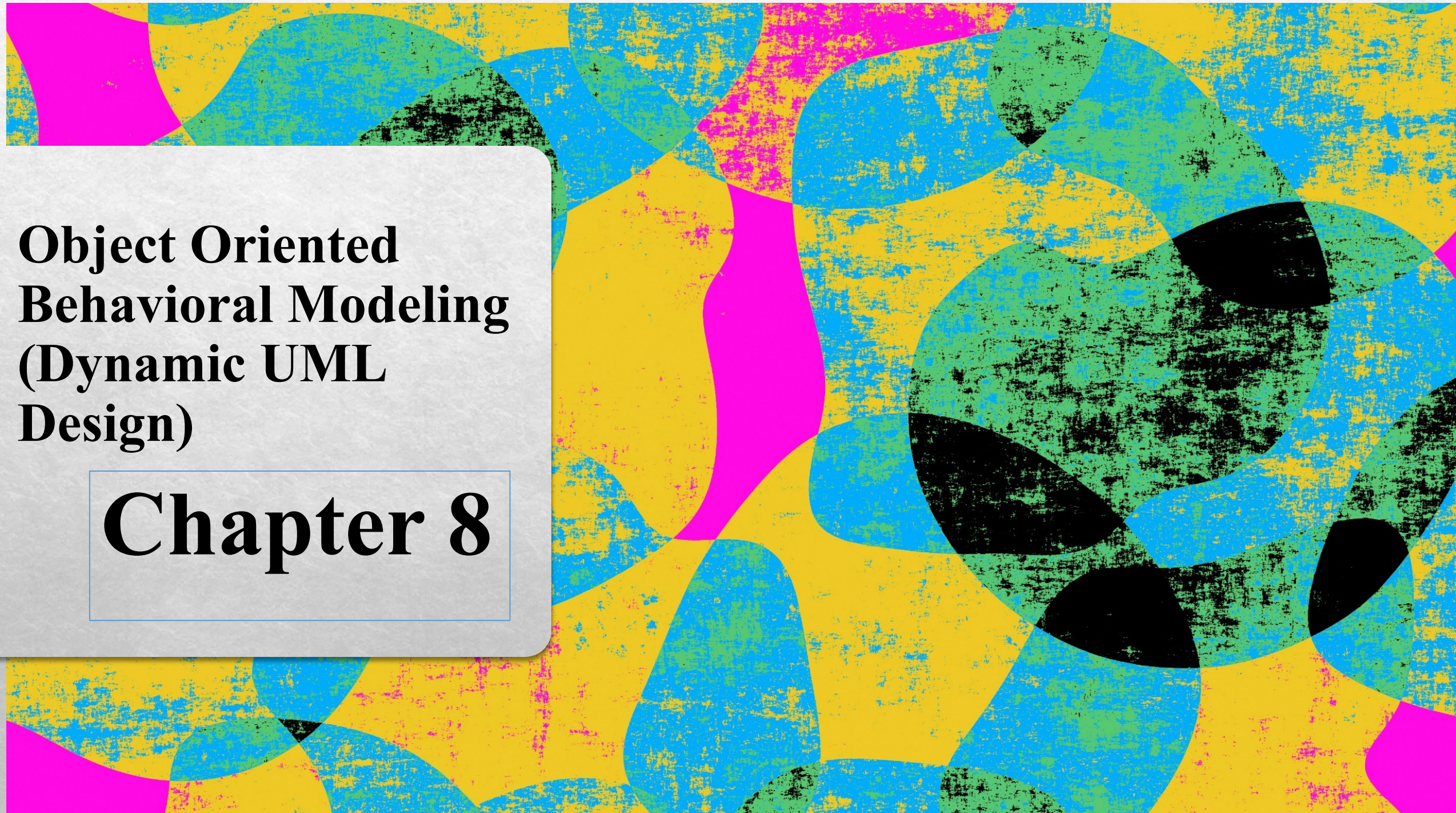


# Object Oriented Behavioral Modeling (Dynamic UML Design)

# Chapter 8

# UML - Behavioral Modeling

- Dynamic modeling refers to the object interactions during runtime.

❑**Behavioral (or Dynamic) view**:

- Emphasizes the dynamic behavior of the system by showing collaborations among objects and changes to the internal states of objects.
- This view includes **sequence diagrams**, **collaboration diagrams** **activity diagrams**, and **state machine diagrams**.

# Behavioral Diagrams

- Used to visualize, specify, construct, and document the dynamic aspects of a system.
- It shows how the system behaves and interacts with itself and other entities (users, other systems).
- They show how data moves through the system, how objects communicate with each other, how the passage of time affects the system, or what events cause the system to change internal states.
- Since behavior diagrams illustrate the behavior of a system, they are used extensively to describe the functionality of software systems.

