




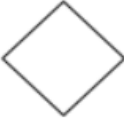
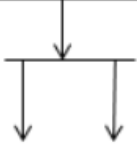


UML diagram types that are used for system modeling

- **Activity diagrams**, which show the activities involved in a process or in data processing .
- **Use case diagrams**, which show the interactions between a system and its environment.
- **Sequence diagrams**, which show interactions between actors and the system and between system components.
- **Class diagrams**, which show the object classes in the system and the associations between these classes.
- **State diagrams**, which show how the system reacts to internal and external events.

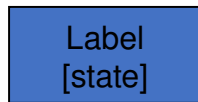
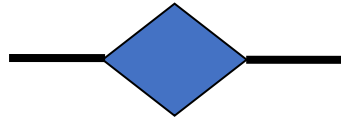
Activity Diagram

- The activity diagram is a flowchart to represent the flow of control among activities in a system.
- The flow of operation can be sequential, branch or concurrent. The activity diagram sometimes considered as the flowchart. Although the diagram looks like a flowchart ,they are not.
- can represent (multiple) flows directly with the Activity diagram
- Activity diagram is used to
 - Draw the activity flow of a system.
 - Describe the sequence from one activity to another.
 - Describe the parallel, branched and concurrent flow of the system.

Notations of Activity Diagram

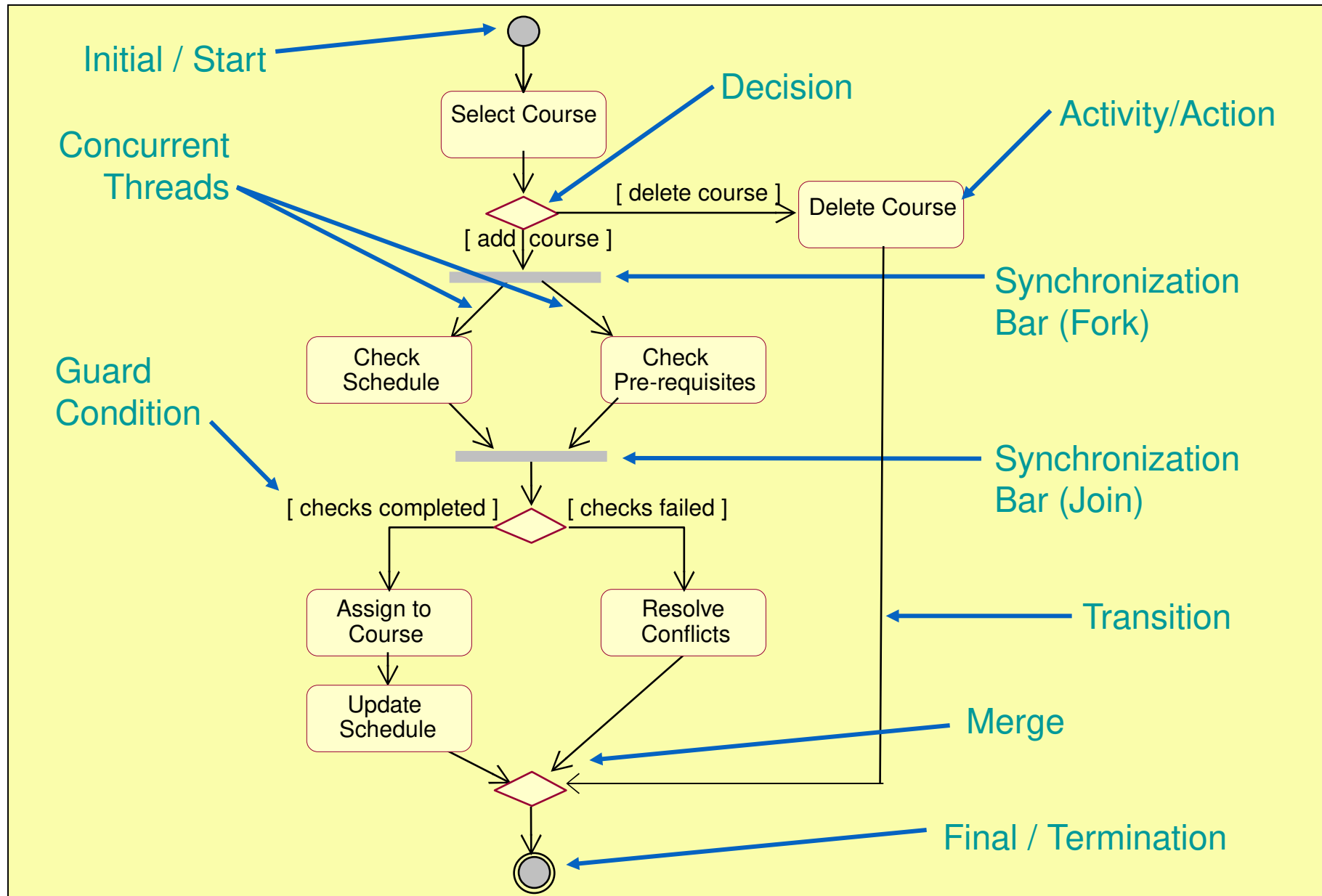
Sr. No	Name	Symbol
1.	Start Node	
2.	Action State	
3.	Control Flow	
4.	Decision Node	
5.	Fork	
6.	Join	
7.	End State	

