Information System Design
IT60105

Lecture 09

Interaction Diagrams - I

Lecture #09

- What is a sequence diagram?
- Basic components in any sequence diagram and their notations
- Illustration of applications in Modeling
- Collaboration diagrams

Interaction Diagrams

- Interaction diagrams model how groups of objects collaborate in some behavior

- There are two types of interaction diagrams
  - Sequence diagrams
  - Collaboration diagrams

Sequence Diagram

What is a Sequence Diagram?

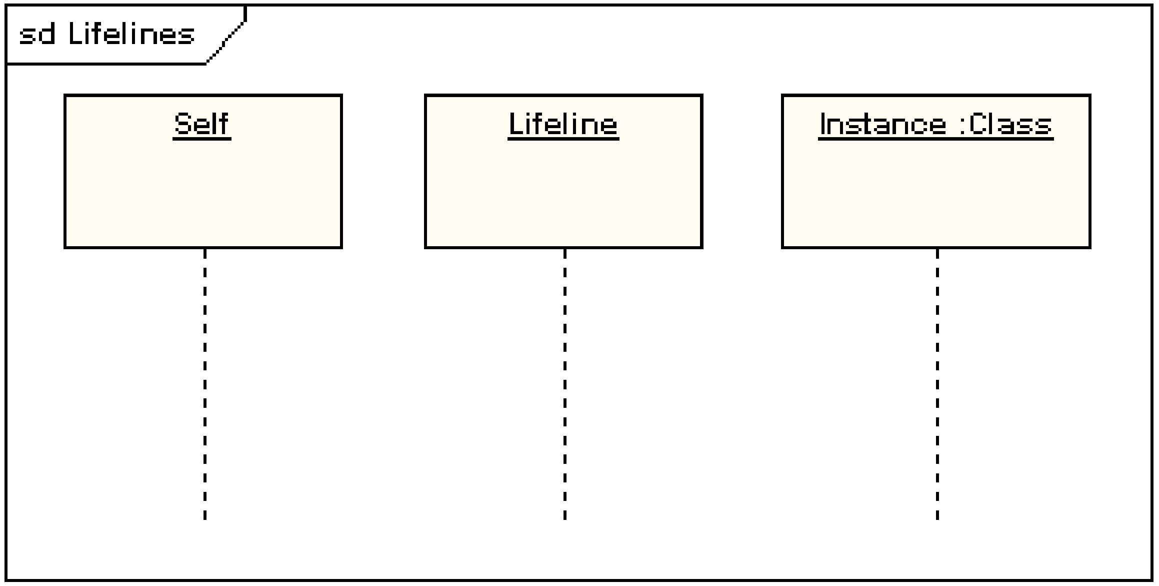
- A sequence diagram shows object interactions arranged in time sequence
- It depicts the object and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario
- Typically, a sequence diagram captures the behavior of a single activity or a use case

Basic of a Sequence Diagram

- A sequence diagram is a two dimensional chart
- The chart is read from top to bottom
- The objects participating in the interaction are shown at the top of the chart as boxes attached to a vertical-dashed line
- Inside the box the name of the object is written with a colon separating it form the name of the class and both the name of the class and object are underlined
- Some times an anonymous object (only class name and underlined) is also used

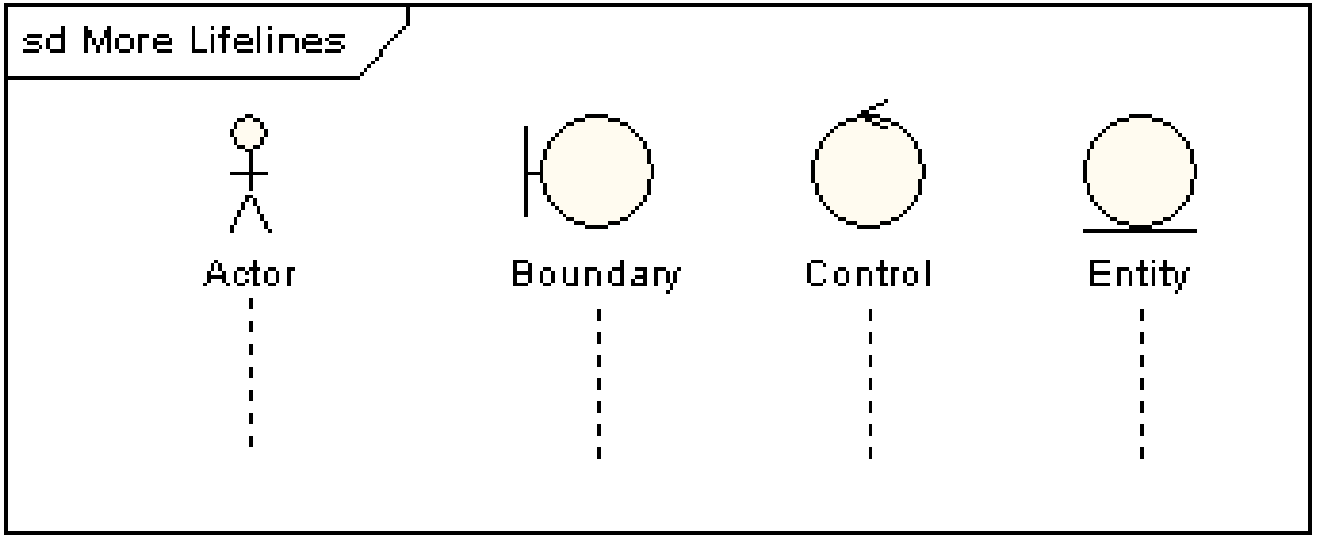
Life Line in a Sequence Diagram

- A lifeline represents an individual participant in a sequence diagram. A lifeline will usually have a rectangle containing its object name. If its name is self then that indicates that the lifeline represents the classifier which owns the sequence diagram

