Sequence Diagrams

Show Sequence of Interactions Between Objects

There are many ways of organizing the UML diagrams.

Can be organized as the following:

1. **Structural diagrams:**
   - to show the building blocks of your
   - Ex: Class diagram
2. **Behavioral diagrams:**
   - to show how your system responds to requests or otherwise evolves over time.
   - Ex: Use case diagram
3. **Interaction diagrams:**
   - Is a type of behavioral diagram.
   - to depict the exchange of messages within a *collaboration* (a group of cooperating objects).
   - Ex: Sequence diagram & Collaboration diagram

UML diagrams

- A series of diagrams describing the *dynamic behavior* of an object-oriented system.
  - A set of messages exchanged among a set of objects within a context to accomplish a purpose.
- Often used to model the way a use case is realized through a sequence of messages between objects.
- The purpose of Interaction diagrams is to:
  - Model interactions between objects
  - Assist in understanding how a system (a use case) actually works
  - Verify that a use case description can be supported by the existing classes
 - Identify responsibilities/operations and assign them to classes

Interaction Diagrams

- Illustrates how objects interacts with each other.
- Emphasizes time ordering of messages.
- Can model simple sequential flow, branching, iteration, recursion and concurrency.
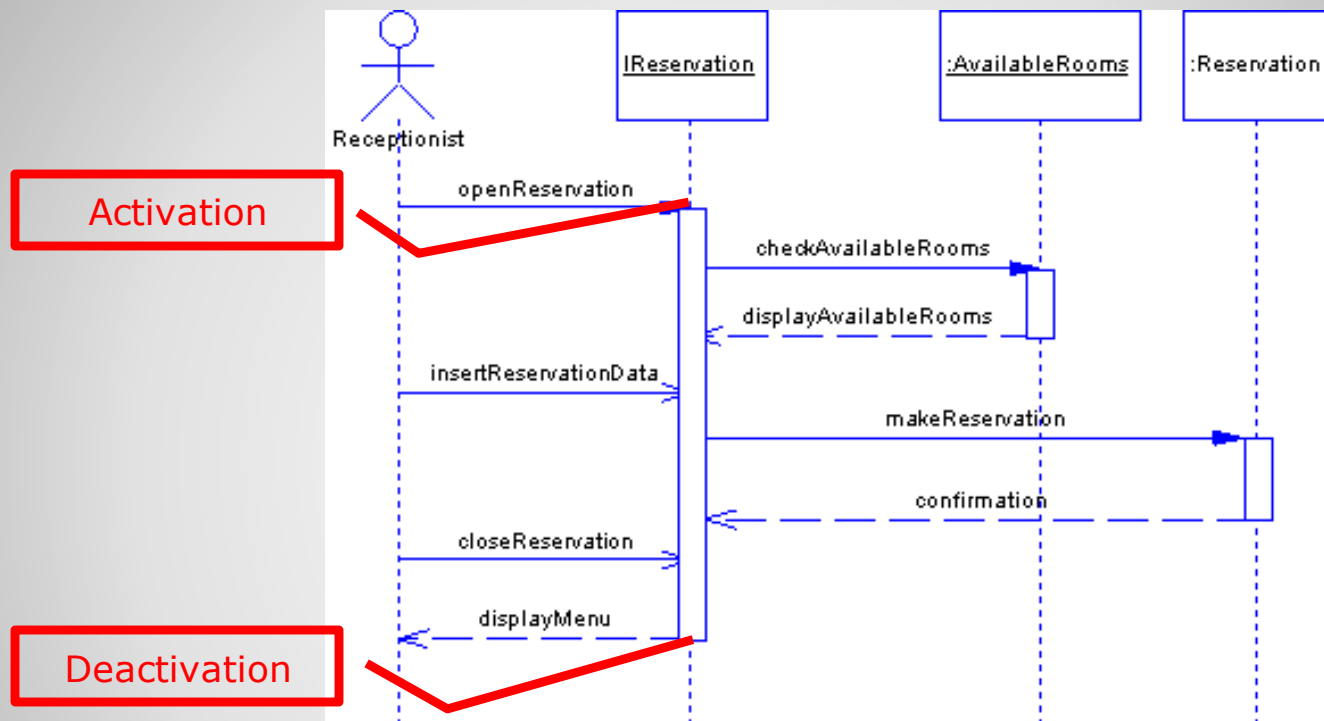
Sequence Diagrams

- ***Classifier*** abstract category of system modeling in UML (e.g. <u>actor</u>, <u>object</u>, <u>interface</u>, component, package, etc.)
- ***Messages*** are represented by an arrow. Describes a control flow in the system; it determines the sequence and place of execution of operations; messages are arranged according to the sequence of their appearance – the later they occur the lower they appear on the diagram
- ***Activations & Deactivation*** are represented by narrow rectangles. It shows time period during which the classifier performs an operation
- ***Lifelines*** are represented by dashed lines. The "X" mark at the end of the lifeline indicates the point at which the object ceases to exist in the system