Activity Diagrams

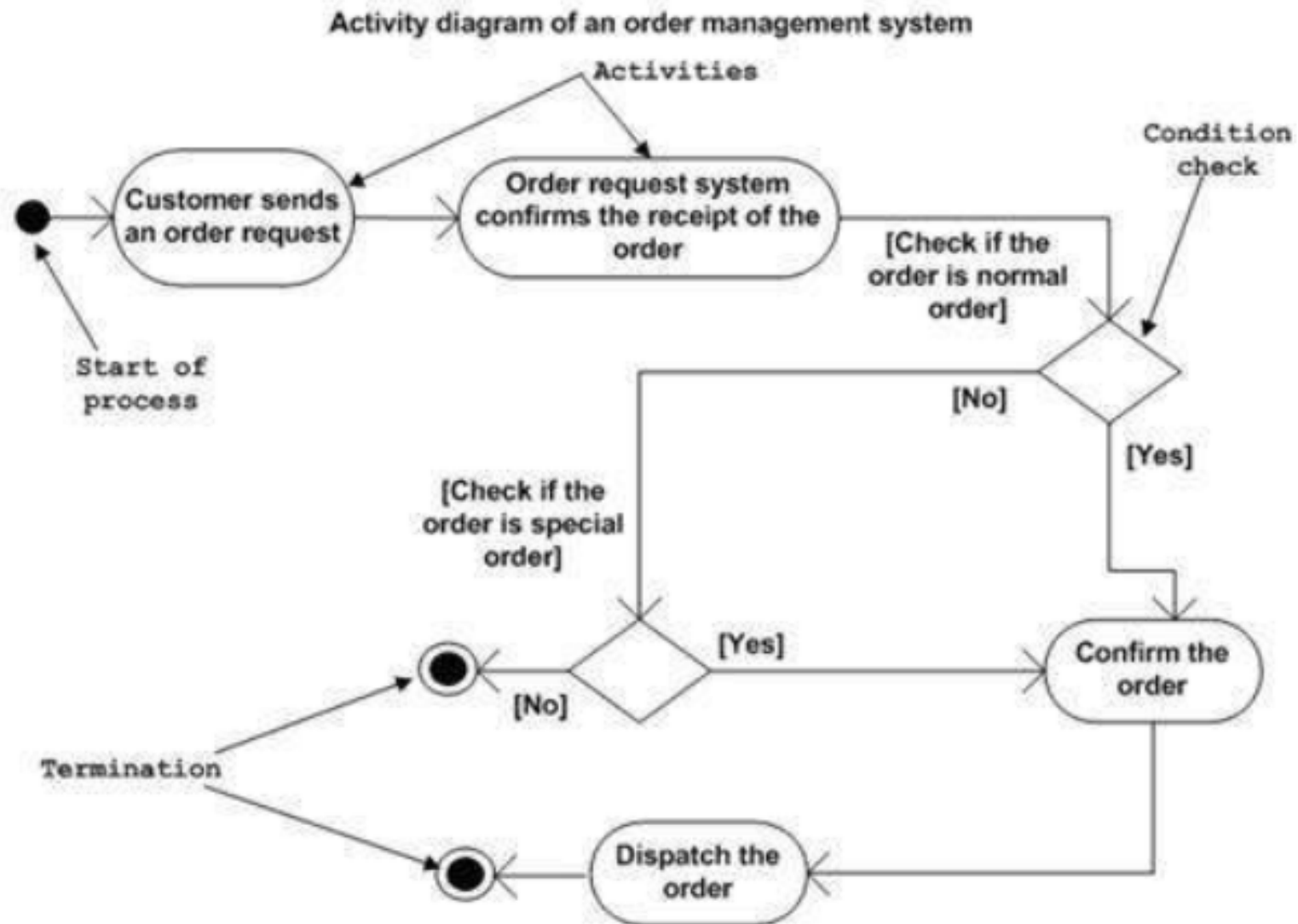


- Summary of Notation:
- **Action/Activity state** – Action states cannot be decomposed, Activity states may be (UML 1.5 - as of UML 2.0 replaced with *Activity frames*)
- **Transition** – control flow; a transition is triggered upon completion of some activity
- **Decision/Merge point** – standard if-else style logic; also supports iteration. Guard conditions indicated in brackets in each transition.
- **Object node** – may be (typically not) included to show where an object's state may change.
- **Synchronization bar** – supports fork/join semantics for concurrent processing
- **Swimlanes** – partition by responsibility, not thread

Example: Activity Diagram

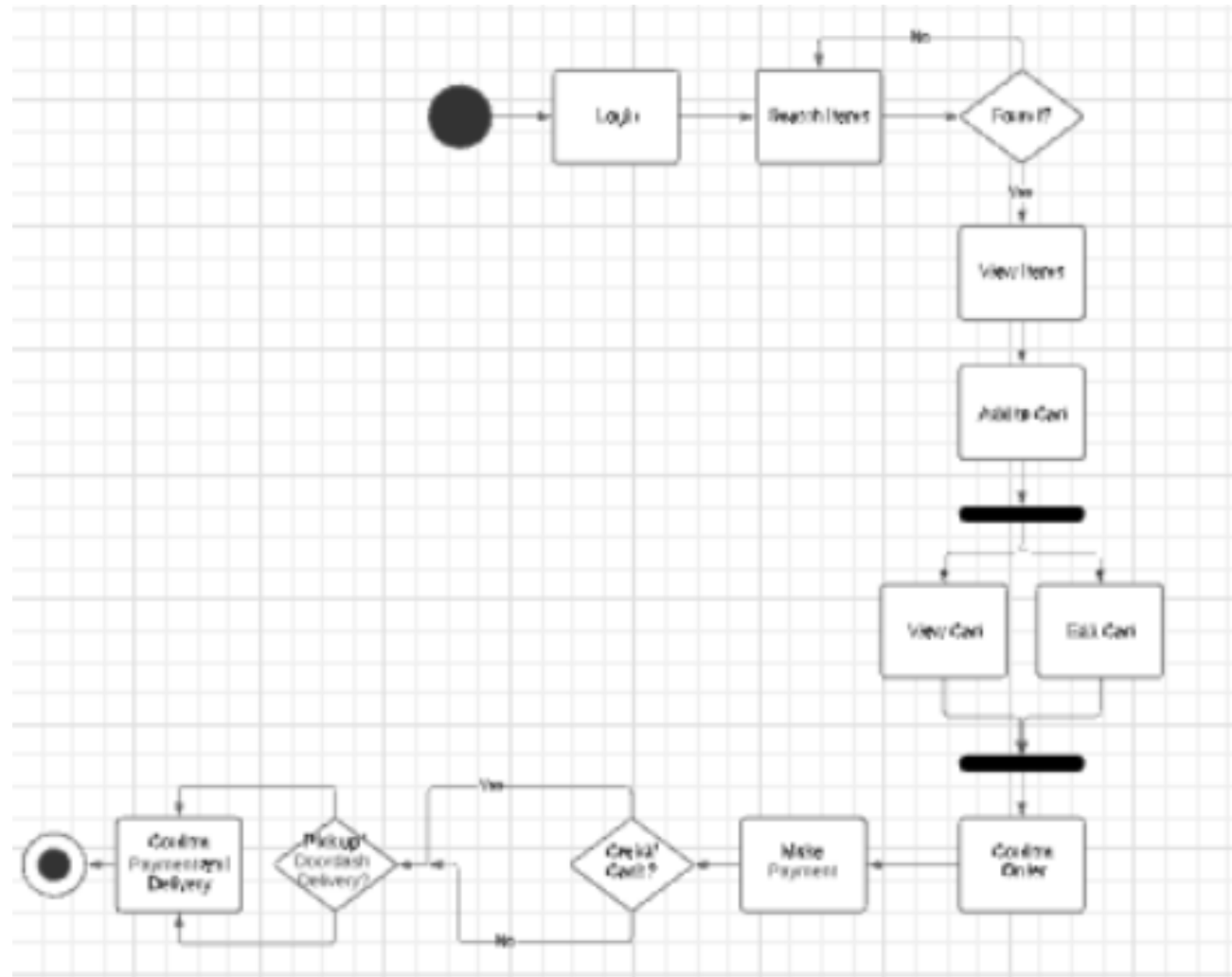


Activity diagram of order management system

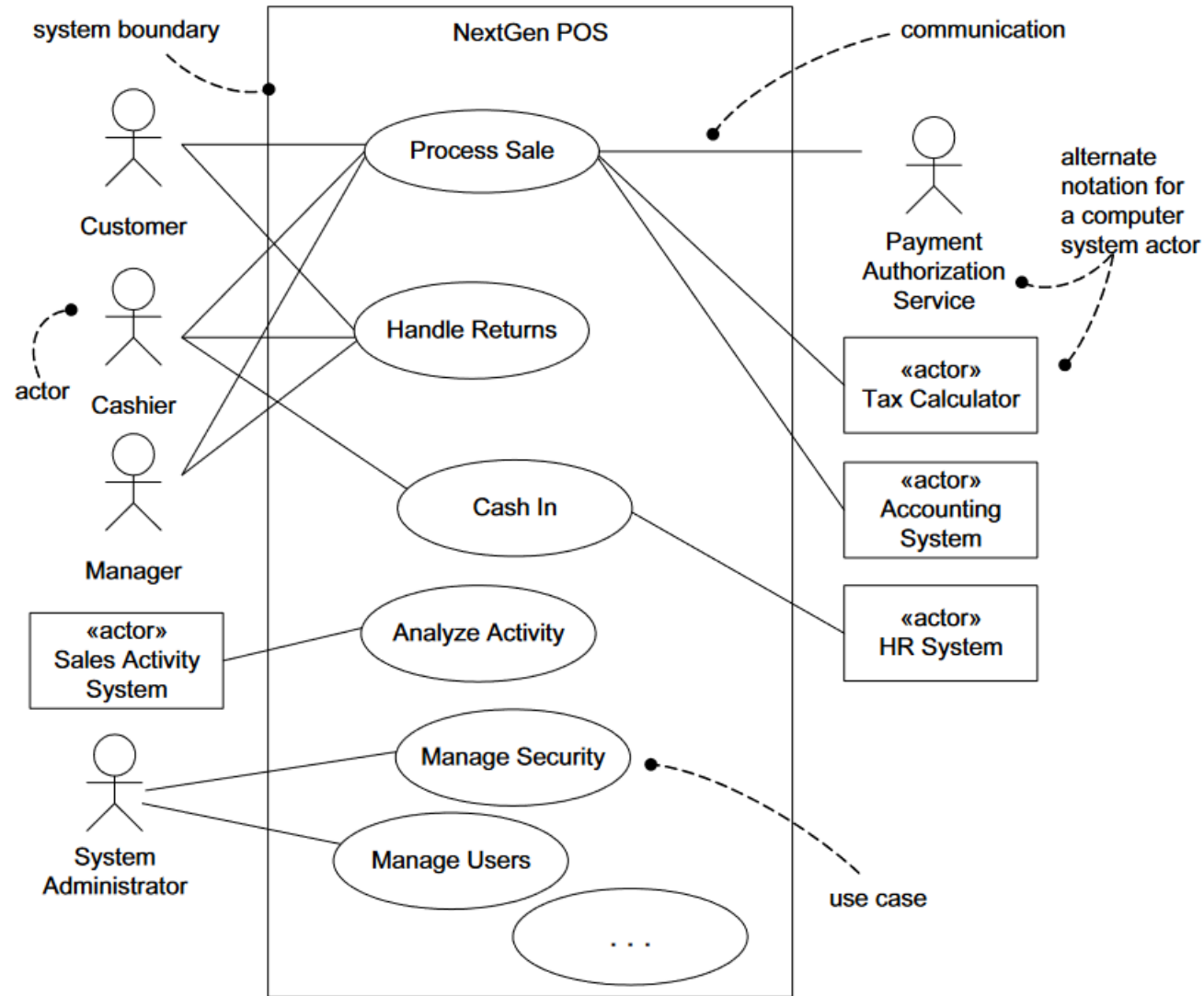


Example:

