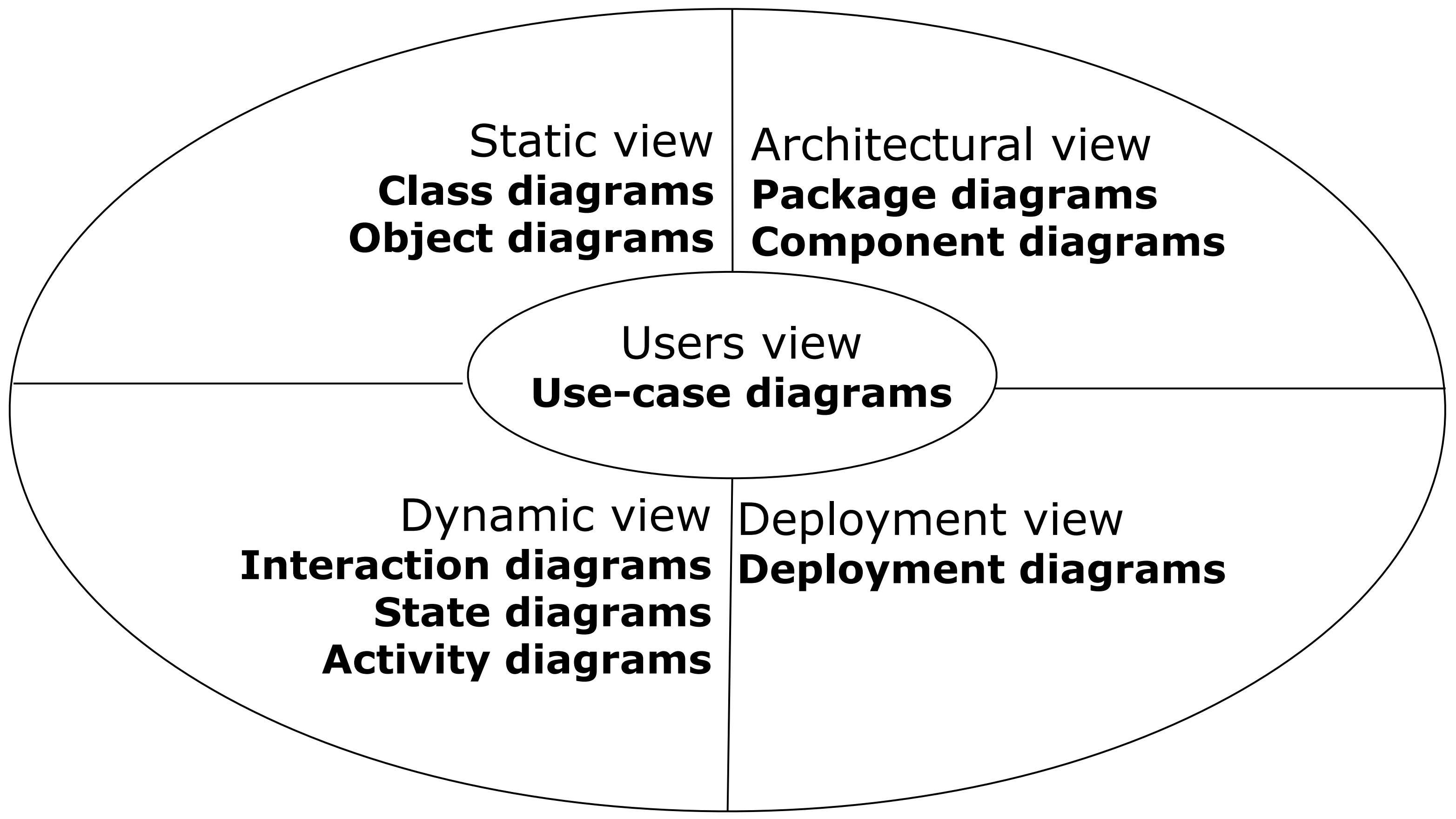
Modelling dynamic behaviour

- Activity diagrams
- State diagrams
- Interaction diagrams



Interaction diagrams

- The essential elements of an interaction diagrams
  - Objects
  - Actors
  - Messages

- Actions between objects and actors are
  - message sendings
  - object creations and destructions

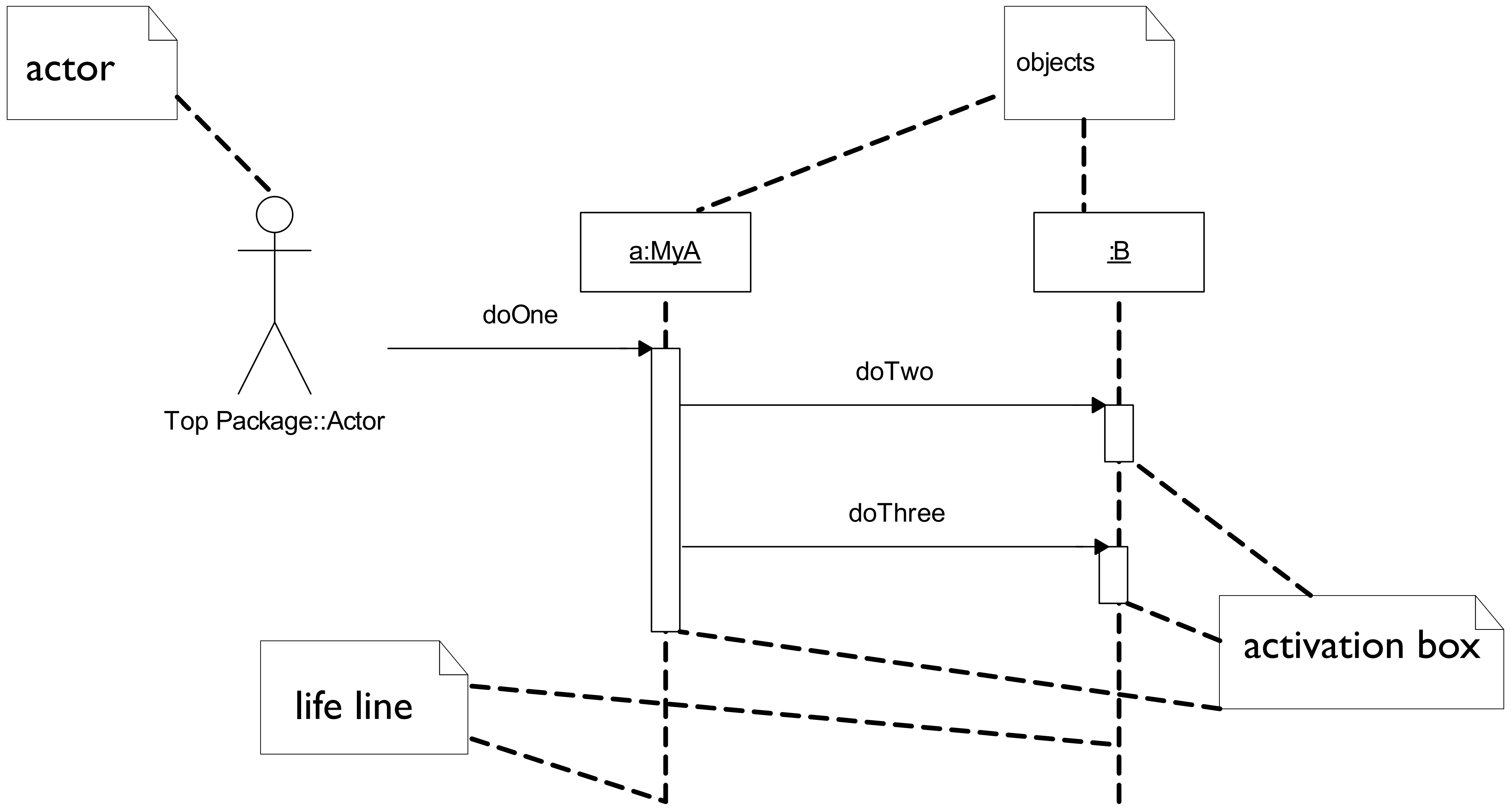
- **Two types of interaction diagrams**
  - **Sequence diagrams**
    - The temporal sequence of interactions
  - **Collaboration diagrams**
    - An instance of class diagram

Sequence diagram

- A **sequence diagram** describes the **temporal sequence** of exchanges of messages between objects and the actor to perform a certain task
  - The **actor** who initiates interactions is usually found on the far left
  - The **objects** are placed horizontally on the diagram
  - The vertical dimension represents time
  - Each object or actor is associated with a **life line** representing the time where the object or actor is
  - An **activation box** represents the object activation period

Sequence diagram

- Notation



Sequence diagram

- Messages
  - Message is the medium of communication between objects
  - The general form of message

[guard]message(parameters)

- **guard**: a condition must be satisfied in order to send the message
- **message**: the identifier of the sent message
- **paramaters**: a list of parameter values

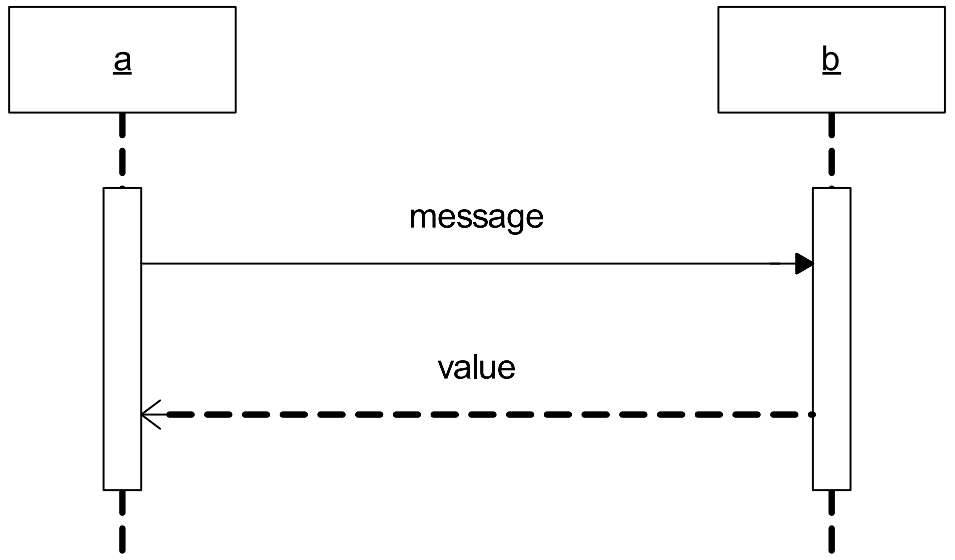
- Note: guard and parameters can be omitted

Sequence diagram

- The return values
  - Sending a message to an object cause **the execution of a method** of this object
    - This method can optionally return a value
  - The return values may be omitted or be explicitly described
    - either as the following form

[guard]value := message(parameters)

- or by a return message that represents graphically

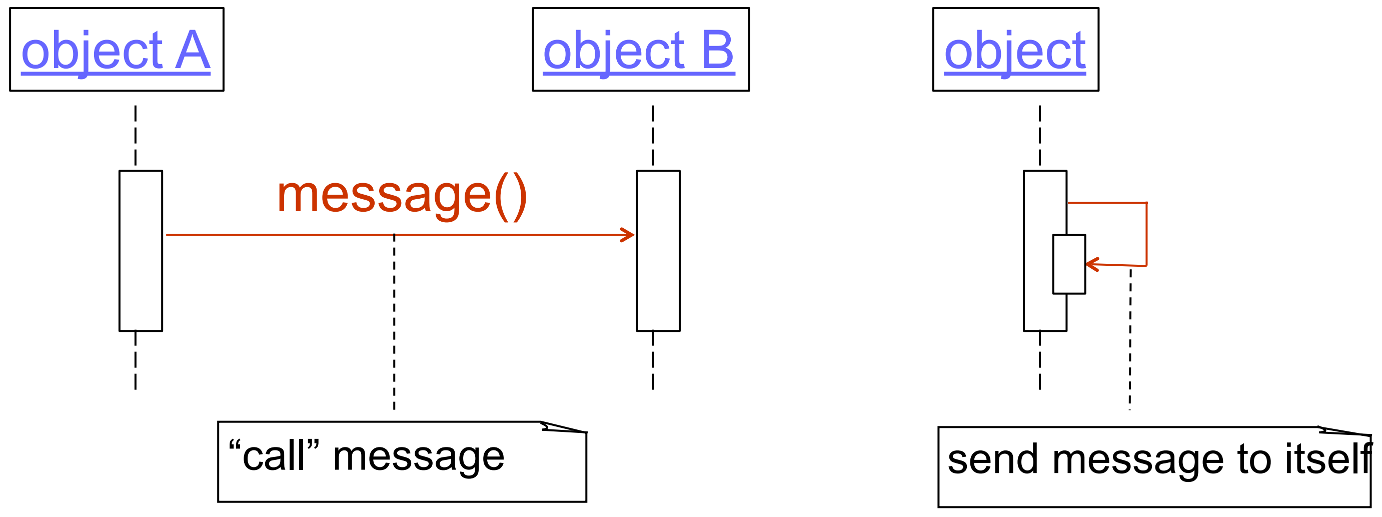


Sequence diagram

- Types of message
  - “call” message
  - “return” message
  - “send” message
  - “create” message
  - “destroy” message

