Lecture 3

Use Case Descriptions

Sequence Diagram

Objectives

The student should know:

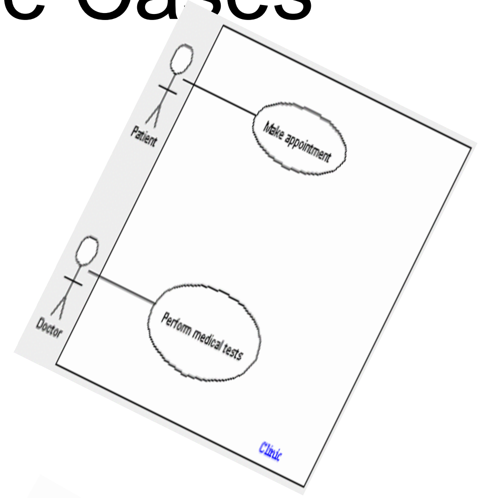
- Structure of a Use Case Description
- How to describe the Main and Alternate Path Scenarios in a Use Case Description
- Purpose of a Sequence Diagram
- Building Blocks of Sequence Diagrams
- How to produce Sequence Diagrams based on a scenario

Use Case

- There are 2 aspects to Use Cases

1. Use Case Diagram

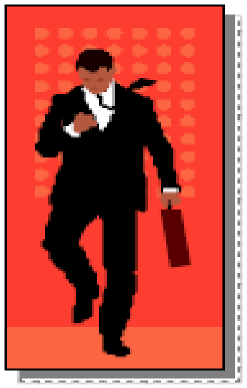
2. Use Case Description



What is a Use Case?

A use case describes how a type of user (called an "actor") uses a system to achieve a goal.

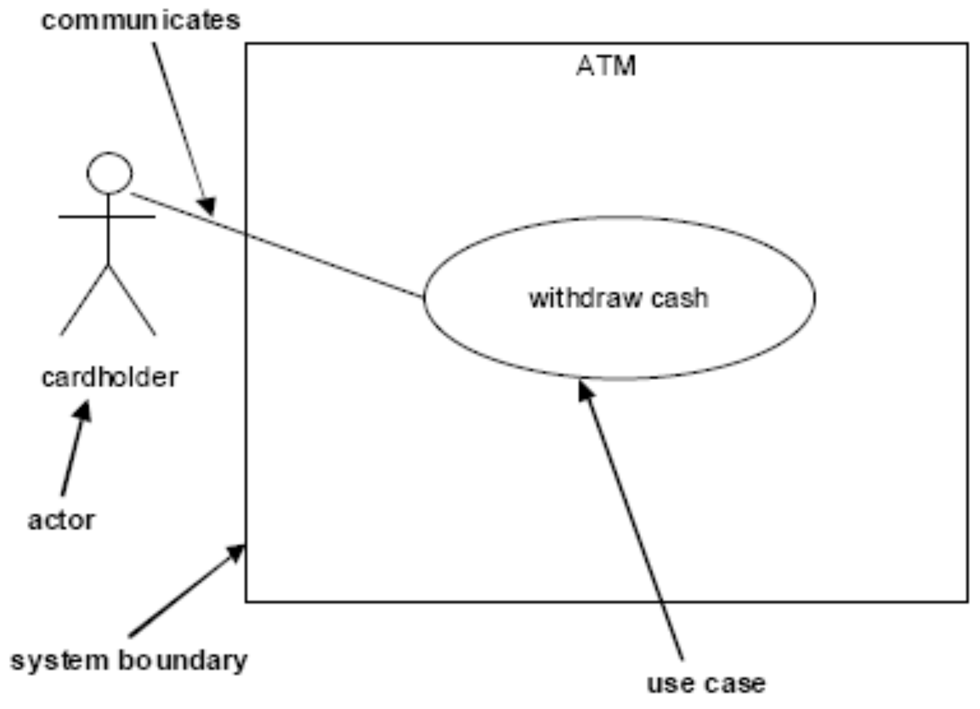
Example:
A bank cardholder might need to use an ATM to get cash out of their account.


Actor

System

Goal

Use Case Diagram



Use Case Specification

- Next step is to write a use case specification

Use Case Specification

Detailed description of the behaviour of each use case

Used as input for design and development and for writing test cases

Use Case Specification Document

Sections:

Use Case name

- Unique identifier for the use case.
- Write in **verb-noun format** (e.g. ***Borrow Books, Withdraw Cash***)
- Should **describe an achievable goal** (e.g. ***Register User***)

Goal

Briefly describes what the user intends to achieve with this use case.

Pre-conditions:

Pre-conditions that need to be satisfied for the use case to perform.

Post-conditions:

Define different states in which you expect the system to be in, after the use case executes.

Use Case Specification - Sections

Triggers
Describes the event that causes the use case to be initiated.
e.g. customer presses button

Actors: List the actors in this use case

Primary Path

- Each use case should convey a ***primary scenario***, or typical sequence of events that will occur when the use case is executed.
- Number each step in the primary path.
- Also called: Main Success Scenario

Typical Sequence of Steps

1. **System prompts the user to log on.**
2. **User enters his name and password.**
3. **System verifies the logon information.**
4. **System logs user on to system.**

Use Case Specification - Sections

Alternative Paths

- Use cases may contain **alternative scenarios**, which are variations on the main theme.
- Indicate, using a numbering system, at which point they differ from the primary scenario, and, if appropriate, where they rejoin.
- Document If this use case **<<extends>>** from other use cases
- Document if this use case **<<includes>>** the functionality of other use cases.

Typical Sequence of Steps

1.System prompts the user to log on.
2. User enters his name and password.
3. System verifies the logon information.
4. System logs user on to system.

Alternate Path

3 a. User enters incorrect login information and system cannot verify the user

Use Case Description – Withdraw Cash

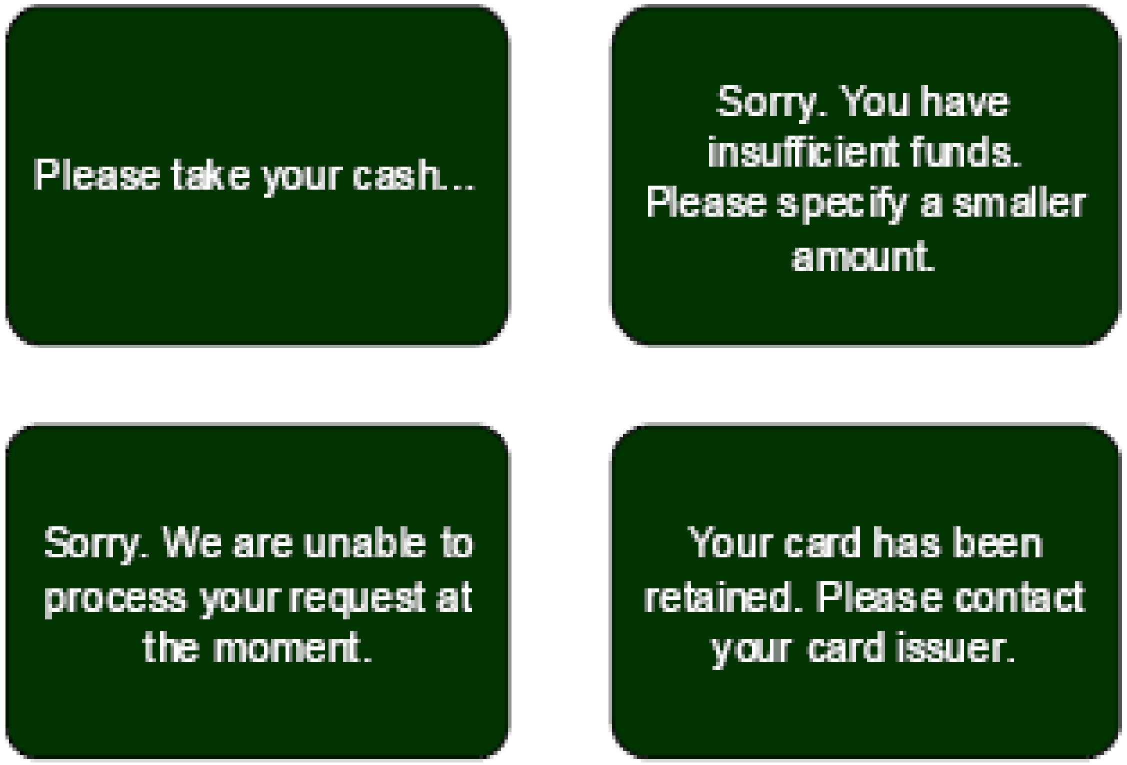
Cardholder	ATM
1 •select "withdrawal" option	
	2 •display withdrawal options
3. •specify amount	
	4. •check cardholder has sufficient funds
	5. •eject card
	6. •prompt cardholder to take card
7. •take card	
	8. •dispense amount
	9. •prompt cardholder to take cash
10. •take cash	
	11. •debit cardholder's account
	12. •thank cardholder
	13. •display welcome and await next cardholder

What is a Use Case Scenario?

- When you want to withdraw cash from an ATM, you don't always get it.
- Sometimes you get it.
- Other times you don't.
- What can happen?

- ------------------------
- ------------------------
- -----------------------

Use Case Alternate Scenarios



Please take your cash...

Sorry. You have insufficient funds. Please specify a smaller amount.

Sorry. We are unable to process your request at the moment.

Your card has been retained. Please contact your card issuer.