UML sequence diagrams
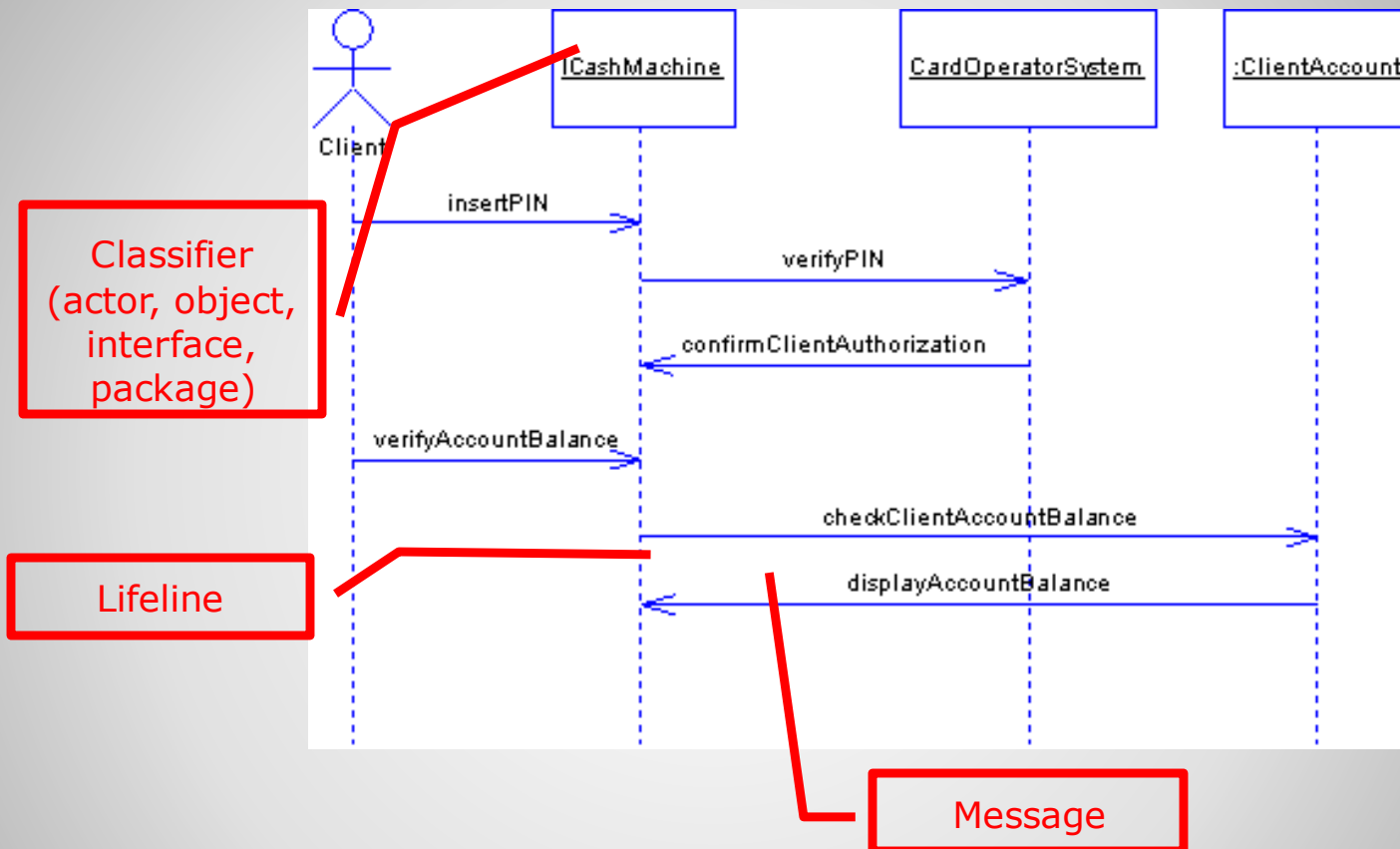
Sequence Diagrams –basic concepts

Sequence Diagrams –basic concepts

- Types of messages

- Creating and destroying objects

- Guard conditions

Sequence Diagrams

Message types:

- **synchronous message –** passes control from the sender classifier to the receiver classifier
- **asynchronous message –** does not pass control, does not wait for an answer from the receiver, may continue processing
- **return message –** indicates control return to the sender classifier after synchronous message and may also initiate a certain operation
- **self message** – message sent by the classifier to itself resulting in calling its own operation; self message is a certain kind of iteration, which creates a nested execution specification
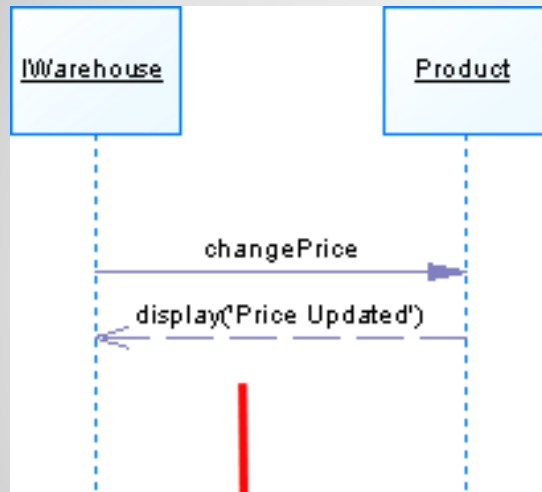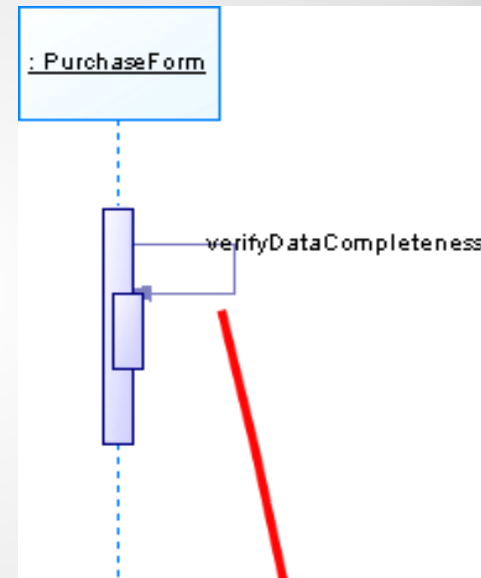
Sequence Diagrams
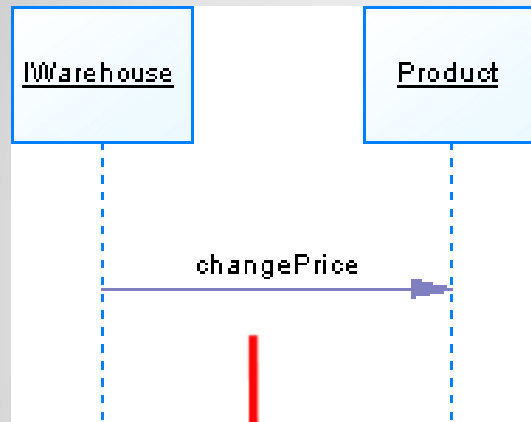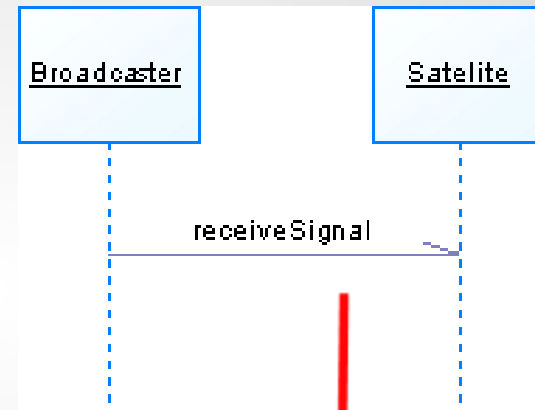
Sequence Diagrams – Message