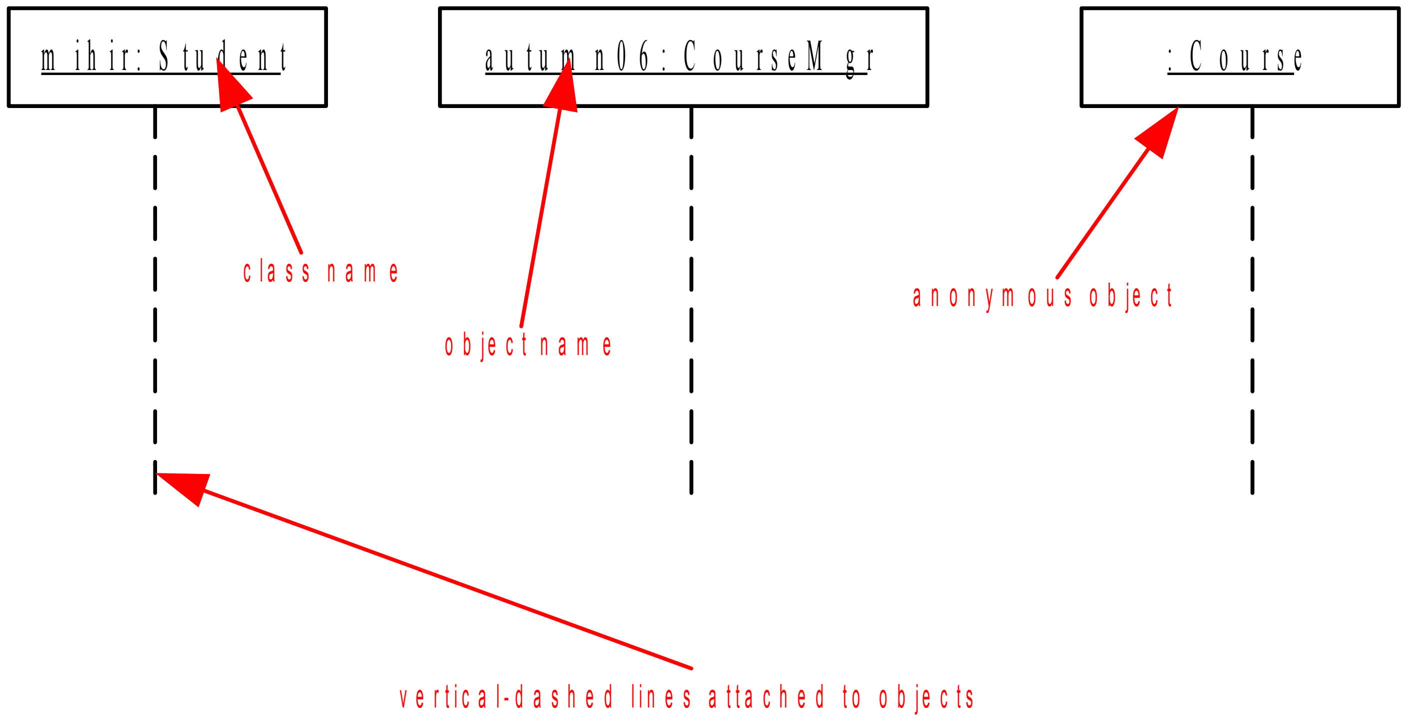


Life Line in a Sequence Diagram

- Sometimes a sequence diagram will have a lifeline with an actor element symbol at its head. This will usually be the case if the sequence diagram is owned by a use case. Boundary, control and entity elements form robustness diagrams can also own lifelines