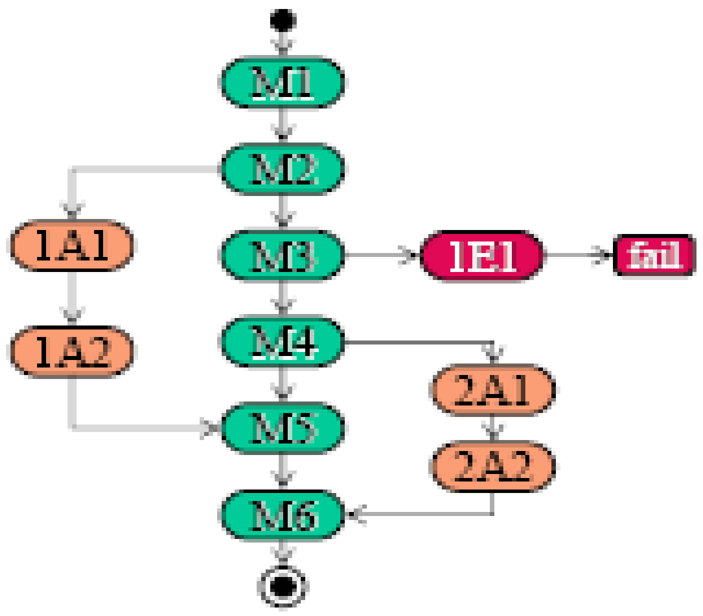
Use case scenarios for withdrawing cash from an ATM

Use Description may have:

Some have:

- *main*
- *alternate*
- *exceptional*

flow



Write out the Main Success Scenario and then you brainstorm each step of ways
(a) in which things can go wrong or,
(b) less commonly, alternative ways in which things can go right.

Use Case Scenario

Document the main scenario, the alternative scenario and variations in the following use case description:

A customer browses the catalog and adds the desired items to their shopping basket.
When the customer wishes to pay, he/she enters the shipping and credit card information and confirms the sale.
The system checks the authorisation on the credit card and confirms the sale both immediately and with a follow-up email.

Use Case Scenario – Buy Product

Step	Action
	PRIMARY PATH
1.	**Customer browses catalog & selects items to buy**
2.	**Customer goes to the check out**
3.	**Customer fills in shipping information (address, next-day or 3-day delivery)**
4.	**System presents full pricing information, including shipping**
5.	**Customer fills in credit card information**
6.	**System authorises purchases**
7.	**System confirms sale immediately**
8.	**System sends confirming e-mail to customer**
	ALTERNATIVE PATHS
3a.	**Customer is regular customer** (Use Case)
	1. **System displays current shipping, pricing & billing information**
	2. Customer may accept or override these defaults, returns to Step 6
6a.	**System fails to authorise credit purchases (Use Case)**
	1. Customer must reenter credit card information or may cancel

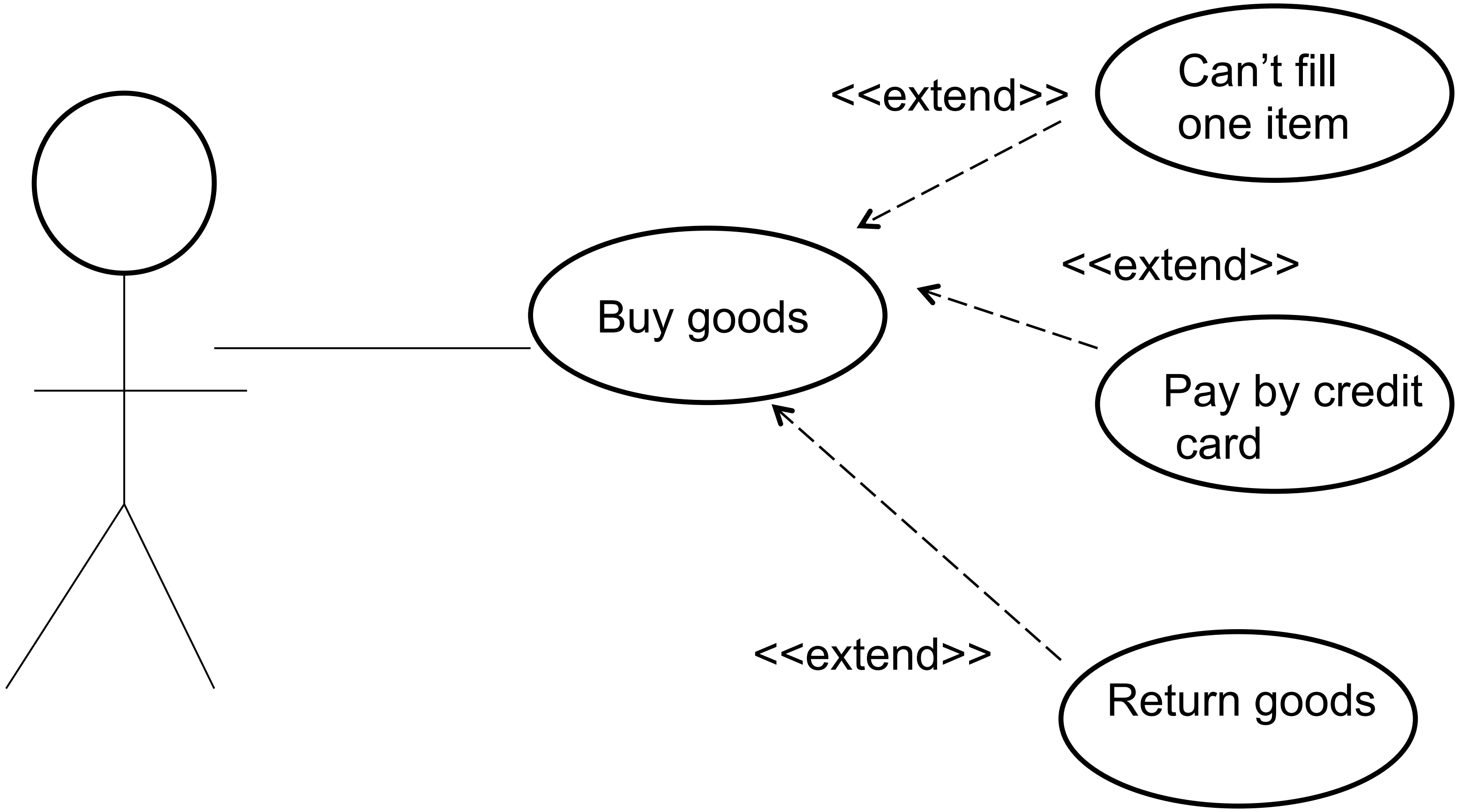
Draw the Use Case for this scenario

More Examples of Use Case Descriptions

Buy goods

- Goal: Buyer issues request directly to our company, expects goods to be shipped and billed.
- Precondition: We know buyer, their address etc.
- Success End Condition: Buyer has goods; we have money for goods
- Failed End Condition: We have not sent the goods. Buyer has not spent the money.

Buy Goods



Buy Goods Use Case Specification

Main Success Scenario:

1. Buyer rings in with purchase request
2. Company captures buyer’s name, address, registered goods etc.
3. Company gives buyer information on goods, prices, delivery dates etc.
4. Buyer signs for order.
5. Co. creates order, ships order to buyer.
6. Co. ships invoice to buyer.
7. Buyer pays invoice.

Extensions:

3a. Co. is out of one of the ordered items:

3a1. Renegotiate order

4a. Buyer pays directly by credit card (use case)

7a. Buyer returns good

7a1. Handle returned goods (use case)

Use Case Specification

- Each step takes the form "A does X", where A is an actor and X is an action.
- If a use case is included, you will find a step that has the form "include U", where U is another use case.
- Extension 3a is executed instead of 3, if condition "*company is out of one of the ordered items*" is fulfilled.

Hotel Reservation Use Case Spec.

Goal: Reserve room(s) at a hotel

1. Reservation_Maker (RM) asks to make a reservation.
2. RM selects hotel, class and room type.
3. System provides price to RM.
4. RM asks for reservation.
5. RM provides name, email.
6. System makes reservation & allocates tag to reservation.
7. System reveals tag to RM.
8. System creates & sends confirmation email.

Hotel Reservation – Extensions

- Extension:
- 3. Room not available

    a. System offers alternatives

    b. RM selects from alternatives

  3b. RM rejects alternatives

    - Fail

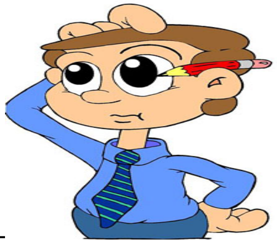
  4. RM rejects offer

    - Fail

  5. Customer already on file

  – Resume 6

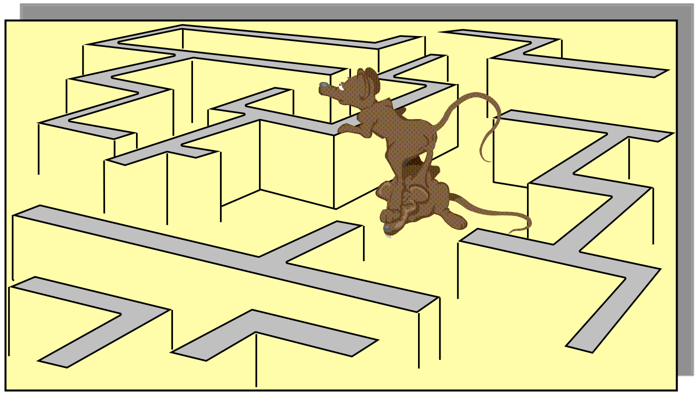
Exercise



- What are the main path & alternate path scenarios in the previous use case specification?