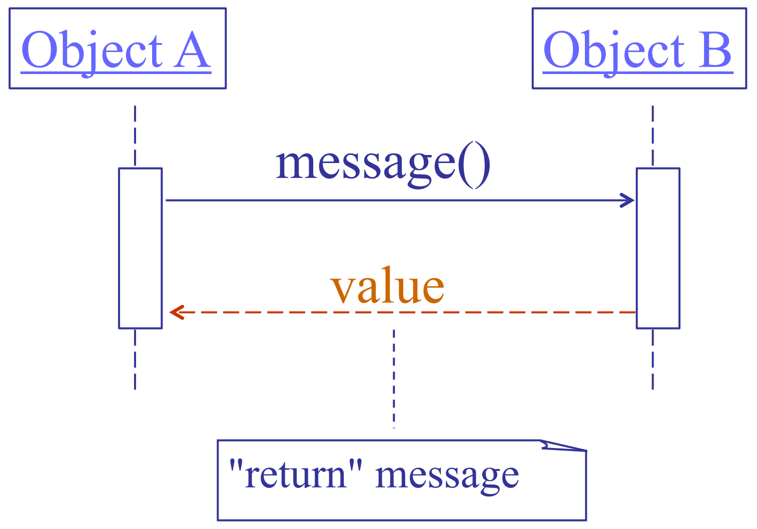
Sequence diagram

- "call" message
  - A "call" message invokes an operation/method of the object
  - A "call" message is a **synchronous message**: the object that sends the message must wait for the termination of the execution of the message before doing other tasks
  - An object can send message to itself
  - Notation



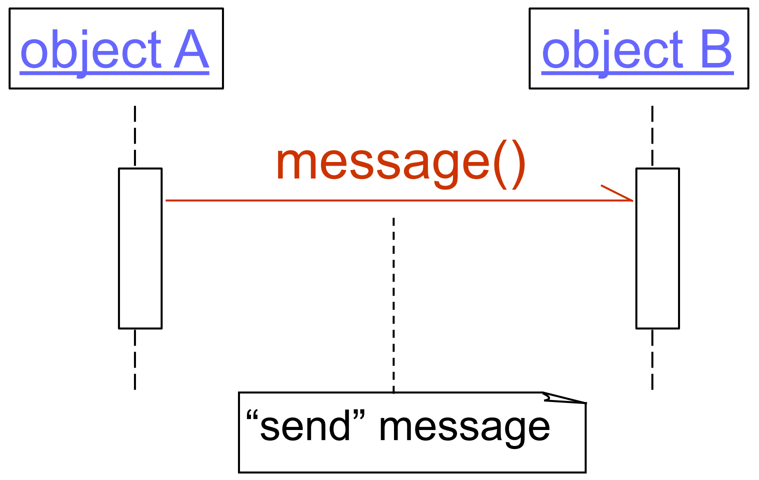
Sequence diagram

- The “return” message returns a value for the calling object
- Notation



Sequence diagram

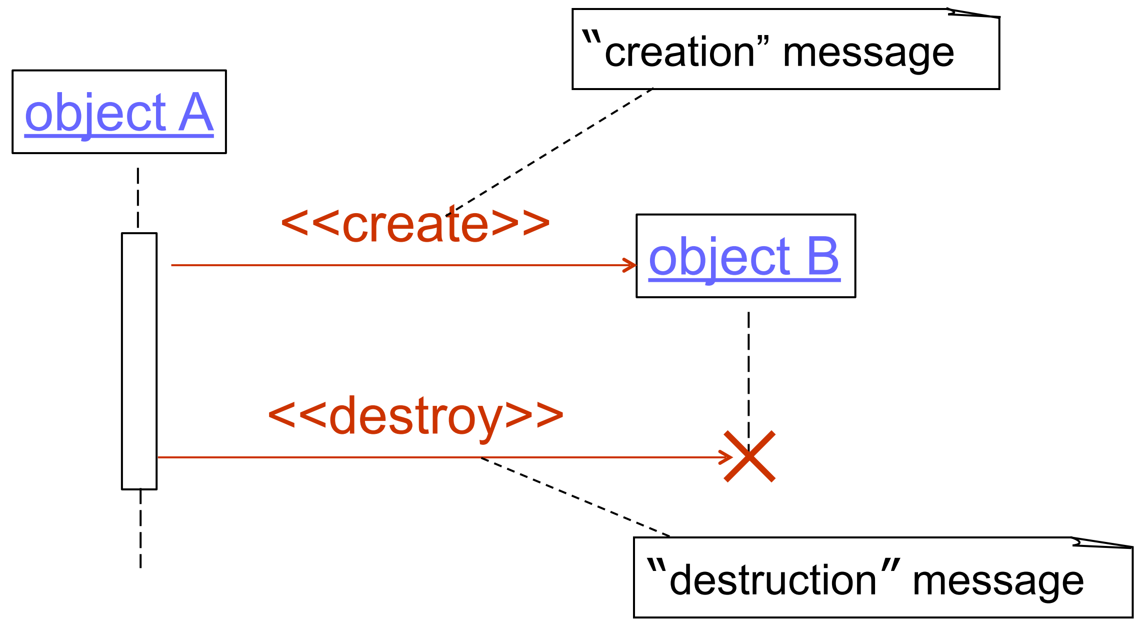
- “send” message
  - A “send” message sends a signal to an object
  - A “send” message is an **asynchronous message**: once the object sends the message, it expects nothing and continues to do other tasks
  - Notation



  - Asynchronous message is often used in multi-threaded environment
    - For example, *Thread.start(), Runnable.run()* in Java

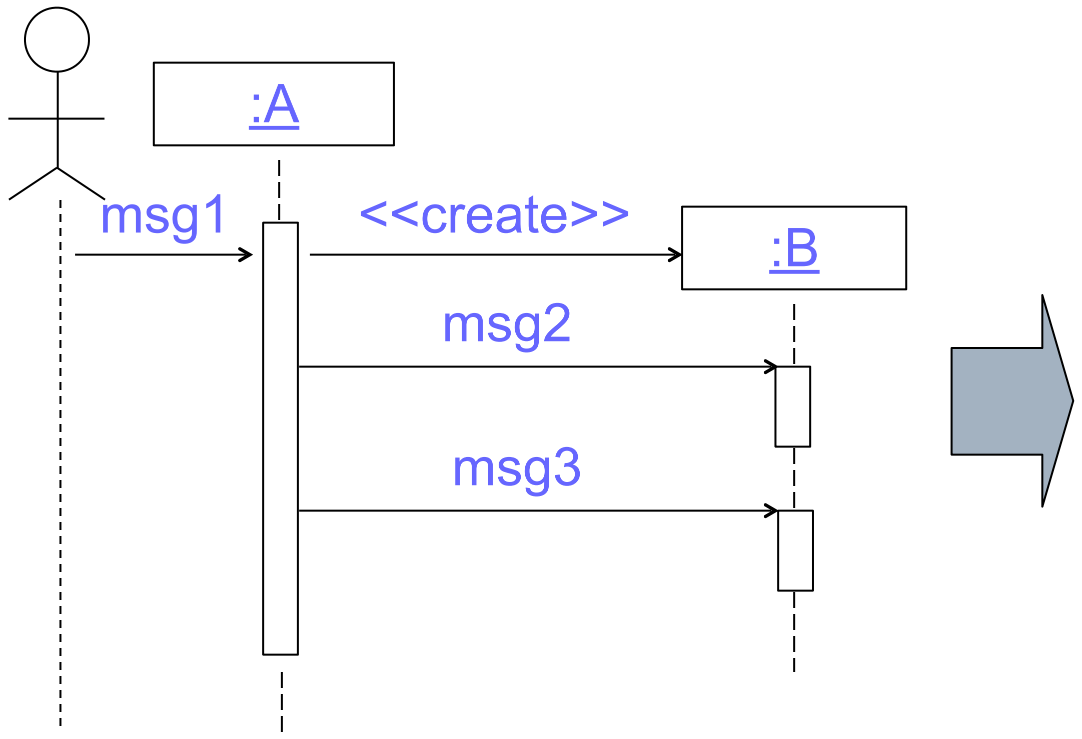
Sequence diagram

- “creation” message
  - invokes the creation method of object (constructor)
- “destruction” message
  - invokes the destruction message of message (destructor)
- Notation



Sequence diagram

- Example
  - The sequence diagram and the corresponding code

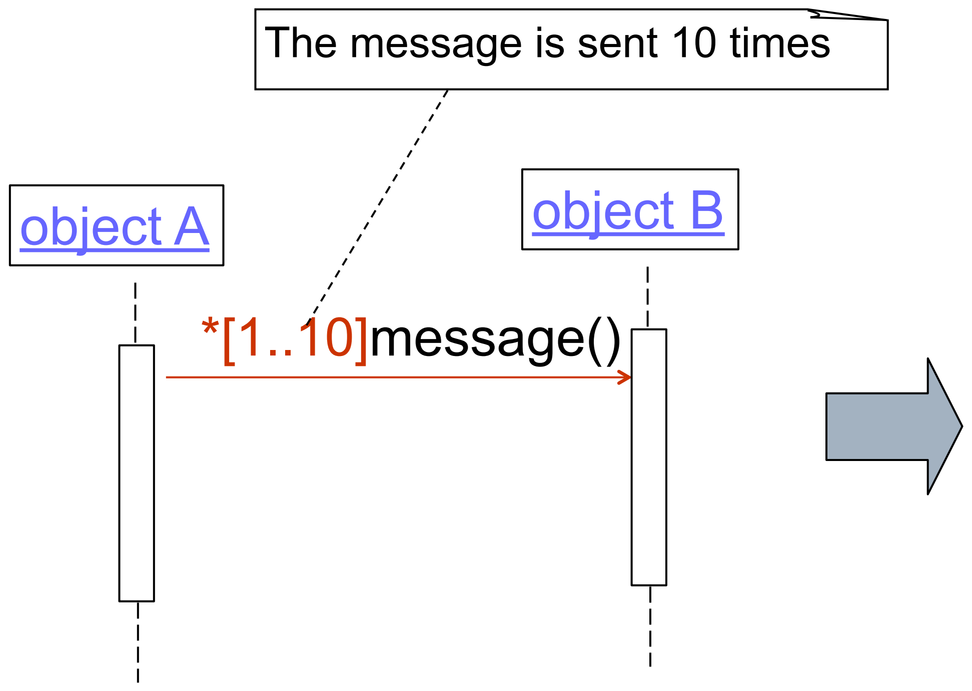


```
public class A
{
  private B objB;
  public void msg1()
  {
     objB = new B();
     objB.msg2();
     objB.msg3();
   }
}

public class B
{
  …
   public void msg2() { … }
   public void msg3() { … }
}
```

Sequence diagram

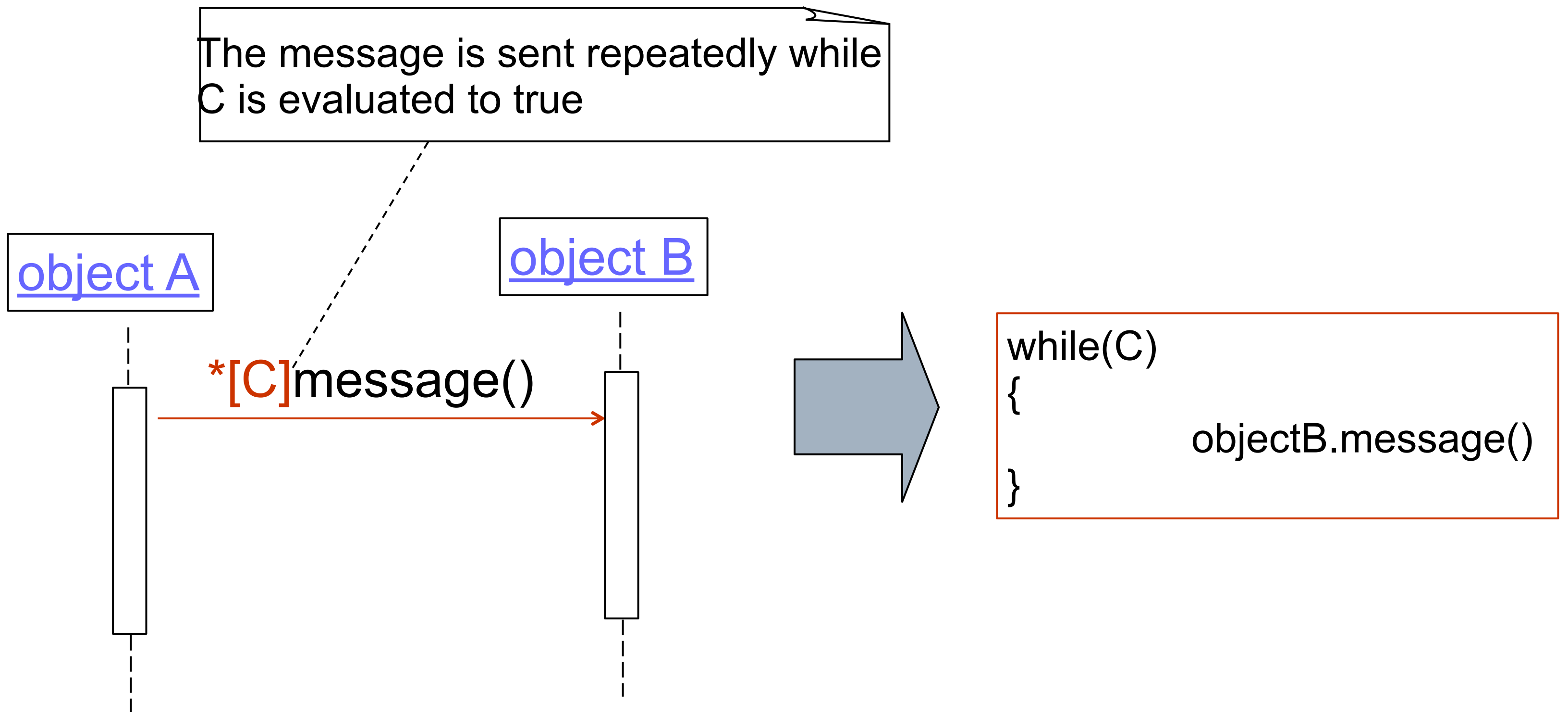
- A message can be **sent iteratively**
- Example



```
for(i = 1; i<= 10; i++)
{
            objectB.message()
}
```