# Interaction Diagrams

- UML Specifies a number of interaction diagrams to model dynamic aspects of the system
- By Dynamic aspects we mean that the
  - Messages moving among objects/classes
  - Flow of control among objects
  - Sequences of events
- **Interaction Diagrams** - Set of objects or roles and the messages that can be passed among them. The sub types are
  - Sequence Diagrams
  - Communication Diagrams or Collaboration diagram

# Sequence Diagram

# Sequence Diagram

- An illustration of different parts of system interact with each other to carry out any function and its order when a particular use case is executed, **like what messages are sent and received .**

# Sequence Diagram

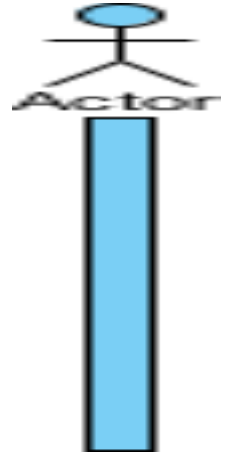
- Shows the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario emphasizing on time ordering.
- Show concurrent processes and activations
- Show time sequences that are not easily depicted in other diagrams
- Typically used during analysis and design to document and understand the logical flow of your system

# Sequence Diagram…

- **Time** in a sequence diagram is all a about ordering, not duration.
- Before start drawing a sequence diagram it is necessary to draw the use case diagram and a comprehensive description of what the use case does.
- Object symbol in sequence diagram represents a class or object demonstrating how an object will behave in the system.
- Objects and classes are communicated each other by sending and receiving messages

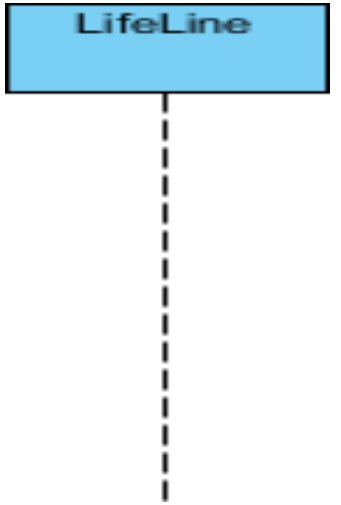
# Basic symbols and components of Sequence Diagram

## Actor




- Who or what is going to using our system like , Customer , staff, admin, bank, etc. internally or externally.

## Lifeline

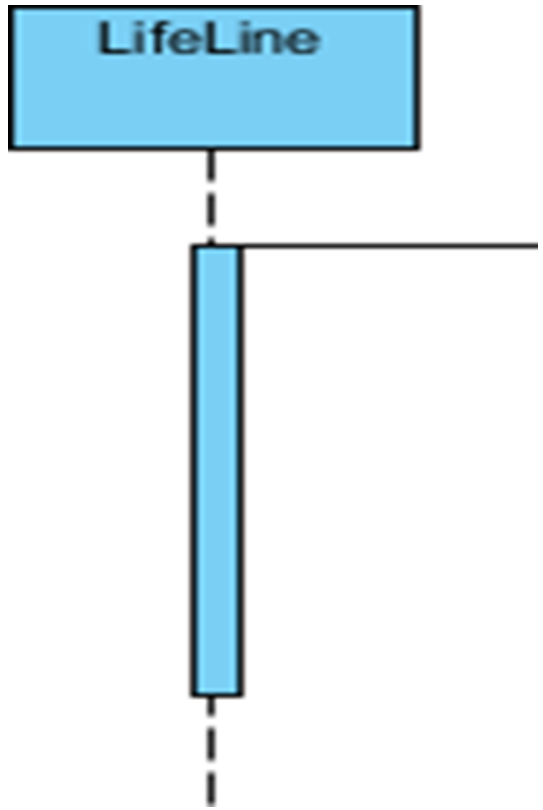



- A lifeline represents an individual participant in the Interaction.