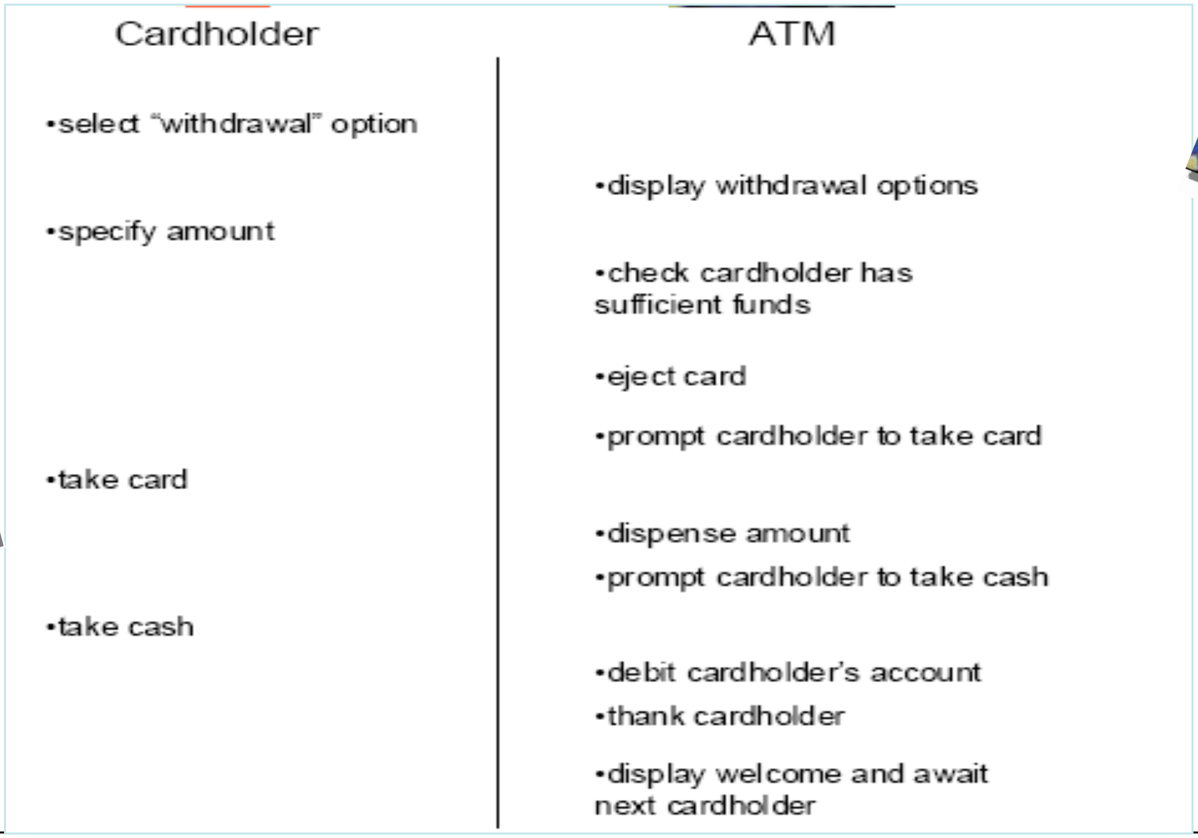
- Draw the Use Case diagram.

Sequence Diagrams



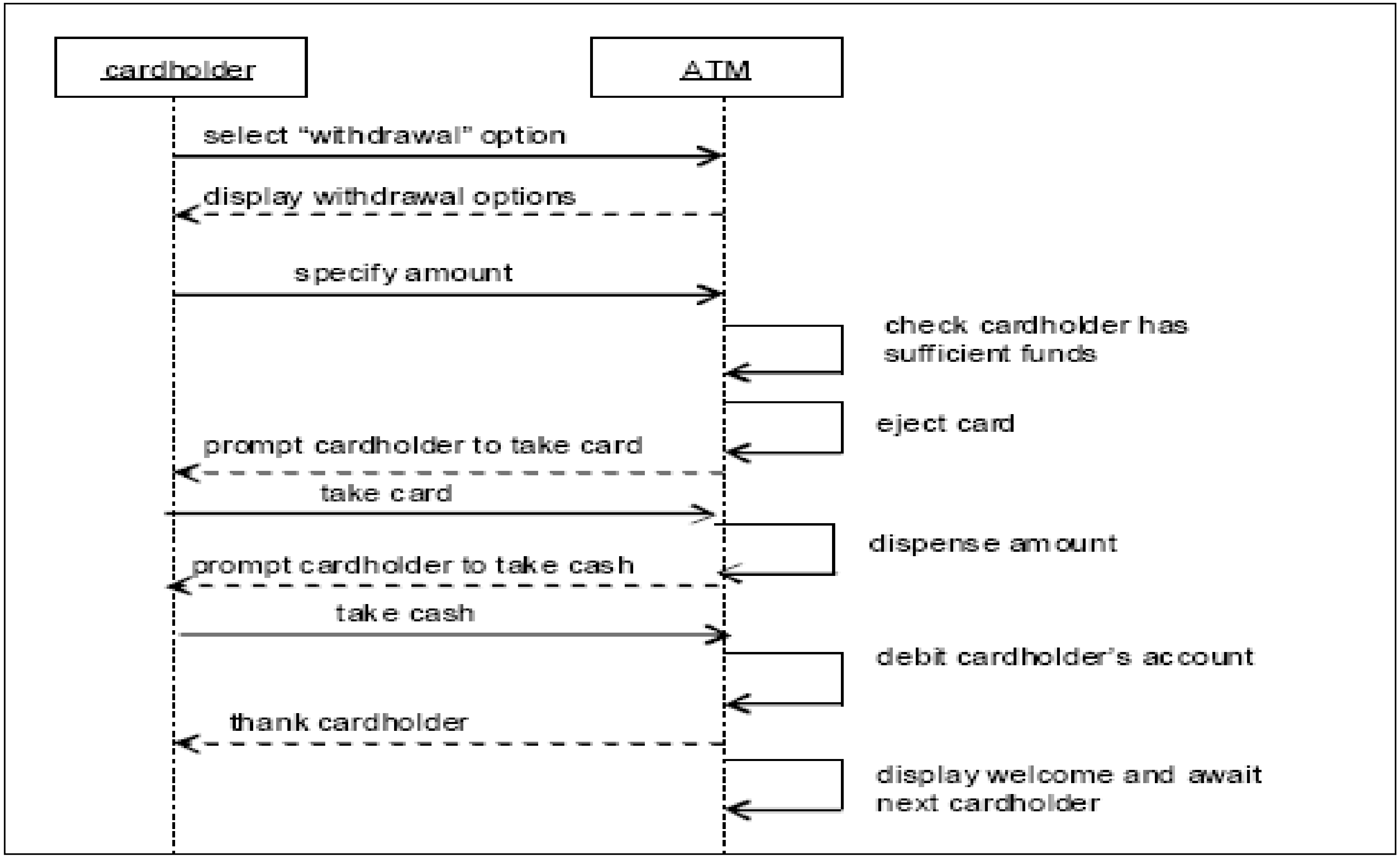
Sequence Diagrams

- You can easily model a Use Case scenario using a Sequence Diagram



Cardholder	ATM
•select "withdrawal" option	
	•display withdrawal options
•specify amount	
	•check cardholder has sufficient funds
	•eject card
	•prompt cardholder to take card
•take card	
	•dispense amount
	•prompt cardholder to take cash
•take cash	
	•debit cardholder's account
	•thank cardholder
	•display welcome and await next cardholder

Sequence Diagram


cardholder
ATM
select "withdrawal" option
display withdrawal options
specify amount
check cardholder has sufficient funds
eject card
prompt cardholder to take card
take card
dispense amount
prompt cardholder to take cash
take cash
debit cardholder's account
thank cardholder
display welcome and await next cardholder

Withdraw cash from ATM

Sequence Diagrams

- Purpose

  To display the exchange of messages between objects within a limited period of time

A sequence diagrams uses objects in its diagram, so you first need to get some objects.

How do you find the objects?

Finding Objects in Sequence Diagrams

- Basic idea is …..you scan your use case and scenario descriptions for nouns.

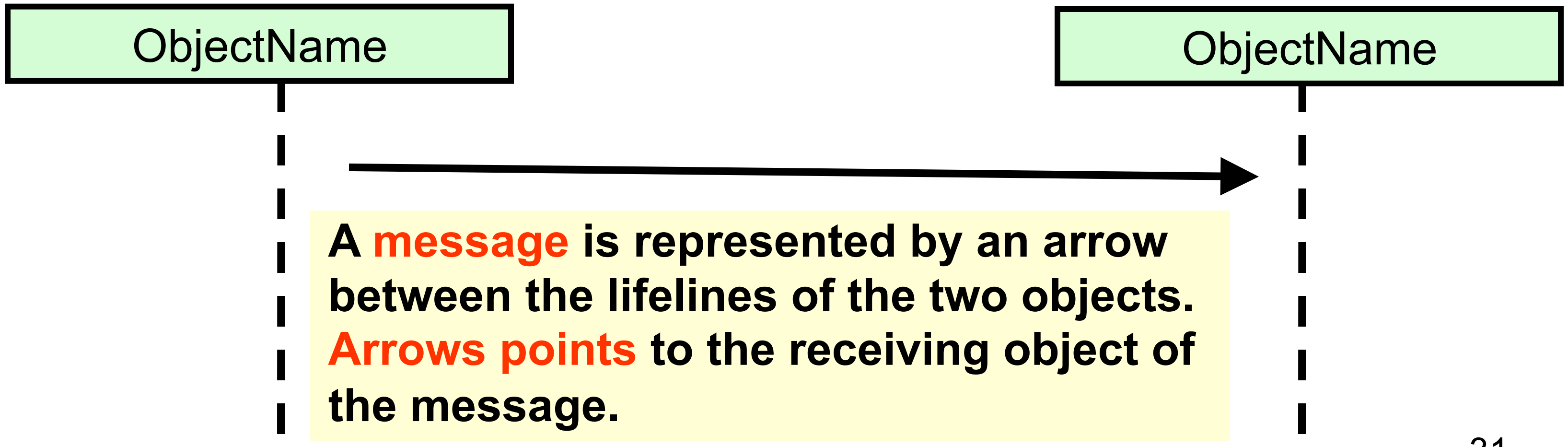
- Find the nouns that could become the classes and subclasses in your software.