Sequence diagram

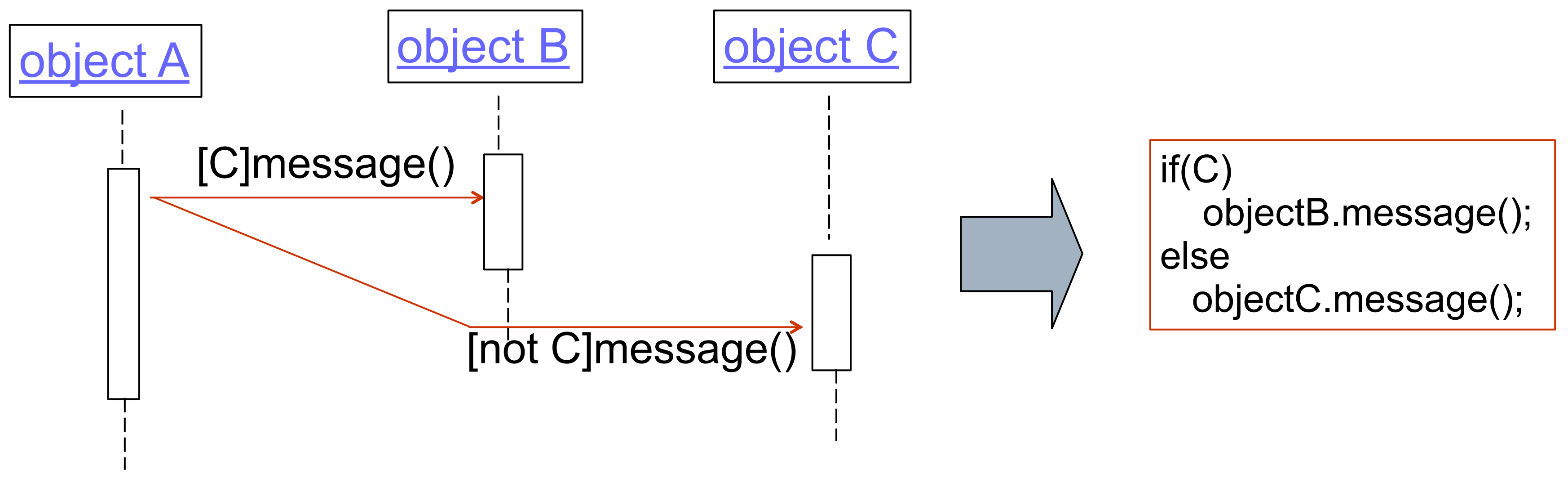
- A message can be sent iteratively based on a condition
- Example



```
while(C)
{
                objectB.message()
}
```

Sequence diagram

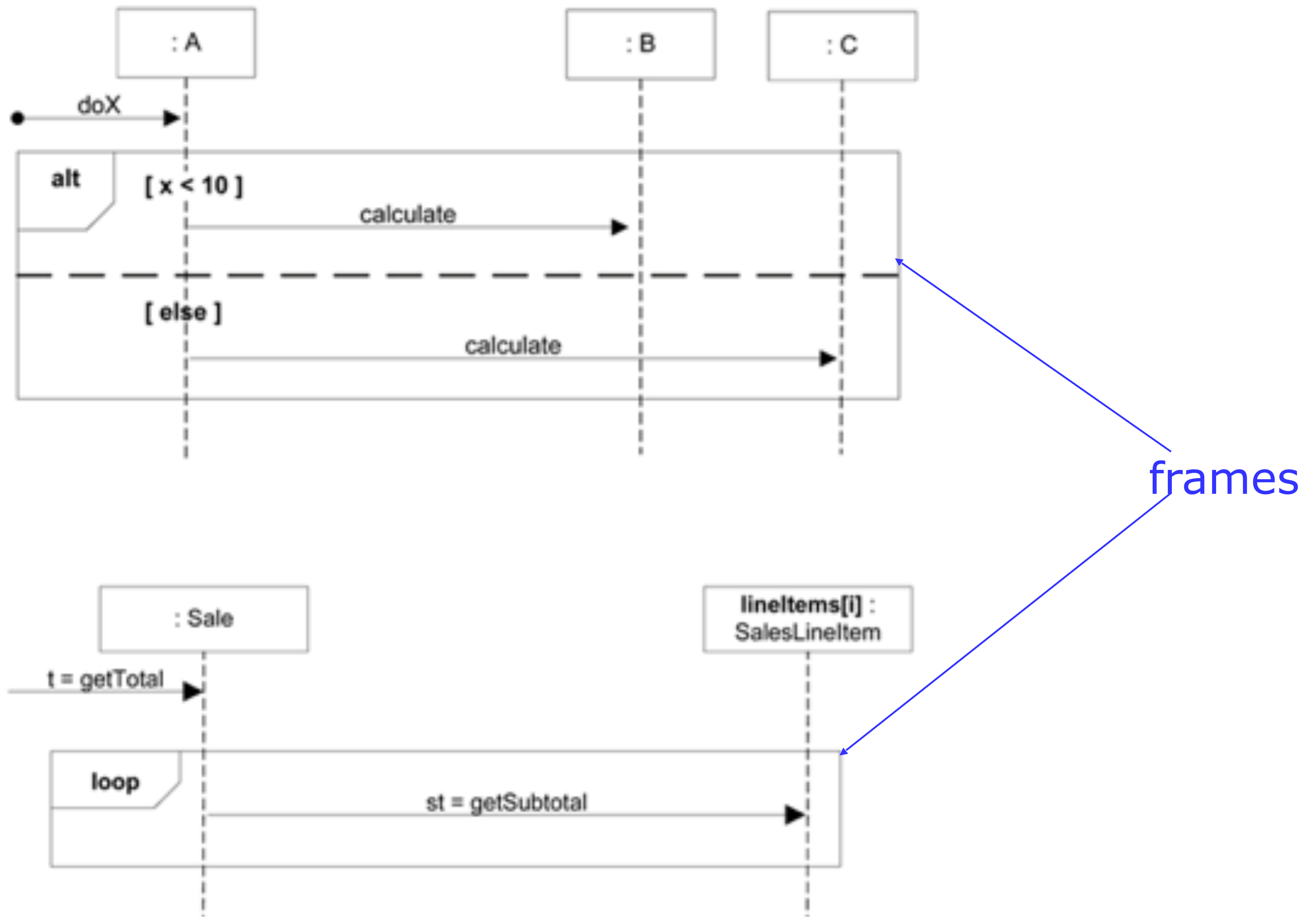
- The sending of a message can depend on a **decision**
- Example



```
if(C)
    objectB.message();
else
   objectC.message();
```

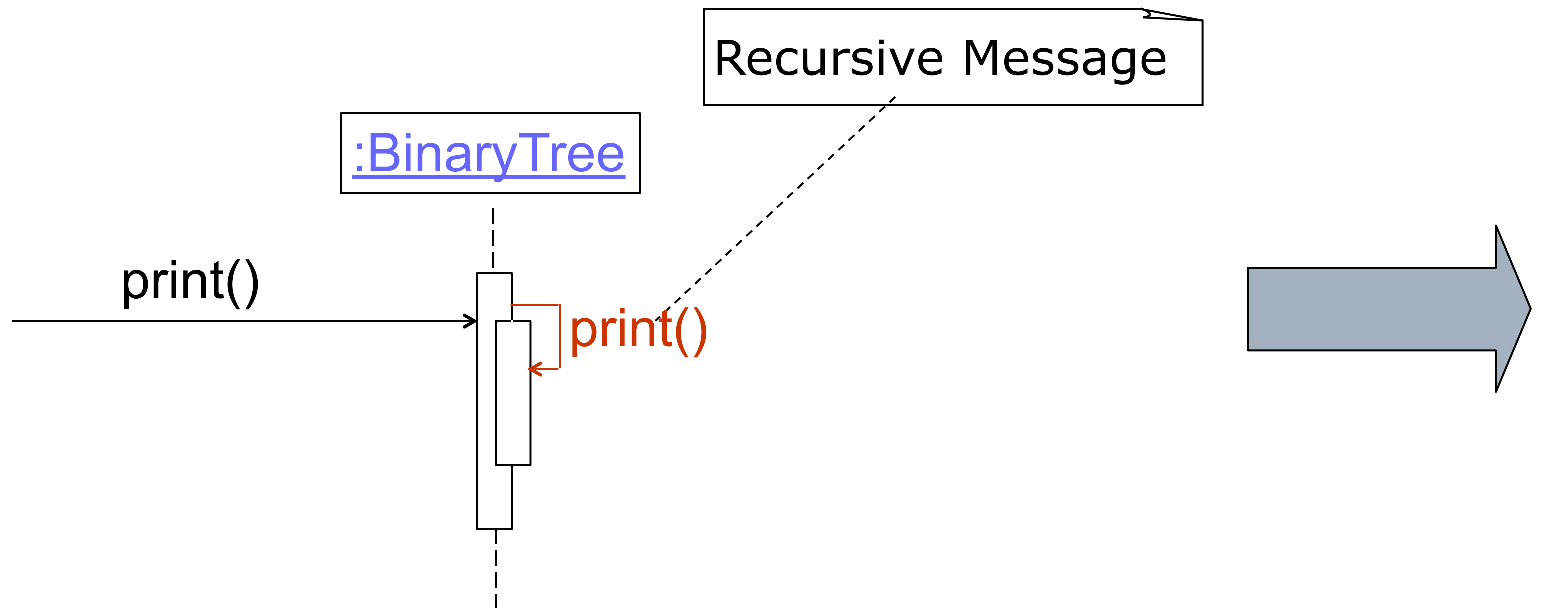
Sequence diagram

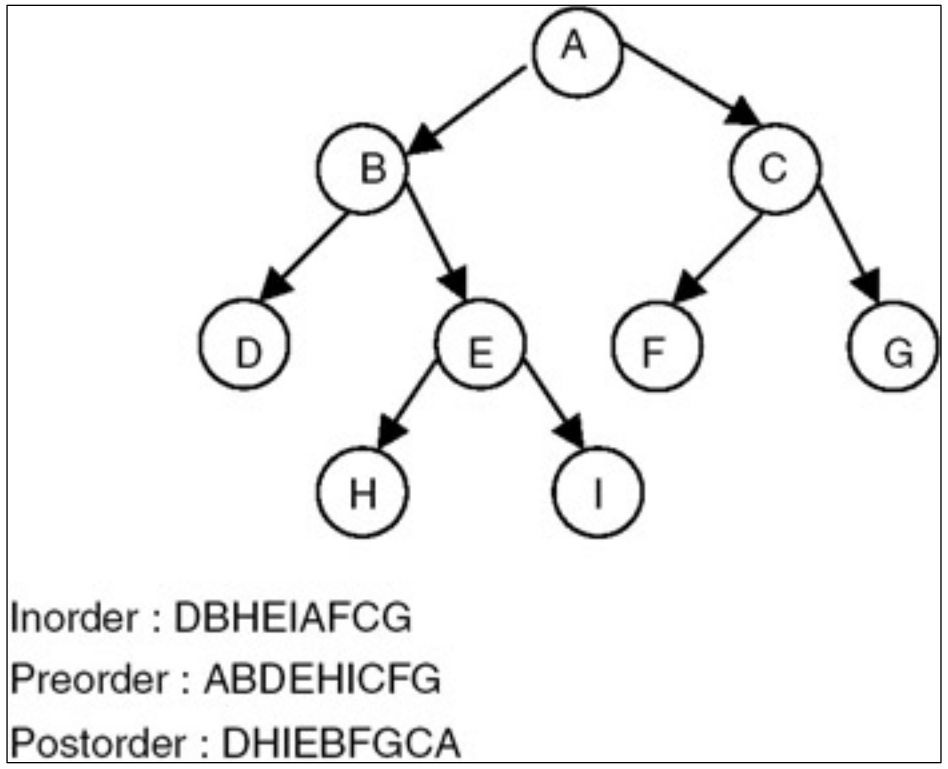
- Note: UML 2.x notations allow the use of frames to represent the conditions or iterations