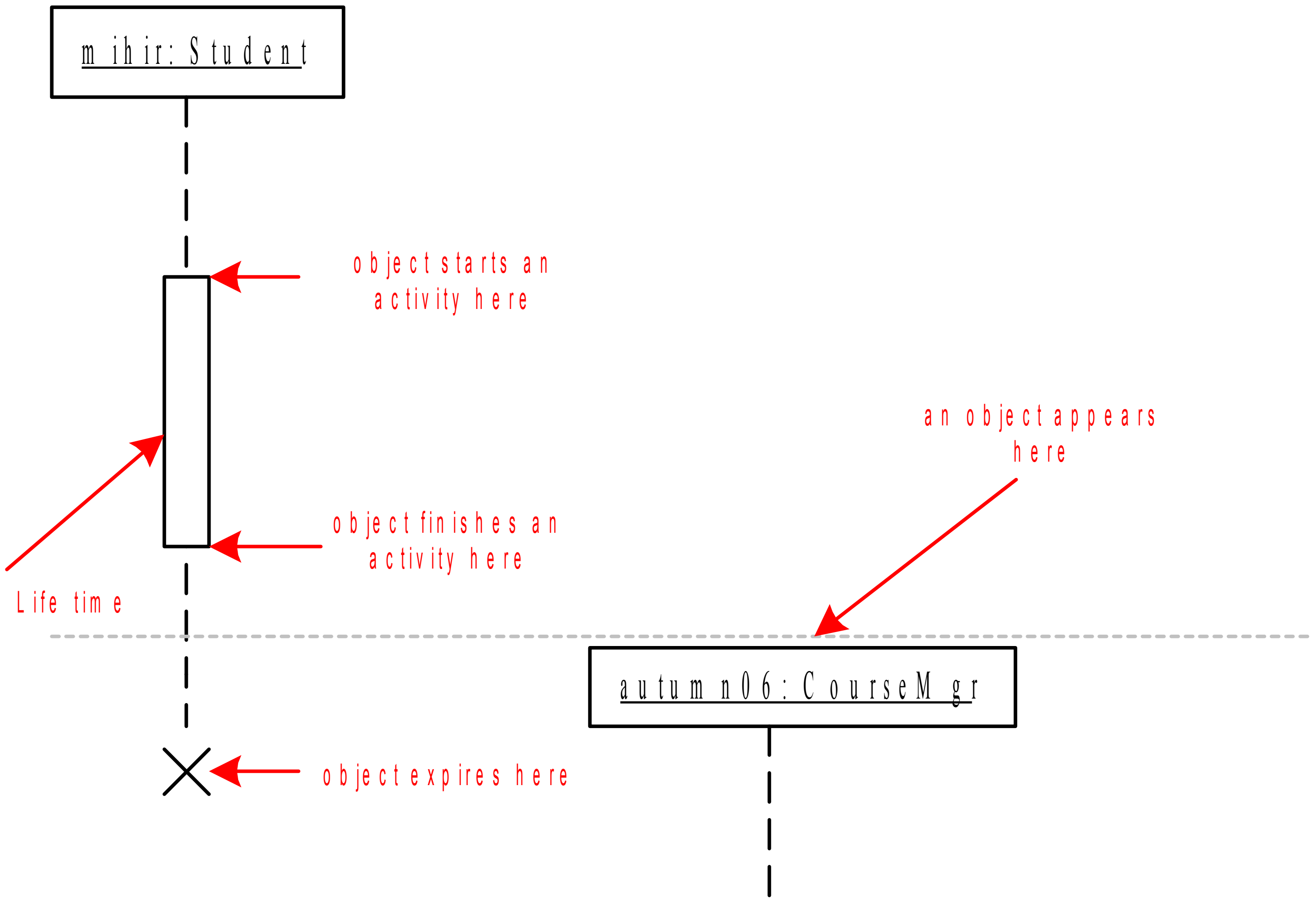
Example: Objects and Life Line



Objects and Life Time

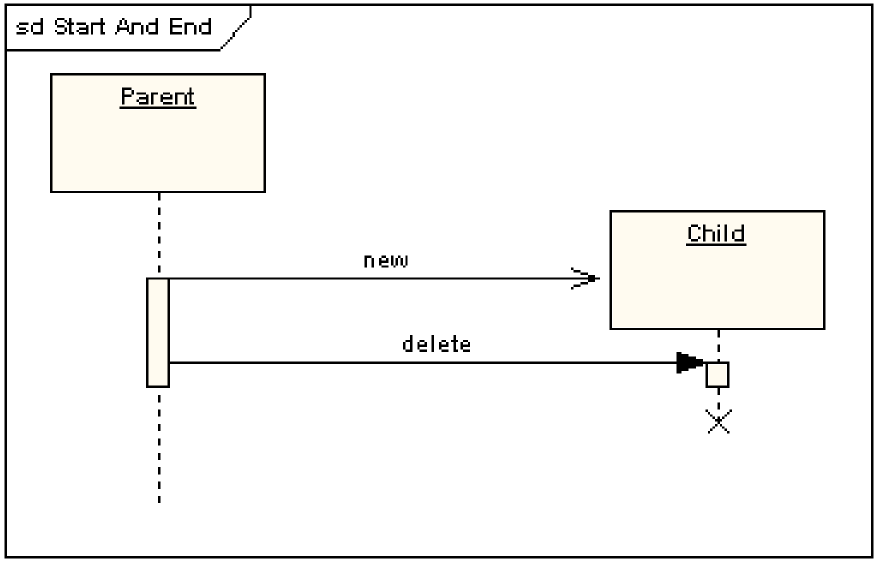
- The objects appearing at the top signifying that the object already existed when the use case execution was executed.
- However, if some object is created during the execution of the use case and participates in the interaction, then that object should be shown at the appropriate place on the diagram where it was created
- The vertical dashed line in the sequence diagram is called the object's life time. The life time indicates the existence of the object at any particular point of time
- A rectangle is used on the life time to indicate the activation symbol and implies that the object is active as long as the rectangle exists

Example: Object's Life Time



Start and End of Life Line

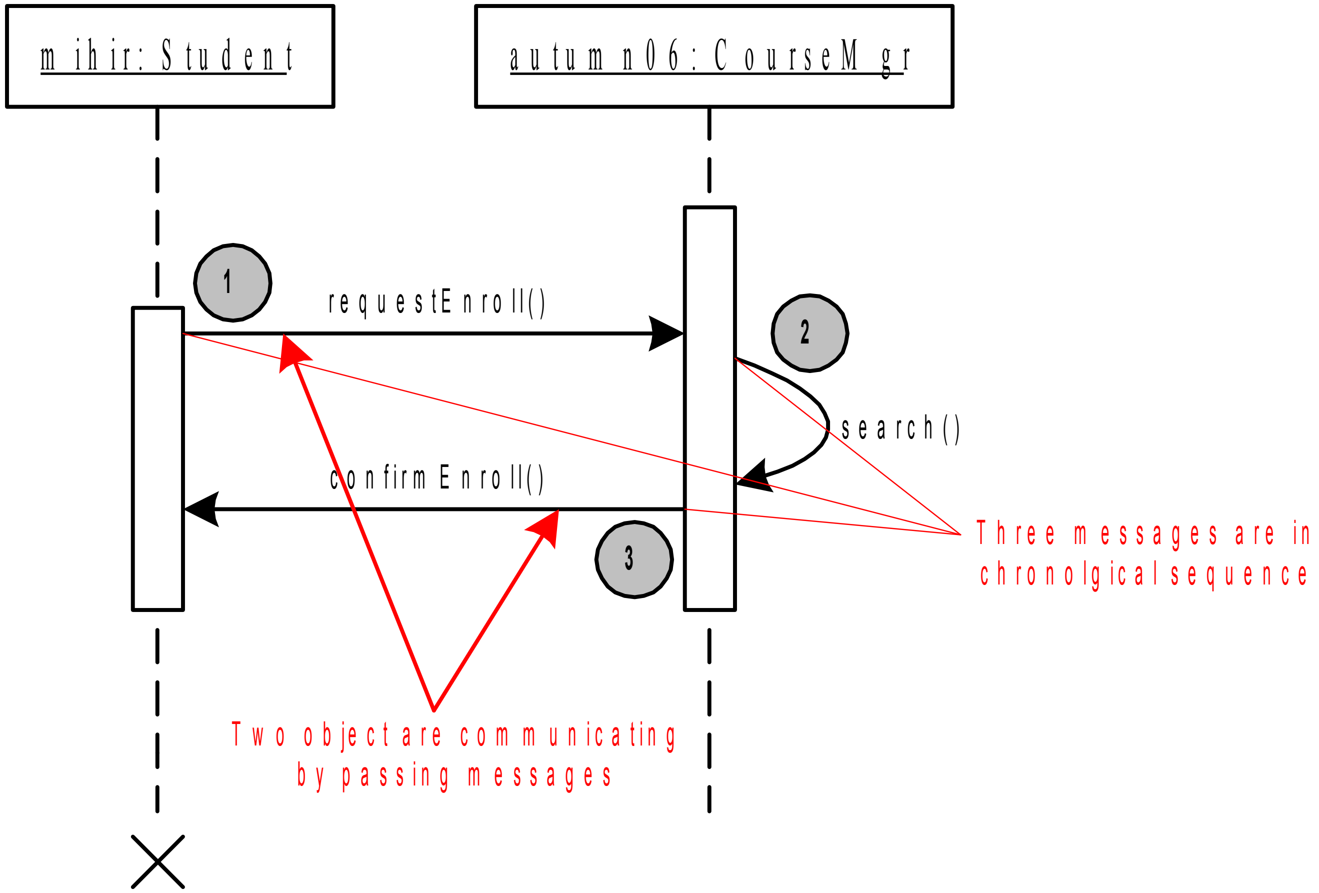
- A lifeline may be created or destroyed during the timescale represented by a sequence diagram. In the latter case, the lifeline is terminated by a stop symbol, represented as a cross. In the former case, the symbol at the head of the lifeline is shown at a lower level down the page than the symbol of the object that caused the creation. The following diagram shows an object being created and destroyed