Organisation of Sequence Diagram

- Sequence diagrams are organized according to time.
- Time progresses as you go down the page.
- Objects involved in the operation are listed from left to right, according to when they take part in the message sequence

Objects and Lifelines

- Participating objects are shown as boxes on top of vertical lines, the lifeline of an object.
- Each vertical dotted line is a **lifeline**, representing the length of time that an object exists.



A message is represented by an arrow between the lifelines of the two objects. Arrows points to the receiving object of the message.

Object Names

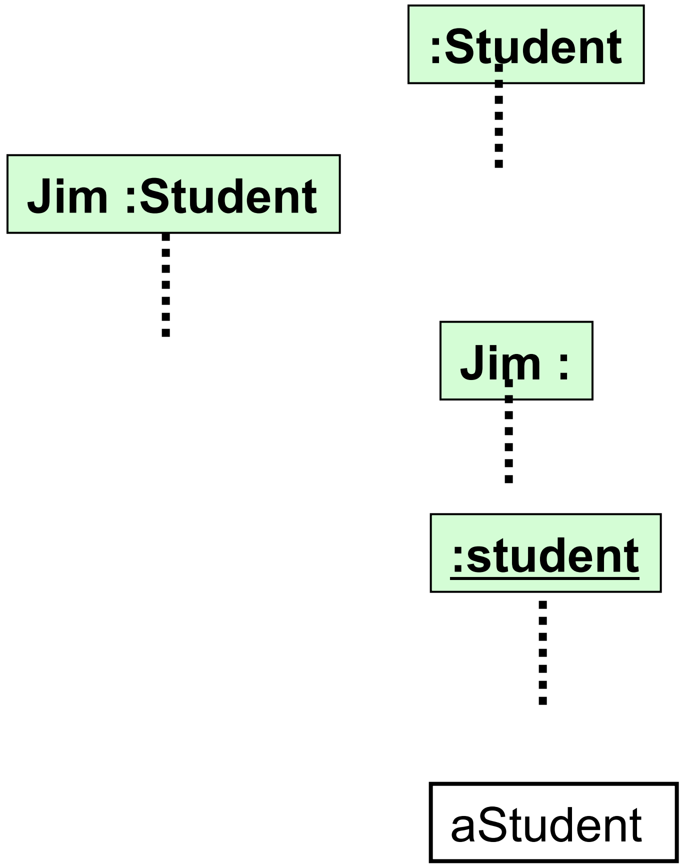
- Objects can be named in **<u>5</u>** different ways:

:className

objectName:className

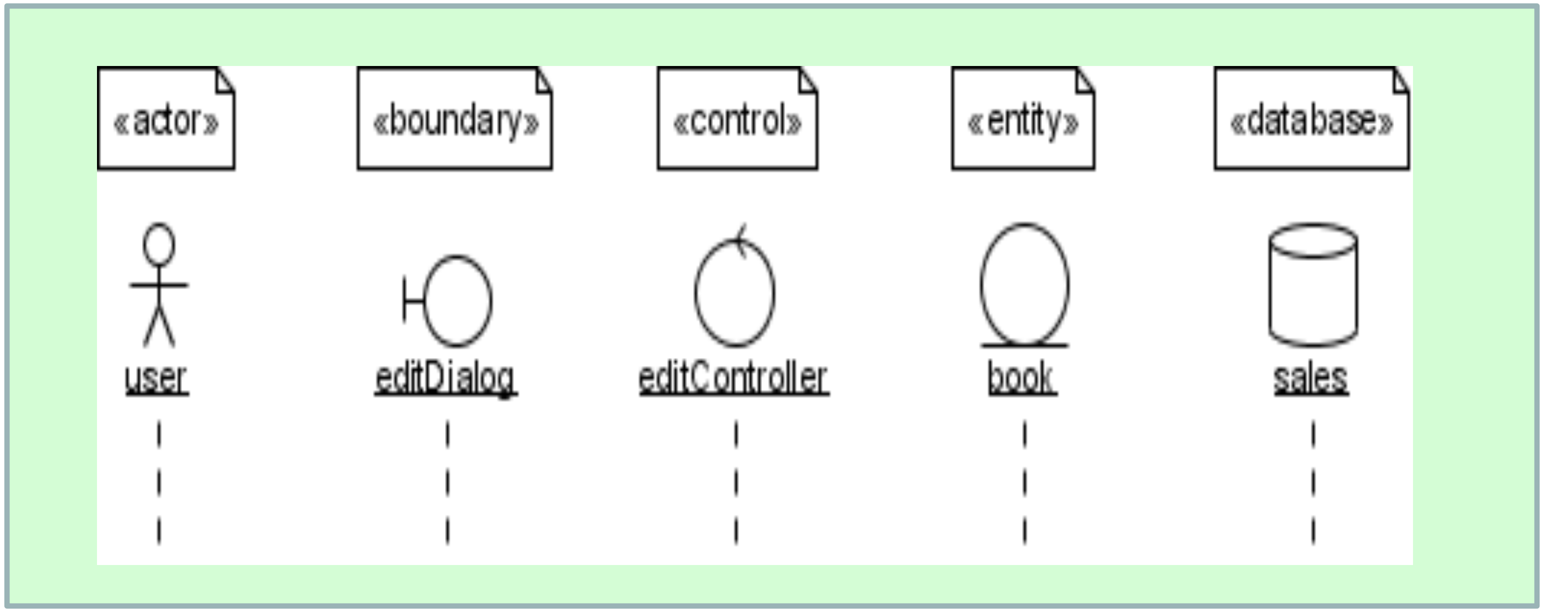
objectName:

<u>:className</u>



Object Names

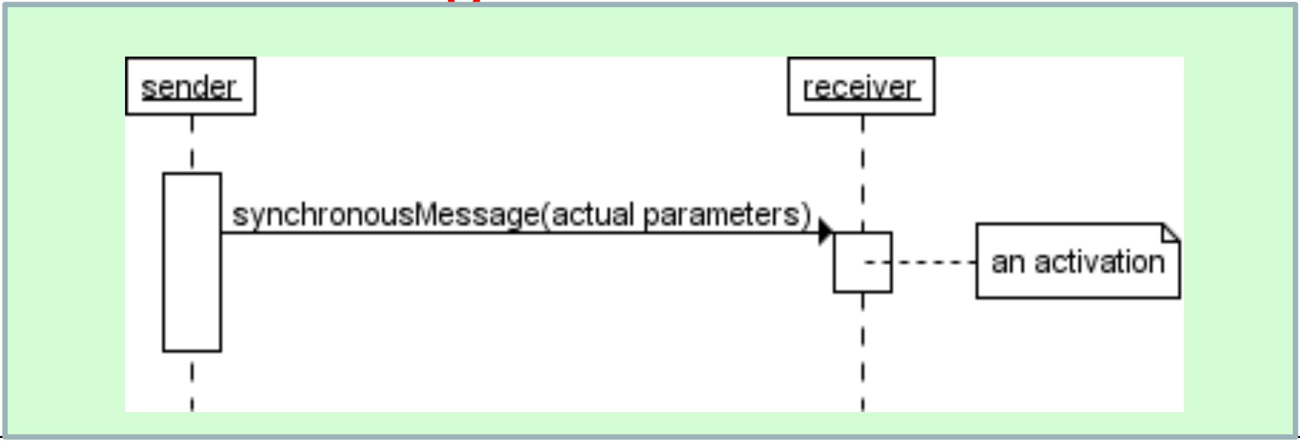
Objects can be shown as icons also with
<< >> around the name



Synchronous message

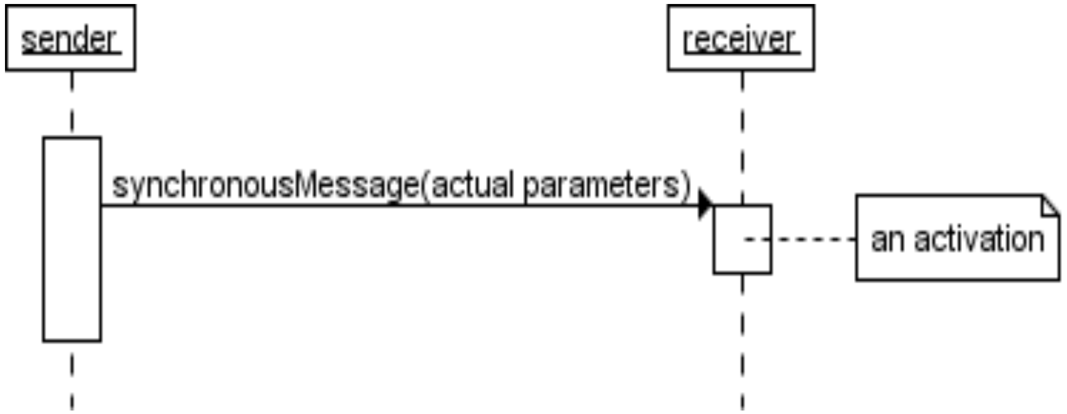
•Used when the sender waits until the receiver has finished processing the message, only then does the caller continue

•Use a closed and filled arrowhead to show a synchronous message.