Sequence diagram

- A message can be called **recursively**
- Notation

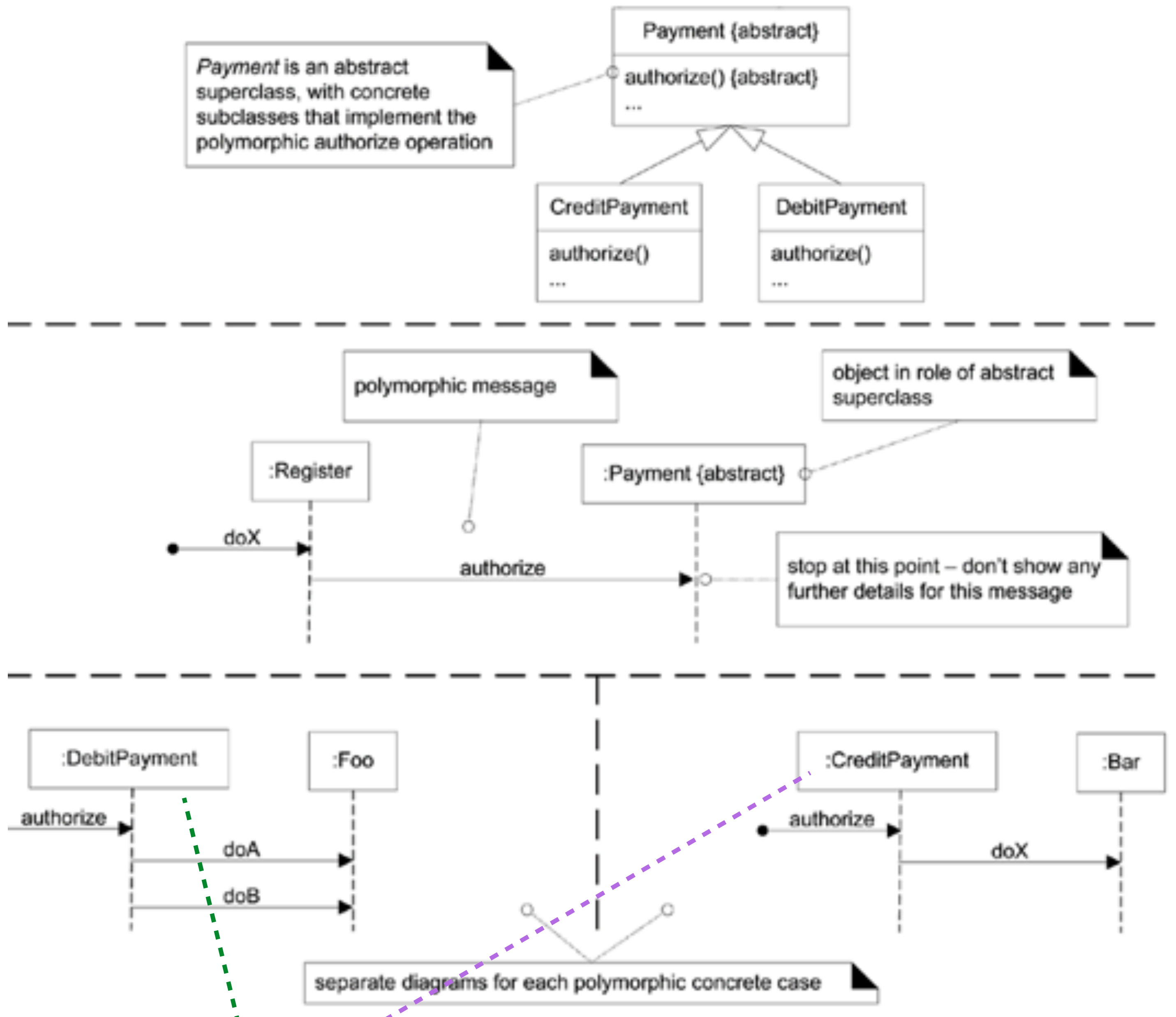


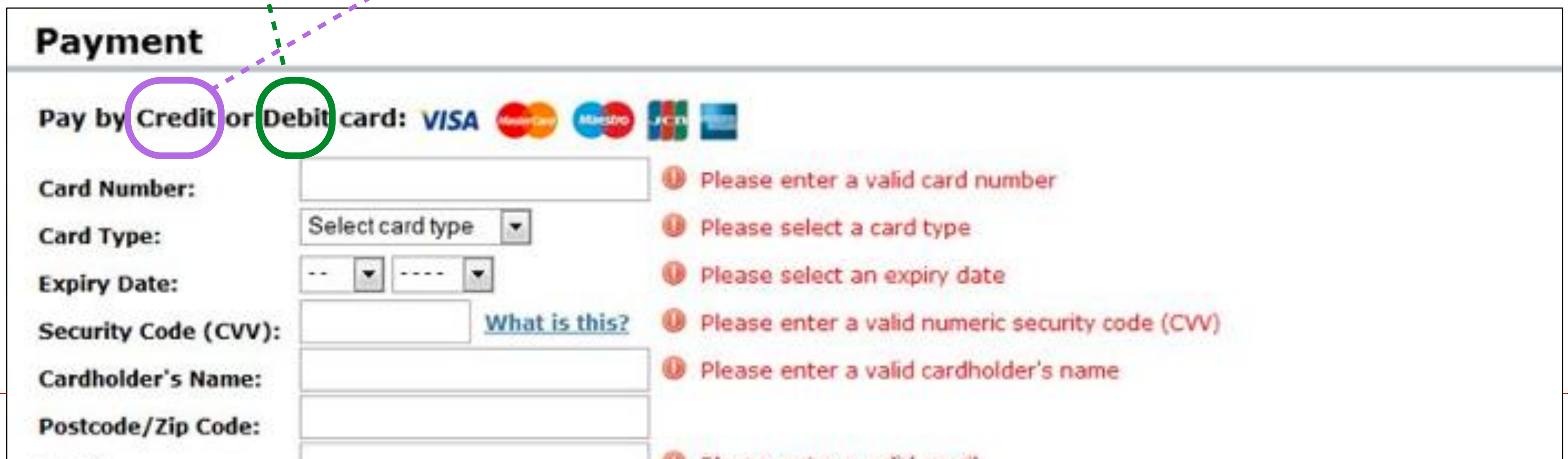


Inorder : DBHEIAFCG
Preorder : ABDEHICFG
Postorder : DHIEBFGCA

Sequence diagram

Modelling a polymorphic message


Payment is an abstract superclass, with concrete subclasses that implement the polymorphic authorize operation
Payment {abstract}
authorize() {abstract}
...
CreditPayment
authorize()
...
DebitPayment
authorize()
...
polymorphic message
object in role of abstract superclass
:Register
:Payment {abstract}
doX
authorize
stop at this point – don't show any further details for this message
:DebitPayment
:Foo
authorize
doA
doB
:CreditPayment
:Bar
authorize
doX
separate diagrams for each polymorphic concrete case



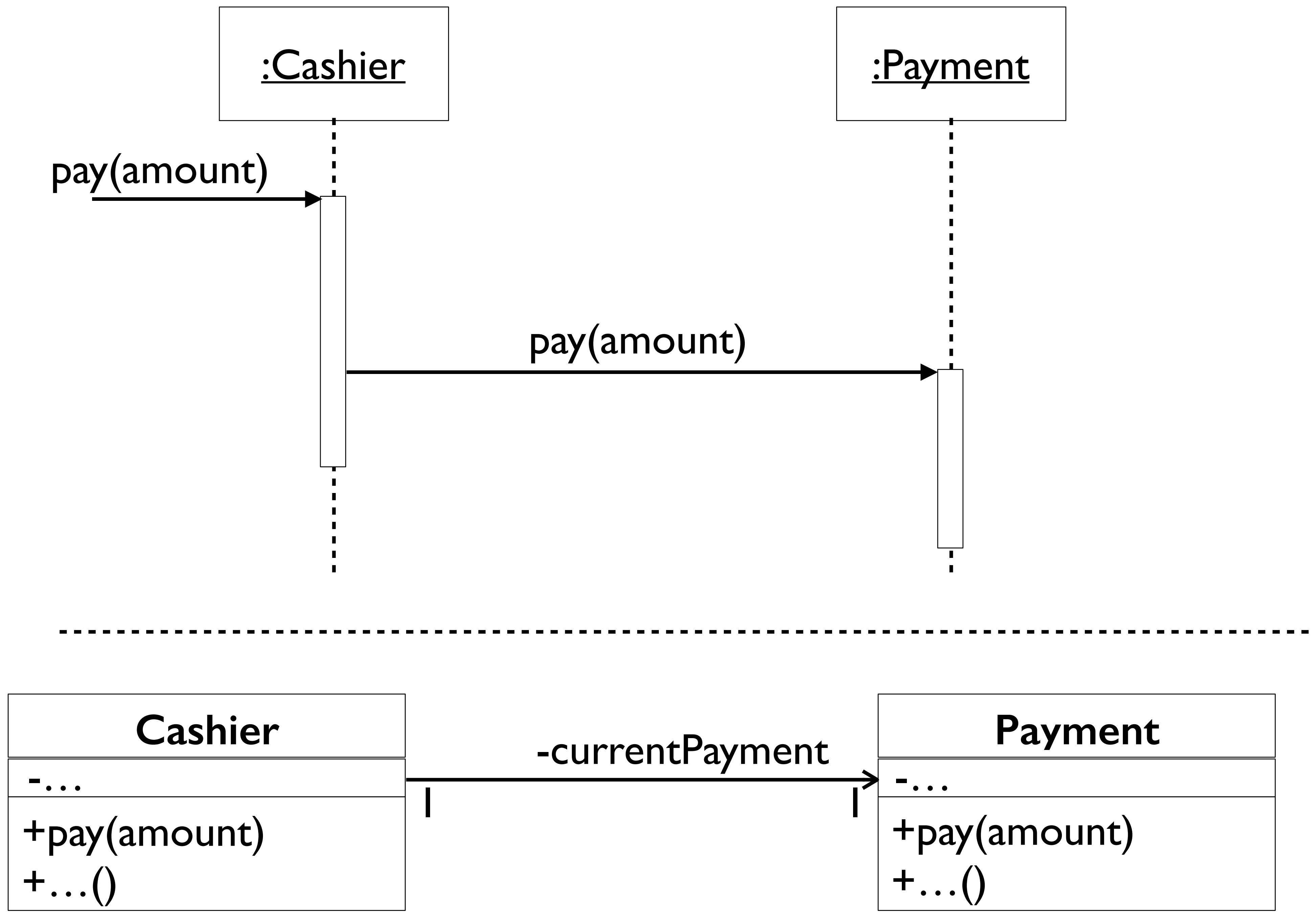
Payment

Pay by Credit or Debit card:

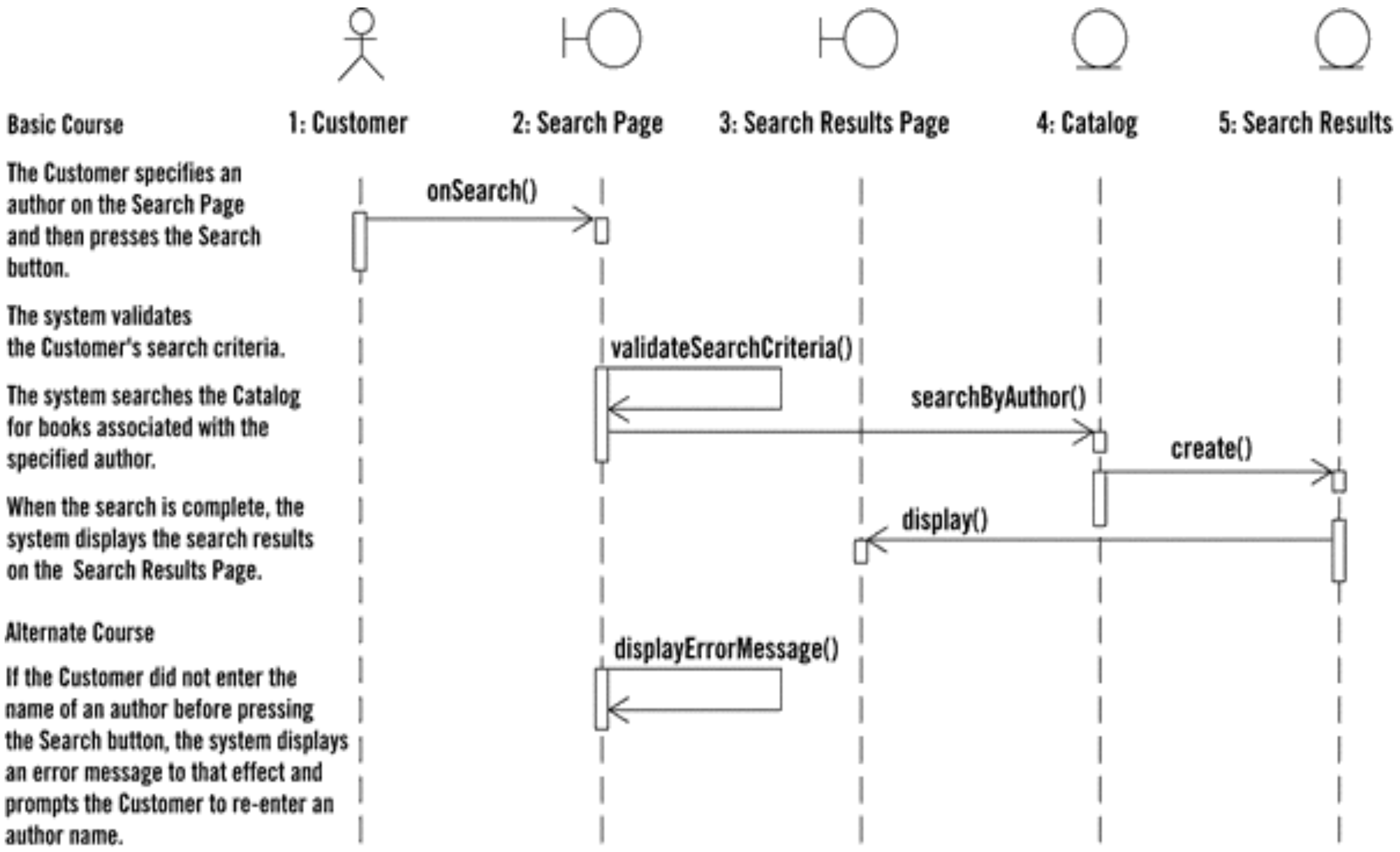
Card Number:		Please enter a valid card number
Card Type:	Select card type	Please select a card type
Expiry Date:	-- / ----	Please select an expiry date
Security Code (CVV):	What is this?	Please enter a valid numeric security code (CVV)
Cardholder's Name:		Please enter a valid cardholder's name
Postcode/Zip Code:		

Sequence diagram

- Relationship between class diagram and sequence diagram



Sequence diagram from use-case



Basic Course

The Customer specifies an author on the Search Page and then presses the Search button.