# Sequence Diagram Notation…..

## Activations

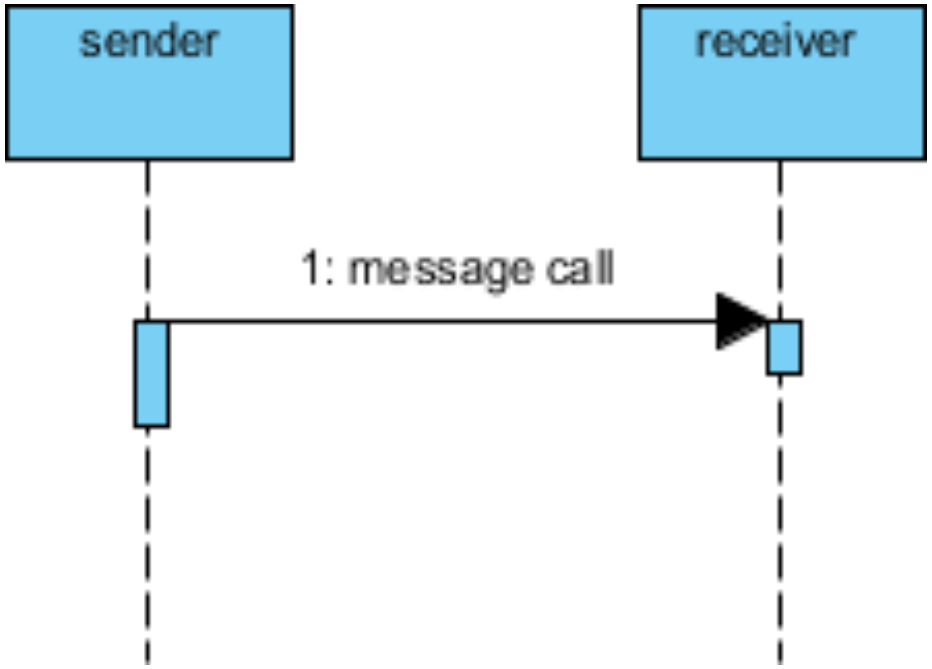
- A thin rectangle on a lifeline represents the period during which an element is performing an operation.
- The top and the bottom of the of the rectangle are aligned with the initiation and the completion time respectively



# Sequence Diagram Notation…..

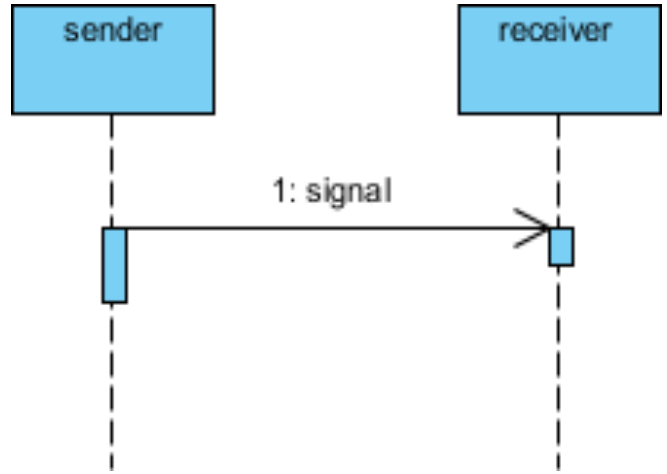
## Types of Messages

- A **synchronous message** (typically an operation call) is shown as a solid line with a filled arrowhead. It is a regular message call used for normal communication between sender and receiver.



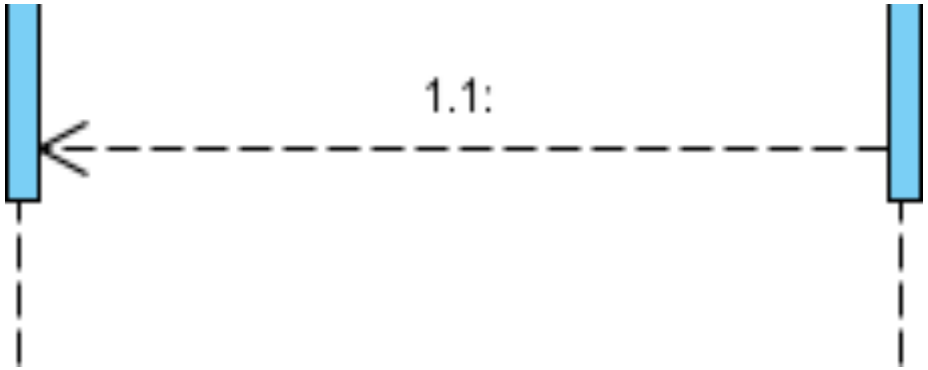
# Sequence Diagram Notation…..

- An **asynchronous message** has a solid line with an open arrowhead. A signal is an asynchronous message that has no reply.




**Return Message**

- A type of message that represents the pass of information back to the caller of a corresponded former message.