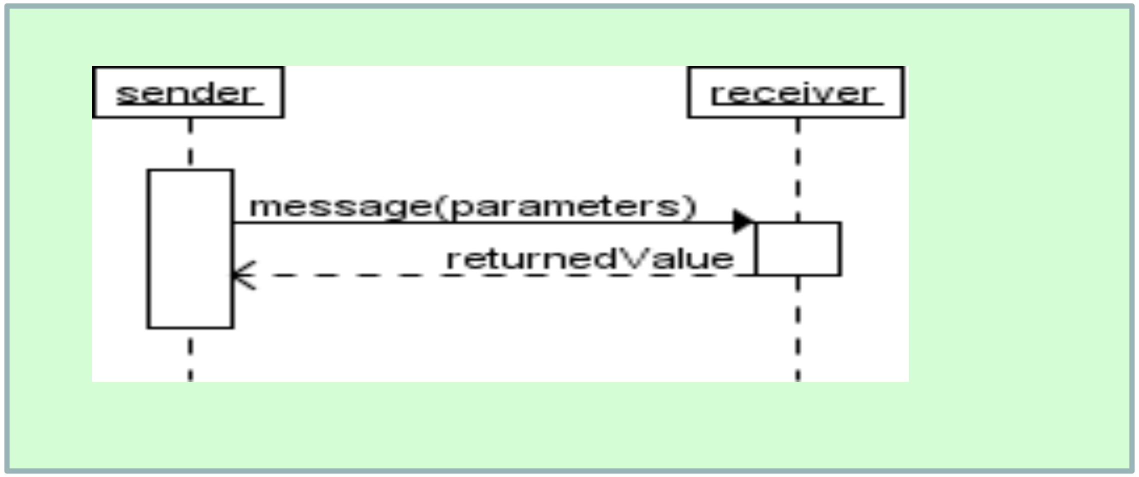
Activation Bar

- White rectangles on a lifeline are called activation bars
- Indicate that an object is responding to a message.
- Starts when the message is received and ends when the object is done handling the message.



Return Message

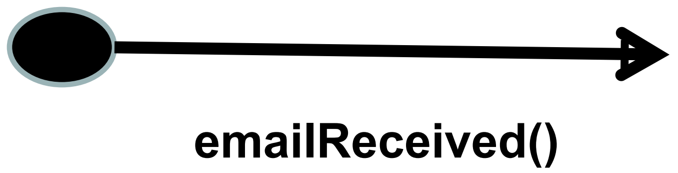
•Use a dashed arrow from receiver to sender to return control to the sender.
•Optionally, a value that the receiver returns to the sender can be placed near the return arrow.



•Return messages may be left out in diagrams.

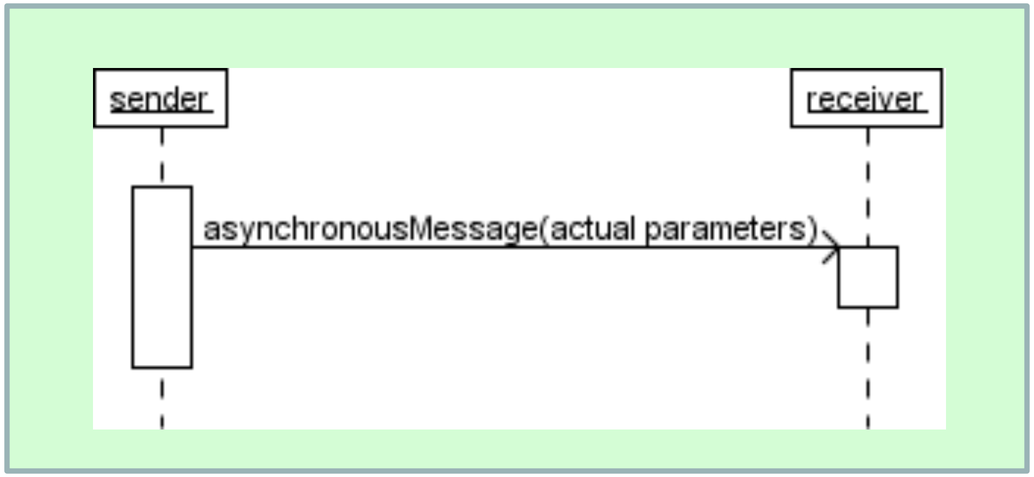
Found Message

- A message sent from outside the diagram is shown as arrowed line with a message with a filled-in circle at the end.

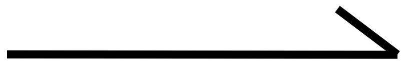


Asynchronous Messages

- The sender does not wait for the receiver to finish processing the message, it continues immediately.
- <u>Example:</u> Messages sent to a receiver in another process, calls that start a new thread, real time process
- An <u>open arrowhead</u> is used to indicate that a message is sent asynchronously.

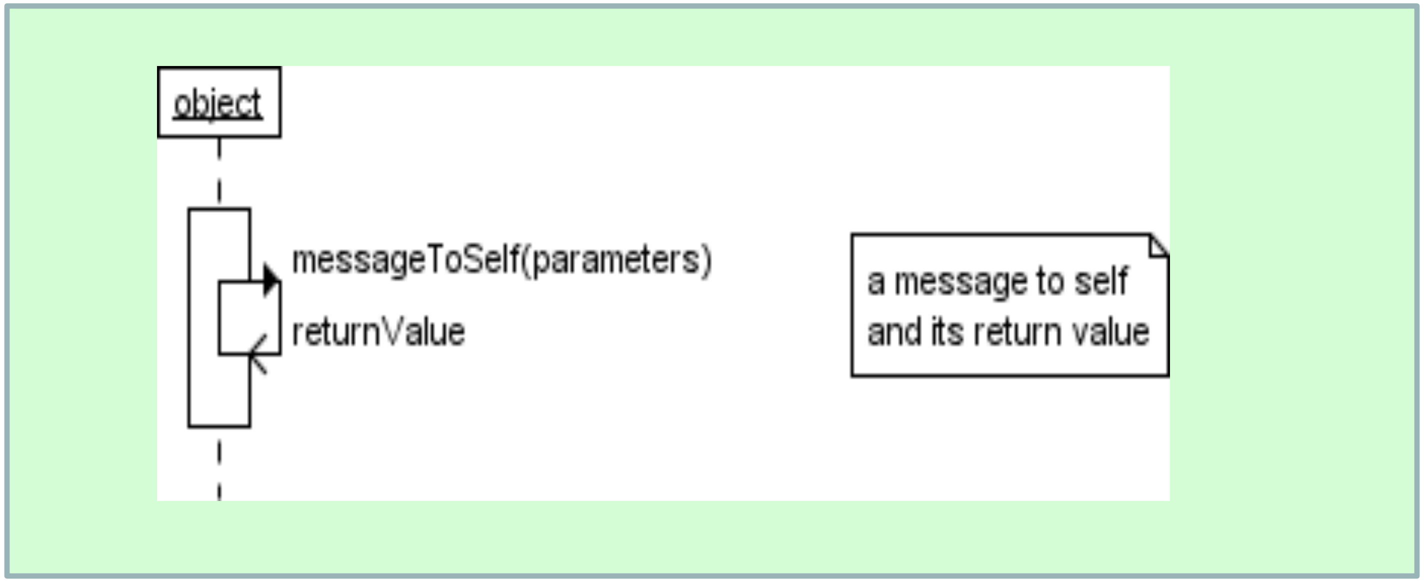


Also show as a half arrow head in other versions of UML



Message to Self

- A message that an object sends itself.



Messages

- A call message is labelled by the name of the operation being called.

- You may also include:
  - **Parameters** to be passed to the object
    e.g. bookOrder(String bookId)

  - **Condition** under which the operation is called
    e.g. [hasCopy] bookOrder()
    - bookOrder will only be called if [hasCopy] is true

Messages cont.

- An **iteration marker**, *, if the message is to be passed to a number of objects

  e.g. *[for all order lines] reduceStockQty()

  - *All order lines linked to the order will receive the message reduceStockQty()*

- The **return type** of the message

  e.g. makeReservation void()