- While doing online food processing following activities are taken place:
 - User login
 - Search food
 - View food choices
 - Add to cart(view/edit cart)
 - Confirm order
 - Make payment(cash,credit/debit card)
 - Select delivery(doordash,pickup)
 - Confirmation message
 - end



Use case diagram of NextGen POS



Sequence diagrams

- Sequence diagram shows the order in which interactions take place. The diagram is used to depict the interaction between several objects in a system.
- Sequence diagrams are part of the UML and are used to model the interactions between the actors and the objects within a system.
- A sequence diagram shows the sequence of interactions that take place during a particular use case or use case instance.
- The objects and actors involved are listed along the top of the diagram, with a dotted line drawn vertically from these.
- Interactions between objects are indicated by annotated arrows.

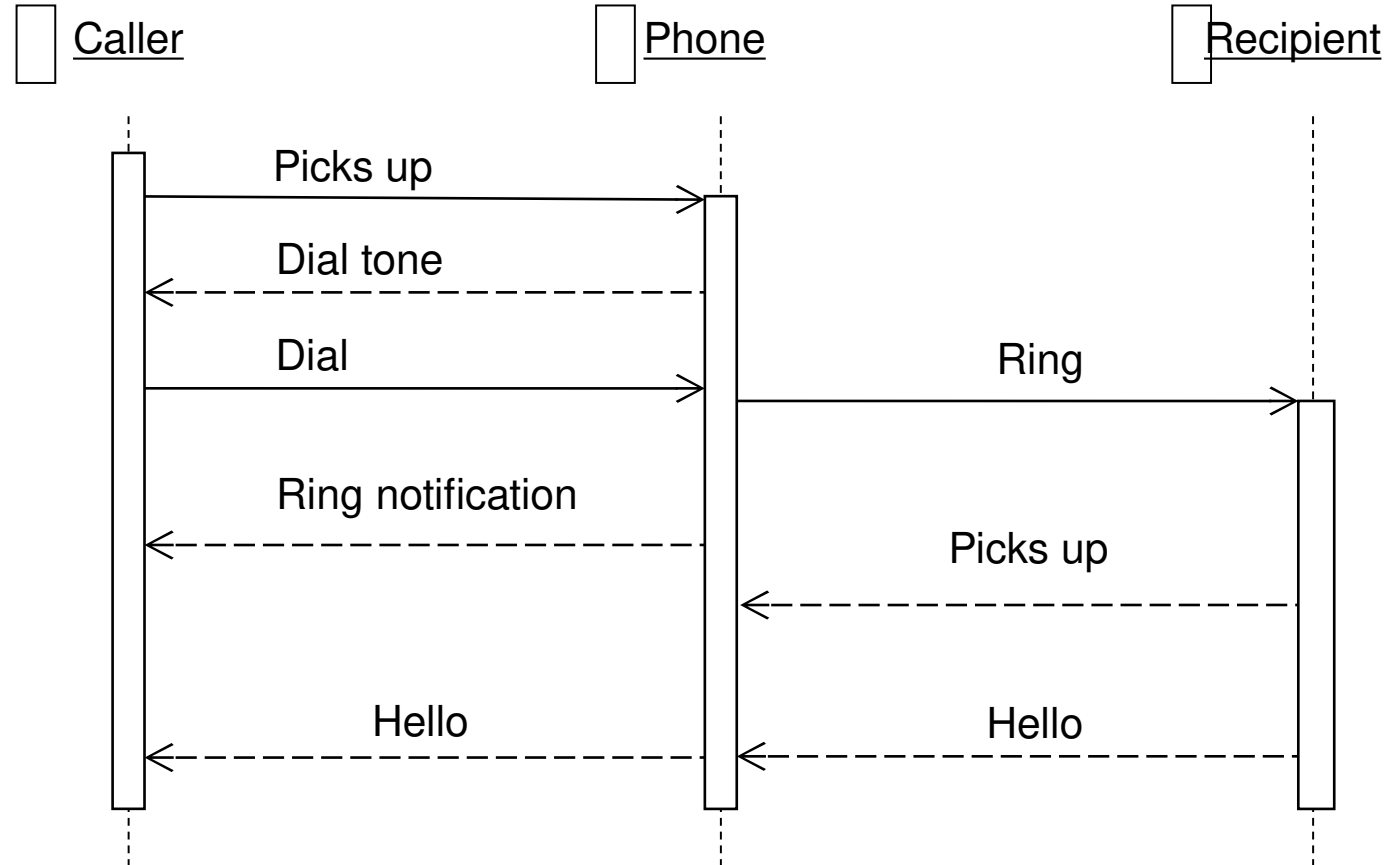
Sequence Diagram

- Describe the flow of messages, events, actions between objects
- Show concurrent processes and activations
- Show time sequences that are not easily depicted in other diagrams
- Typically used during analysis and design to document and understand the logical flow of your system
- **Emphasis on Time Ordering**

Sequence Diagram Key Parts

- **participant**: object or entity that acts in the diagram
 - diagram starts with an unattached "found message" arrow
- **message**: communication between participant objects
- the **axes** in a sequence diagram:
 - **horizontal**: which object/participant is acting
 - **vertical**: time (down -> forward in time)

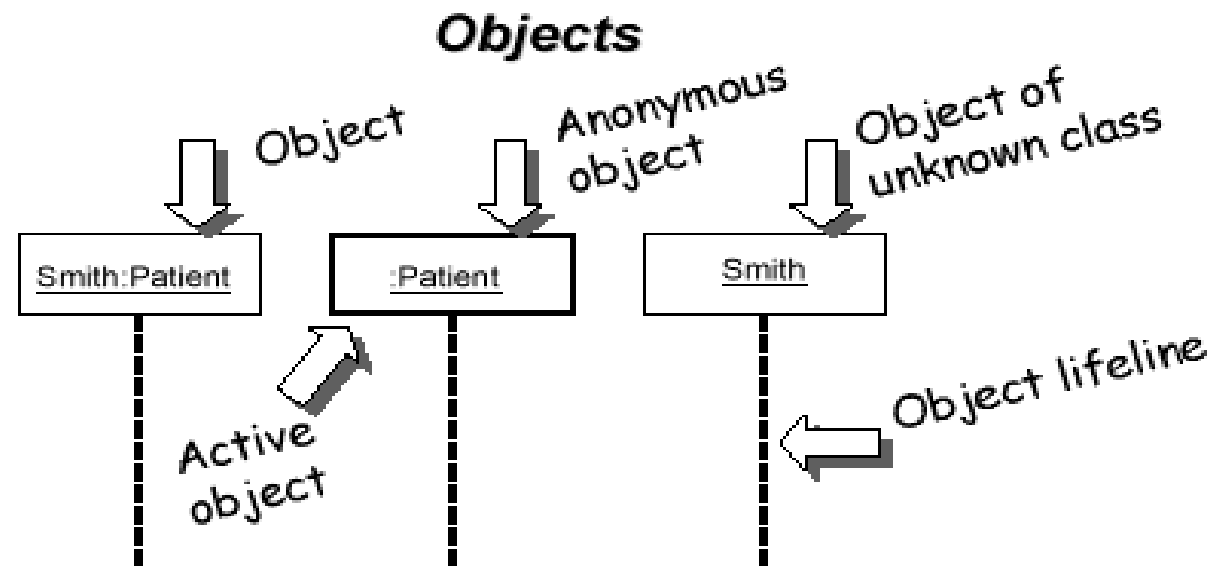
Sequence Diagram (make a phone call)