Return message

Self message

Sequence Diagrams – Message
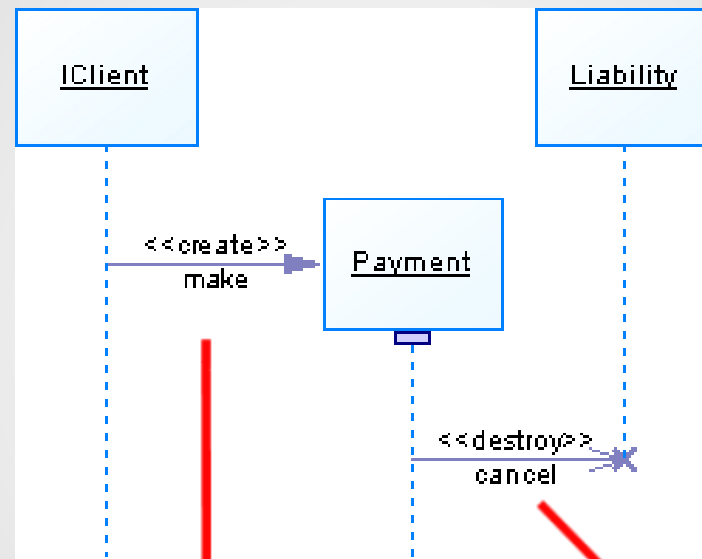
Other message types:

- **lost message** – message sent from a known sender to an unknown receiver (temporary message)
- **found message** – message whose sender is unknown (may be an external signal, stimulus)
- **balking message** – message which will not be handled by the receiver classifier if it cannot be handled immediately
- **timeout message** – similar to balking message although sender classifier is willing to wait for handling the operation for a specified period of time

Sequence Diagrams – Message

Creating and destroying objects:

- **"create" stereotype message** – results in creation of an object, which is situated below the primary existing classifiers, corresponding with the time of its creation
- **"destroy" stereotype message** – results in destruction of an object

Sequence Diagrams – Objects
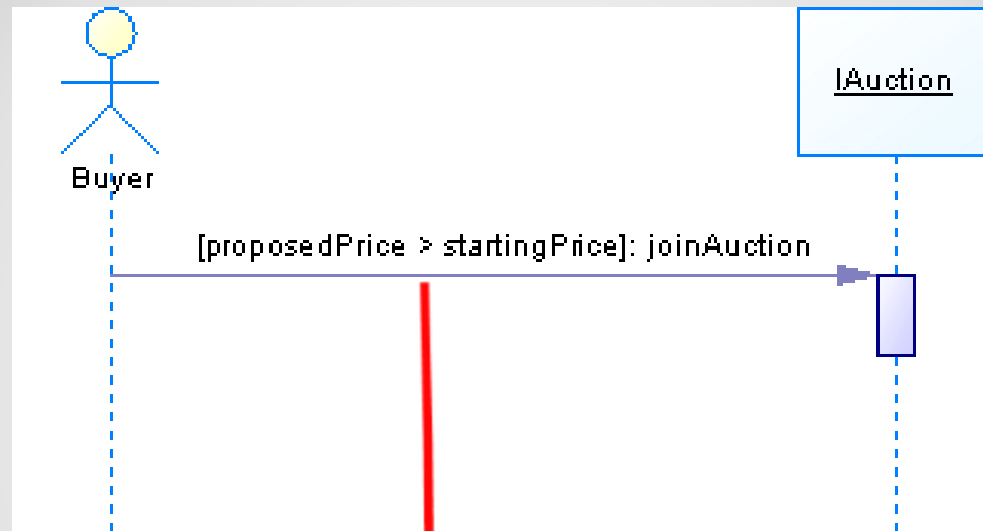
Sequence Diagrams – Objects

16

- **Guard condition** – a criterion connected with the message, on fulfillment of which depends the execution of a specified operation.
- If a condition referring to a certain message is not met the operation indicated by the message is not executed.
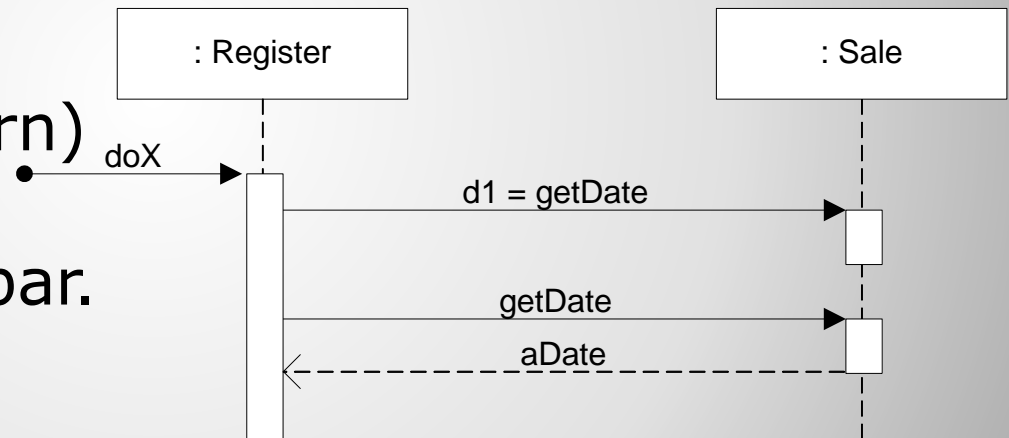- Conditions are placed in square brackets before the message name