Creation and Destruction of Objects

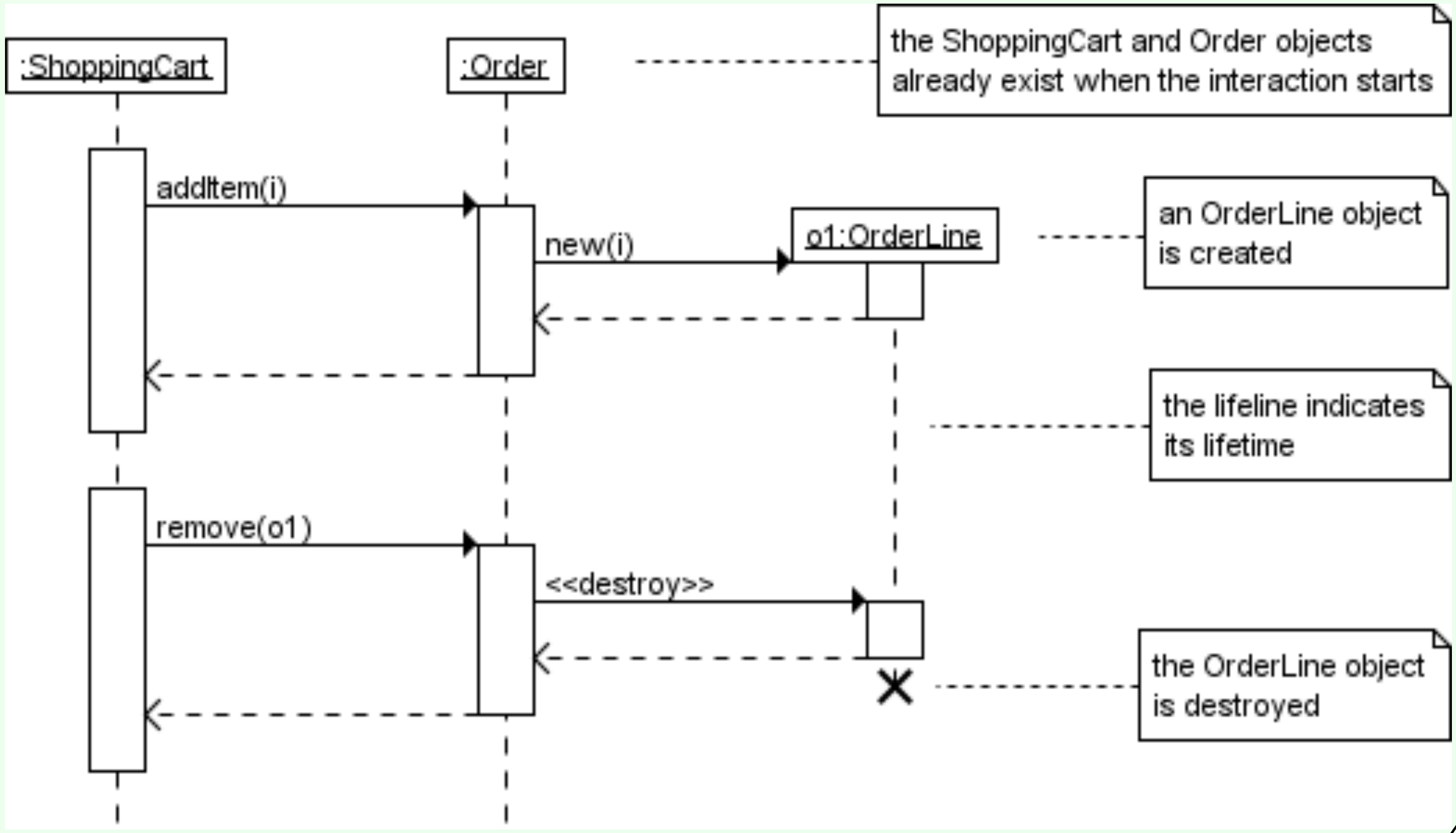
- Objects that exist at the start of an interaction are placed at the top of the diagram.

- Any objects that are created during the interaction are placed further down the diagram, at their time of creation.

An object's lifeline extends as long as the target exists.

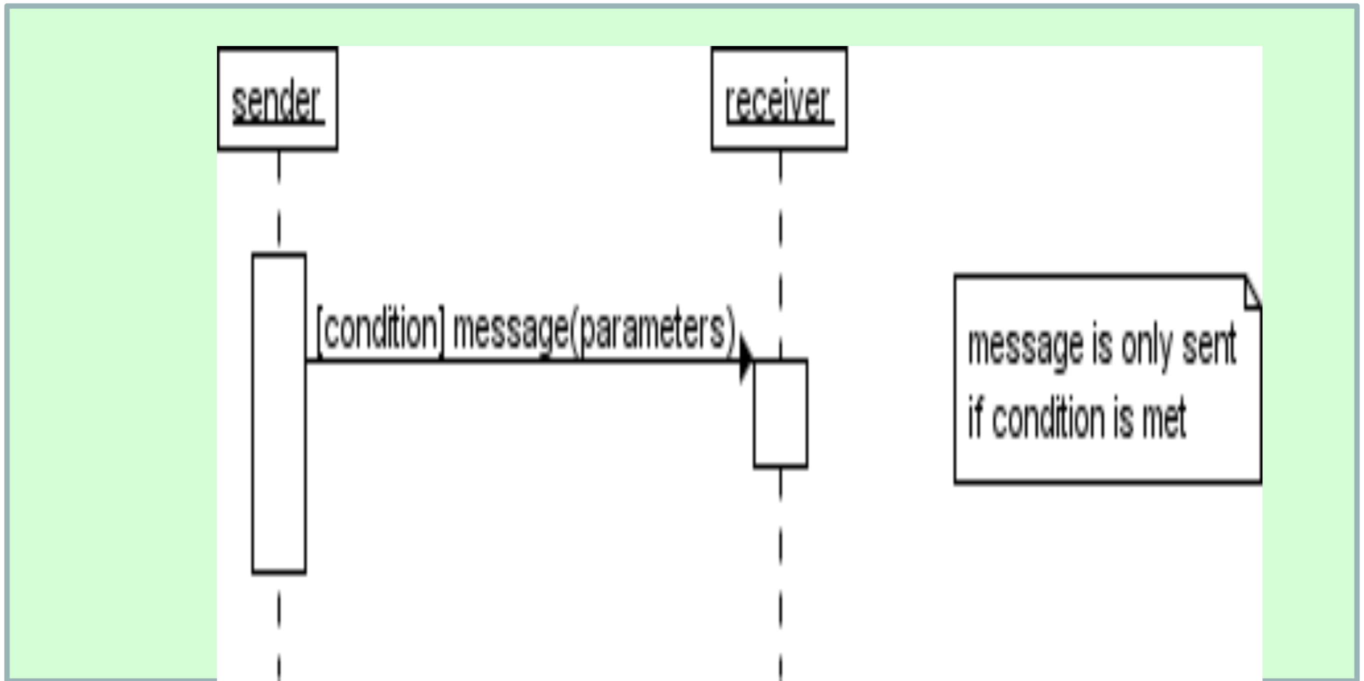
- If an object is destroyed during the interaction, the lifeline ends at that point in time with a big X sign.

Creation and Destruction of Objects


:ShoppingCart
:Order
the ShoppingCart and Order objects already exist when the interaction starts
addItem(i)
new(i)
o1:OrderLine
an OrderLine object is created
the lifeline indicates its lifetime
remove(o1)
<<destroy>>
the OrderLine object is destroyed

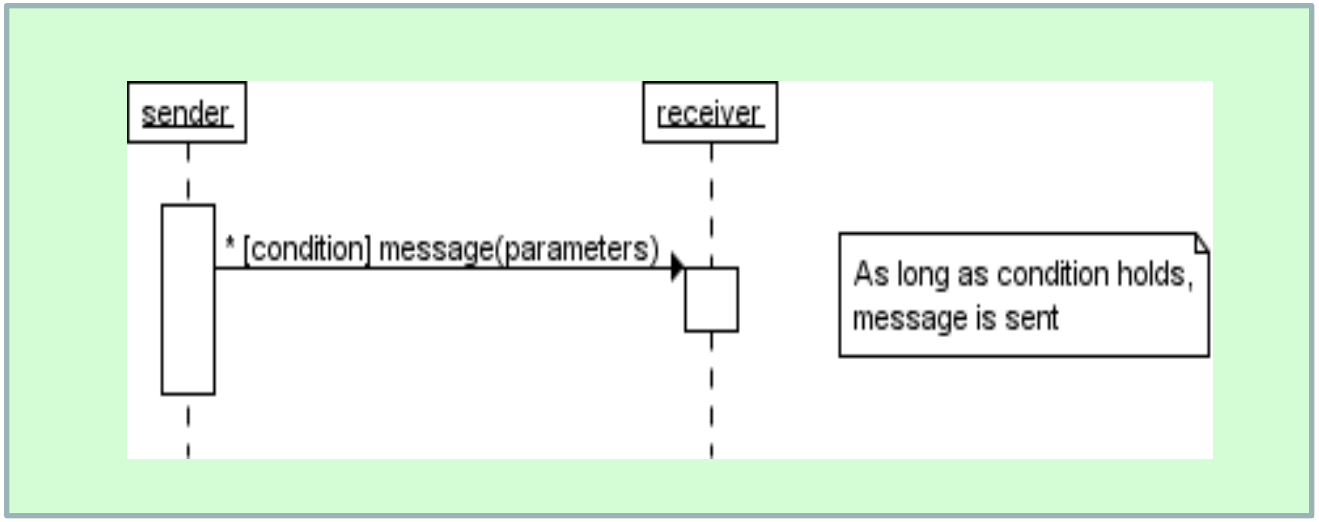
Conditional Interaction

•A message can include a guard
•Guard is simply a condition shown between square brackets.
• Signifies that the message is only sent if a certain condition met


sender
receiver
[condition] message(parameters)
message is only sent if condition is met