Messages in Sequence Diagrams

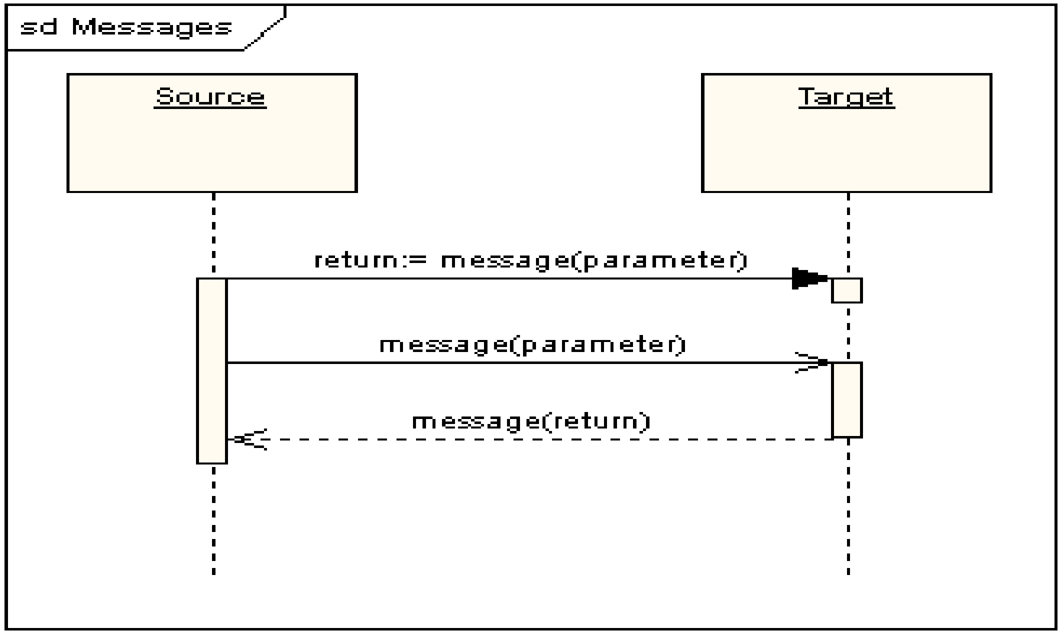
- Two objects in a sequence diagram interacts with passing messages between them
- Each message is indicated as an arrow between the lifelines of two objects
- The order of message is very important in the sequence diagram. They should appear in chronological order from top to the bottom. That is, reading the diagram from the top of the bottom would show the sequence in which the message occurs
- Each message is labeled with the message name

Example: Message in a Sequence Diagram


mihir: Student
autumn06: CourseMgr
1
requestEnroll()
2
search()
confirmEnroll()
3
Three messages are in chronolgical sequence
Two object are communicating by passing messages

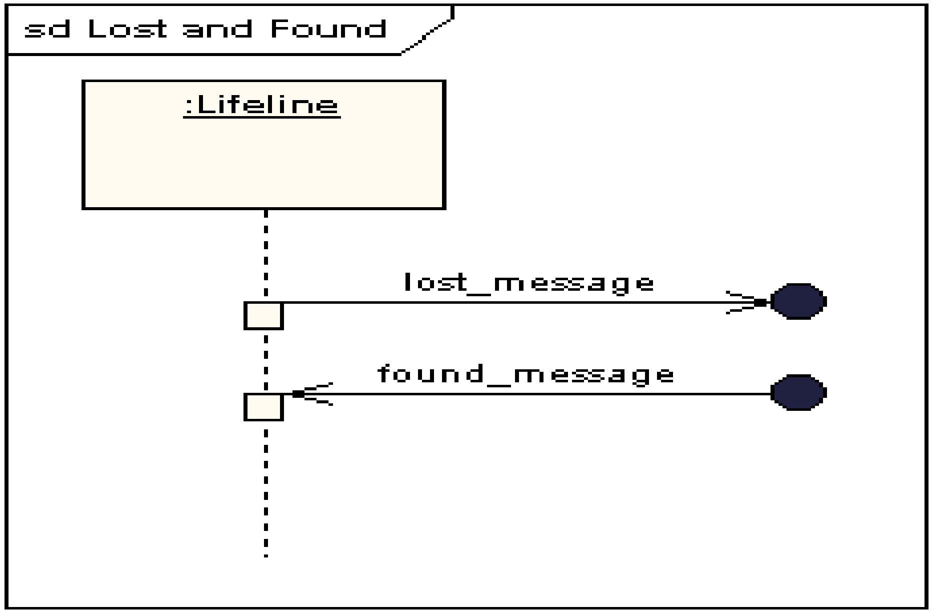
More on Messages

- Messages can be synchronous or asynchronous; call or signal. In the following diagram, the first message is a synchronous message (denoted by the solid arrowhead) complete with an implicit return message; the second message is asynchronous (denoted by line arrowhead) and the third is the asynchronous return message (denoted by the dashed line)