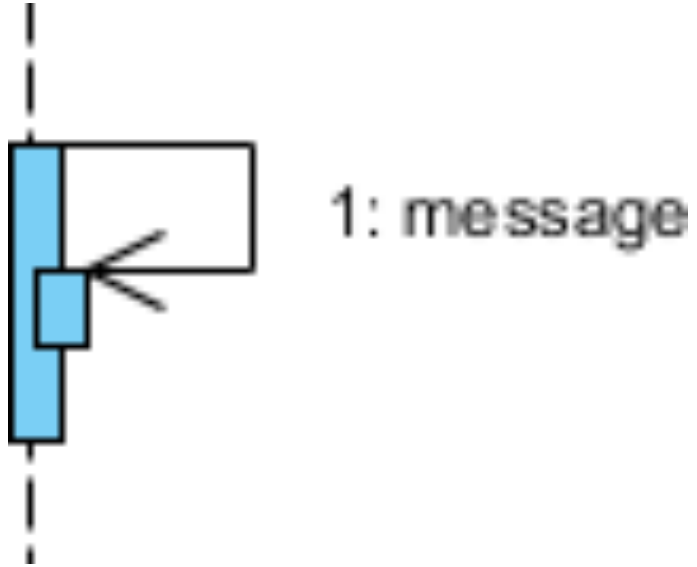
# Sequence Diagram Notation…..

## Self Message

- A message defines a particular communication between Lifelines of an Interaction.
- Self message is a kind of message that represents the invocation of **message of the same lifeline**.


## Recursive Message

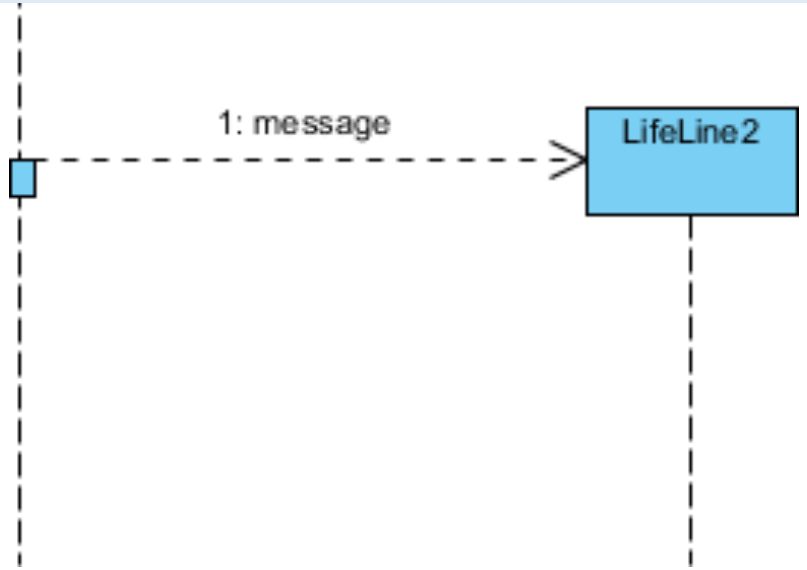
- Sometimes an object needs to communicate to the same class method, i.e., **when an object sends a message to itself.**
- It's target points to an activation on top of the activation where the message was invoked from.



# Sequence Diagram Notation…..

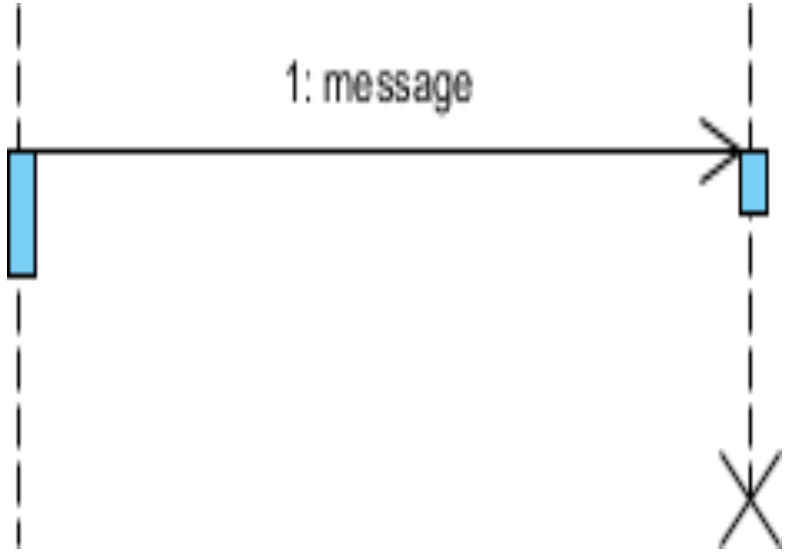
## Create Message

- A kind of message that represents the instantiation of (target) lifeline.




## Destroy Message

- Is a kind of message that represents the request of destroying the lifecycle of target lifeline.



