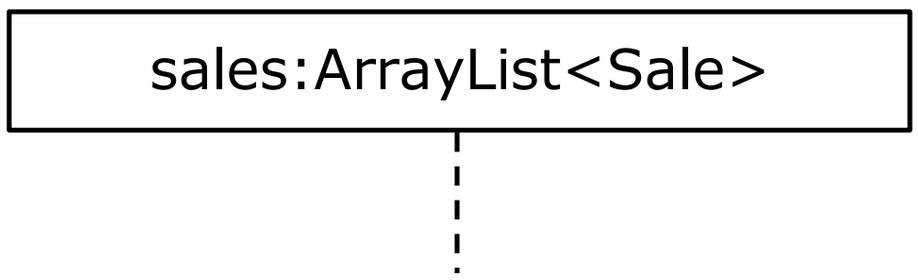
```
ArrayList<Sale> sales = …;
Sale sale = sales.get(i);
```

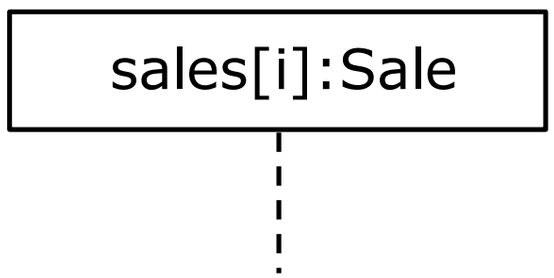
Lifeline box representing one instance of class Sale, selected from the sales ArrayList<Sale> collection.

Common Notations for UML Interaction Diagrams

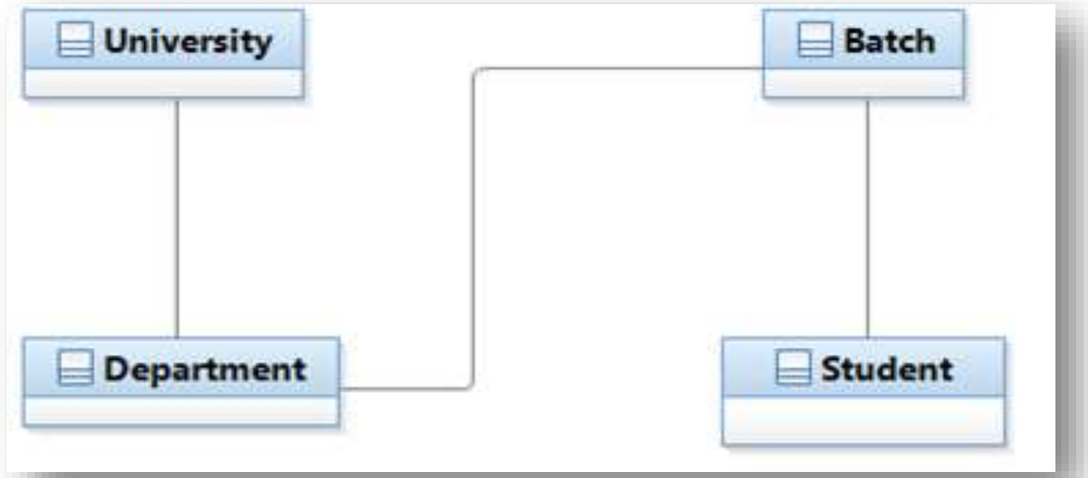


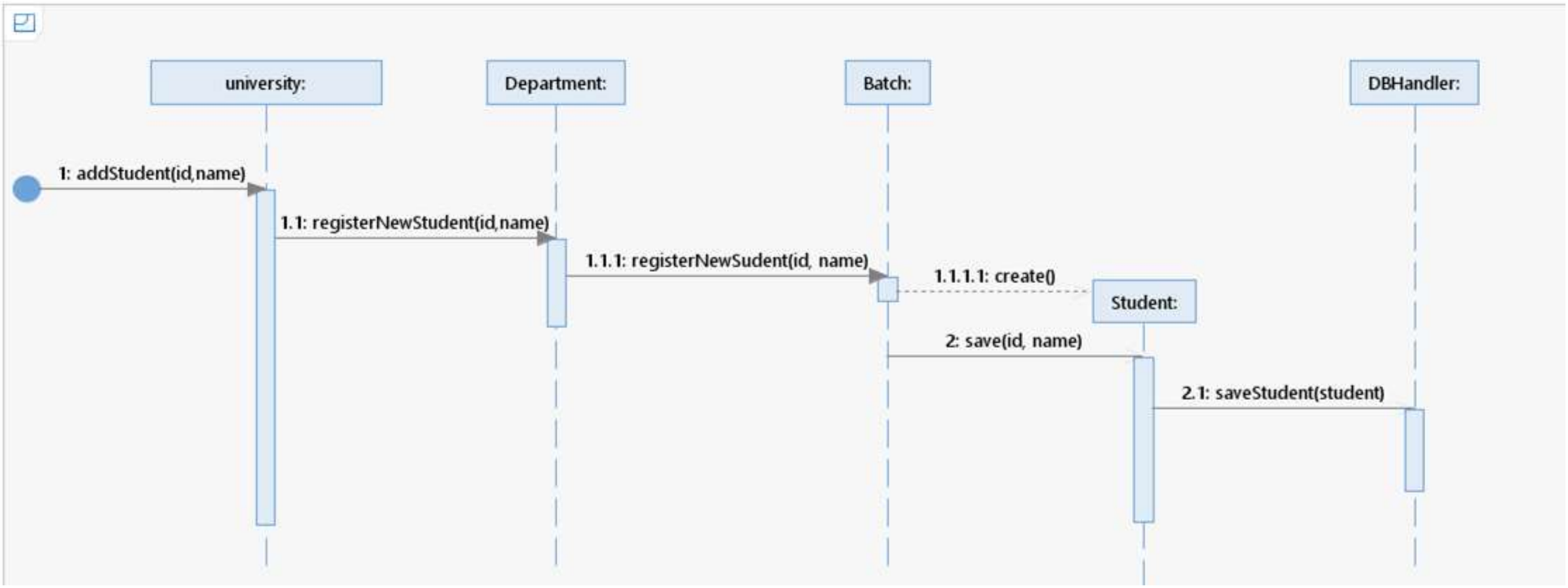






Sequence Diagram Example





Common Notations for UML Interaction Diagrams - Format for Interaction Messages

"Commonly" Used Grammar:

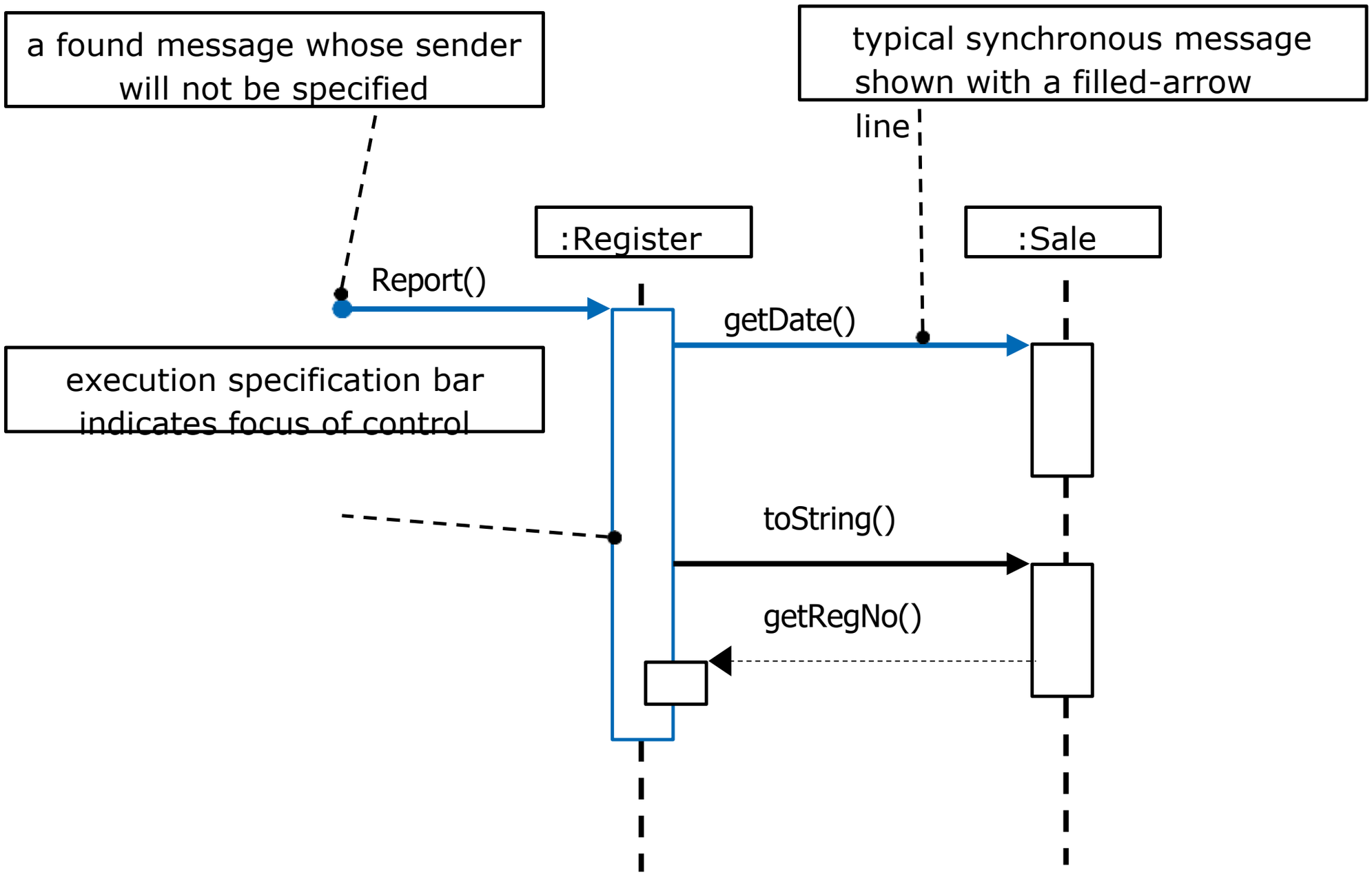
```
return = message(parameter:parameterType):returnType
```

Parentheses are usually excluded if there are no parameters. Type information may be excluded if unimportant.

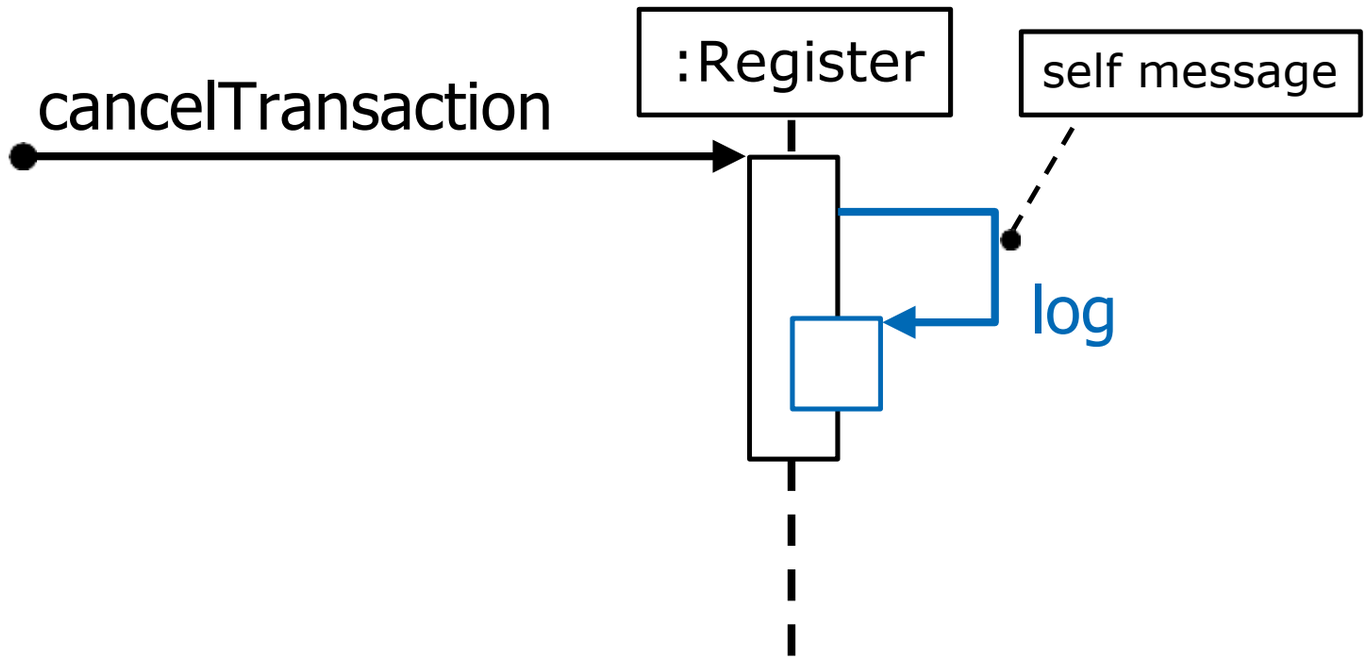
```
initialize(code)
initialize
d = getProductDescription (id)
d = getProductDescription (id : ItemId)
d = getProductDescription (id : ItemId) : ProductDescription
```