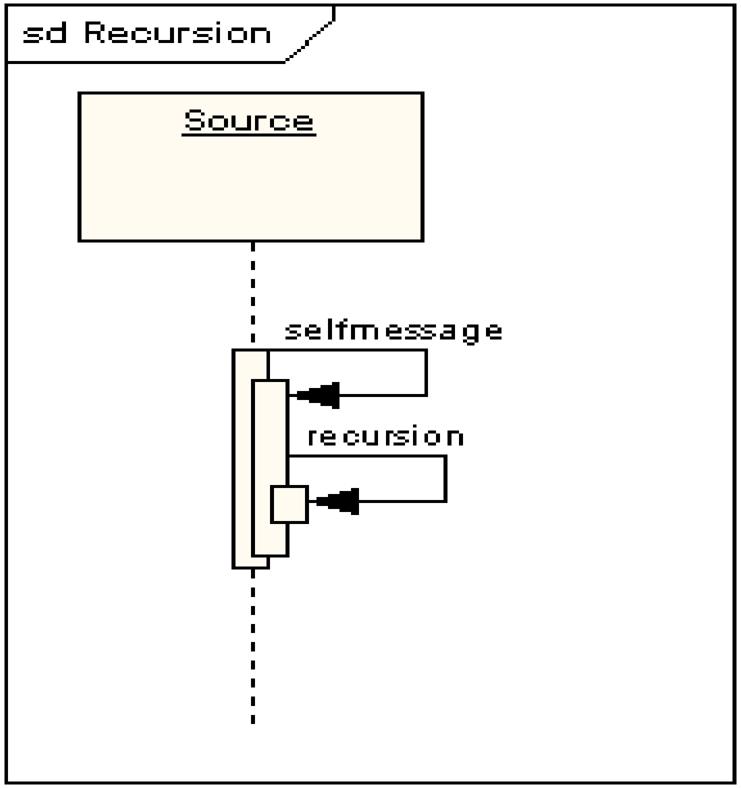
More on Messages

- Messages can be lost or found. Lost messages are those that are either sent but do not arrive at the intended recipient, or which go to a recipient not shown on the current diagram. Found messages are those that arrive from an unknown sender, or from a sender not shown on the current diagram. They are denoted going to or coming from an endpoint element



Self or Recursive Messages

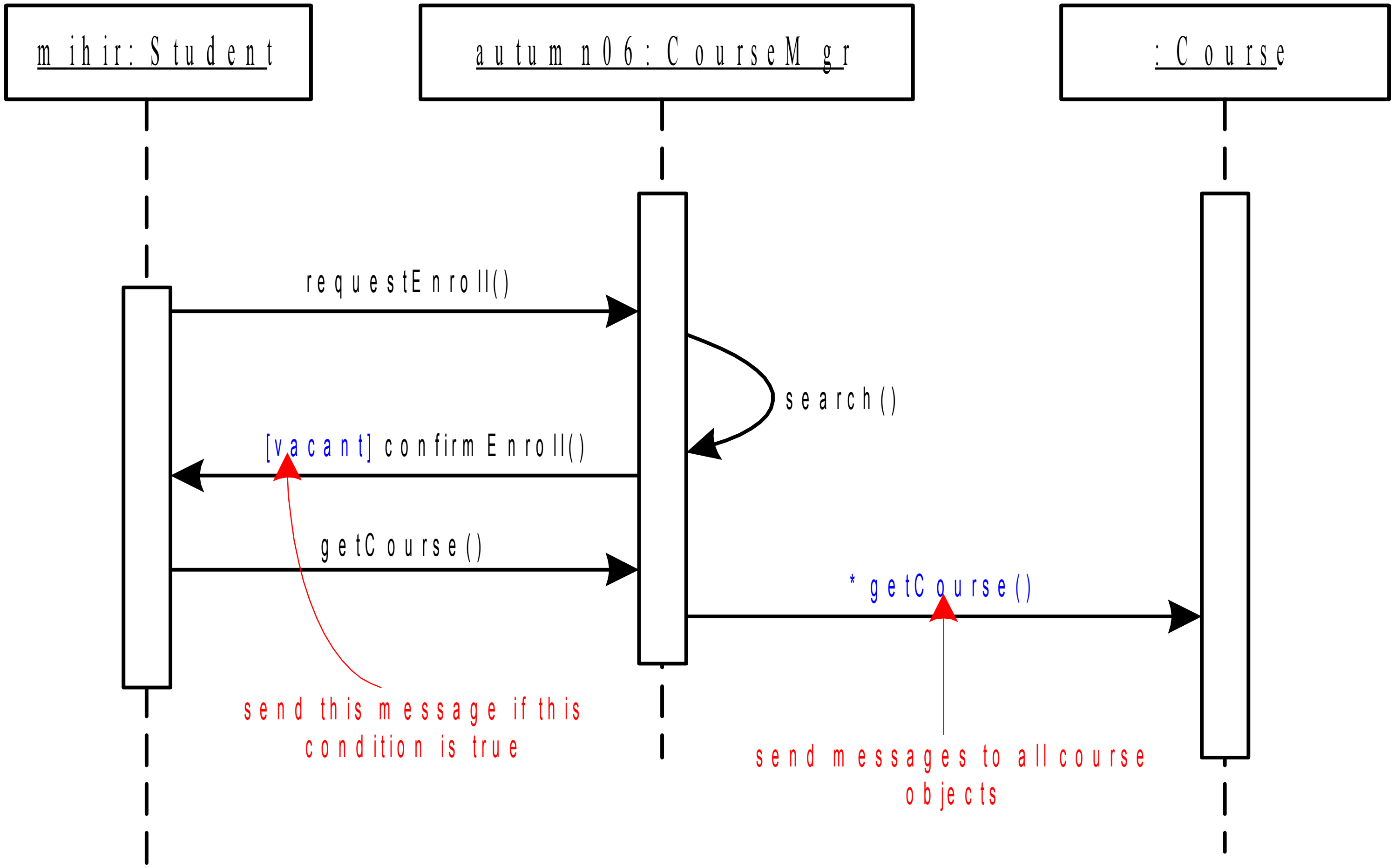
- A self message can represent a recursive call of an operation, or one method calling another method belonging to the same object. It is shown as creating a nested focus of control in the lifeline's execution occurrence