Representing Objects

Squares with object type, optionally preceded by "*name* :"

- write object's name if it clarifies the diagram
- object's "life line" represented by dashed vert. line

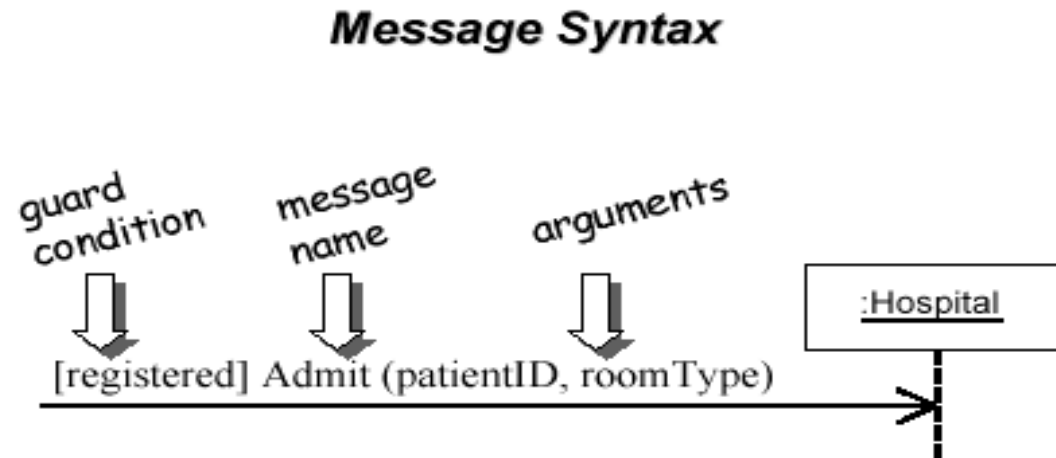


Name syntax: <objectname>:<classname>

Messages Between Objects

messages (method calls) indicated by arrow to other object

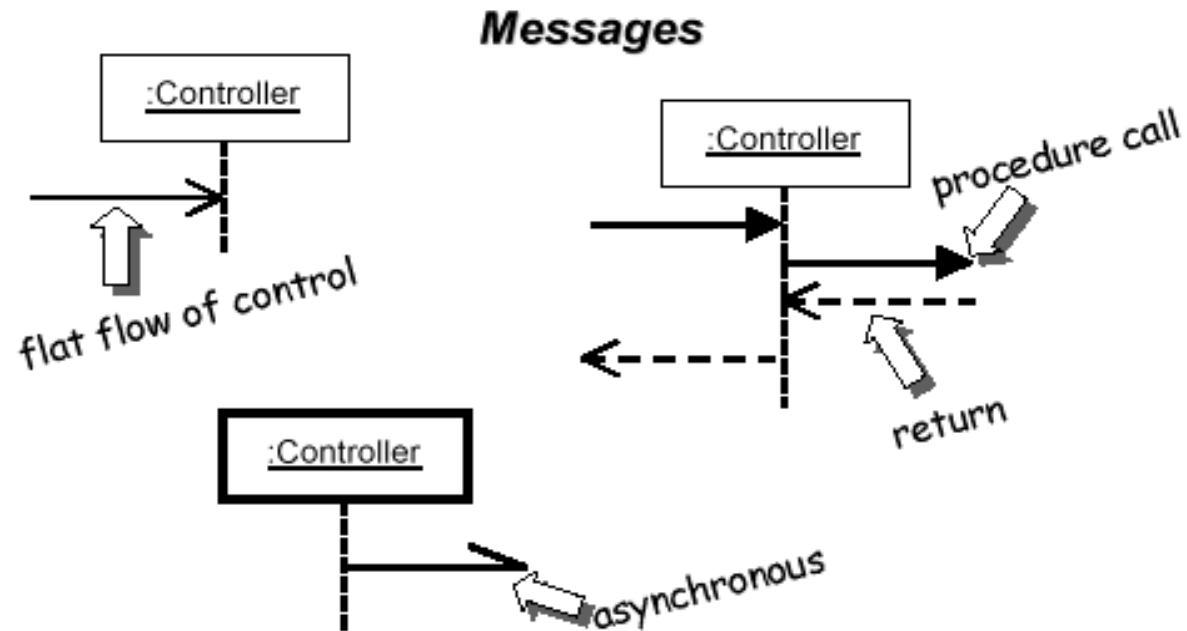
- write message name and arguments above arrow



Messages, continued

messages (method calls) indicated by arrow to other object

- dashed arrow back indicates return
- different arrowheads for normal / concurrent (asynchronous) calls



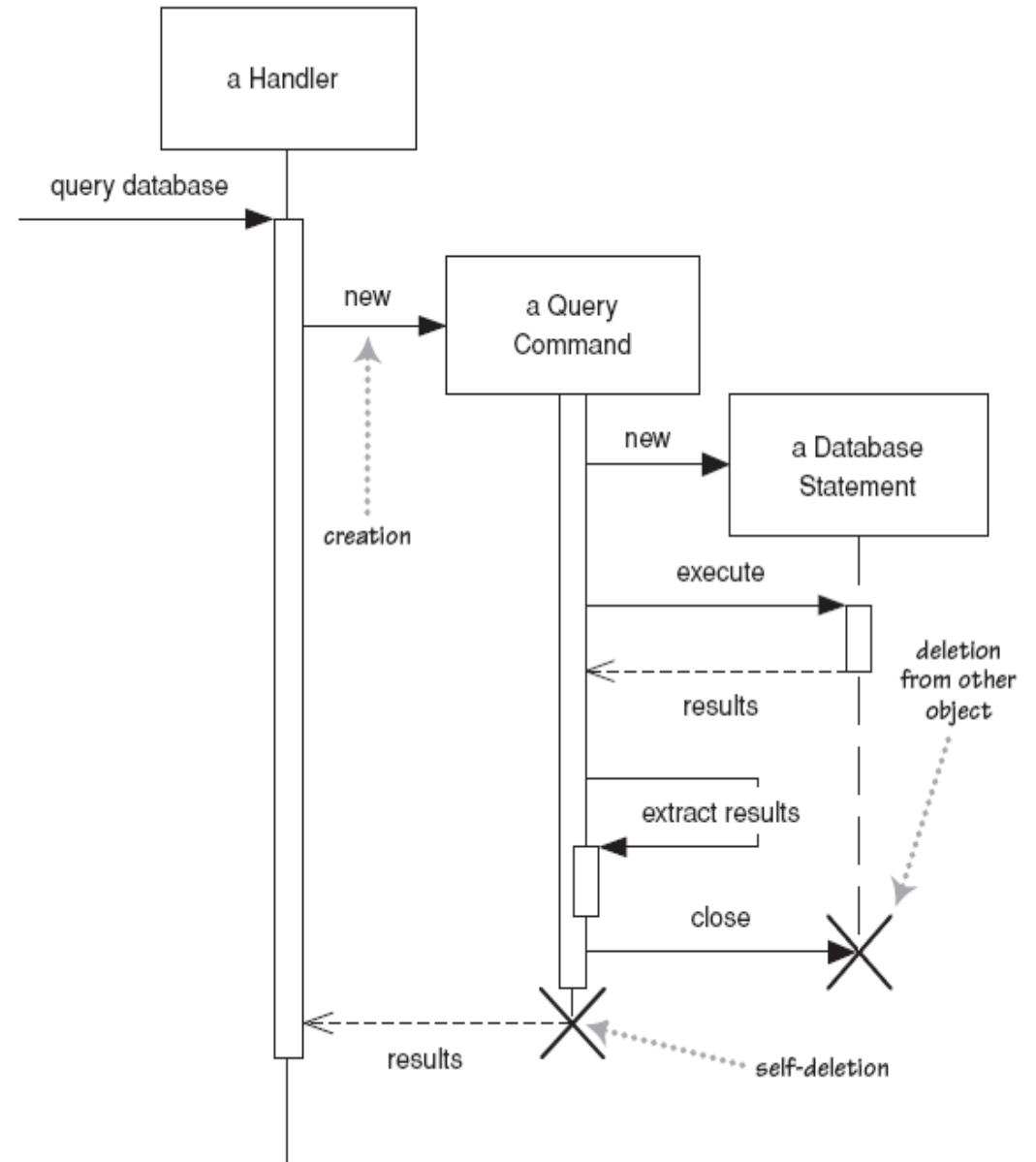
Lifetime of objects

creation: arrow with 'new' written above it

- notice that an object created after the start of the scenario appears lower than the others