UML Sequence Diagrams

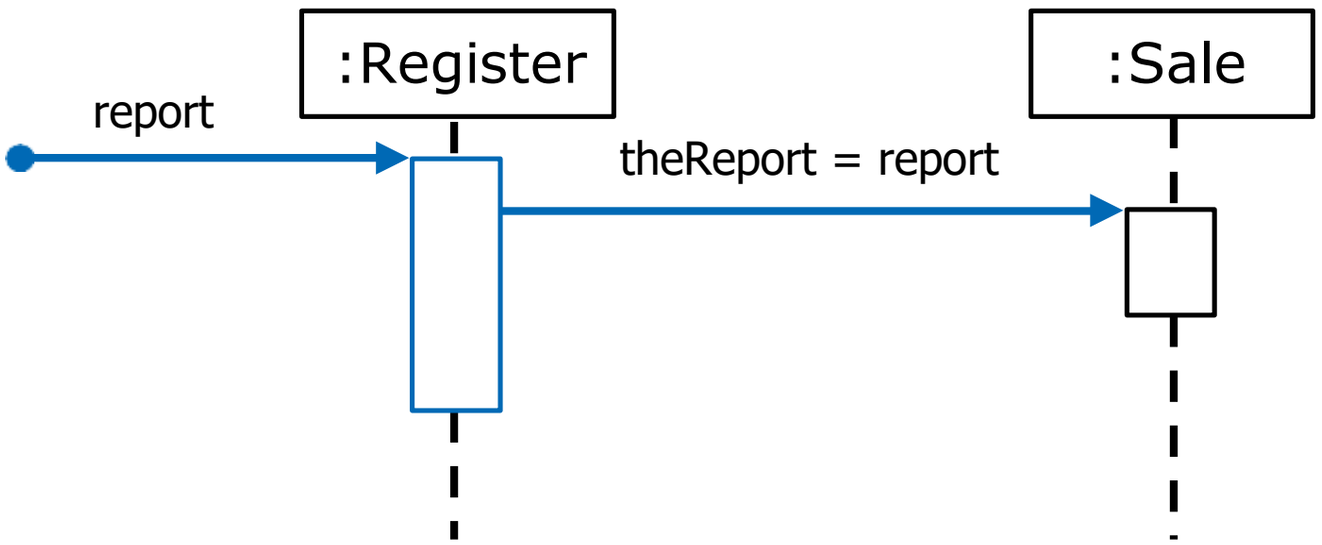
Modeling (Synchronous) Messages



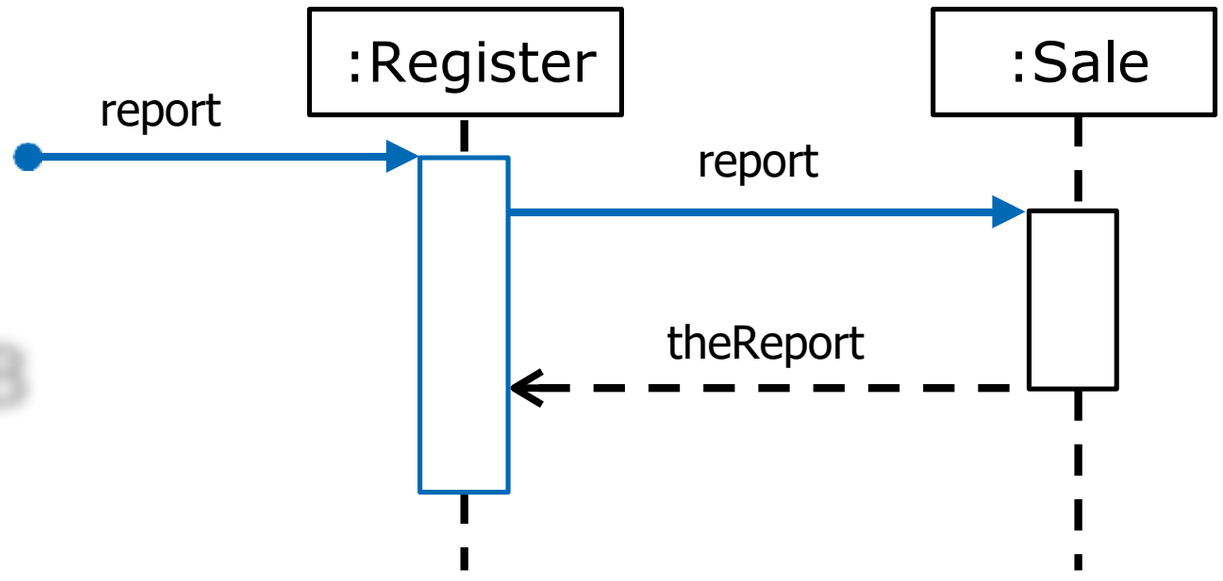
Self messages can be modeled using nested execution specification bars.



To show the return value of a message you can either use the message syntax (A) or use a message line at the end of an execution specification bar (B).