Controlled Messages

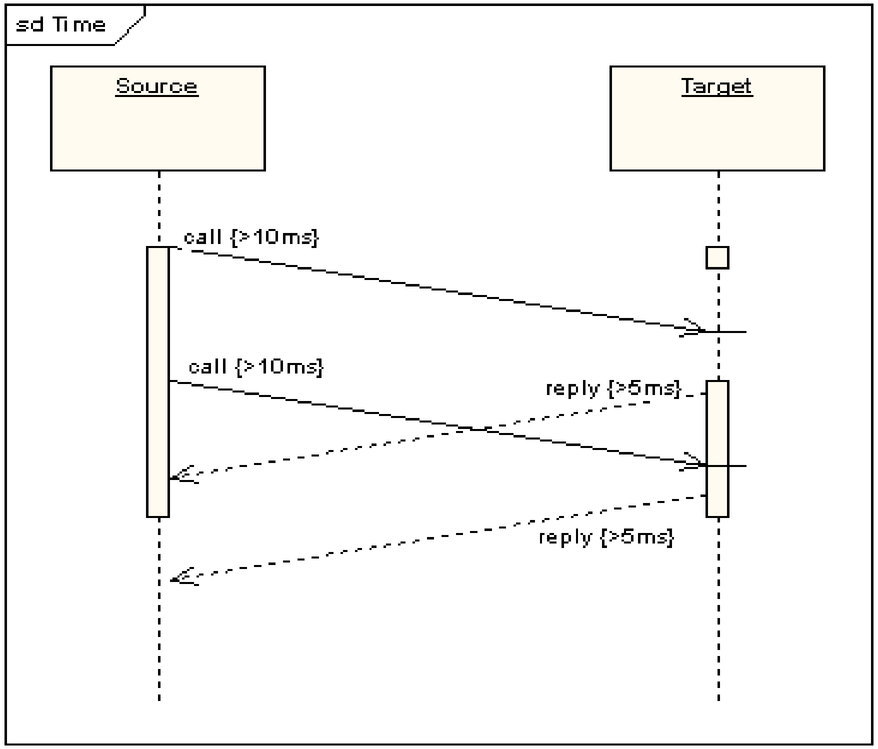
- Some control information can also be included
- Two types of control information are particularly known:
  1. A condition (e.g. [vacant = true]) indicates that a message is sent, only if the condition is true
  2. An iteration marker (*) is used to indicate that the message is to be repeated many times to multiple receiver objects (e.g. when it is required to iterate over a collection or an array of elements)