The system validates the Customer's search criteria.

The system searches the Catalog for books associated with the specified author.

When the search is complete, the system displays the search results on the Search Results Page.

Alternate Course

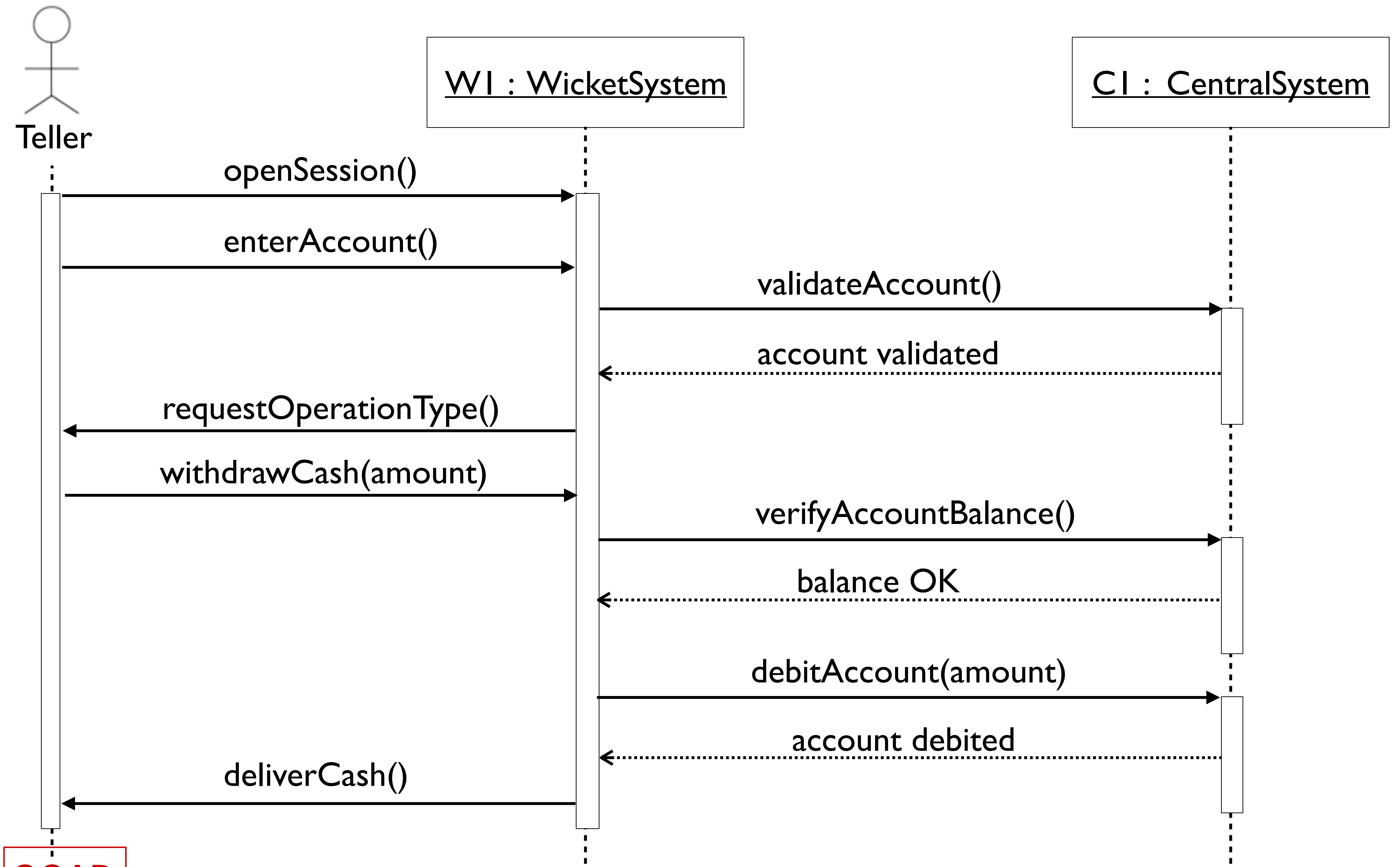
If the Customer did not enter the name of an author before pressing the Search button, the system displays an error message to that effect and prompts the Customer to re-enter an author name.

Sequence diagram

- Example: Cash withdrawal at the bank

Sequence diagram

- Example: Cash withdrawal at the bank

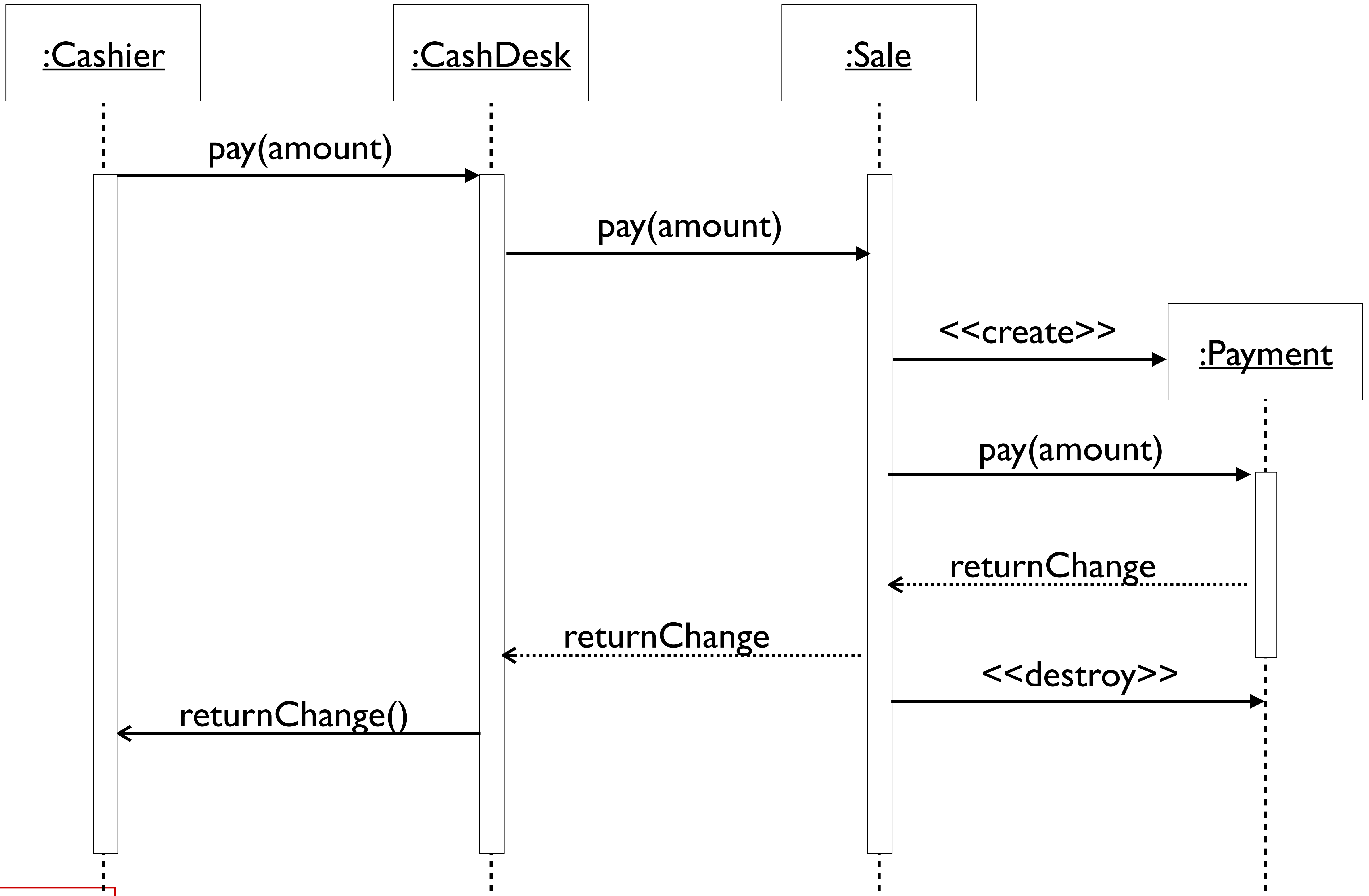


Sequence diagram

- Example: Use-case “cash payment”

Sequence diagram

- Example: Use-case “cash payment”



Why not just code it?

- Sequence diagrams can be somewhat close to the code level. So why not just code that algorithm rather than drawing it as a sequence diagram?
  - a good sequence diagram is still a bit above the level of the real code (not EVERY line of code is drawn on the diagram)
  - sequence diagrams are language-agnostic (can be implemented in many different languages)
  - non-coders can do sequence diagrams
  - easier to do sequence diagrams as a team
  - can see many objects/classes at a time on same page (visual bandwidth)

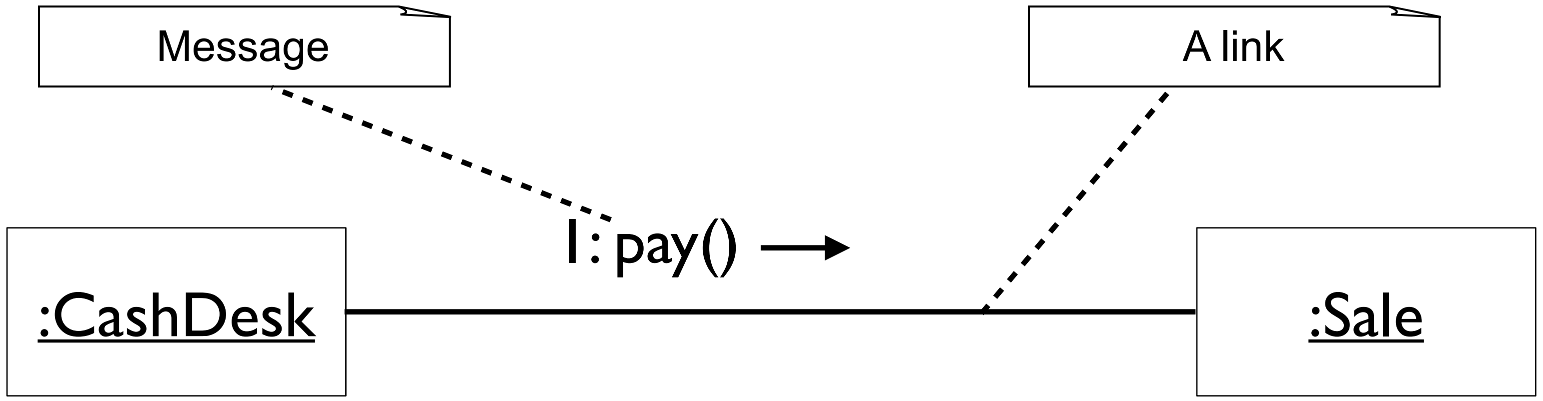
Collaboration/Communication diagram

- A collaboration diagram describes the interaction between objects
  - A collaboration diagram is a graph whose
    - nodes represent object
    - edges represent the communication between objects
  - The temporal ordering of messages is represented by a **numbering** of messages
  - Collaboration diagram is an extension of class diagram

Collaboration diagram

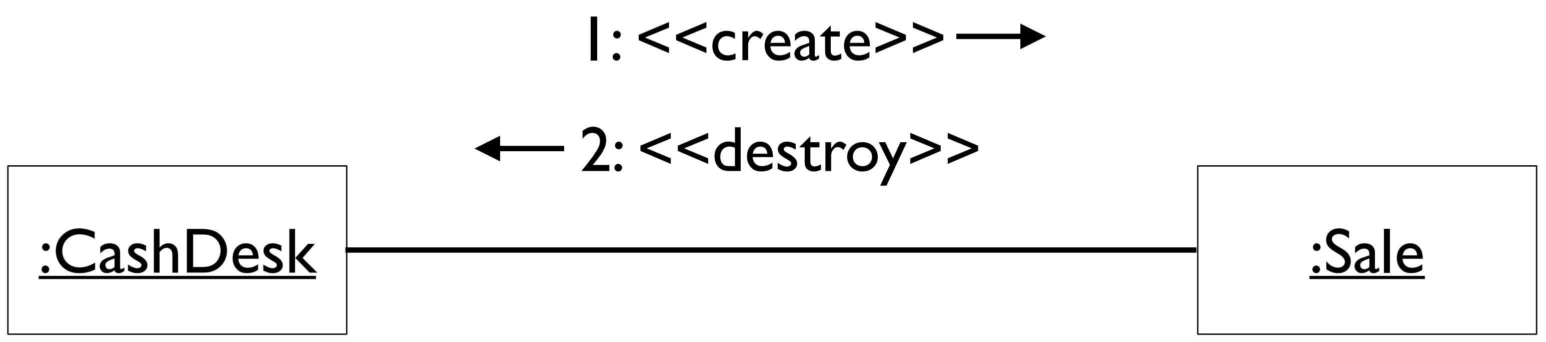
- Links
  - A link shows the sending of a message from an object to another object
  - Formally, a link is an instance of an association

- Messages
  - Each message between objects is presented by an expression of message and an arrow showing the direction of the message