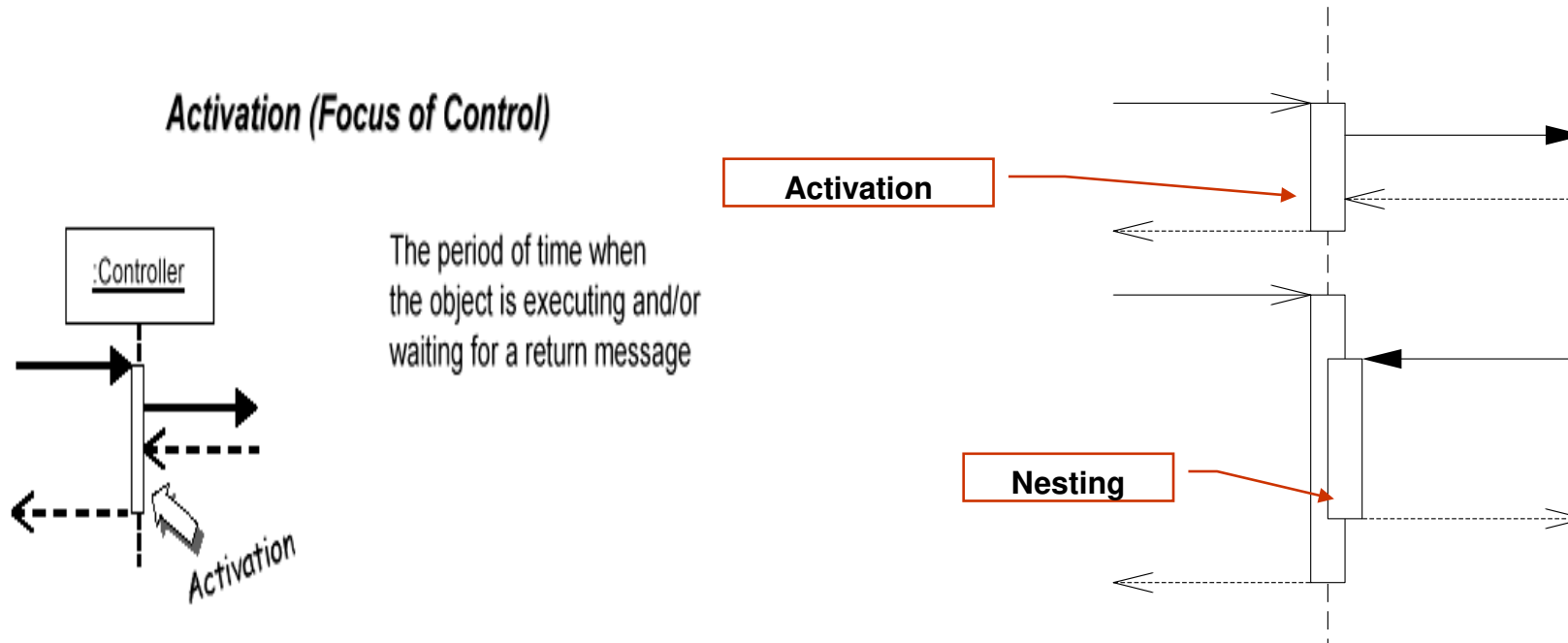
deletion: an X at bottom of object's lifeline

- Java doesn't explicitly delete objects; they fall out of scope and are garbage-collected



Indicating method calls

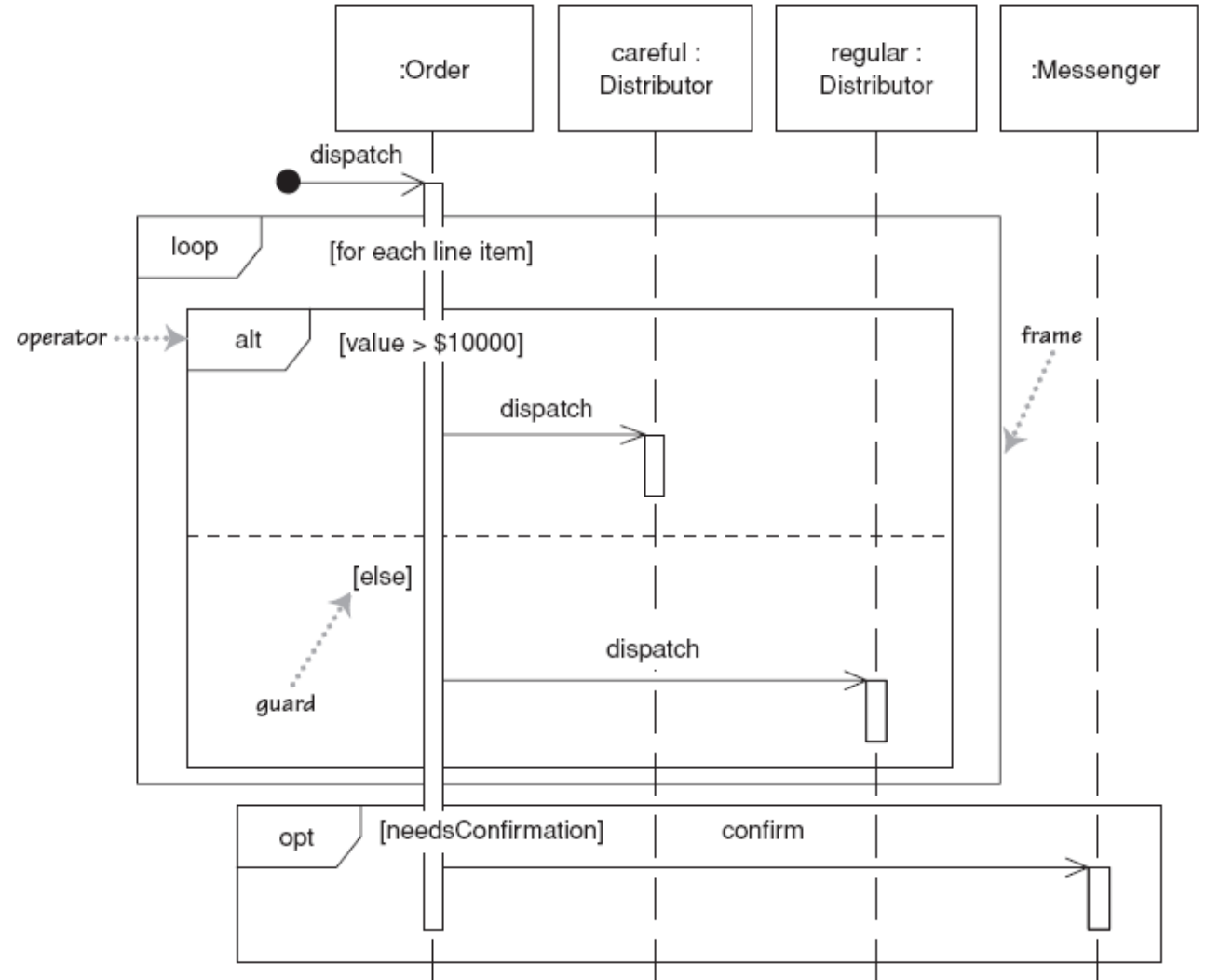
- **activation**: thick box over object's life line; drawn when object's method is on the stack
 - either that object is running its code, or it is on the stack waiting for another object's method to finish
 - nest activations to indicate recursion