Example: Controlled Message



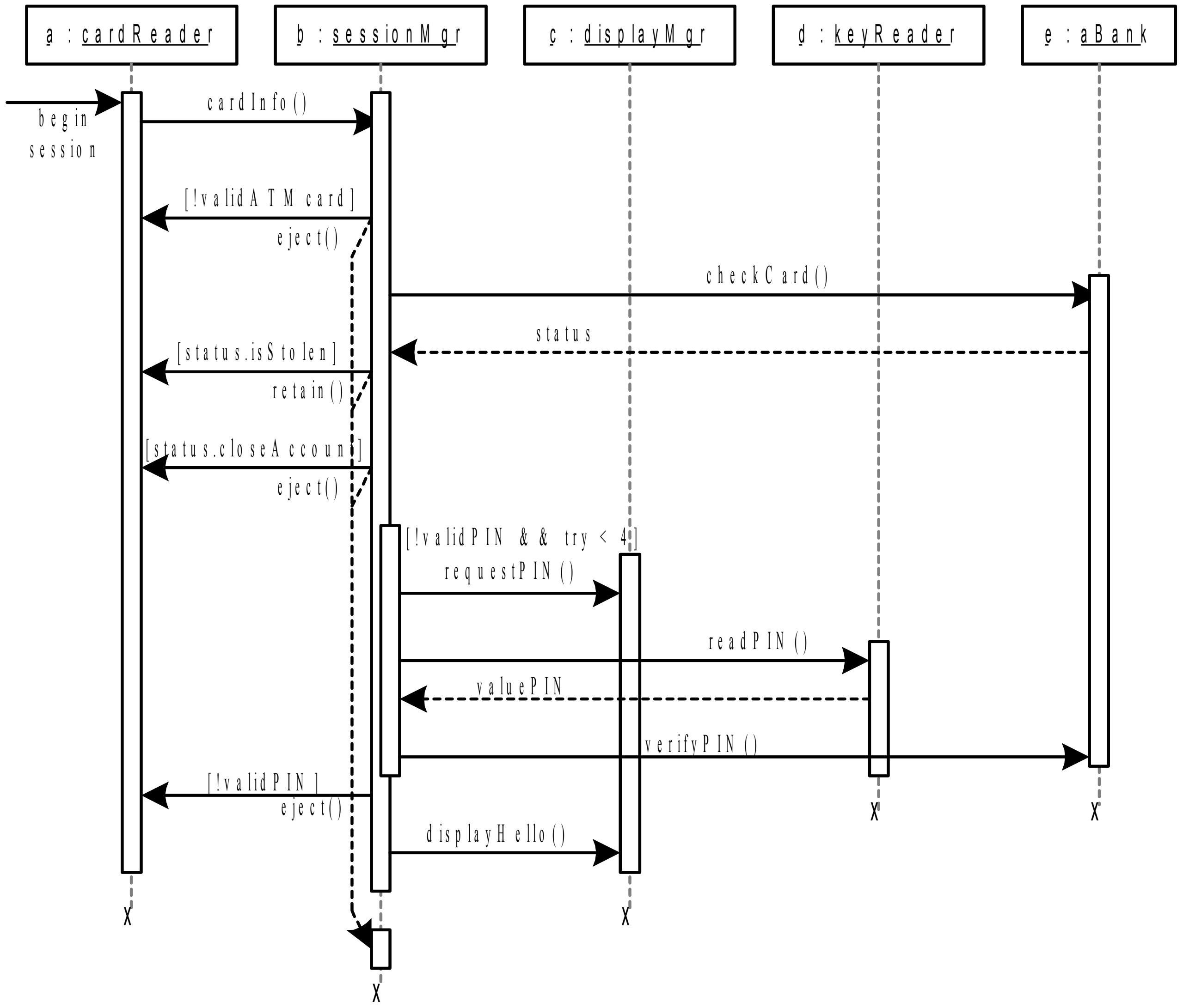
Duration and Timing Constraints

- By default, a message is shown as a horizontal line. Since the lifeline represents the passage of time down the screen, when modeling a real-time system, or even a time-bound business process, it can be important to consider the length of time it takes to perform actions. By setting a duration constraint for a message, the message will be shown as a sloping line



Few Examples

ATM PIN Validation

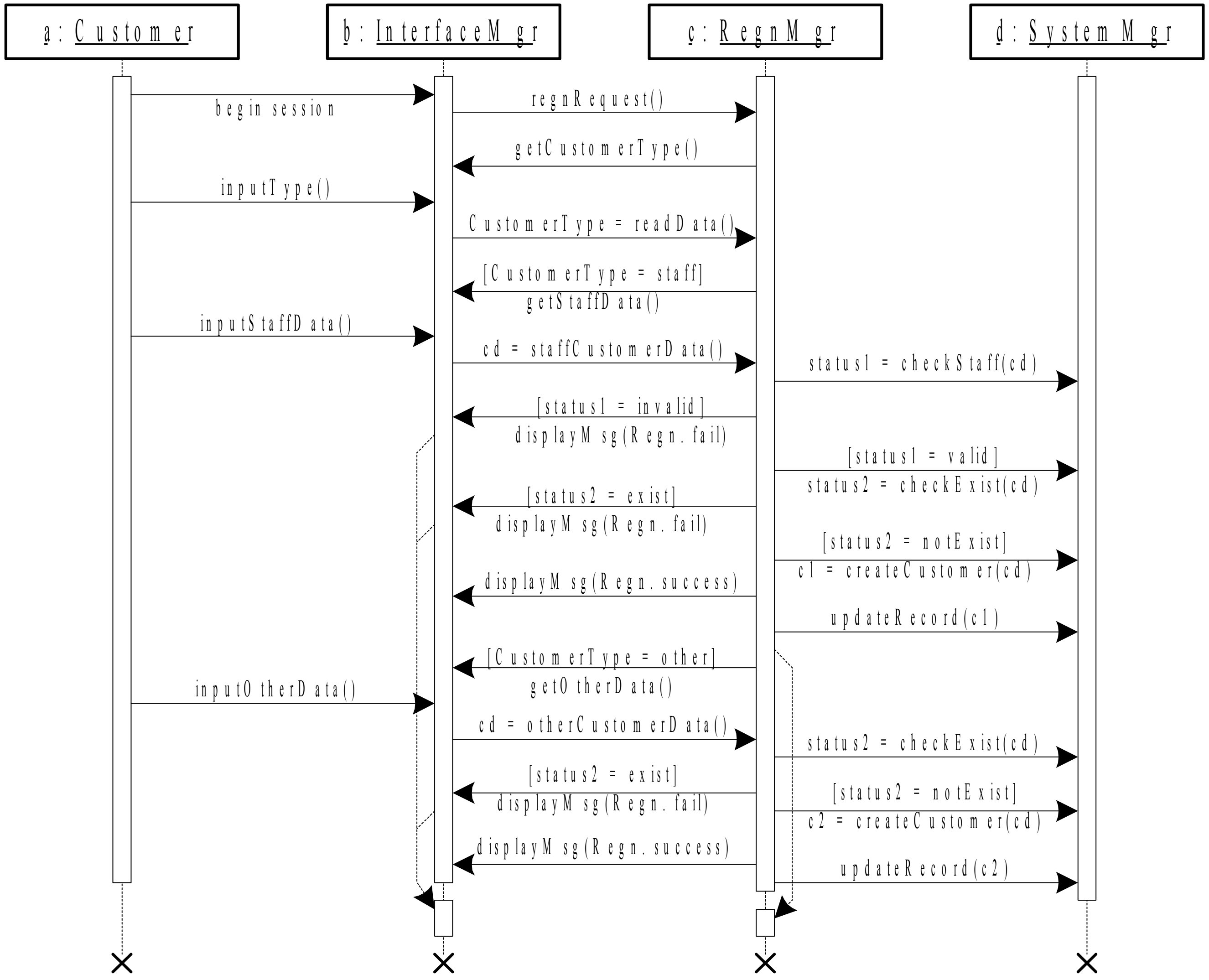


Use Case Registration of OLP

Use case: Registration

- *Scenario 1: Customer is a staff member*
  - Select customer type as staff.
  - Get data for a customer as staff.
  - Check the validity of the staff customer.
  - *Alternative 1.1: Disqualify the validity of a staff*
    - Message “Registration fail”.
  - *Alternative 1.2: Qualify the validity of a staff*
    - Check for already registered customer.
    - *Alternative 1.2.1: Registration exist*
      - Message “Registration fail”.
    - *Alternative 1.2.1: Registration does not exist*
      - Message “Registration successful”.
      - Create a new customer *c*.
      - Update record with *c*.
- *Scenario 2: Customer is other than staff*
  - Select customer type as other.
  - Get data for a customer as other.
  - Check for already registered customer.
  - *Alternative 2.1: Registration exist*
    - Message “Registration fail”.
  - *Alternative 2.2: Registration does not exist*
    - Message “Registration successful”.
    - Create a new customer *c*.
    - *Update record with* c.