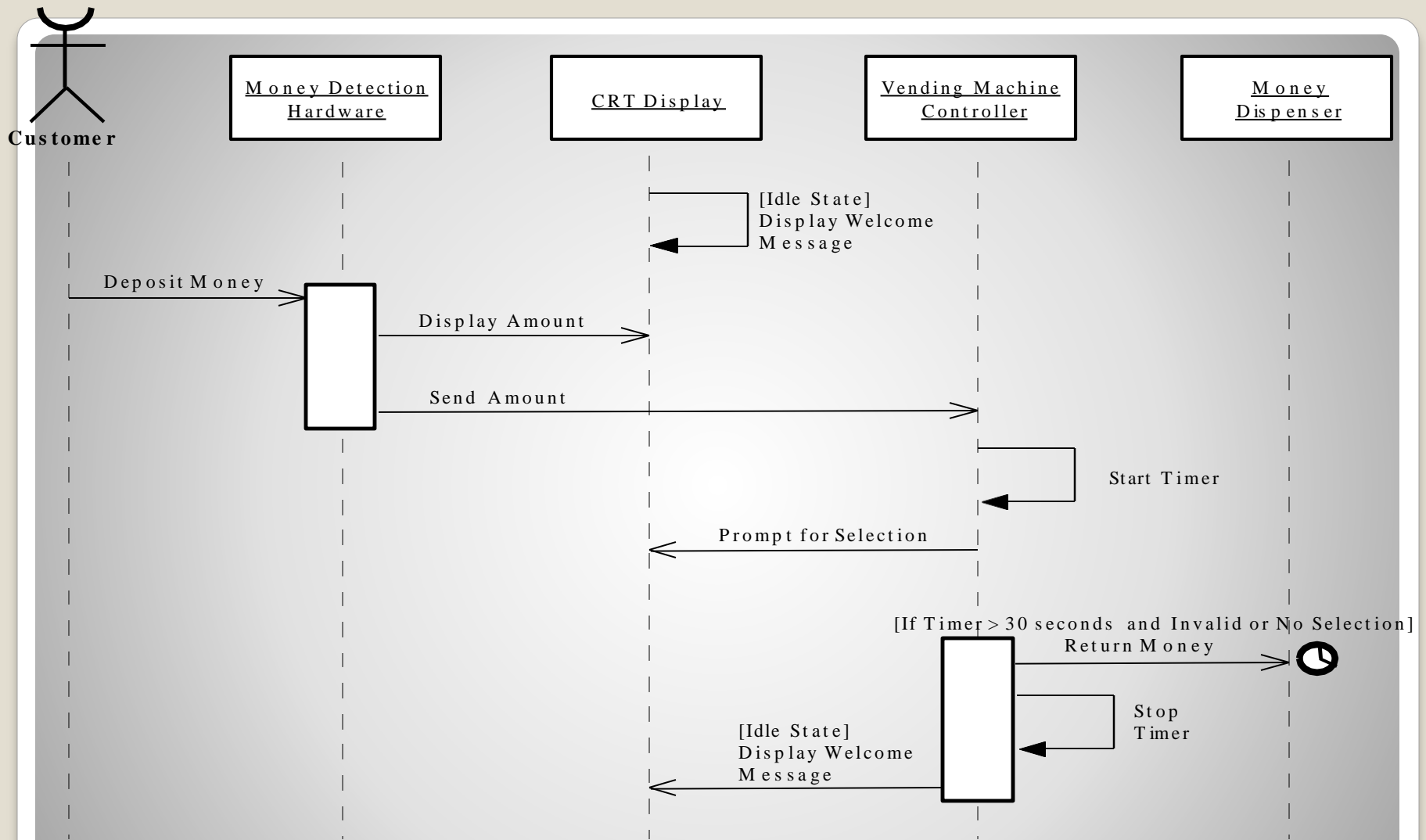# Sequence Diagrams – Condition

Sequence Diagrams – Condition

There are two ways to show return result from a message:

○ Using the message syntax returnVar=message(parameter)

○ Using a reply(or return) message line at the end of an activation bar.

| : Register | | : Sale |

doX

d1 = getDate

getDate

aDate

Reply or Return

Customer

Money Detection Hardware

CRT Display

Vending Machine Controller

Money Dispenser

[Idle State] Display Welcome Message

Deposit Money

Display Amount

Send Amount

Start Timer

Prompt for Selection

[If Timer > 30 seconds and Invalid or No Selection] Return Money

Stop Timer

[Idle State] Display Welcome Message

Sequence Diagram-Timeout Message

```
an Order        a Order Line

        *    prepare()
        ─────────────────▶
        ▲
        │
        │
     Iteration
      marker
```
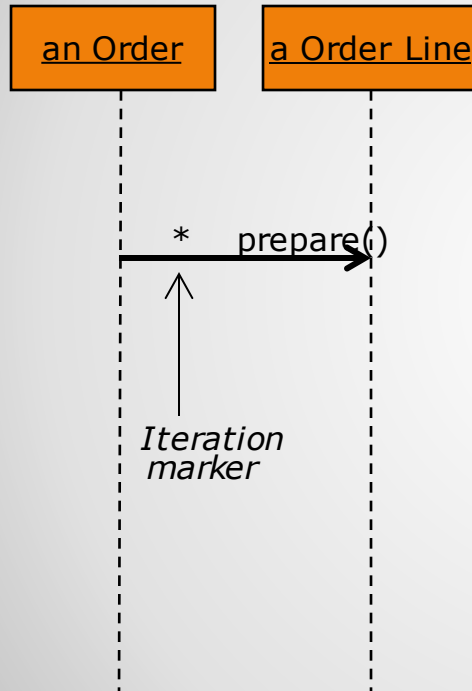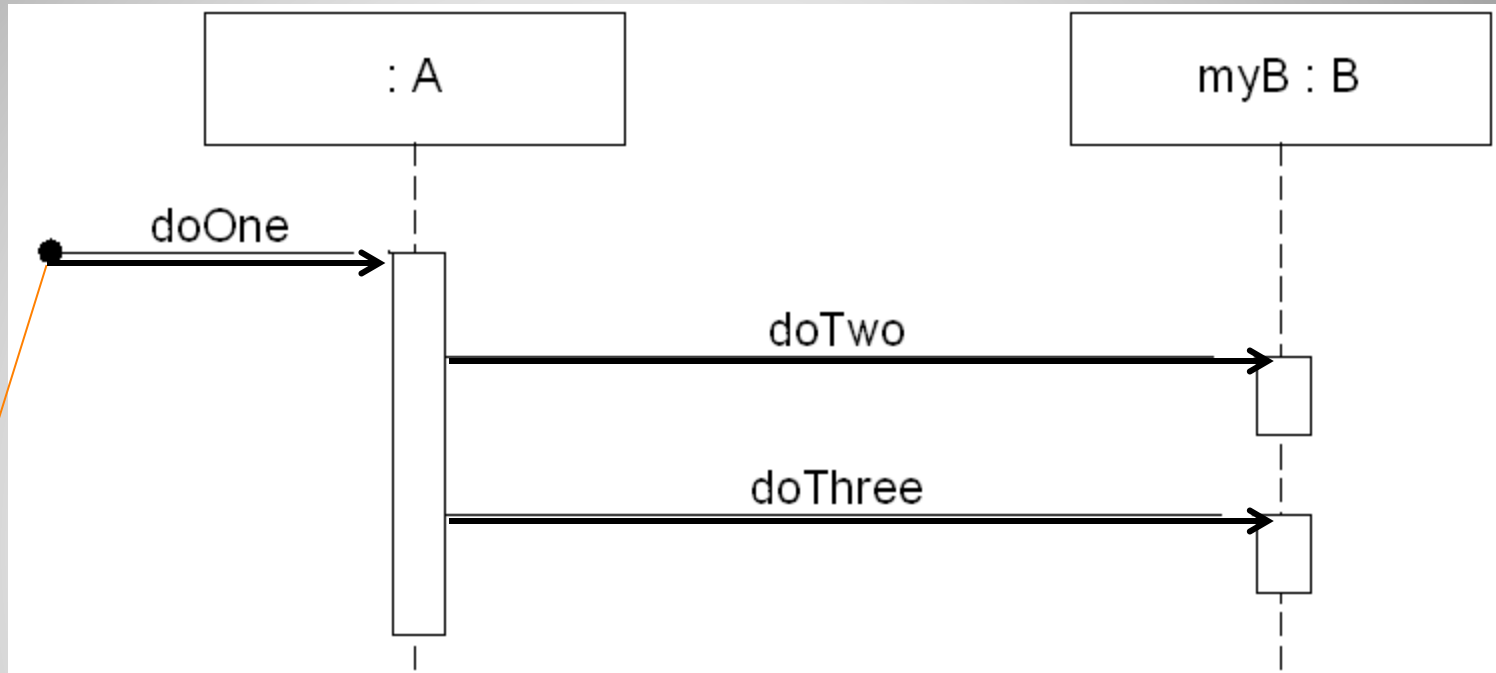
An iteration marker, such as * (as shown), or *[i = 1..n] , indicates that a message will be repeated as indicated.