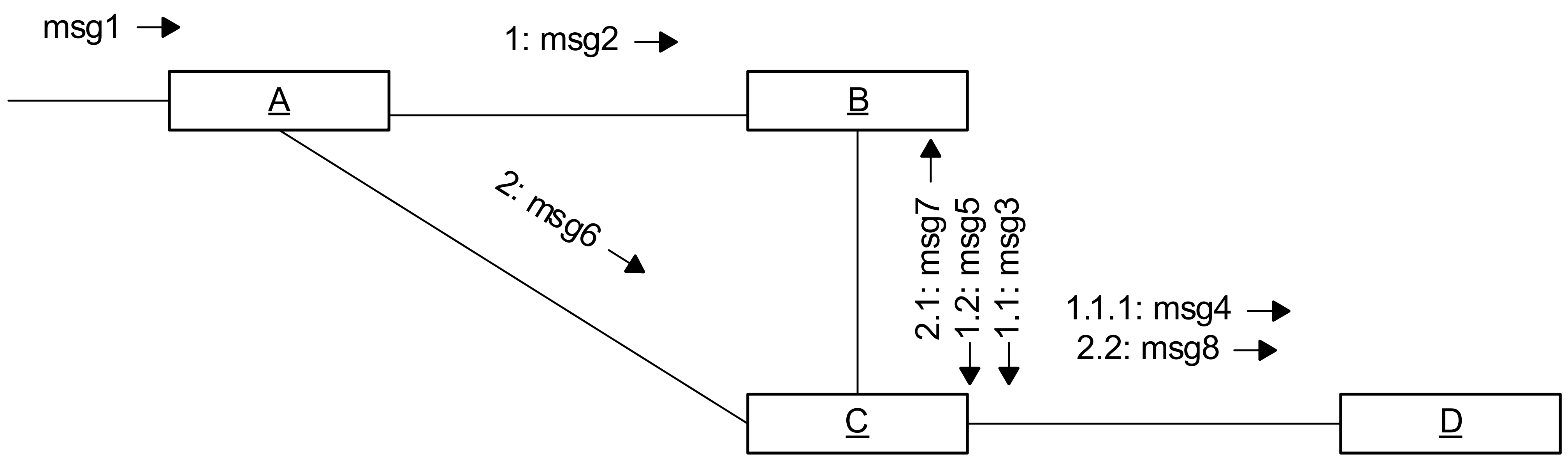


Collaboration diagram

- "creation" message and "destruction" message

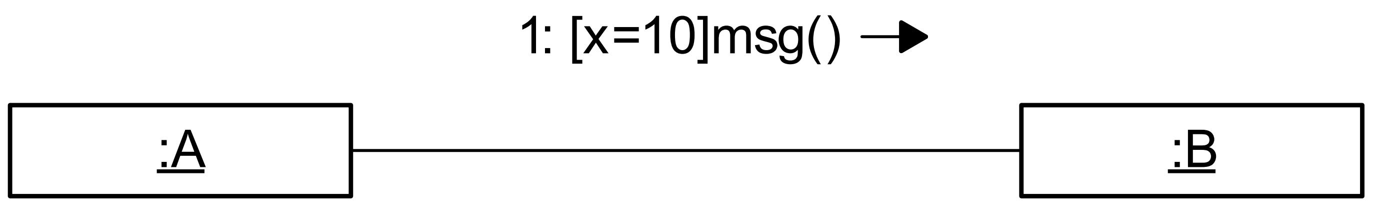


- Message numbering

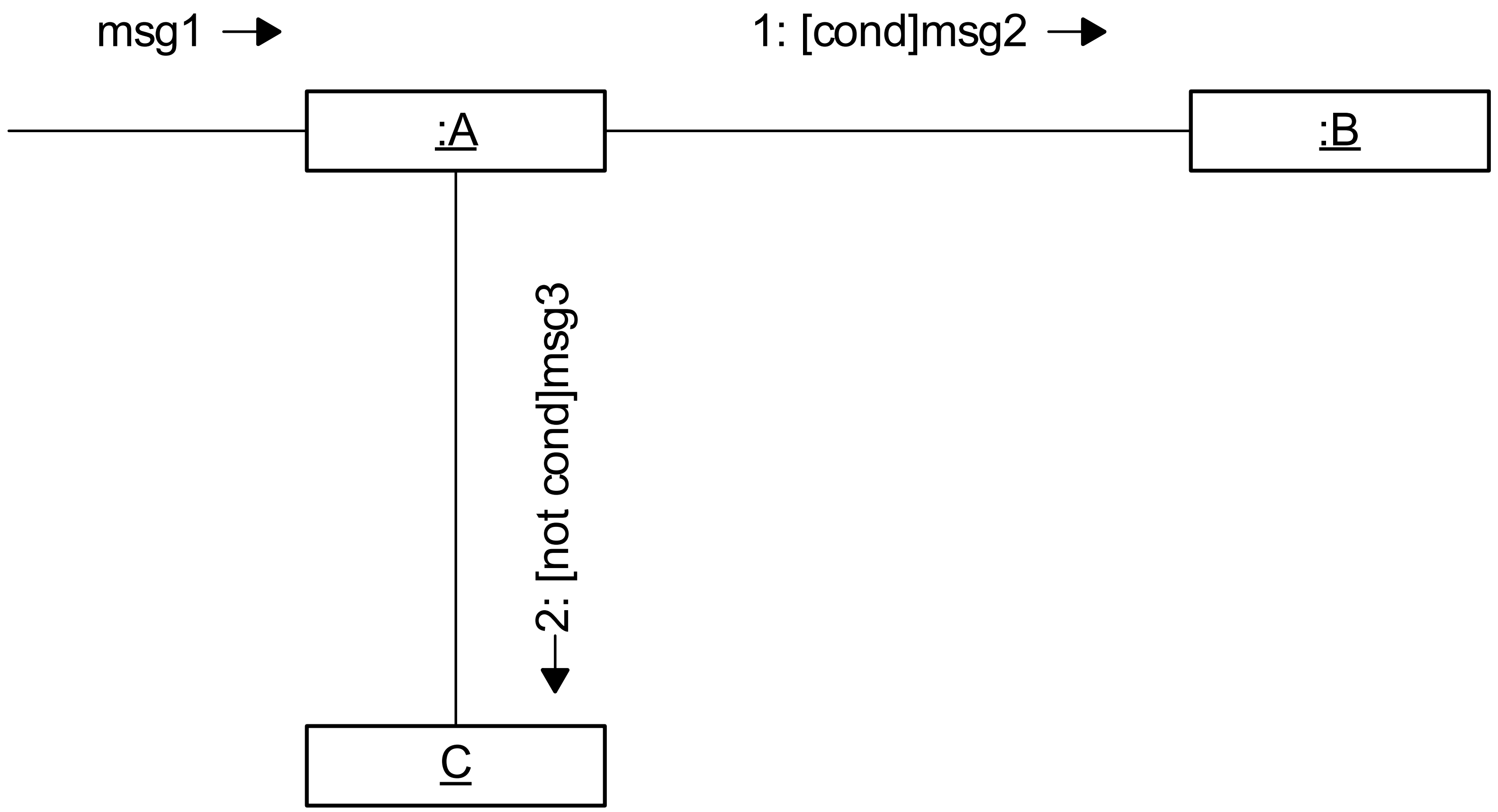


Collaboration diagram

- Conditional message

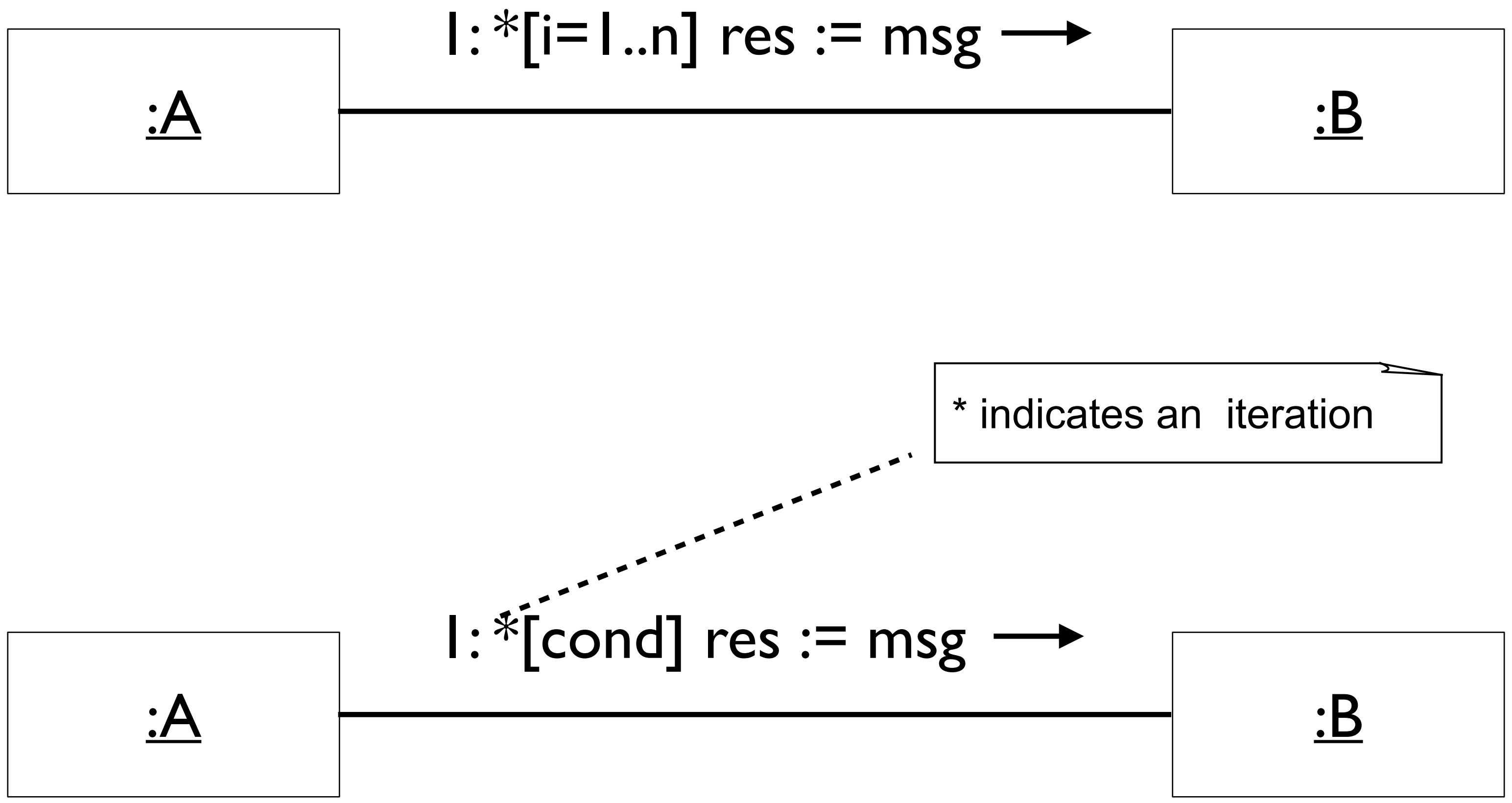


- Modelling a decision



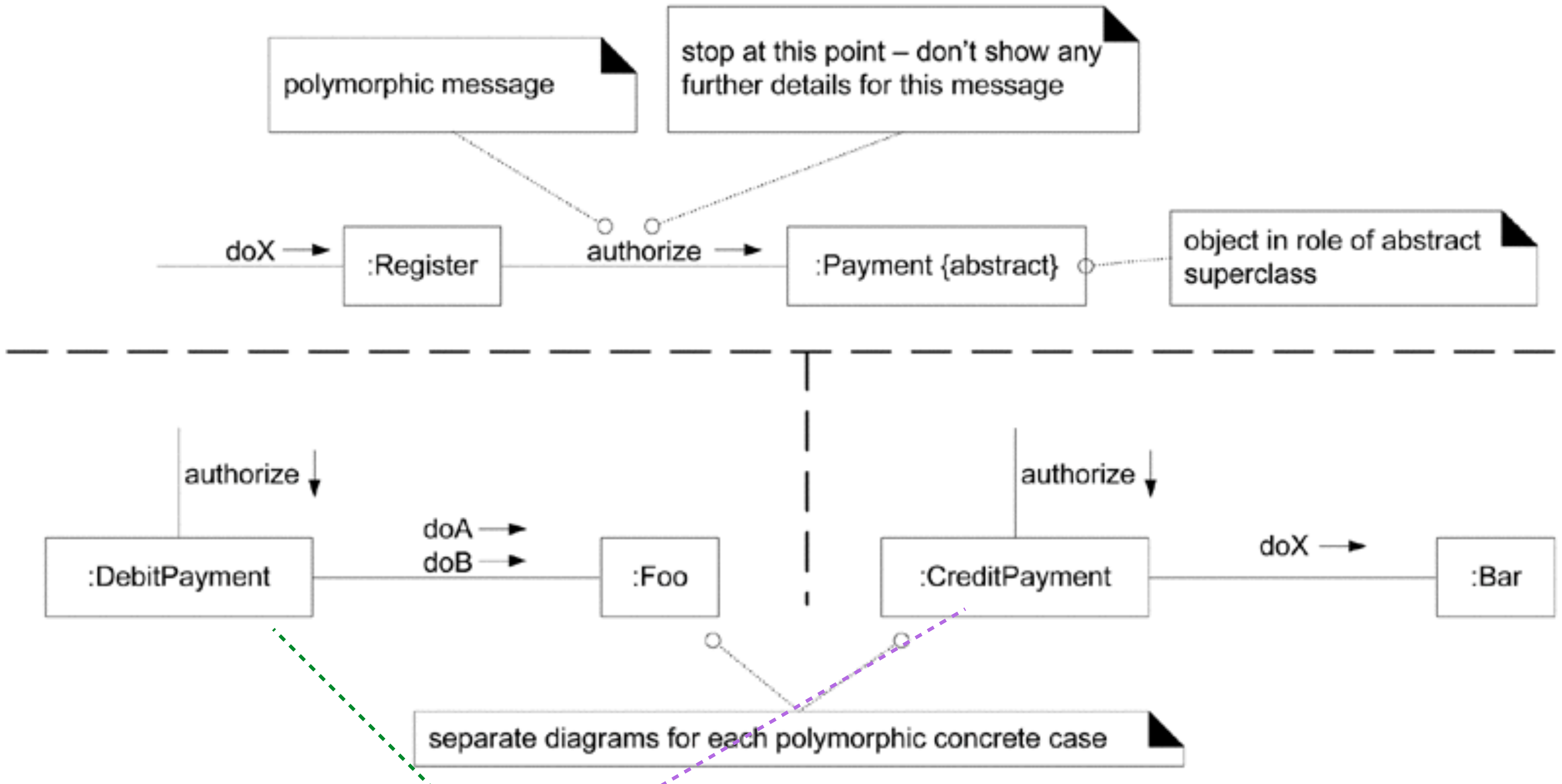
Collaboration diagram

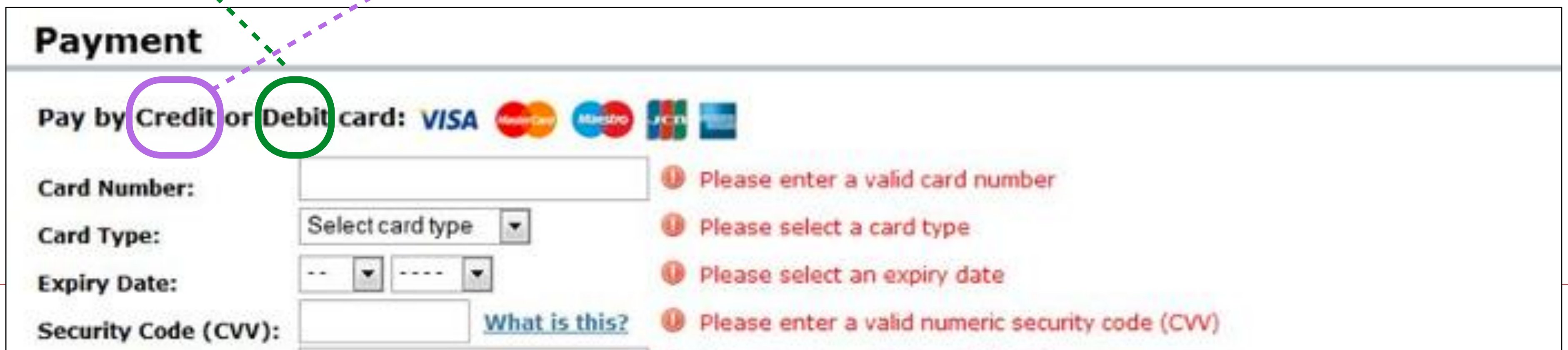
- Modelling an iteration



Collaboration diagram

- Modelling a polymorphic message


polymorphic message