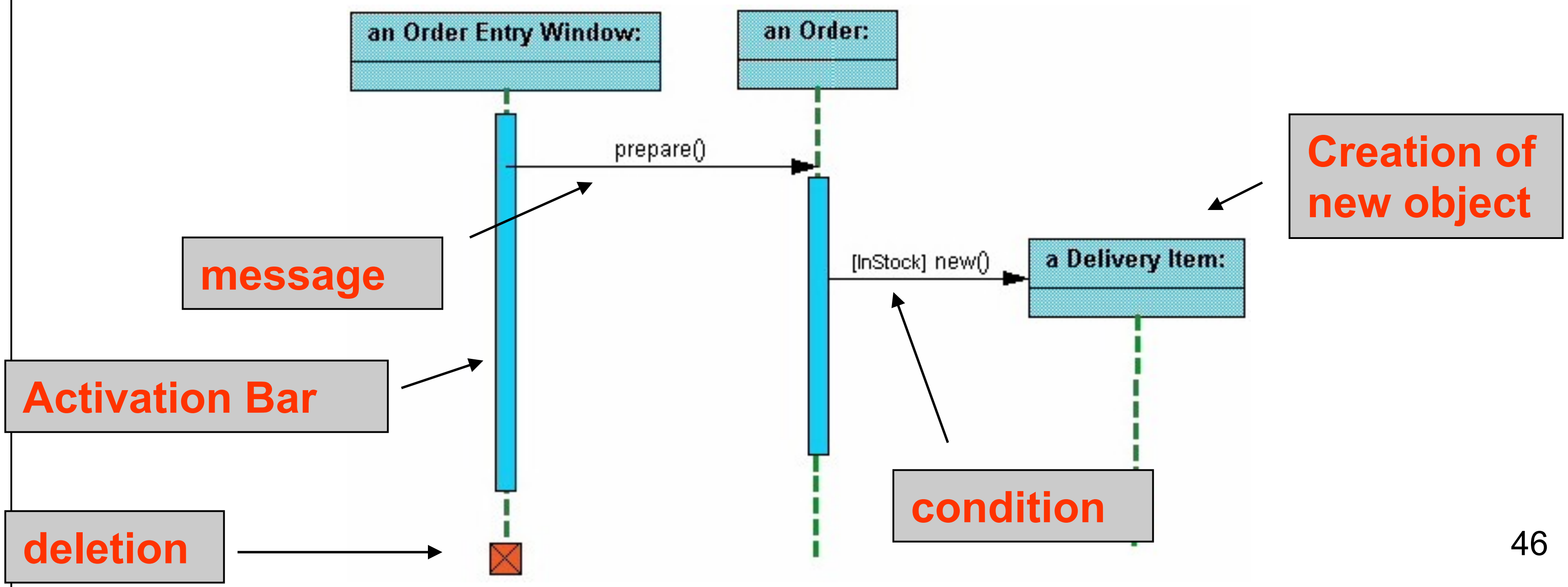
Repeated Interaction

•When a message is prefixed with an asterisk (* symbol), it means that the message is sent repeatedly.
•A guard shows the condition that determines whether or not the message should be sent (again).
•As long as the condition holds, the message is repeated.


sender
receiver
* [condition] message(parameters)
As long as condition holds, message is sent

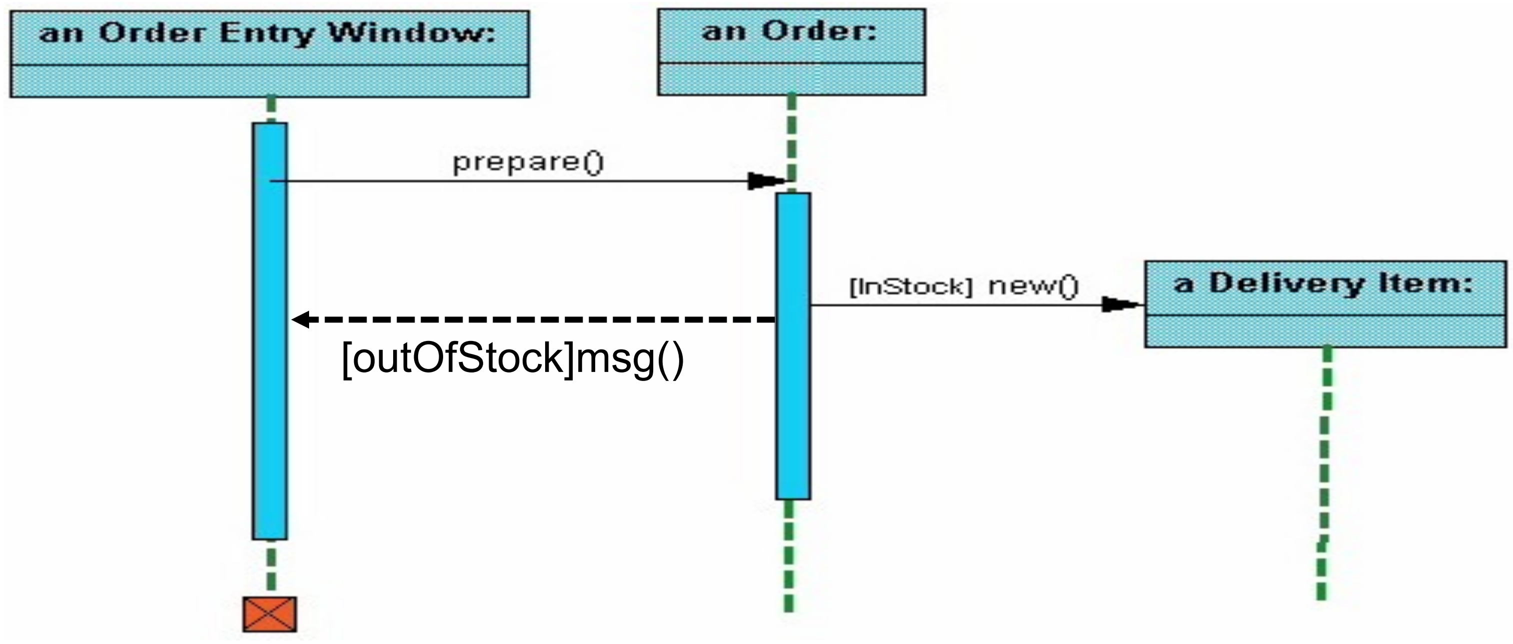
Example – Placing an Order

- The object an Order Entry Window is created and sends a message to an Order object to prepare the order.
- Next the Order object checks to see if the item is in stock
- If the [InStock] condition is met, it sends a message to create a new Delivery Item object

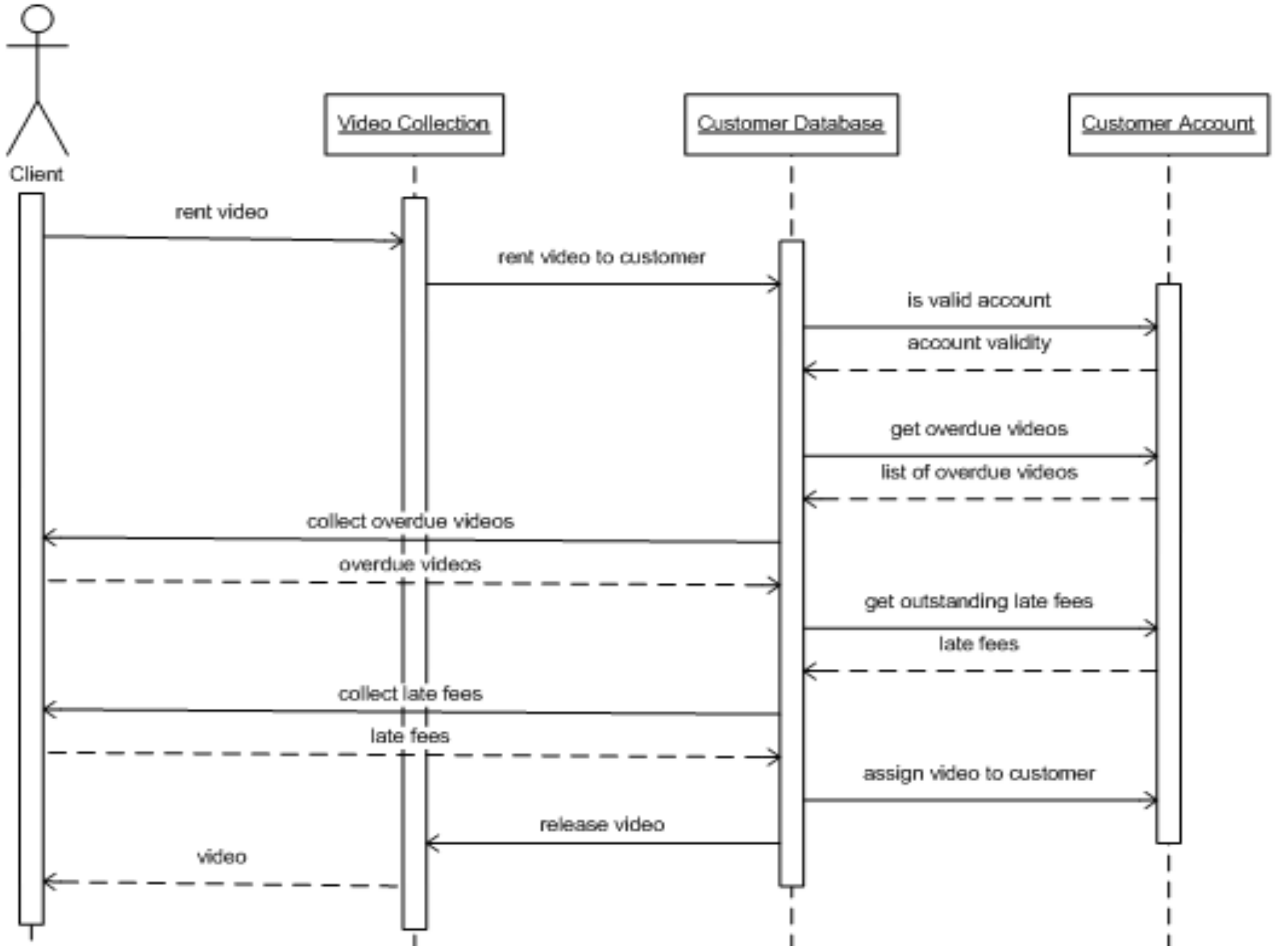


Placing an Order cont.

- Another conditional message is added to the Order object.
- If the item is [OutOfStock] it sends a message back to the Order Entry Window object stating that the object is out of stock.