Sequence Diagram-Iteration

- Illustrate interactions in a kind of fence format, in which each object is added to the right.
- Used to think through concrete details of what message to send, and to whom, and in what order.
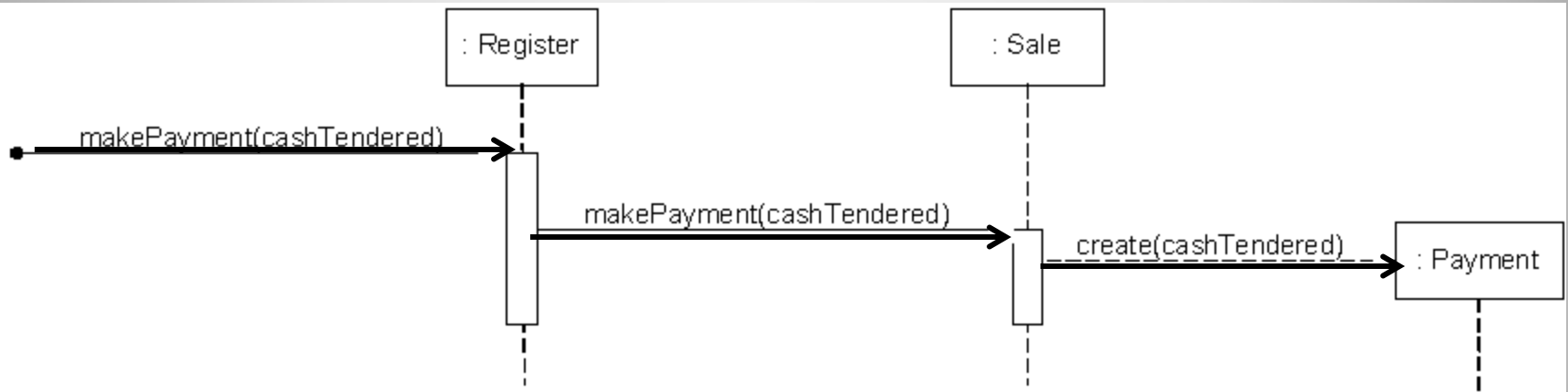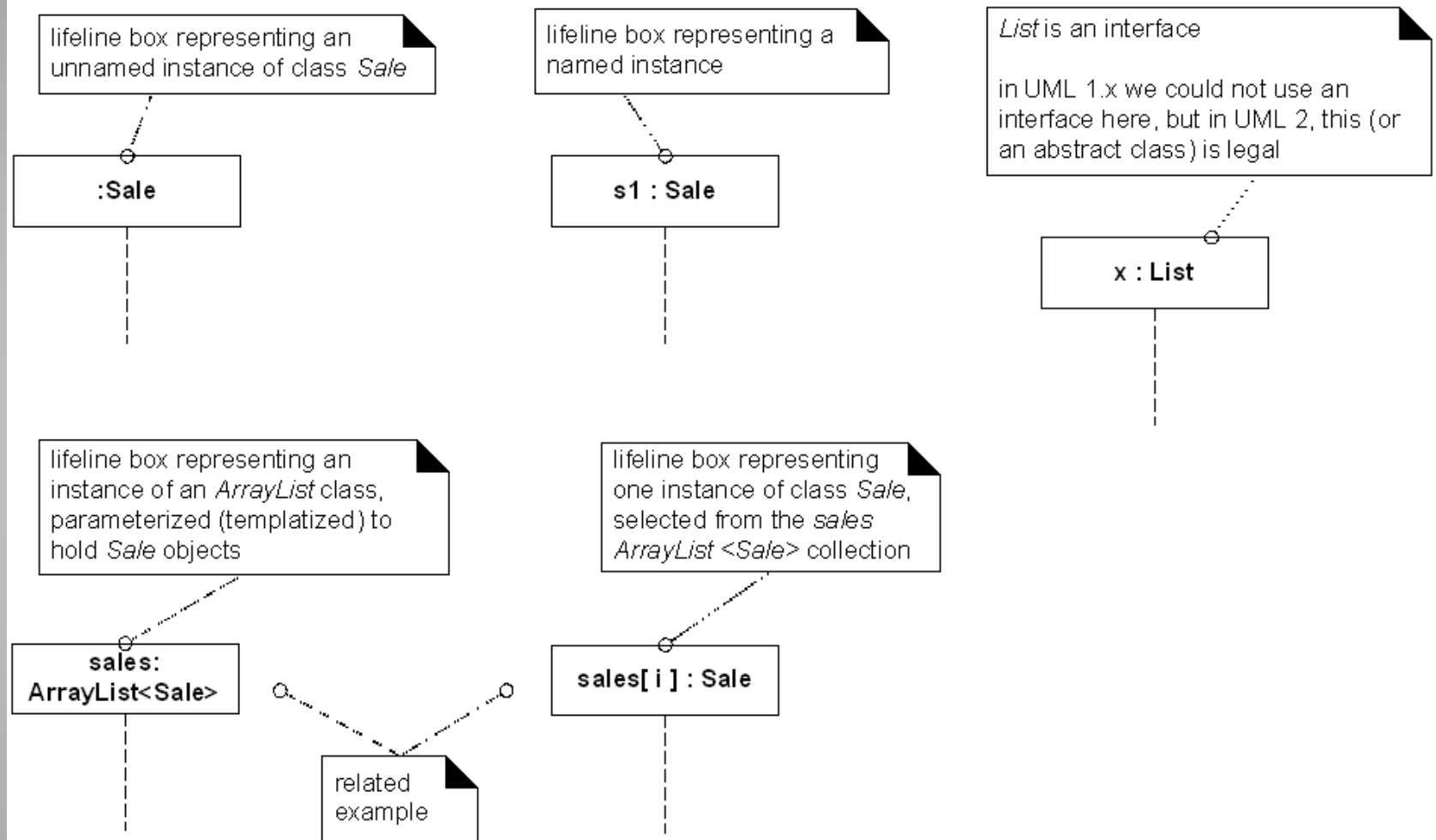
Sequence Diagram