stop at this point – don't show any further details for this message
doX
:Register
authorize
:Payment {abstract}
object in role of abstract superclass
authorize
:DebitPayment
doA
doB
:Foo
authorize
:CreditPayment
doX
:Bar
separate diagrams for each polymorphic concrete case


Payment

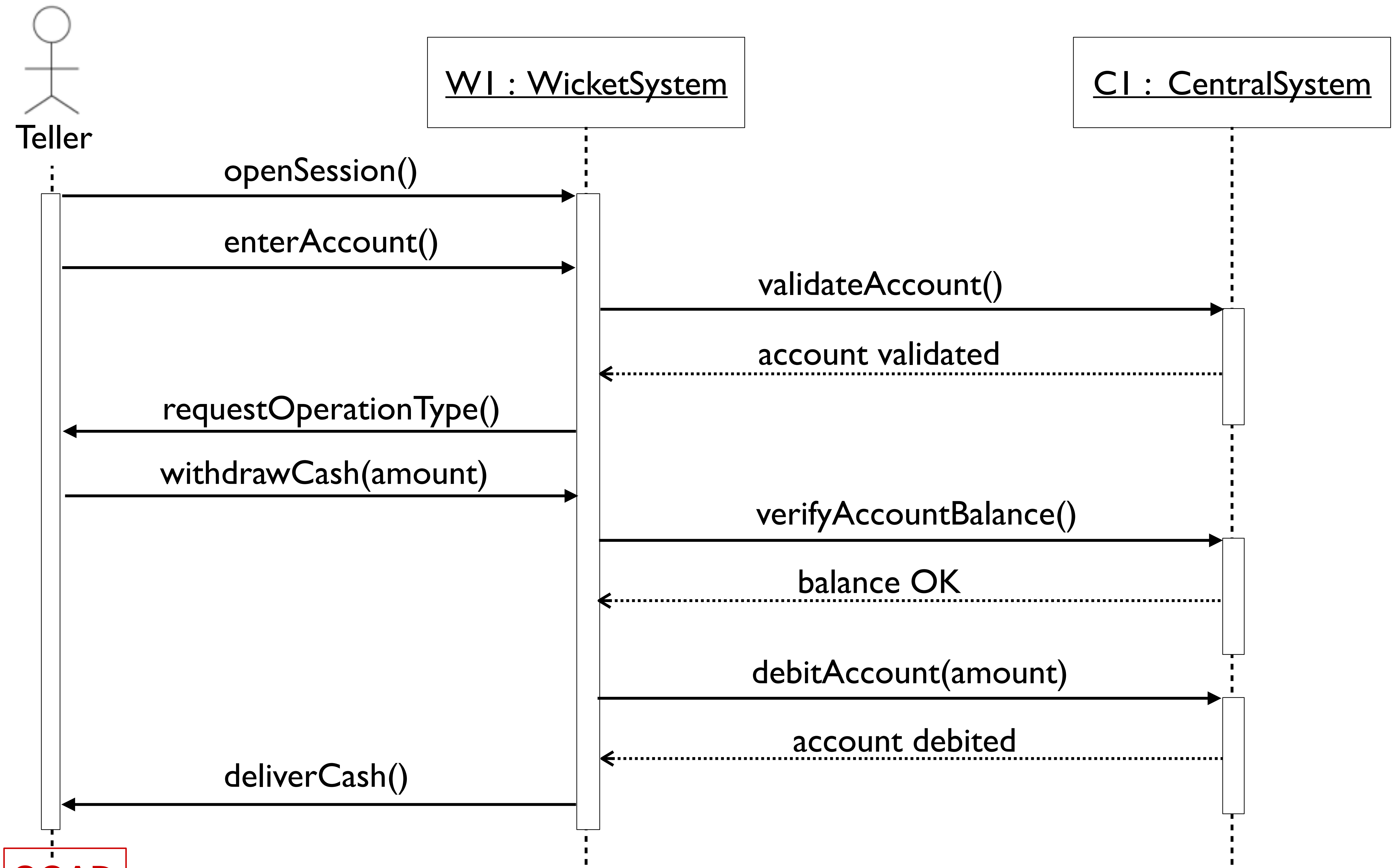
Pay by Credit or Debit card: VISA

Card Number:		Please enter a valid card number
Card Type:	Select card type	Please select a card type
Expiry Date:	-- / ----	Please select an expiry date
Security Code (CVV):	What is this?	Please enter a valid numeric security code (CVV)