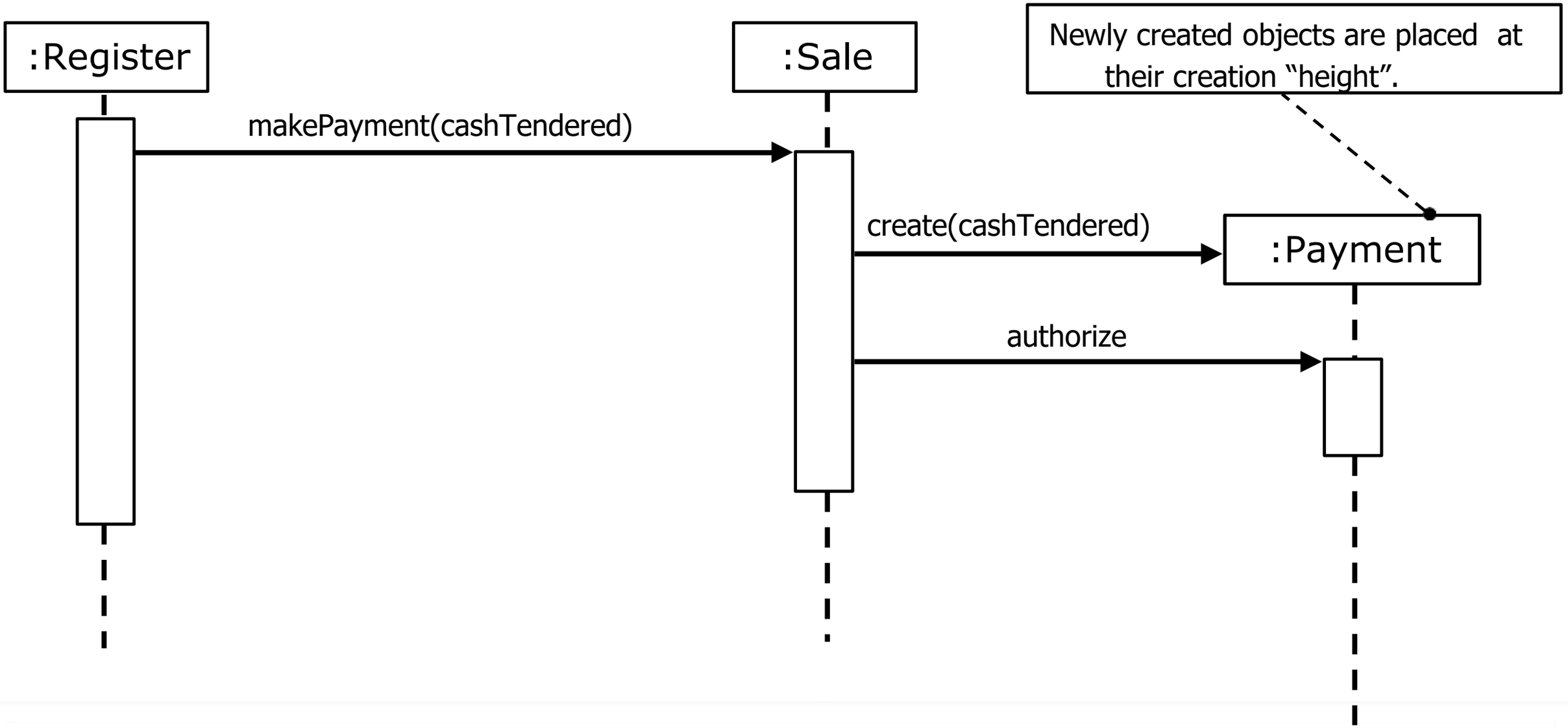


Variant A



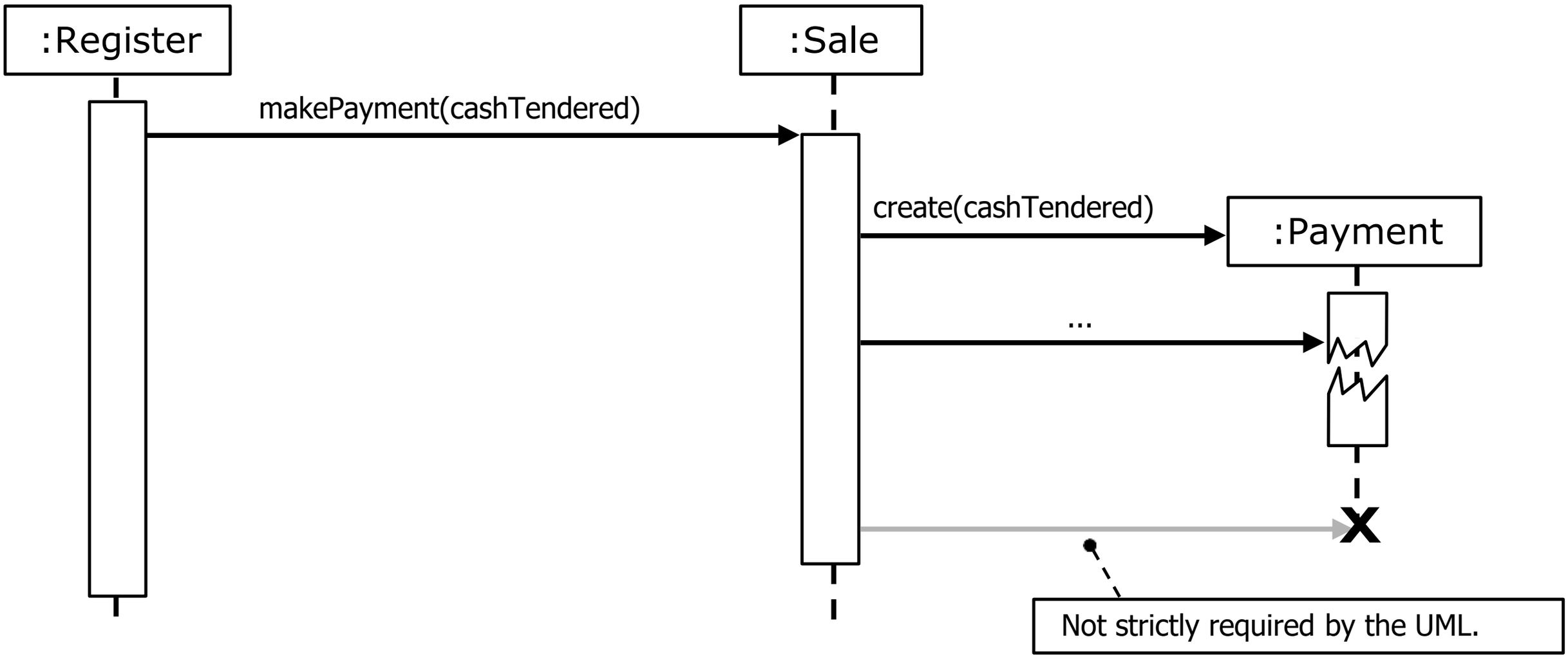
Variant B

Object Instance Creation



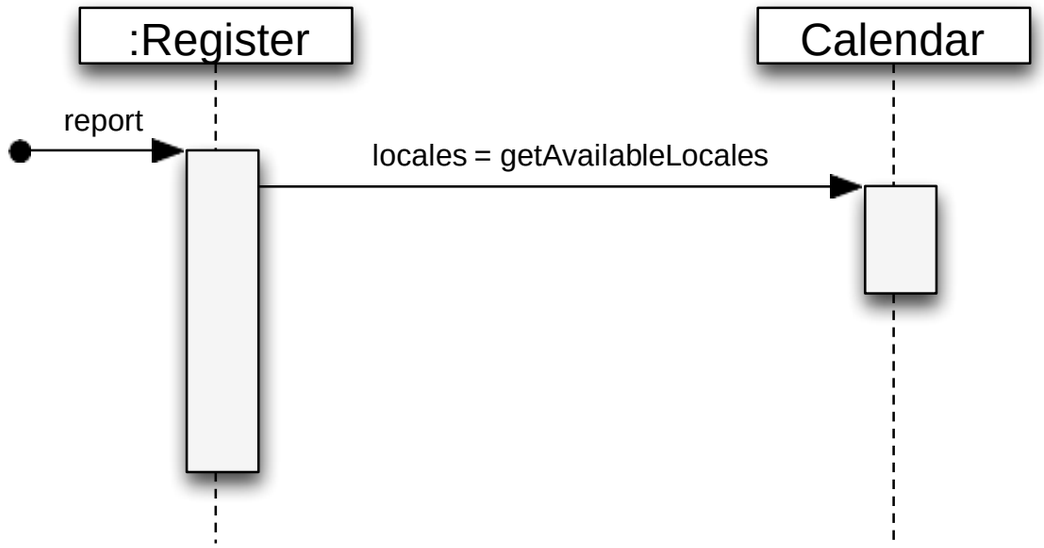
The name create is an *UML idiom*; it is not required.

Object Instance Destruction



The object destruction notation is also used to mark objects that are no longer usable.

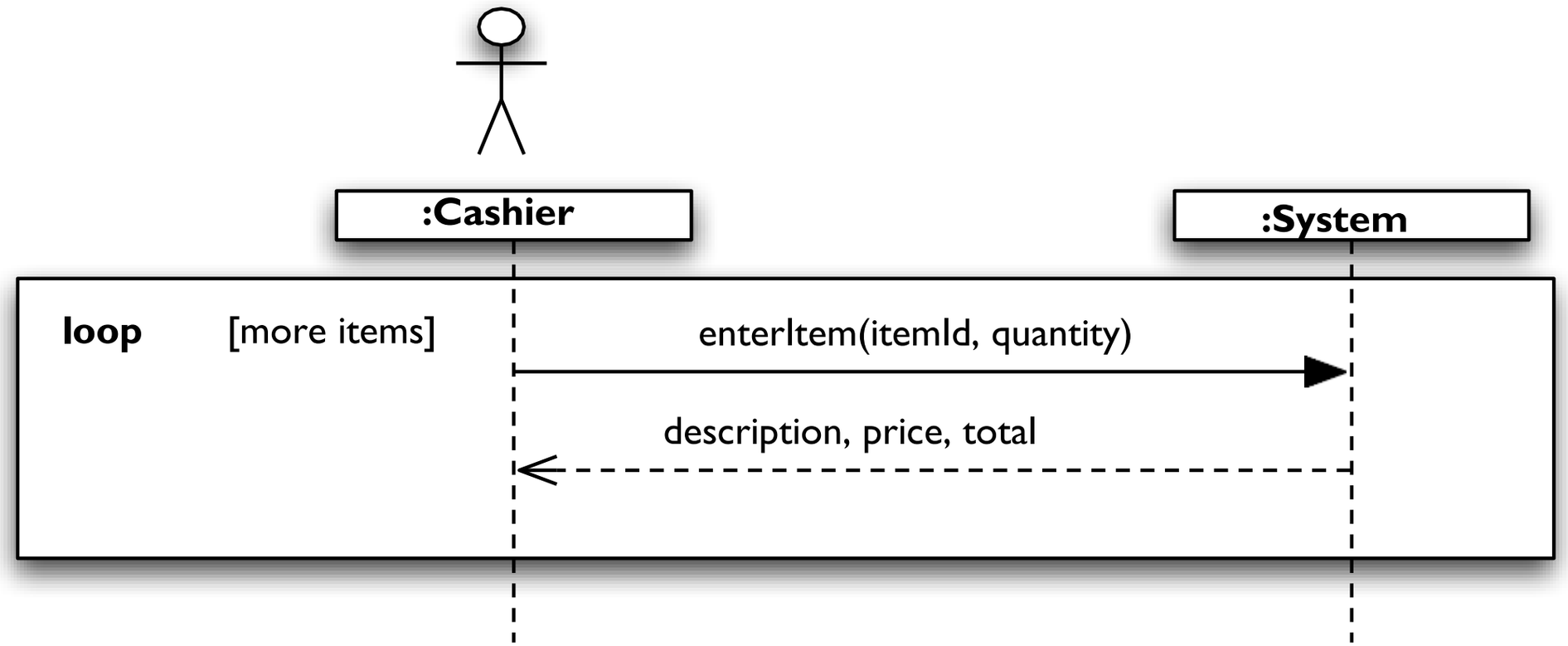
Invoking Static Methods (Class Methods)




```
public      class Register
   {  public      void
      report() {
      Locale[] locales =
      Calendar.getAvailableLocales();
   }
}
```