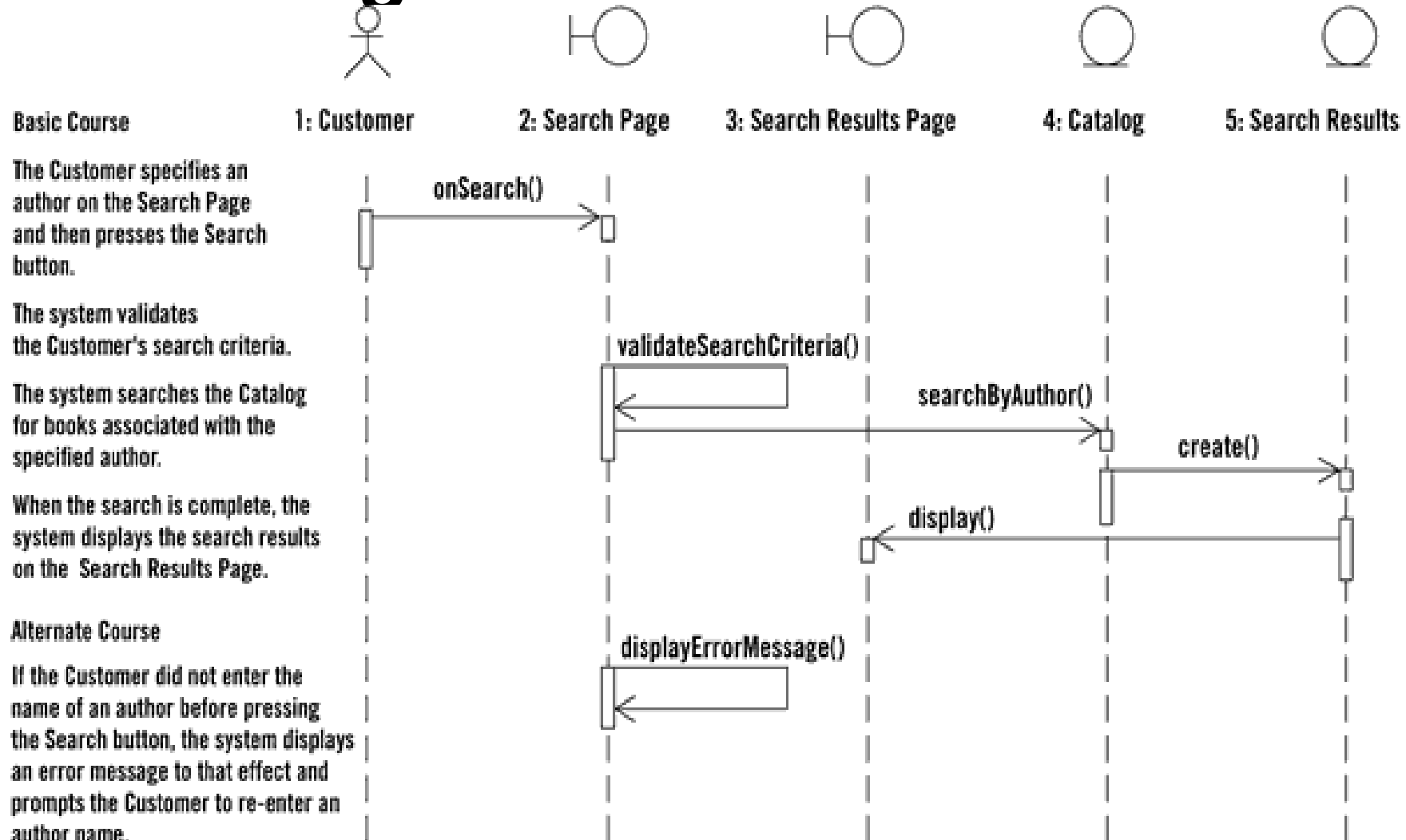
Selection and loop

frame: box around part of diagram to indicate `if` or loop

- `if` -> (opt)
[condition]
- `if/else` -> (alt)
[condition], separated by horizontal dashed line
- `loop` ->
(loop) [condition or items to loop over]