Cash withdrawal at the bank

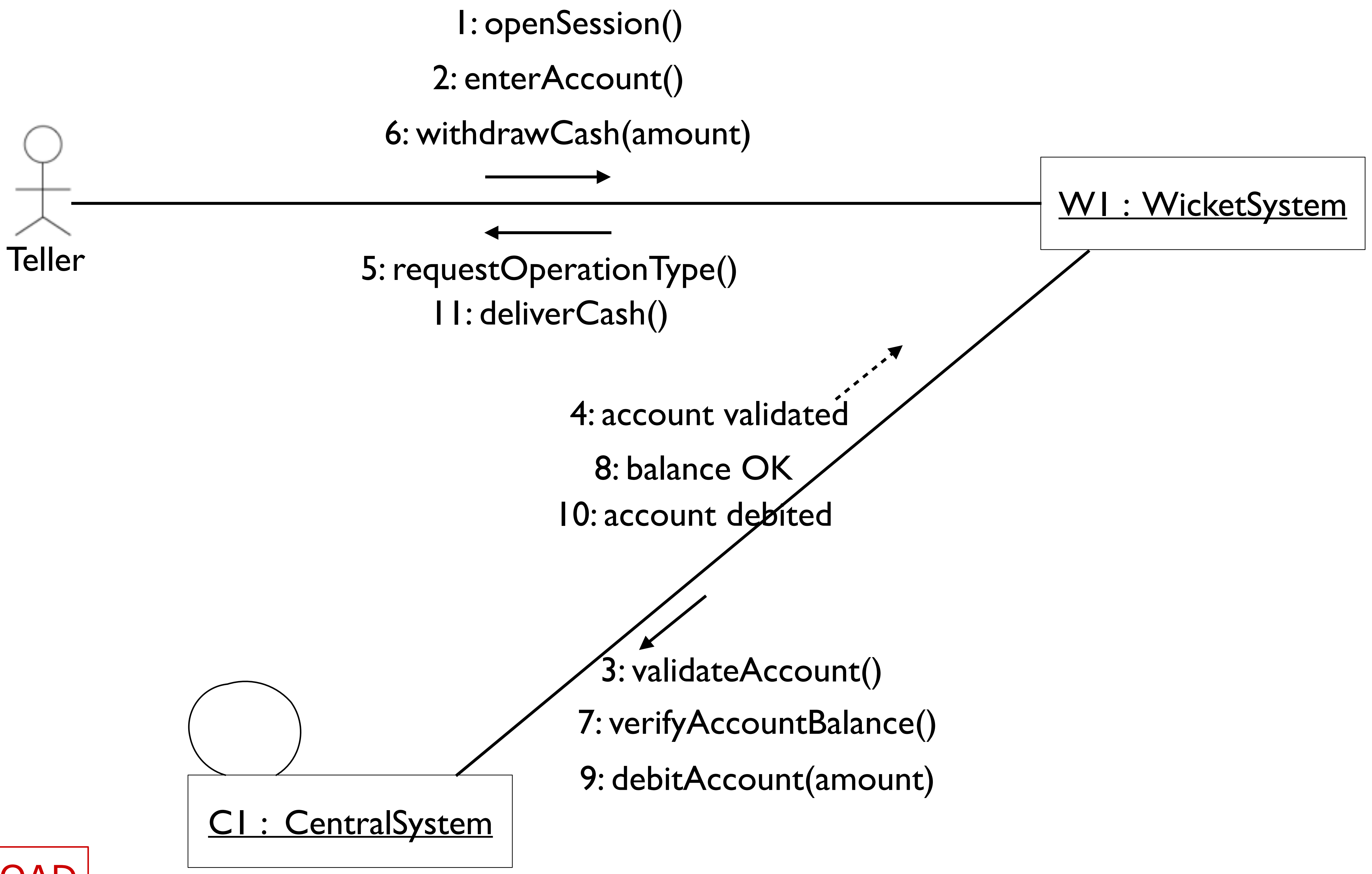
Sequence diagram

- Example: Cash withdrawal at the bank



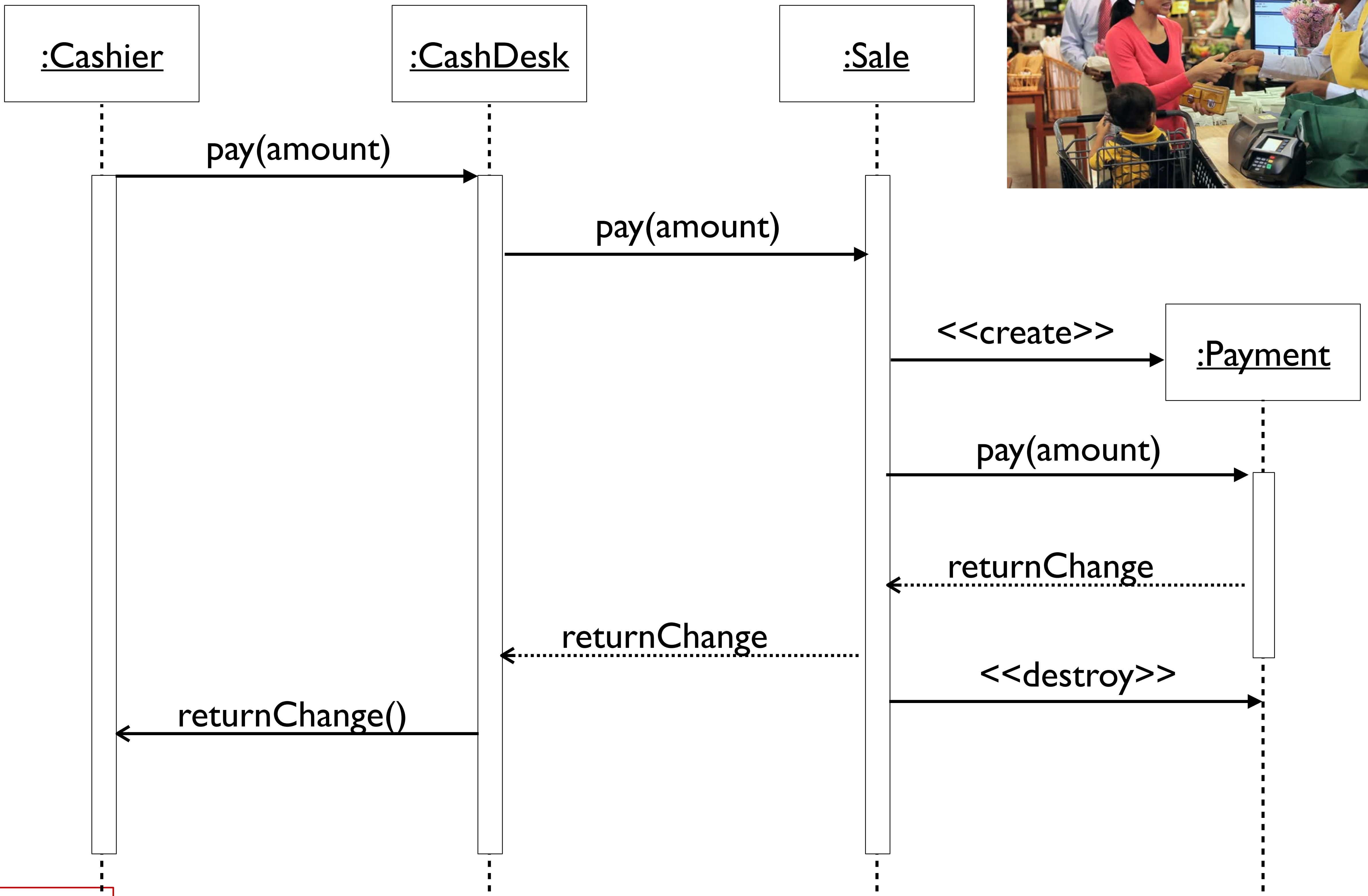
Collaboration diagram

- Example: cash withdrawal in the bank

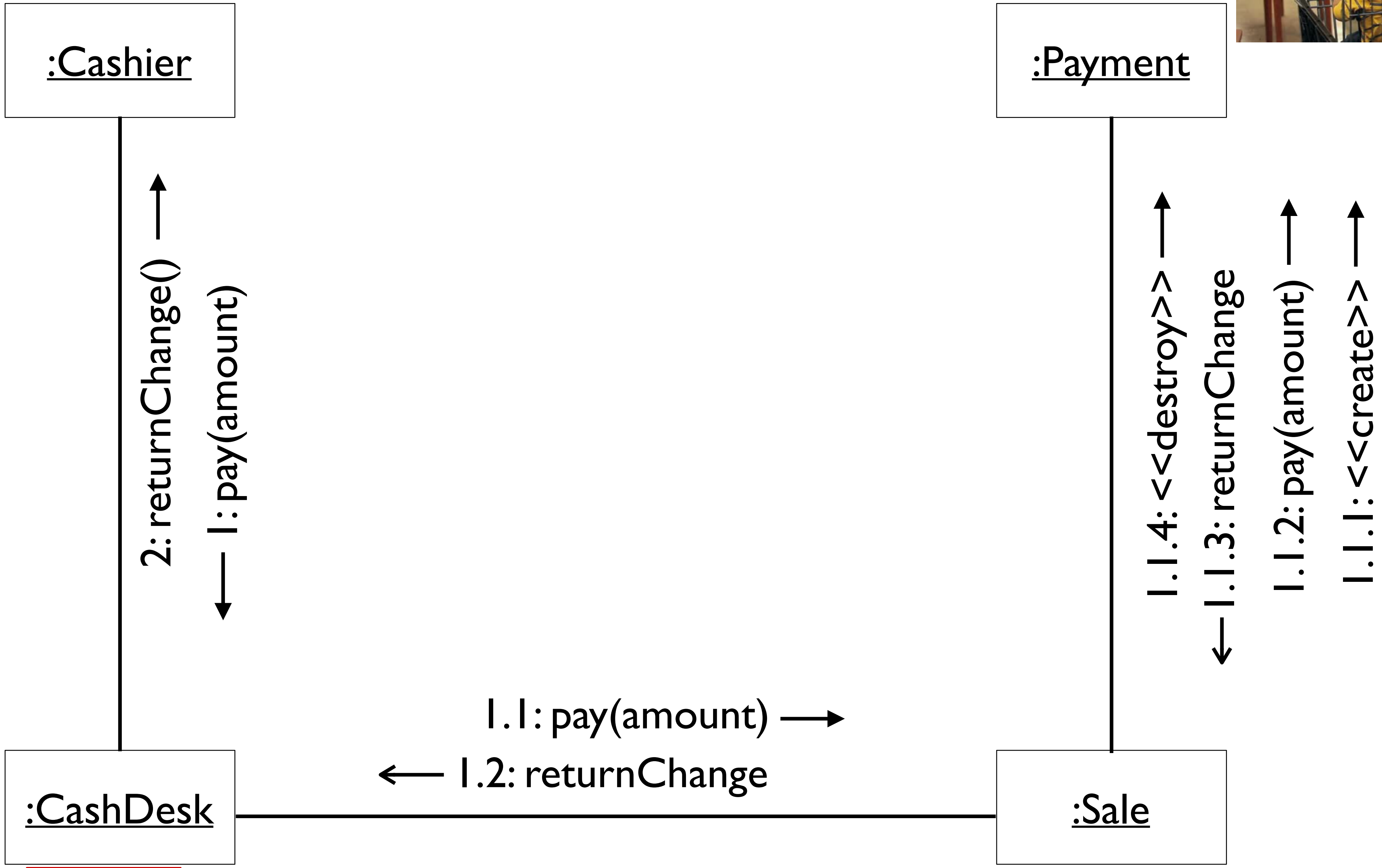


Use-case “cash payment”

Sequence diagram



Collaboration diagram



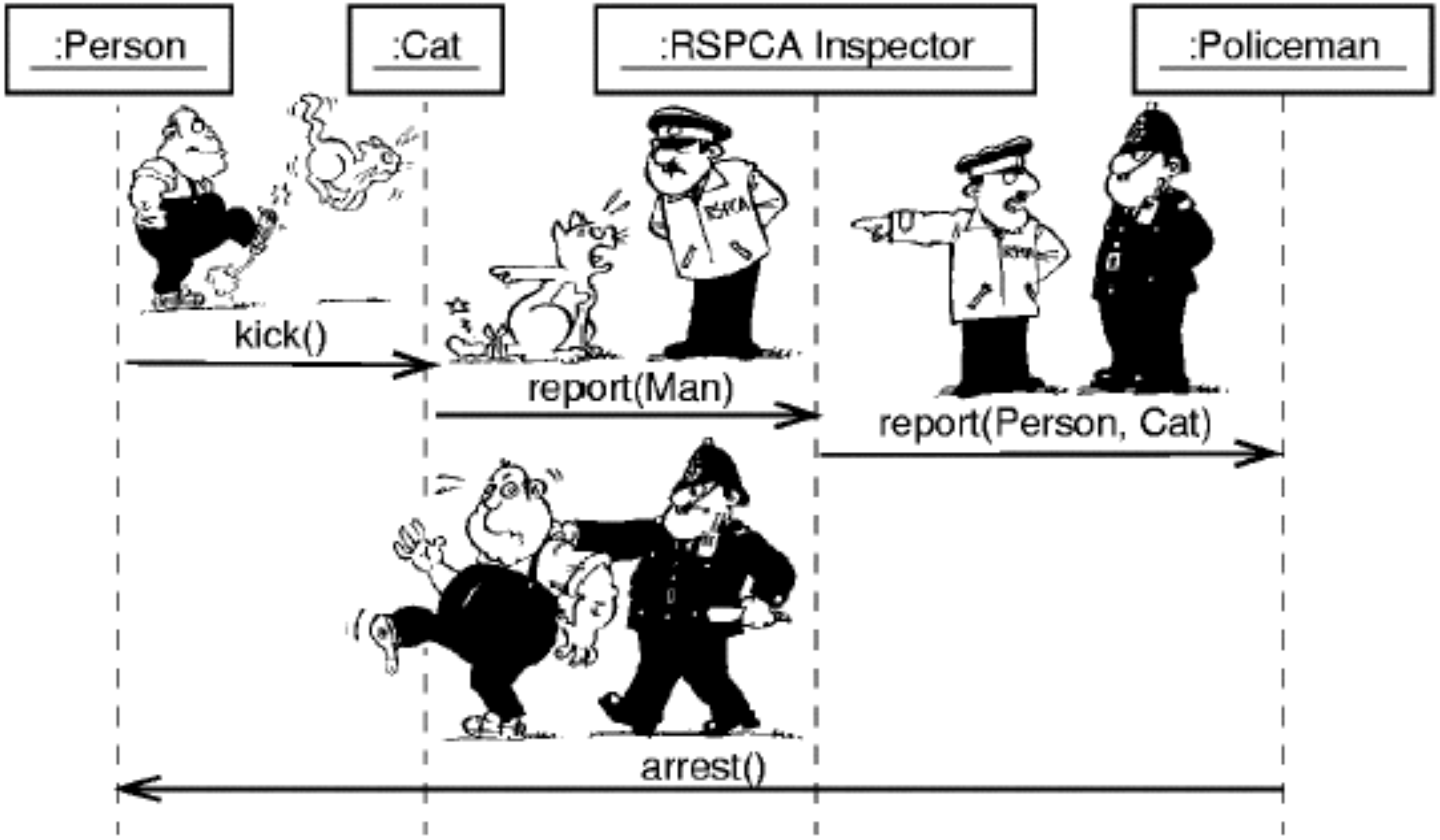
Sequence diagram v.s. Collaboration diagram

- Both sequence diagram and collaboration diagram are alternate representations of an interaction
- Sequence diagram
  - is a graphical view of a scenario
  - shows object interaction in a time-based sequence of what happens first, what happens next
  - establishes the roles of objects and help provide essential information to determine class responsibilities and interfaces
  - is normally associated with a use-case
- Collaboration diagram
  - shows how object associate with each other (objects, links and messages)
  - provides the structural relationships between objects

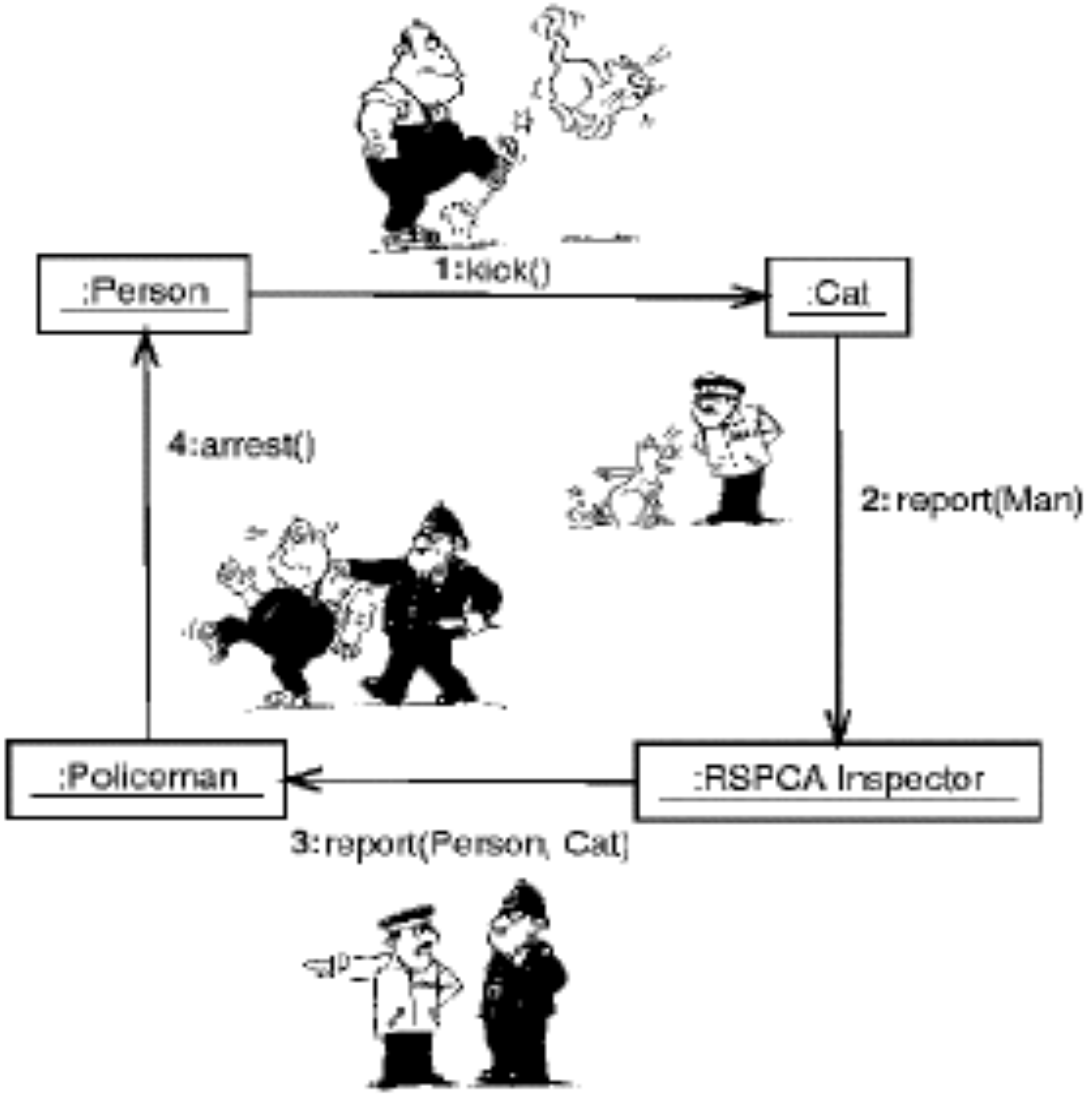
Fun example

:Cat
:Policeman
:Person
:RSPCA Inspector
RSPCA

Fun example: Sequence diagram



Fun example: Collaboration diagram


:Person
1:kick()
:Cat
2:report(Man)
:RSPCA Inspector
3:report(Person, Cat)
:Policeman
4:arrest()