Sequence Diagram

a **found message** whose sender will not be specified

The first message may come from an unspecified participant, and is called a "found message". It is indicated with a ball at the source

Example: Sequence Diagram: makePayment

lifeline box representing an
unnamed instance of class *Sale*

:Sale

lifeline box representing a
named instance

s1 : Sale

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

x : List

lifeline box representing an
instance of an *ArrayList* class,
parameterized (templatized) to
hold *Sale* objects

sales:
ArrayList<Sale>

related
example

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

sales[i] : Sale

Participants with Lifeline Boxes

- Person arrives at desk, gives name to receptionists.

- Receptionists gets map of available Rooms from the system, chooses one.

- Receptionist enters Person's details on Room Management system against chosen room and confirm the reservation.

- Receptionist gives key to Person.
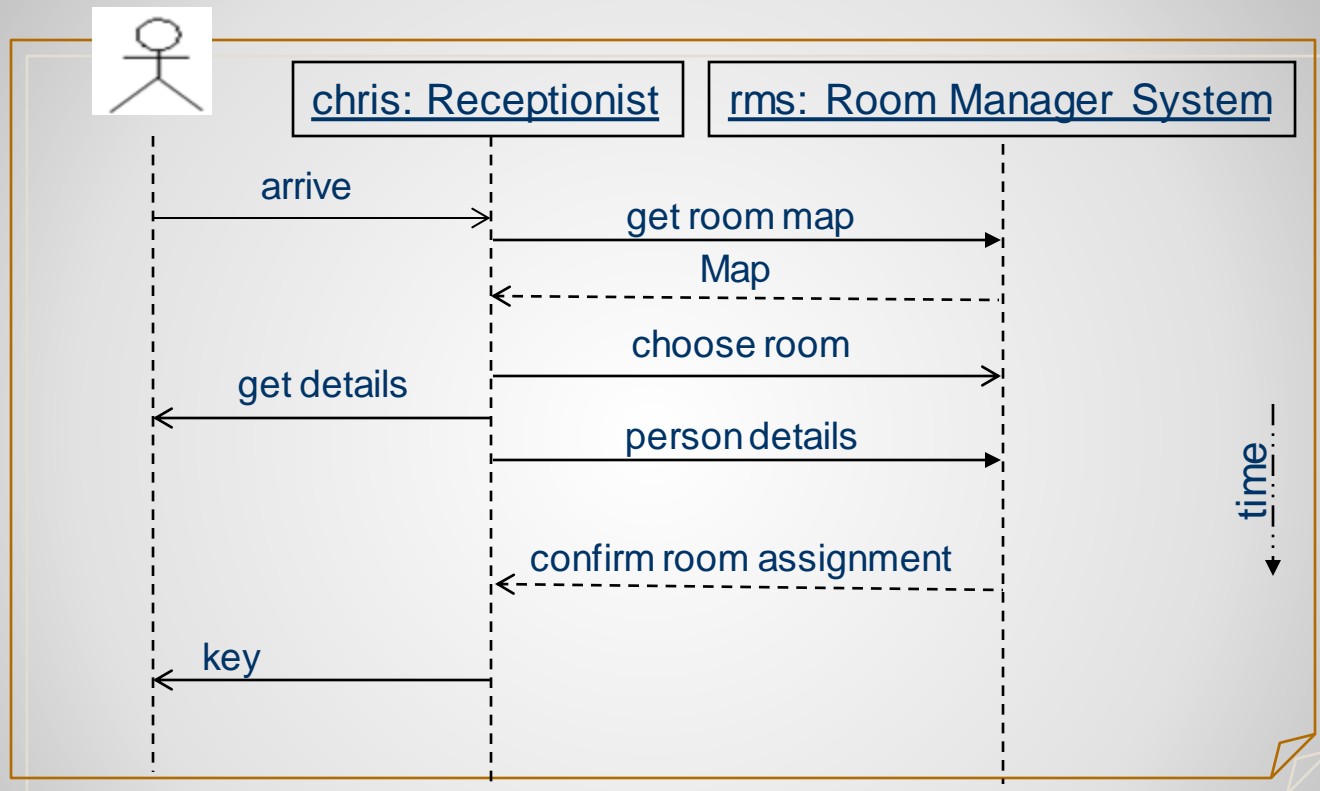
- Joe withdraws $20 from the ATM (flow of events)
  - The process begins when Joe inserts his card into the card reader. The card reader reads the number on Joe's card, then tells the ATM screen to initialize itself
  - The ATM verifies the card against account and prompts Joe for his PIN.
  - Joe enters PIN.
  - Joe's PIN is validated and the ATM prompts him for a transaction
  - Joe selects Withdraw Money
  - The ATM prompts Joe for an amount.
  - Joe enters $ 20.
  - The ATM verifies that Joe's account has sufficient funds and subtracts $ 20 from his account.
  - The ATM dispenses $ 20 and ejects Joe's card

Example for Withdraw

- The convener selects a case on the Disbursement GUI (graphical user interface) screen.
- The Disbursement GUI sends the message QueryCase() to the Disbursement Control object, requesting it to query payment-related details about the case.
- The Disbursement Control object services this request by passing a number of messages to the Case object. These include GetPaymentAmount(), GetPcMember(), and GetPcAccount(). These are requests to retrieve payment and Peace Committee member information relevant to the case.
- The convener approves the disbursement for the case.
- The GUI responds to the approval by sending the message CreatePayments() to the Disbursement Control object.
- The Disbursement Control object responds by sending a Create() message to each required Payment object.
- The Payment object sends a Withdraw() message to the cash account and a Deposit() message to the Peace Committee member account.
- The Disbursement Control object finishes the process by sending the message SetPaidStatus() to the Case object to indicate that payments have been made.

- 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 Result Page.