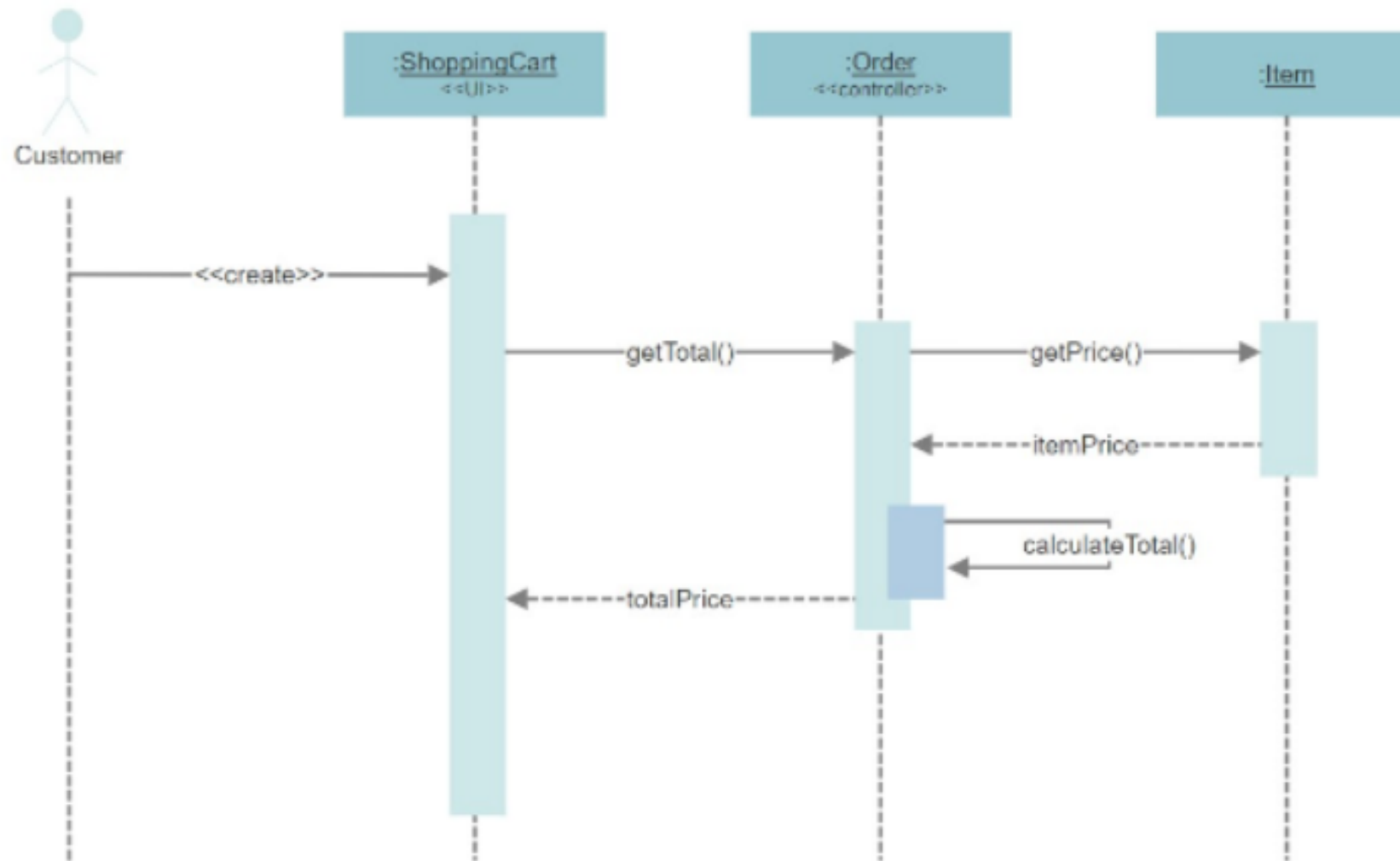


Sequence diagram from use case scenario

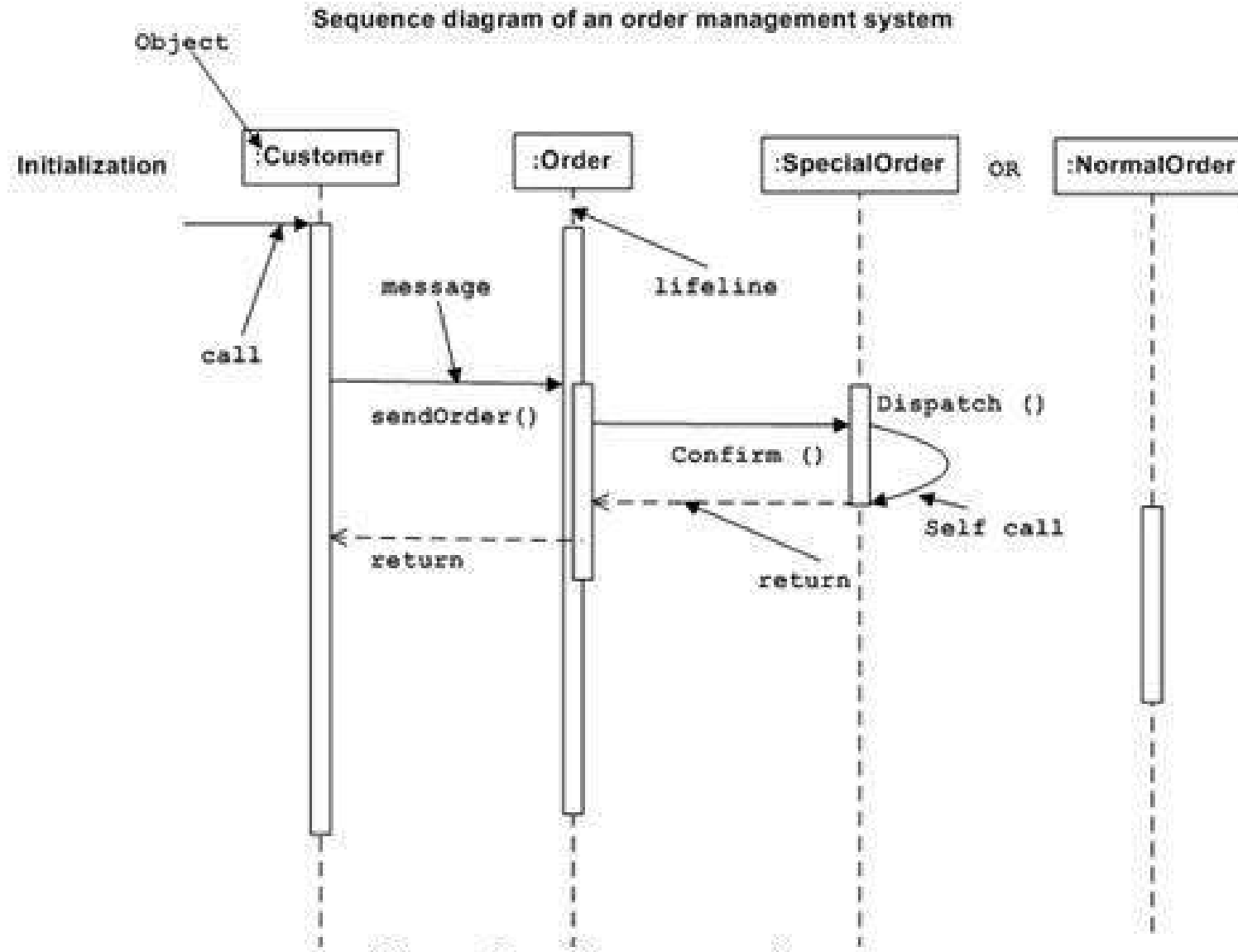


Explanation:

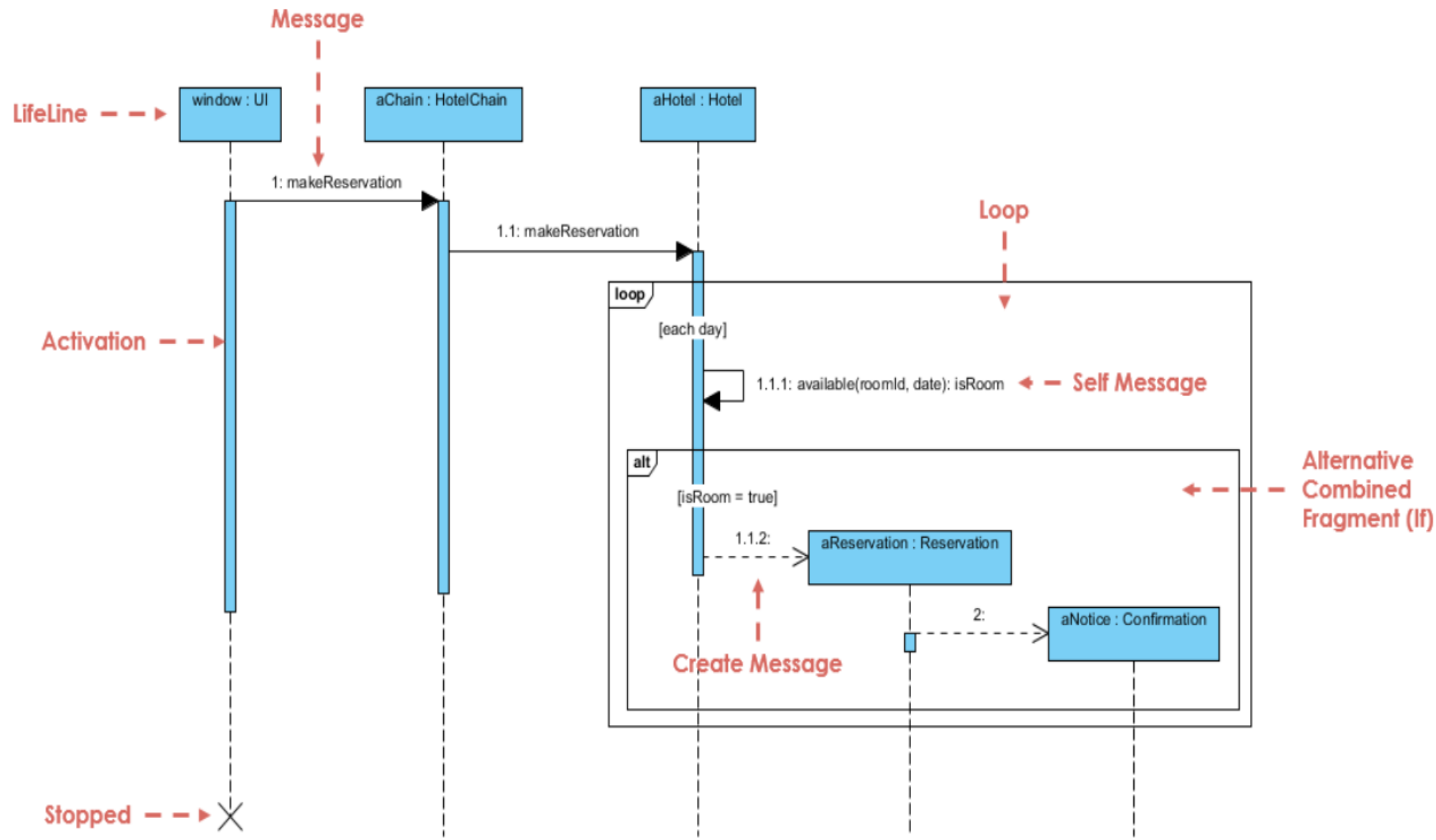
- the customer that logs in to the e-commerce website and adds an item to the shopping cart in order to complete the purchase. Let's take a look at the diagram and try to understand how the task is completed in the system.
- The Shopping cart then asks the Order Controller for the total price of the items in the cart using the `getTotal()` function.
- Once the request has been received, the Order Controller uses the `getPrice()` function for each item to get the item price.
- Once the price for all items in the cart has been received by the Order Controller, the total amount of items in the cart will be calculated using `calculateTotal()` function, which uses the data already available to the Order Controller.
- Once calculated, the `totalPrice` is sent to the Shopping Cart UI so that the customer can proceed with the checkout process.