Corresponding Java Code

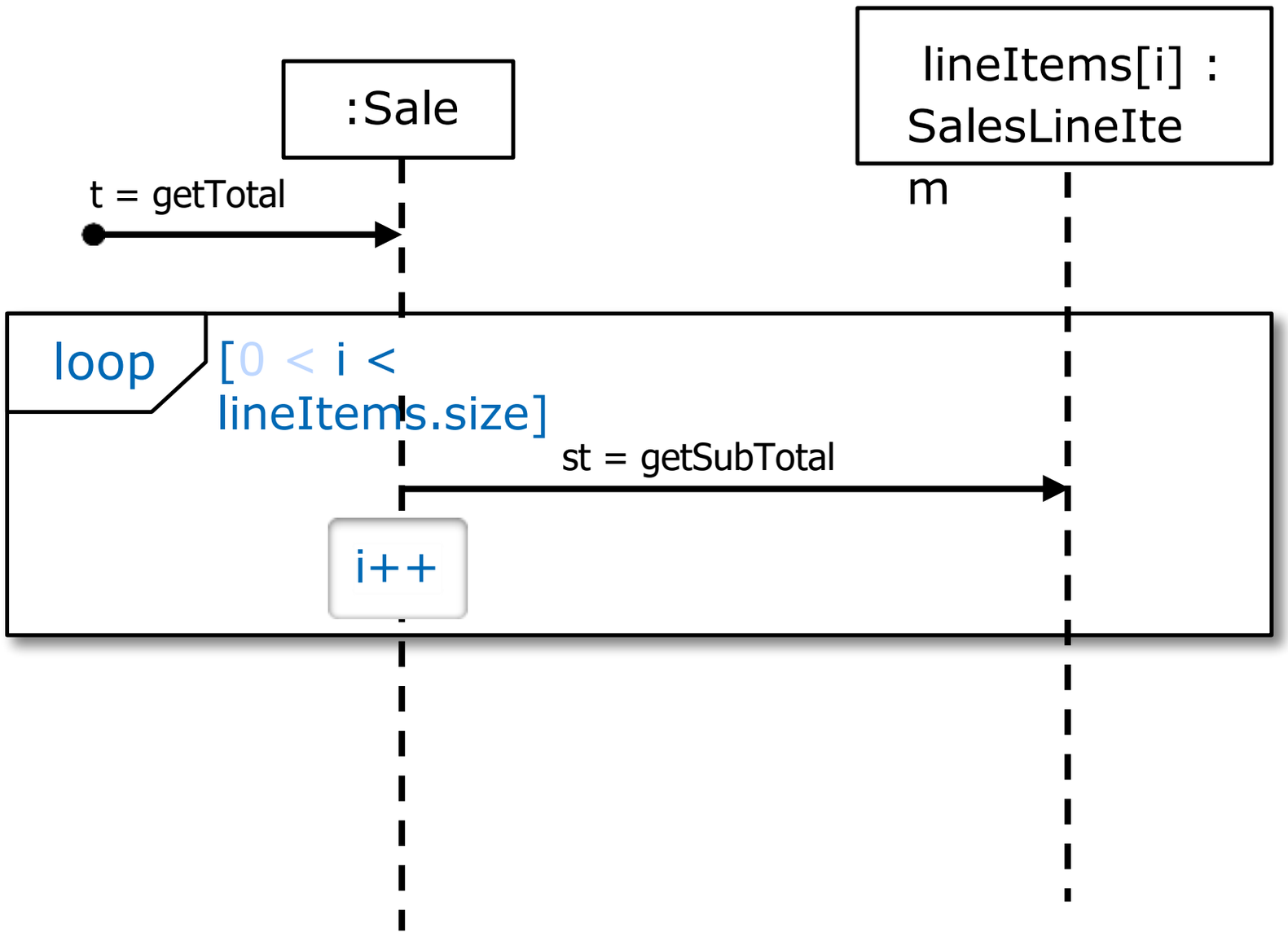
Diagram frames in UML sequence diagrams are used to support - among others - conditional and looping constructs.



How to model the iteration over a collection?

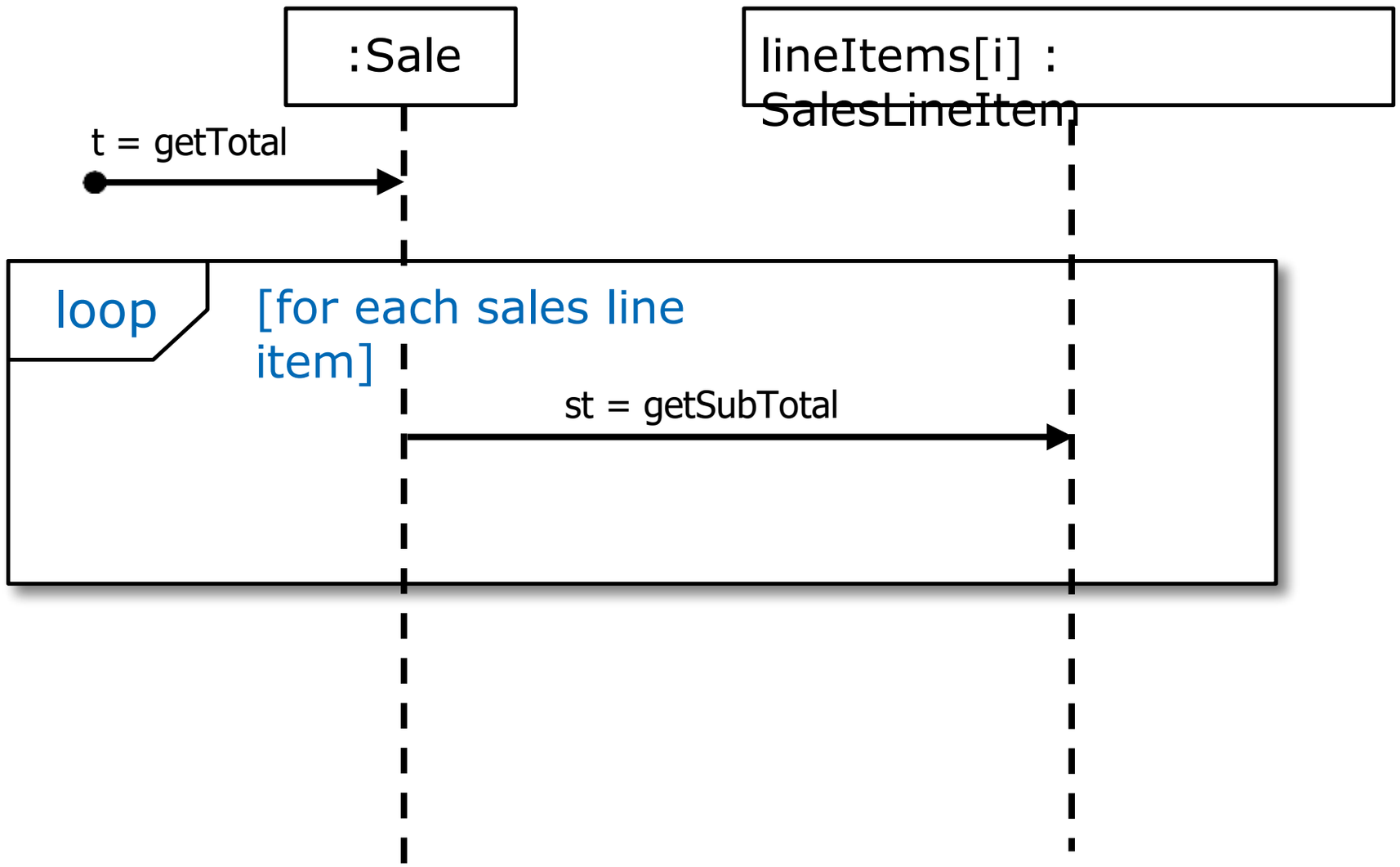
Modeling task: Calculate the total of a sale by summing up the sub totals for each sales line item.