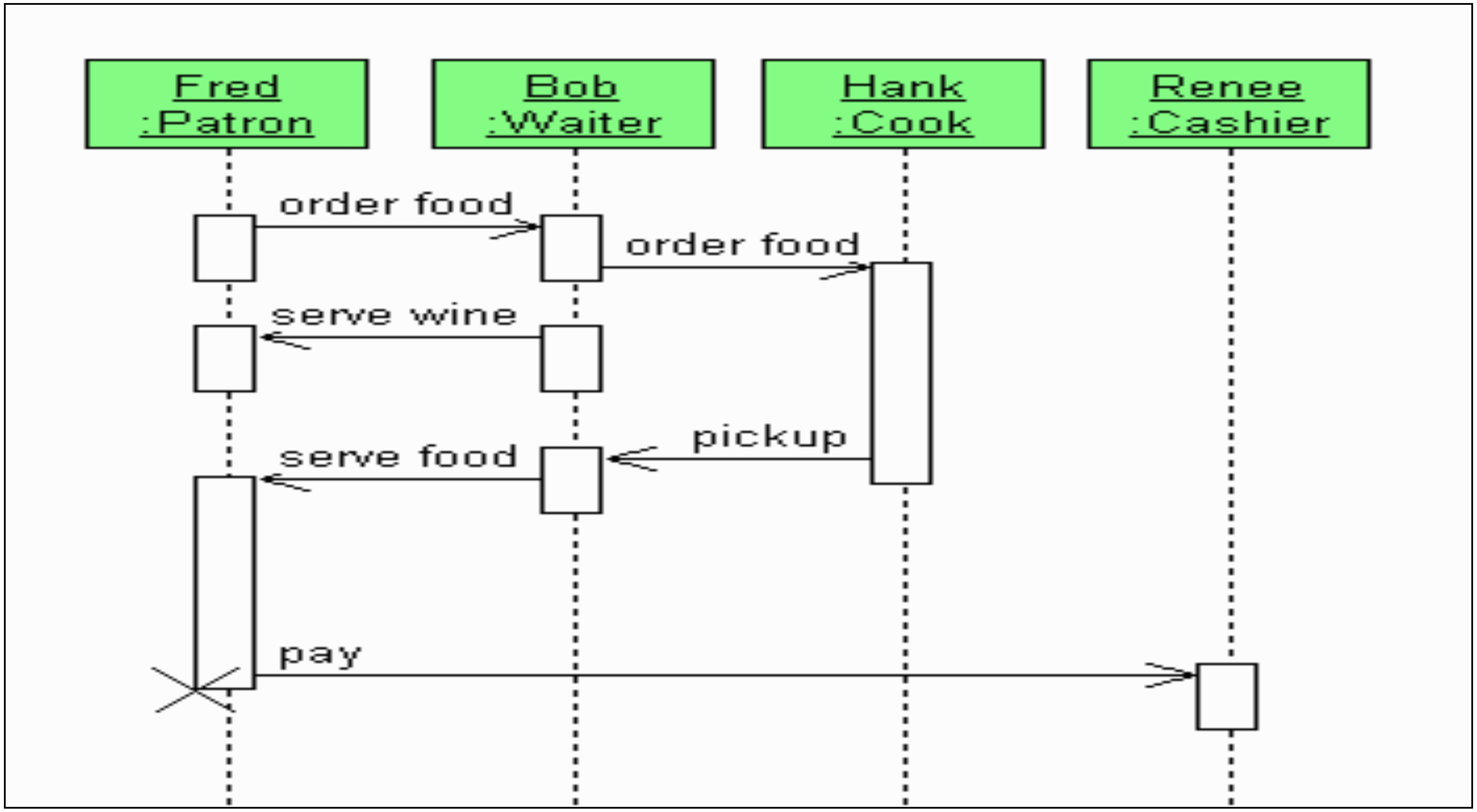
Collecting Video Rentals Example



Collecting Video Rentals Example

- How would you improve the previous example so that it conforms with UML?

Exercise



Fred :Patron
Bob :Waiter
Hank :Cook
Renee :Cashier
order food
order food
serve wine
pickup
serve food
pay

Exercise

- What does the X sign mean?
- ___________
- Which object is created first?
- ___________
- Which object is created last?
- ___________
- What is the difference between the two types of message symbols?

____

____

What do the following call messages mean?

1. shipOrder(String orderId)

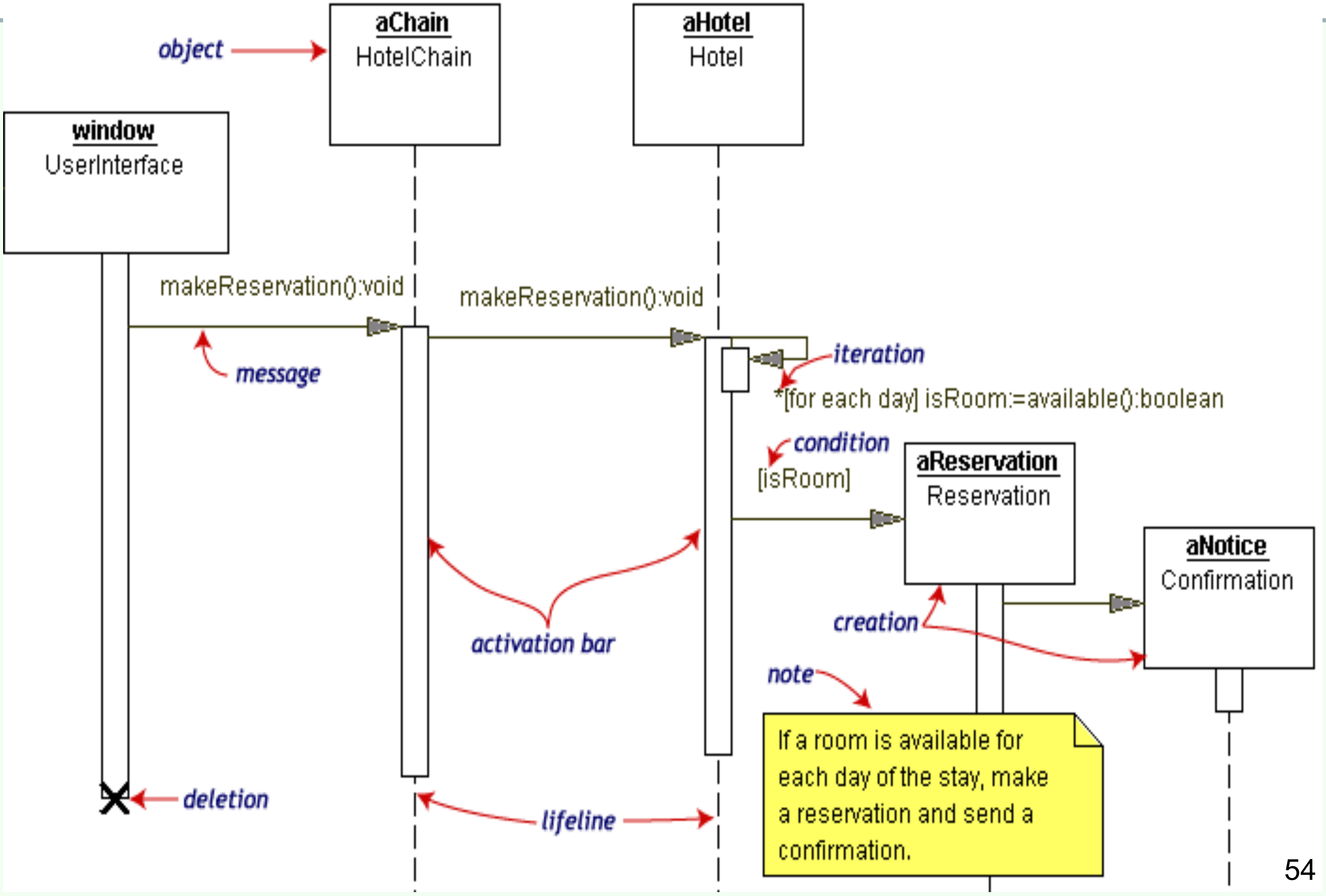
2. [hasStock] shipOrder()

3. *[for all order lines] decrementStockQty()

What do the following mean?

- 1002:bookCopy
- IBM:
- :Customer
- $\boxed{\underline{\text{FileSystem}}}$

Hotel Reservation Example



Hotel Reservation

- **ReservationWindow** object initiates the sequence of messages
- It sends a ***makeReservation() message*** to a **HotelChain.**

- **HotelChain** then sends a ***makeReservation()*** message to a **Hotel**.

- If the **Hotel** has available rooms, then it makes a **Reservation** and a **Confirmation.**

- **Hotel** issues a **self call** to determine if a room is available.
- If so, then the **Hotel** creates a **Reservation** and a **Confirmation**.

- Asterisk on the **self call** means **iteration** (to make sure there is available room for each day of the stay in the hotel).

- Expression in square brackets, [ ], is a **condition.**

- Diagram has a clarifying **note**

- The object is **deleted** – shown with **X**

When to Use Sequence Diagrams

- Excellent tool to analyze and visualize sequences and object connections for a Use Case scenario
- Keep diagrams simple and easy to read
- Use basic notation in diagrams
- No need to show return messages
- Put additional information in comment boxes

Look for **verbs** …messages

Exercise 1

Look for **nouns** ….. Objects

- Draw a Sequence Diagram to show the following:
  - The user sends a message to the oven control unit (ocu) to set the temperature for 200^{o}c.
  - The oven sends a message to the oven light to turn on.
  - The user sends a message to the ocu to turn the fan on.

Look for **verbs**
…messages

Lab Exercise 2

Look for **nouns**
….. Objects

Exercise 2

Draw a sequence diagram:

- When a customer orders items, a purchase order (PO) is created.
- This PO consists of a number of orderlines, each containing details on the items ordered e.g. quantity, price per item etc. Each item on the orderline must be processed.
- This involves checking that the items requested are in stock.
- The order is filled, with whatever items are in stock, and is shipped to the customer.
- You solution should show iterations and conditions.