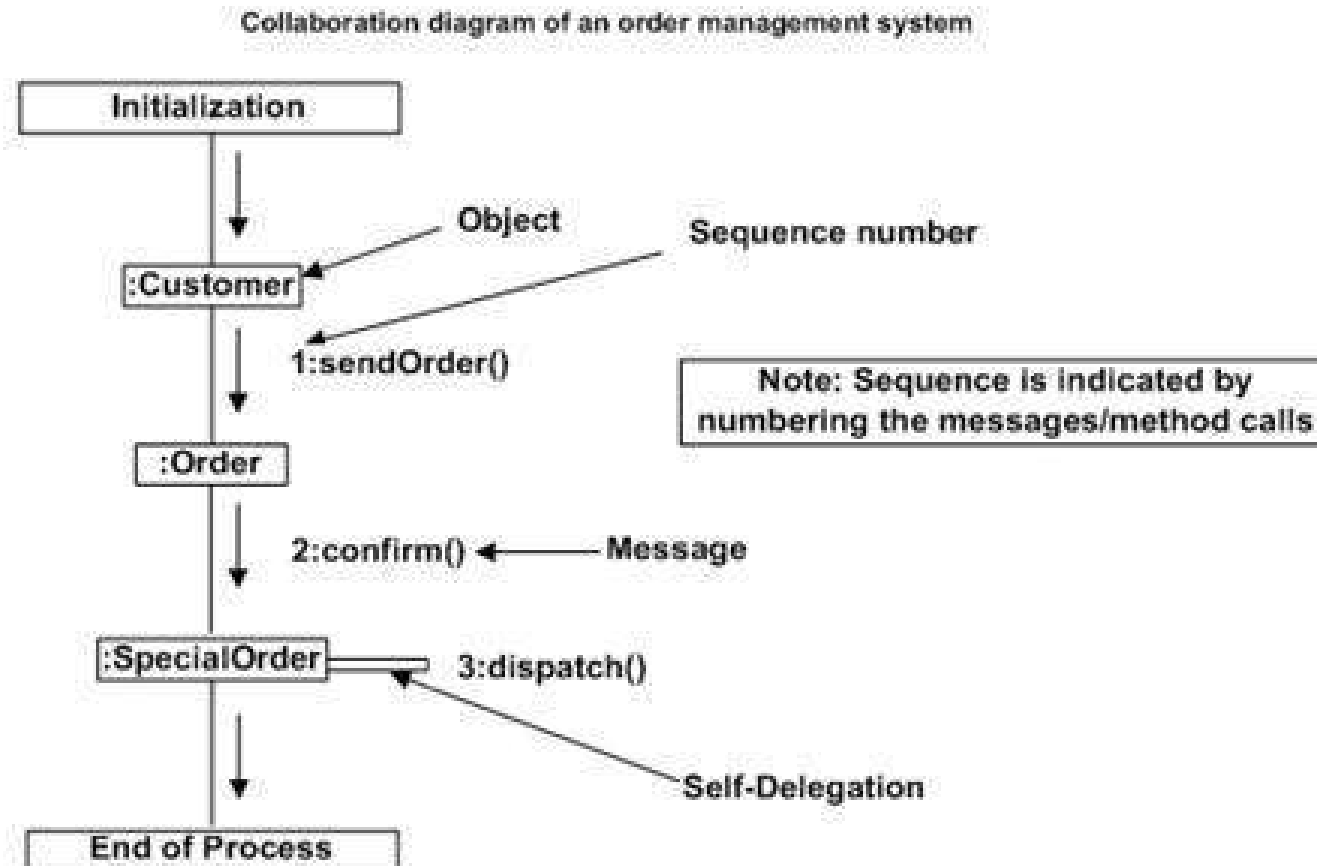
UML Sequence Diagram of Order Management System



Examples:



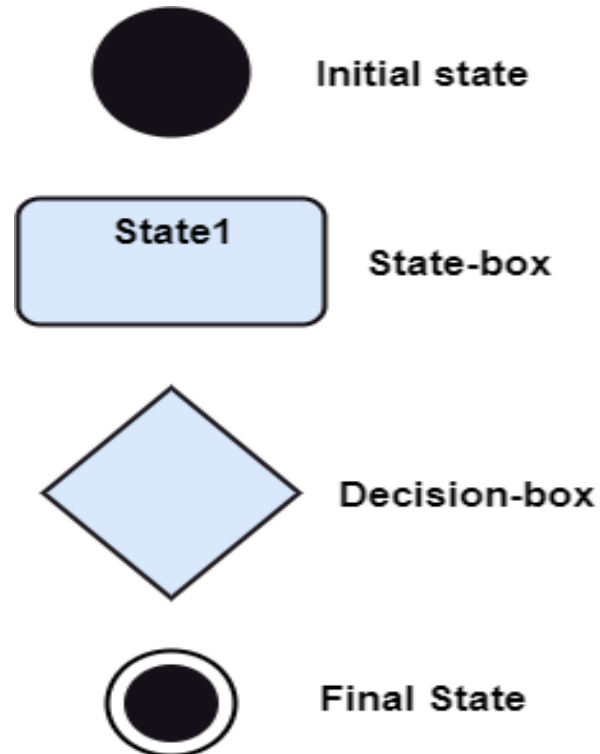
UML Collaboration Diagram of Order Management System



Purposes of State Machines

- To model the dynamic aspect of a system.
- To model the life time of a reactive system.
- To describe different states of an object during its life time.
- Define a state machine to model the states of an object

- **Notations:**

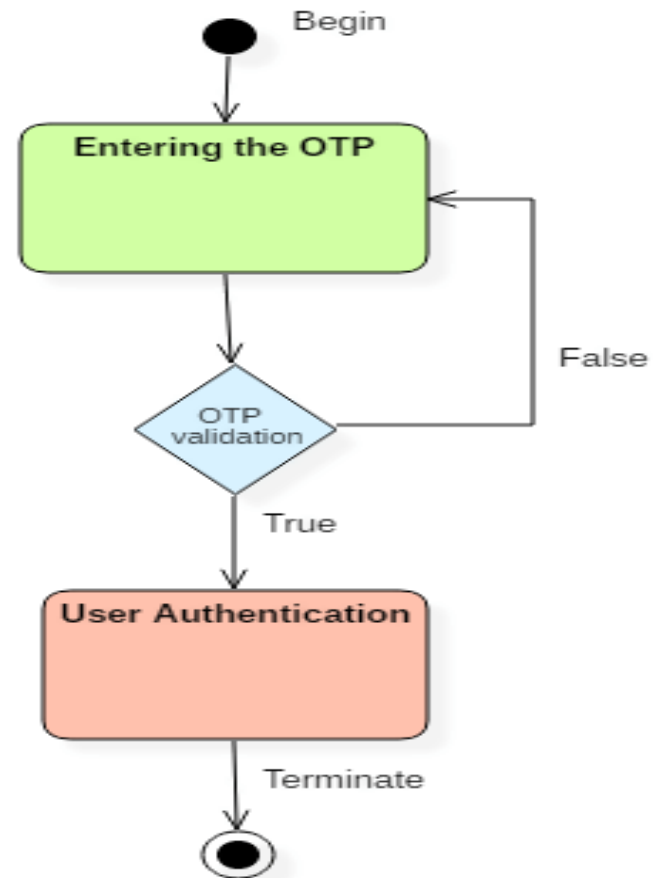


- **Initial state:** It defines the initial state (beginning) of a system, and it is represented by a black filled circle.
- **Final state:** It represents the final state (end) of a system. It is denoted by a filled circle present within a circle.
- **Decision box:** It is of diamond shape that represents the decisions to be made on the basis of an evaluated guard.
- **Transition:** A change of control from one state to another due to the occurrence of some event is termed as a transition. It is represented by an arrow labeled with an event due to which the change has ensued.
- **State box:** It depicts the conditions or circumstances of a particular object of a class at a specific point of time. A rectangle with round corners is used to represent the state box.
- **Types of State**
 - **Simple state:** It does not constitute any substructure.
 - **Composite state:** It consists of nested states (substates), such that it does not contain more than one initial state and one final state. It can be nested to any level.
 - **Submachine state:** The submachine state is semantically identical to the composite state, but it can be reused.

When to use a State Machine Diagram?

- For modeling the object states of a system.
- For modeling the reactive system as it consists of reactive objects.
- For pinpointing the events responsible for state transitions.
- For implementing forward and reverse engineering.

Example:user authentication process



Statechart diagram of an order management system