Use a UML loop frame to iterate over a collection.



Modeling task: Calculate the total of a sale by summing up the sub totals for each sales line item.

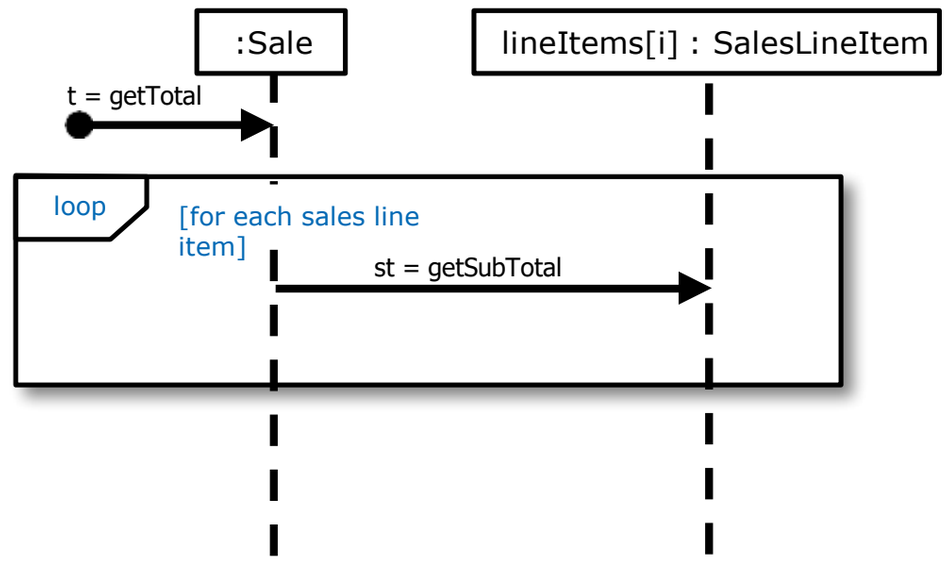
Use a UML loop frame to iterate over a collection.



Modeling task: Calculate the total of a sale by summing up the sub totals for each sales line item.

Java code corresponding to a UML loop frame.

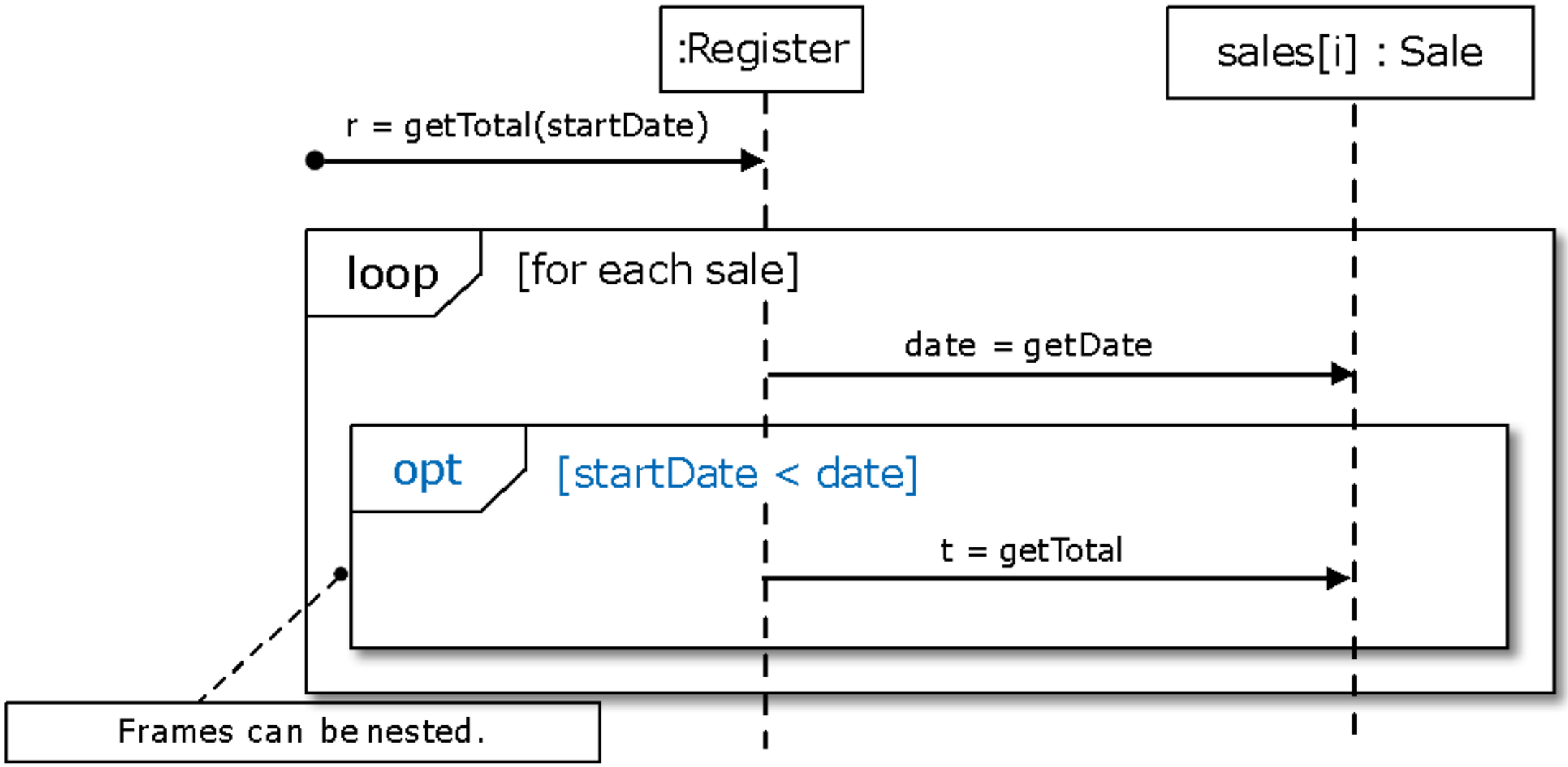

```
public      class
      Sale {
   private     List<SalesLineItem>
   lineItems
   =      new
   public   Money   getTotal()
      {      Money   t   =   new
      Money();    Money  st  =
      null;
      for  (SalesLineItem  lineItem  :  lineItems)
         {  st = lineItem.getSubtotal();
    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

}
```



How to model the sending of a message only if a guard condition matches?


```
Modeling task: Get the sum of all sales that happened today
after 18:00 o’clock.
```

Use a UML opt frame to model the sending of a message if the guard condition matches.