- Alternate Course:
  ◦ If the Customer did not enter the name of 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.
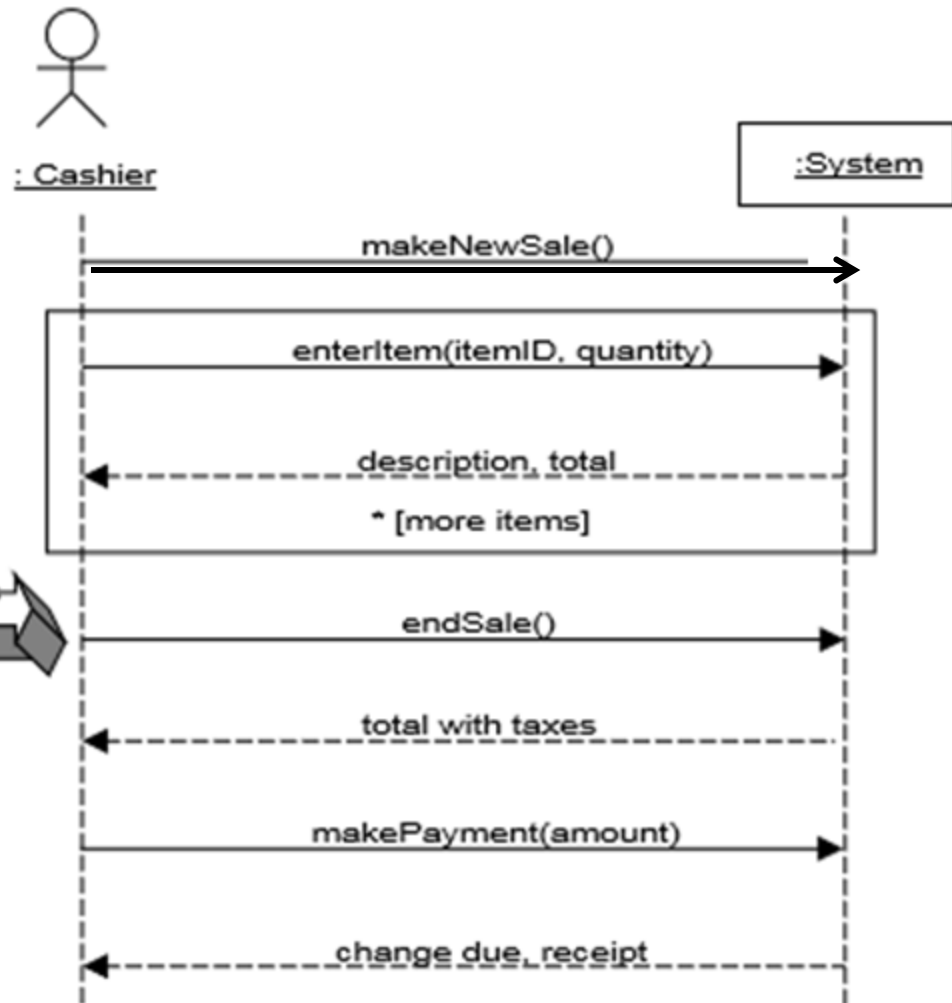
Search an Author

- A system sequence is a picture that shows one particular scenario of a use case, the events that external actors generate, their order, and inter system events.
- All system are treated a black box.
- The emphasis of the diagram is events that cross the system boundary from actors to system.

System sequence Diagram

- **Combined fragment** – is a logically consistent area of interaction, a part of a sequence diagram characterized by specific properties defined by the interaction operator such as loop and a guard.

Sequence Diagrams – Diagram Frames

Frame Operator	Description
alt	Alternative fragment for mutual exclusive 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 execute if the guard is true.
par	Parallel fragments that execute in parallel.
region	Critical region within which only one thread can run.

Frame Operators

```
Class order
Distributor careful;
Distributor regular;
Messenger messenger;
procedure dispatch
foreach (lineitem)
        if (value > $10000K)
                careful .dispatch
        else
                regular .dispatch
end if
end for
if (needsConfirmation)
messenger .confirm
end procedure
end class
```
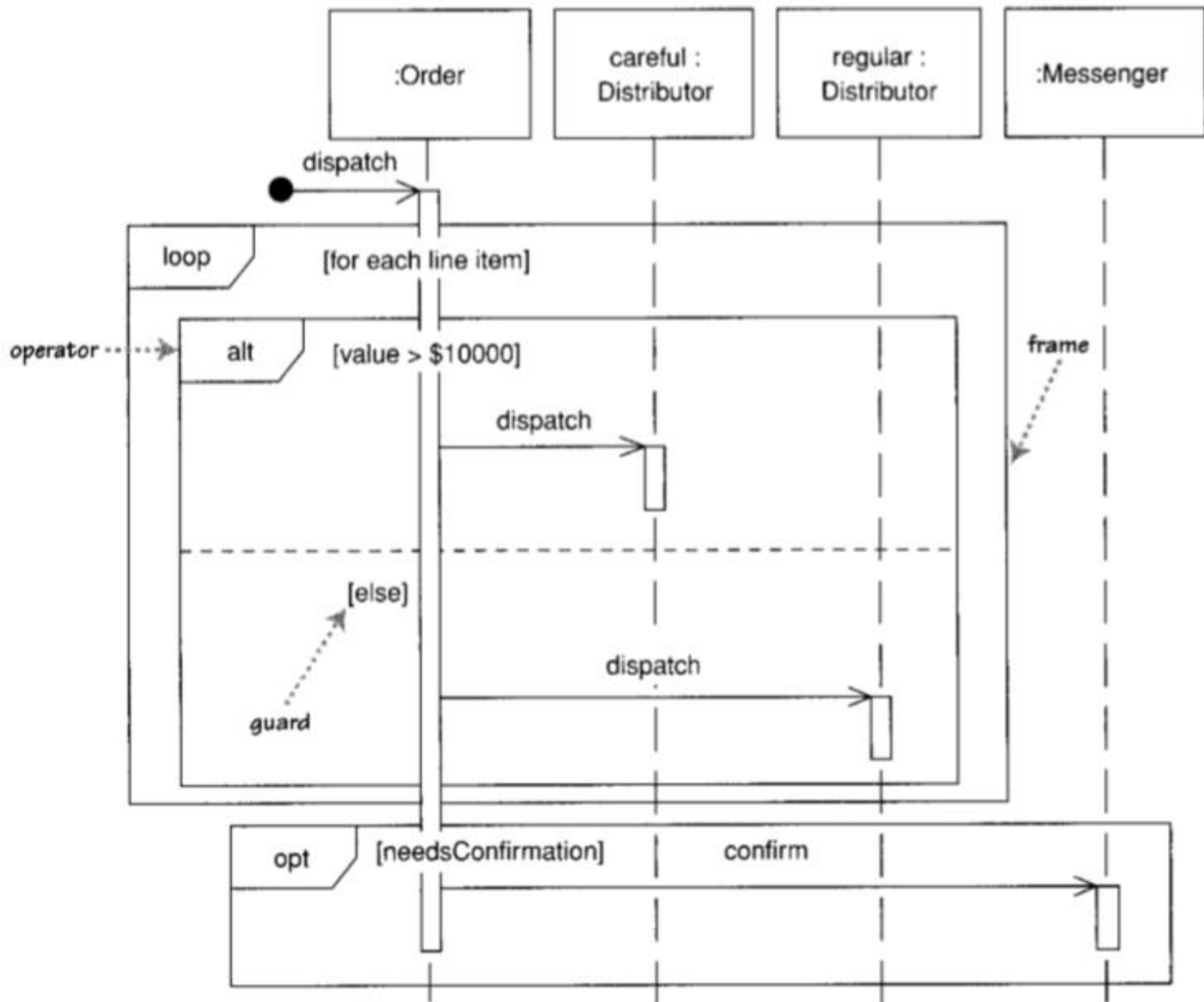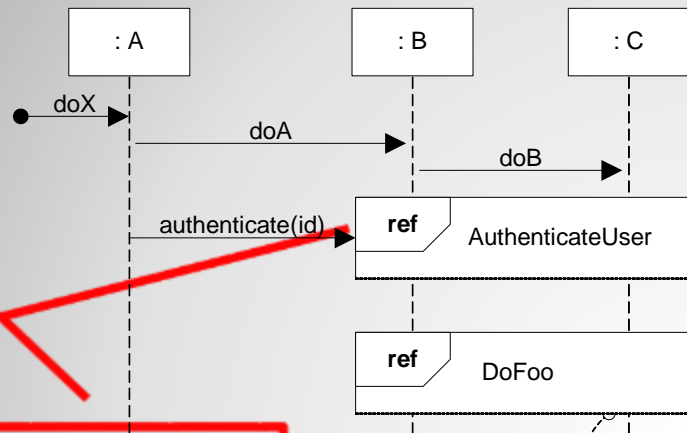
- **Interaction occurrence** – is a reference to a linked interaction diagram, placed within the base diagram
- Interaction occurrences are especially useful in case of extensive sequence diagrams, which refer to other diagrams defined earlier
- Interaction occurrence can be invoked either by a message or by time factor