Sequence Diagram of Registration



Use Case Place Order in OLP

Use case: Order Items

```
Scenario 1: Option is new
   Prompt for “Registration”
   Call “Registration”
   Display registration status
   Exit

Scenario 2:Option is login
   Call “Check In”
   Alternative 2.1: Login is valid
         Prompt for “Item Details”
         Call “Create Order”
         Display order status
      Exit
   Alternative 2.2: Login is invalid
         Display login fail
         Exit
```

Process Order in OLP System

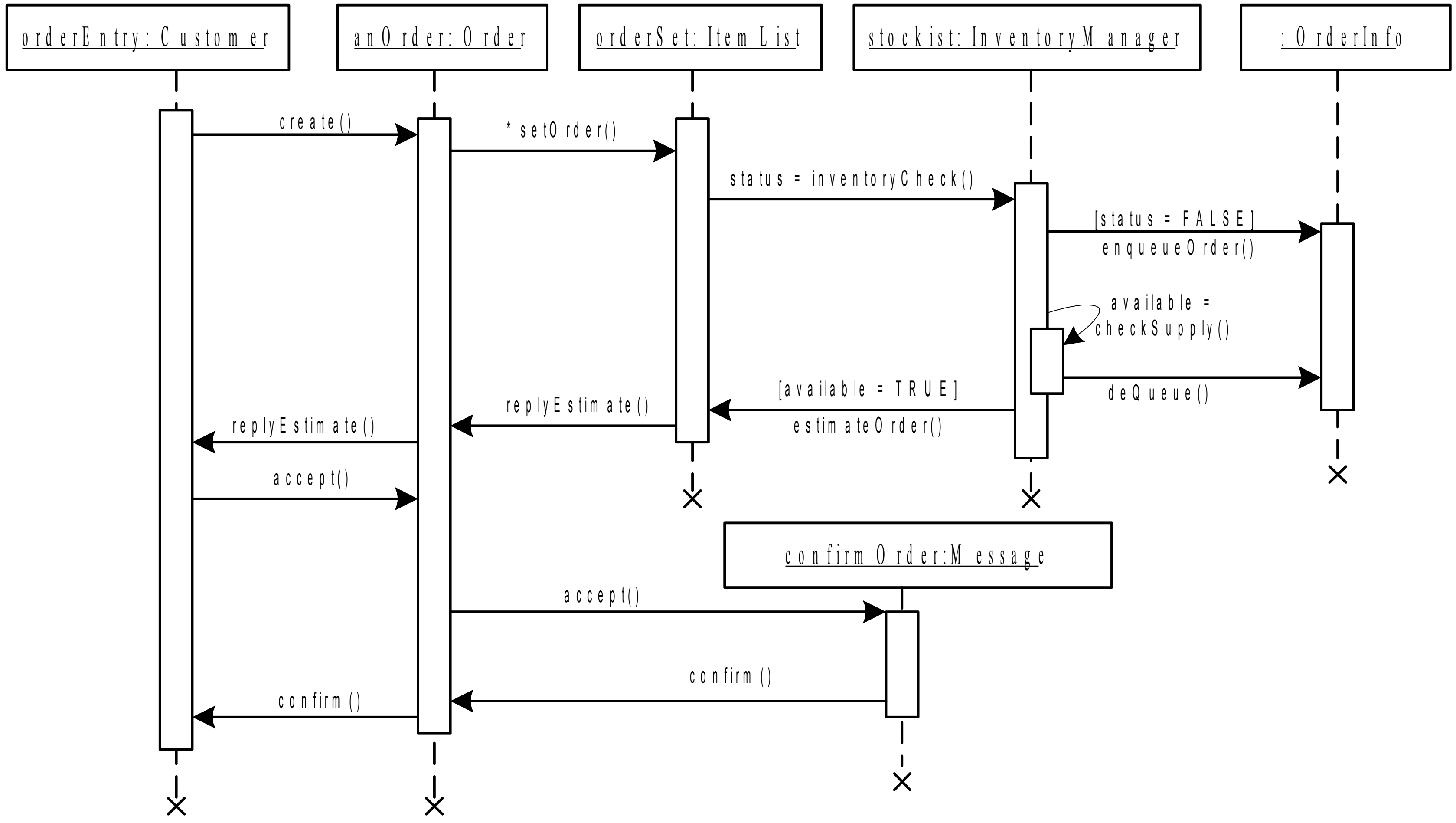
- To illustrate the drawing of a sequence diagram, let us consider the use case "Process Order " in the OLP system
- The "Process Order" use case is proposed to have a following behavior (or scenario)
  - <u>orderEntry: Window</u>
    - this object will get an order from a customer
  - <u>anOrder: Order</u>
    - receives an order from a customer (via orderEntry object)
  - <u>orderSet: ItemList</u>
    - an object is a list of items is to be processed

Sequence Diagram in OLP System

- stockist: InventoryManage
  - object responsible for checking stock, supply stock, request for inventory etc.
- :OrderInfo
  - containing the orders information in a queue
- confirmMessage: Message
  - message objects for sending confirmation message

- The sequence diagram for Process Order use case can be drawn as follows.

Sequence Diagram of Process Order use case in OLP System



Use of Sequence Diagram

- From the sequence diagram of “Process Order” use case, it is evident that the diagram is easy to understand and has immediate appeal. This is the great advantage of the sequence diagram
- However, in some situation, there may be a lot of small methods in different classes, and at times it can be very tricky to figure out the overall sequence of behaviors. in fact, so many details can be resolved during coding only
- The development of sequence diagram would help a designer in determining the responsibilities of the different classes. i.e. what methods should be supported by each class, sequence of message passing etc.

Problems to Ponder

- Obtain sequence diagrams for the following
  - Heap sort
  - Binary Search
  - Dinning Philosophers problem
  - Tower of Hanoi problem
  - "Login Verification" procedure
  - All use cases in OLP