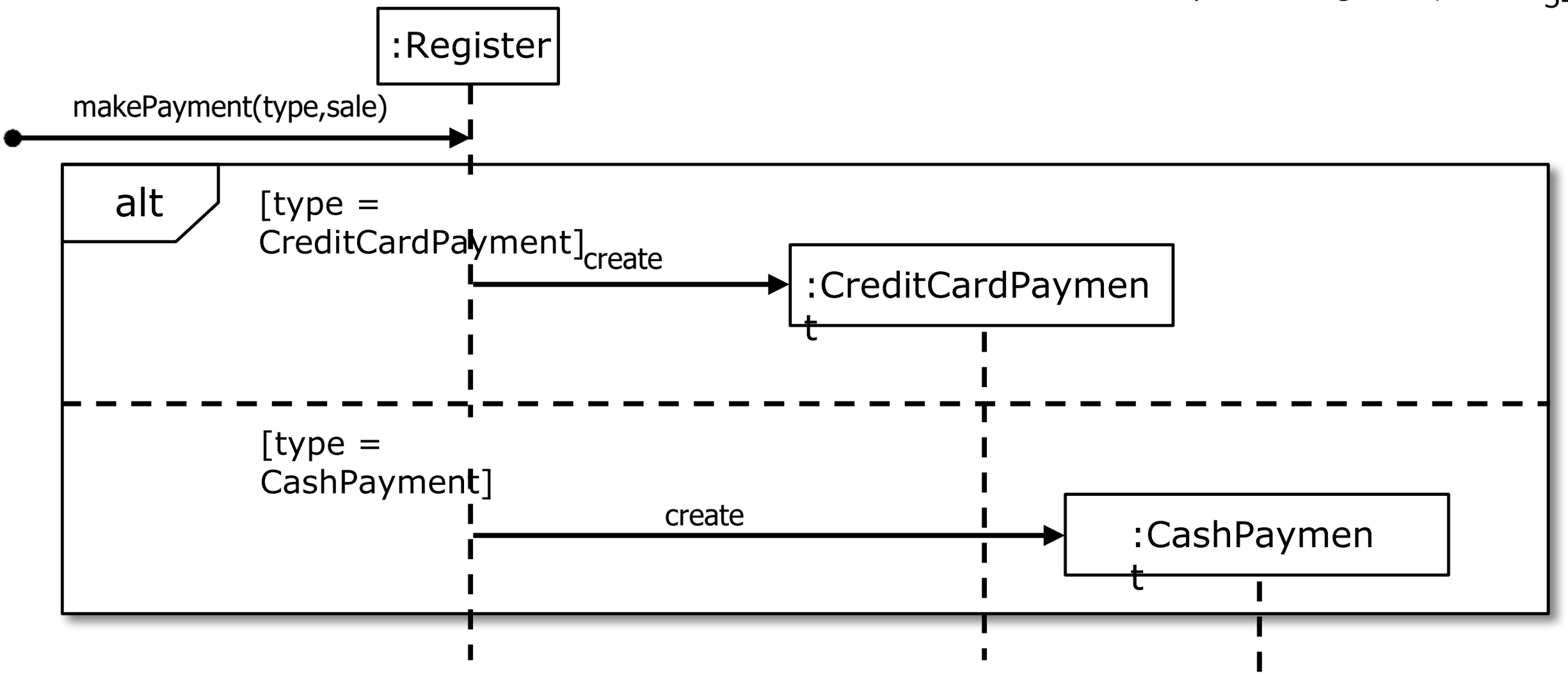


Modeling task: Get the sum of all sales that happend today after 18:00 o'clock.

How to model mutually exclusive alternatives?

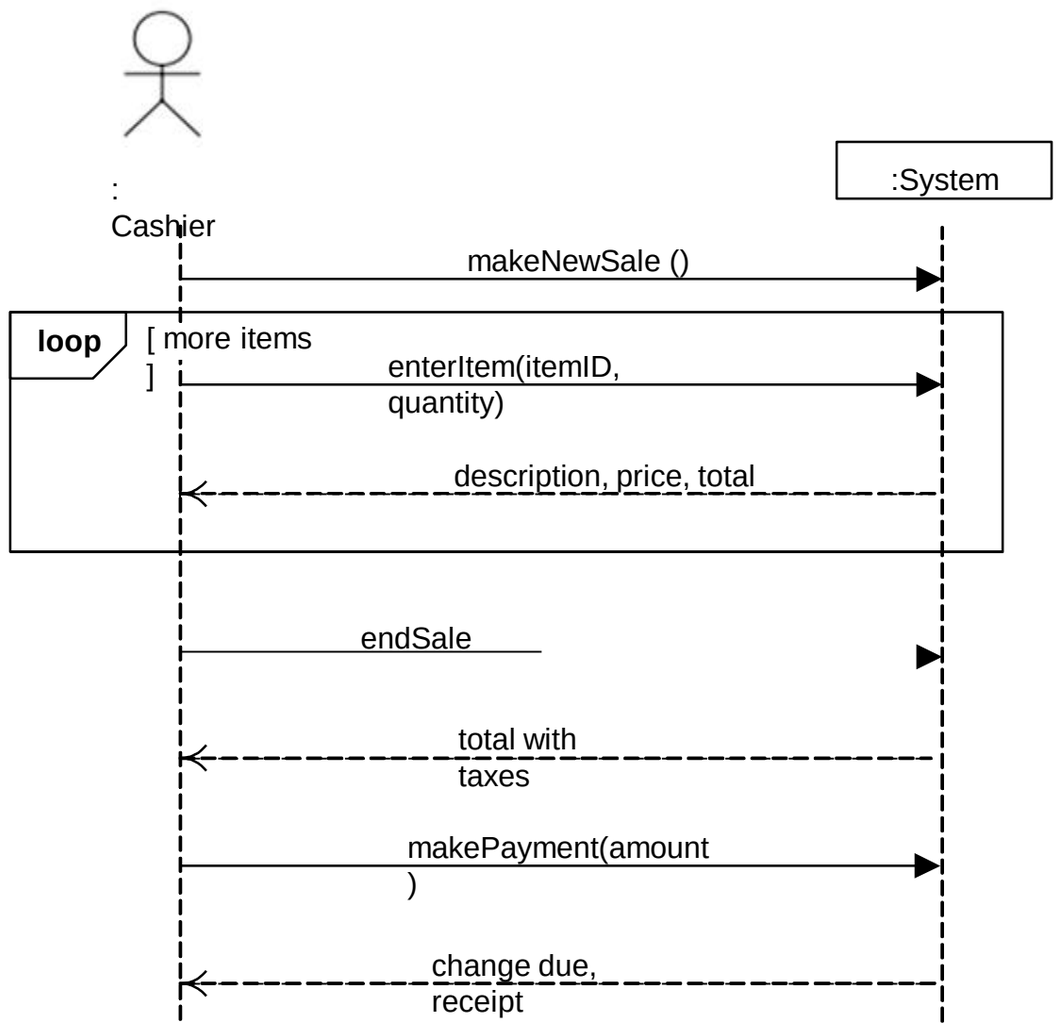
Modeling task: A register should be able to handle credit card payments and cash payments.

Use the UML alt frame to model between 2 and n mutually exclusive alternatives.