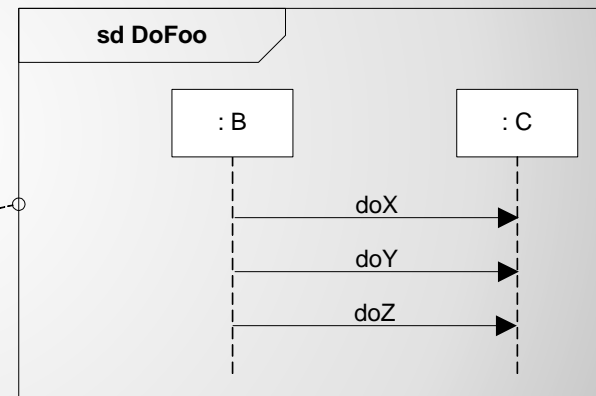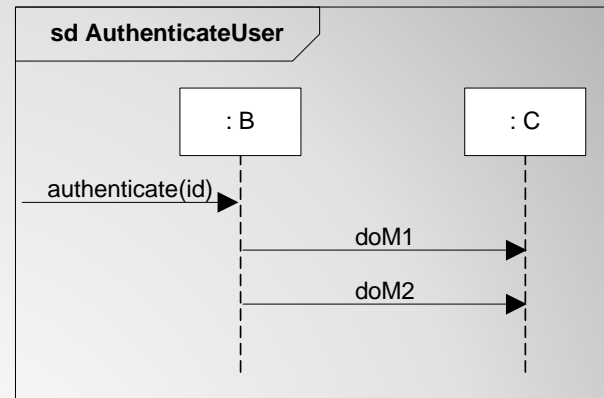# Sequence Diagrams-Interaction Occurrence

Sequence Diagrams – ref

Once a customer has provided the information for an order, the system checks if all required information is there, either given in the order (type of product, quantity, etc.), or in the account (address and payment information). If any information is lacking, the system will inform the customer that it needs to be added before the order can be placed. Once all information is in place, the order is placed in the system, and the credit card information is sent to the bank for approval. If the bank approves the card, the order is finalized, otherwise customer is asked to reenter the credit card.

Example

The scenario begins when the player chooses to start a new round in the UI. The UI asks whether any new players want to join the round; if so, the new players are added using the UI.

All players' hands are emptied into the deck, which is then shuffled. The player left of the dealer supplies an ante bet of the proper amount. Next each player is dealt a hand of two cards from the deck in a round-robin fashion; one card to each player, then the second card.

If the player left of the dealer doesn't have enough money to ante, he/she is removed from the game, and the next player supplies the ante. If that player also cannot afford the ante, this cycle continues until such a player is found or all players are removed.

Poker-Start New Game Round

- The scenario begins after the Start New Round case has completed. The UI asks the first player for a bet. That player chooses to either bet a given amount, or check (no bet). The next player is asked what to do. If the prior player placed a bet, the next player must either match ("see") it, or match it plus add an additional bet ("raise"), or choose not to match and exit the round ("fold"). This continues around the table until an entire pass is made in which all players have either matched all other players' bets or folded. If the next player doesn't have enough money to match the current bet, the player is allowed to bet all of their money. But they can then win only up to the amount they bet; the rest is a "side pot" among the more wealthy players remaining in the round.

Poker-Betting Round

Let's do a sequence diagram for the following casual use case

The scenario begins when the user chooses to add a new appointment in the UI. The UI notices which part of the calendar is active and pops up an Add Appointment window for that date and time.
The user enters the necessary information about the appointment's name, location, start and end times. The UI will prevent the user from entering an appointment that has invalid information, such as an empty name or negative duration. The calendar records the new appointment in the user's list of appointments. Any reminder selected by the user is added to the list of reminders.

If the user already has an appointment at that time, the user is shown a warning message and asked to choose an available time or replace the previous appointment. If the user enters an appointment with the same name and duration as an existing group meeting, the calendar asks the user whether he/she intended to join that group meeting instead. If so, the user is added to that group meeting's list of participants.

Add Calendar Appointment