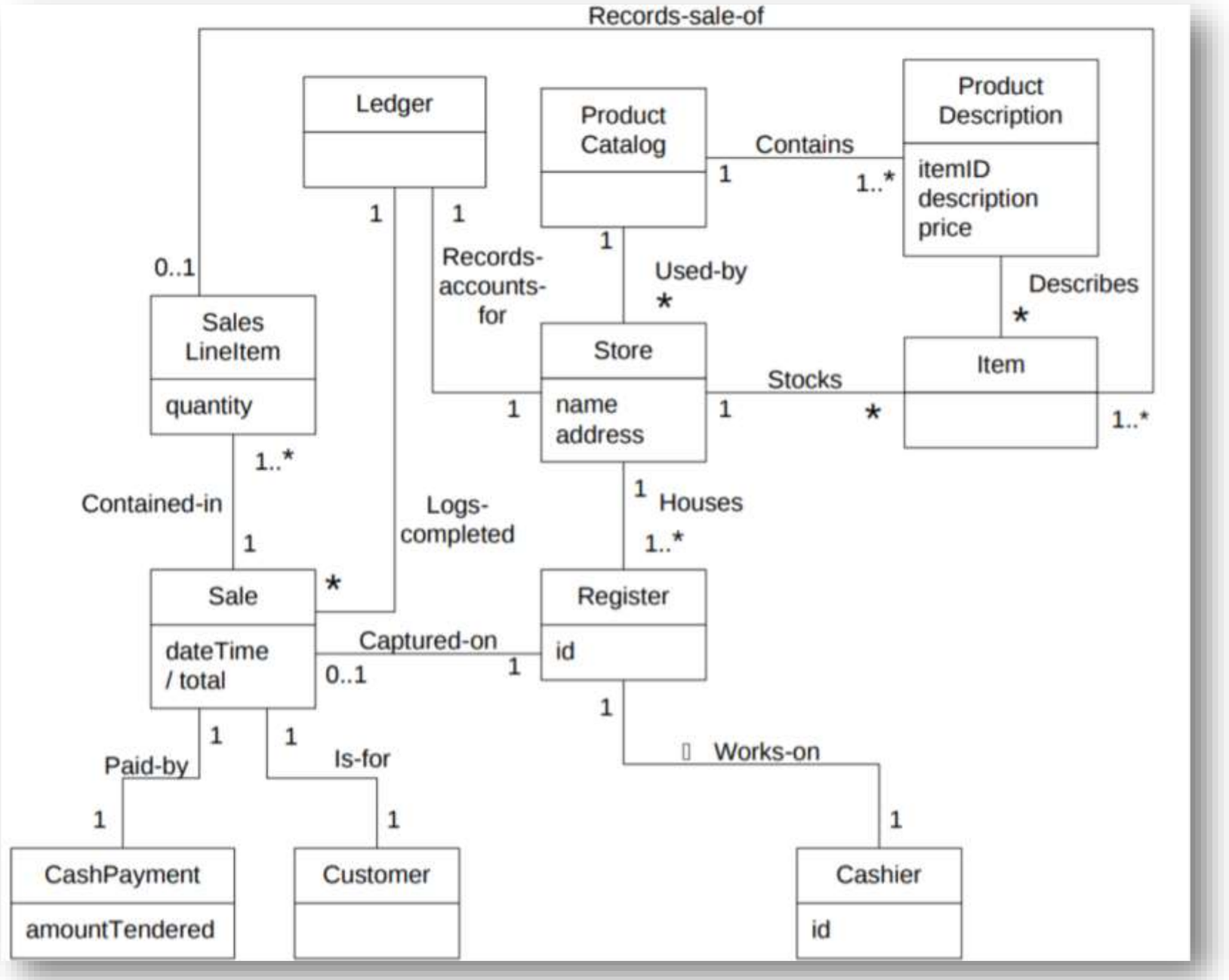
Modeling task: A register should be able to handle credit card payments and cash payments.

Process Sale Scenario

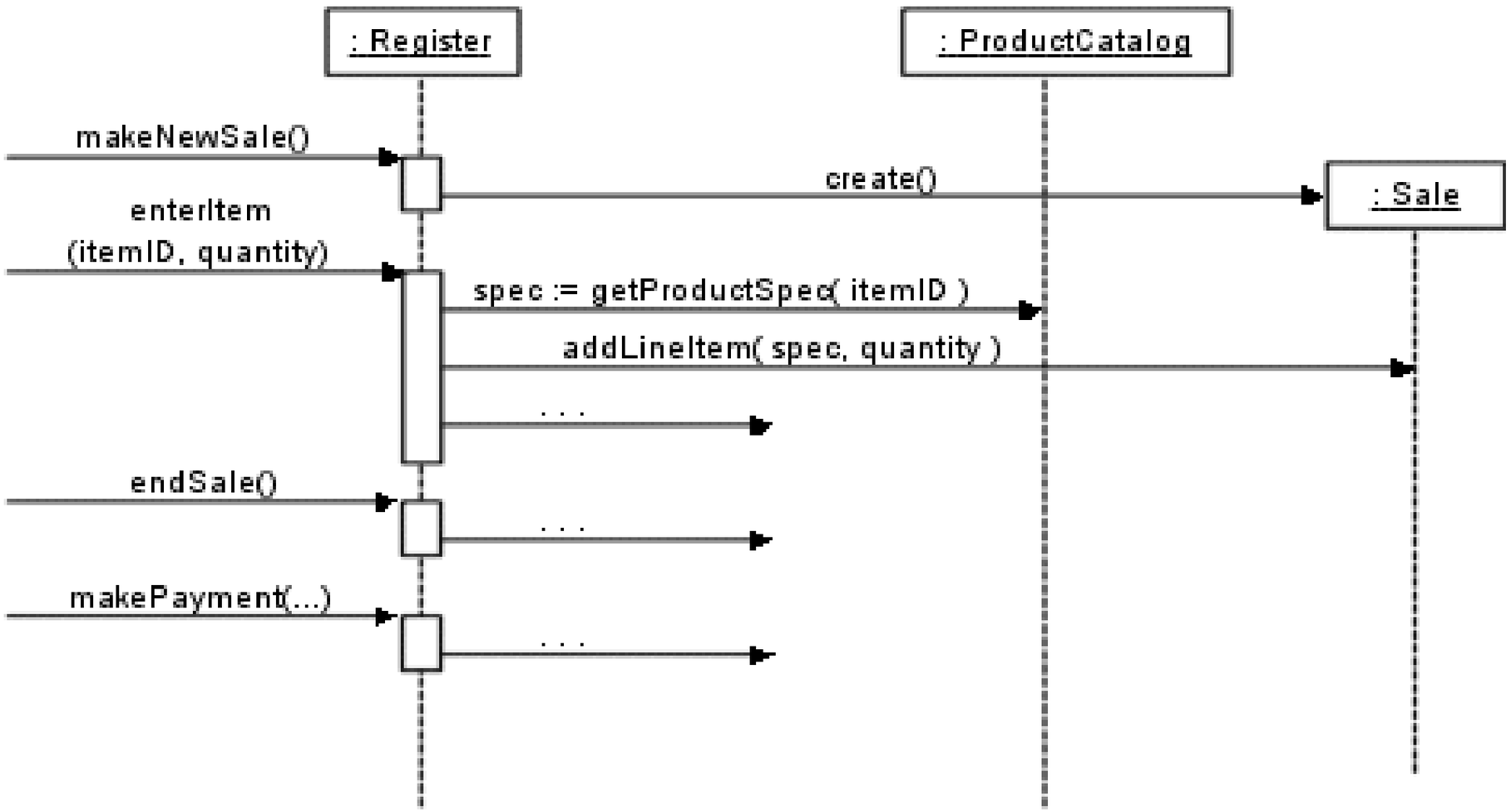


1. **makeNewSale()**
2. **enterItem(itemID, quantity)**
3. **endSale()**
4. **makePayment(amount)**

Domain Model



Process Sale



Summary

The goal of this lecture is to enable you to systematically carry out small(er) software projects that produce quality software.

- Modeling the dynamic behavior is often more rewarding than modeling the static structure w.r.t. understanding a domain
- Modeling the dynamic behavior is often particularly useful if the control-flow is more involved; but only draw the part that is relevant to understand the problem at hand
- The UML is often used informally - this is OK if everyone interprets the diagrams in the same way