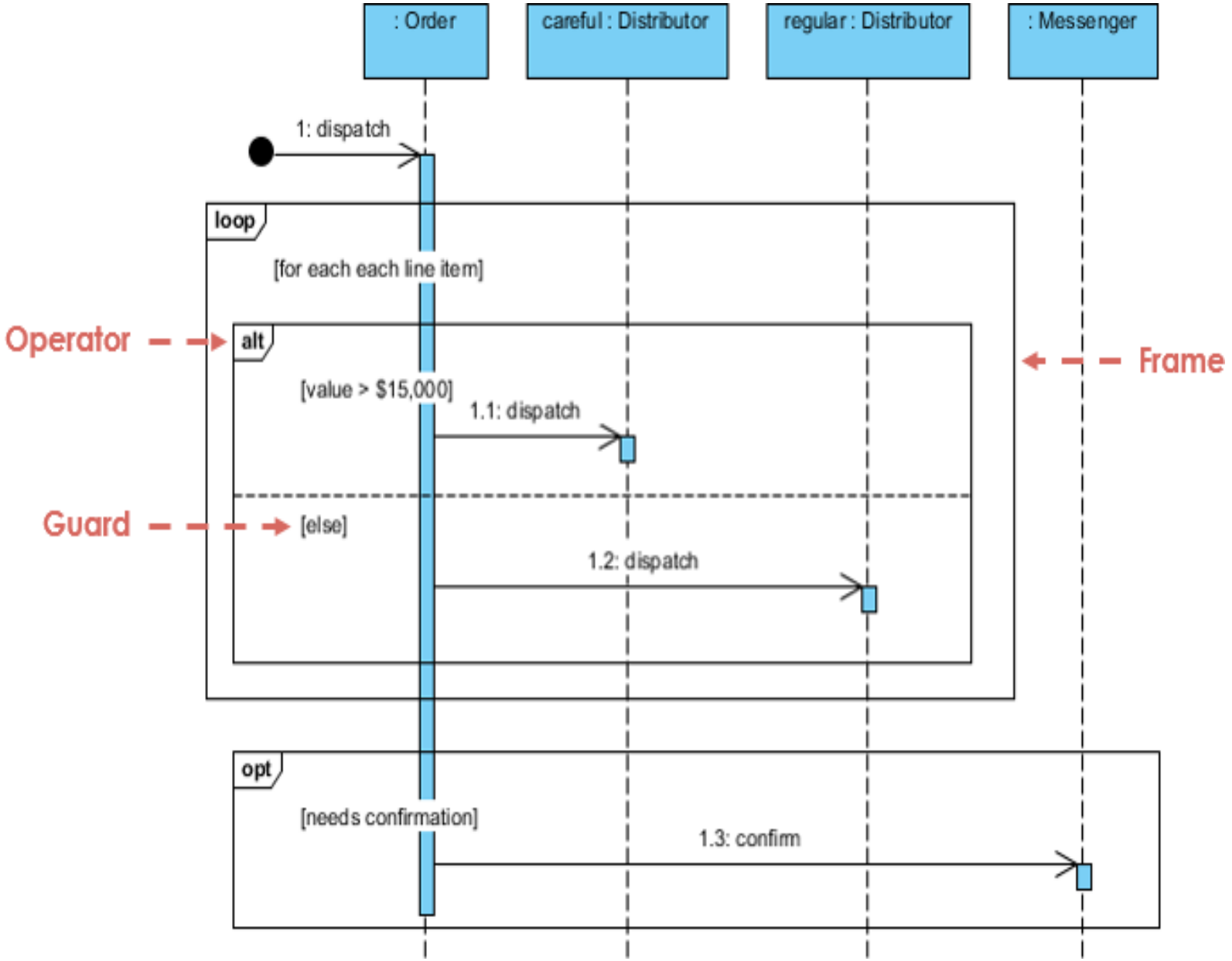
# Sequence Diagram Notation…..

- A sequence fragment is represented as a box, called a combined fragment, which encloses a portion of the interactions within a sequence diagram
- The **fragment operator** (in the top left cornet) indicates the type of fragment
- Fragment types: **loop, alt, opt, par, etc.**

## For instance :-

| | |
|---|---|
| **alt** | Alternative multiple fragments: only the one whose condition is true will execute. |
| **opt** | Optional: the fragment executes only if the supplied condition is true. Equivalent to an alt only with one trace. |
| **par** | Parallel: each fragment is run in parallel. |
| **loop** | Loop: the fragment may execute multiple times, and the guard indicates the basis of iteration. |

# Combined Fragment Example



# Example -1 Sequence diagram to Search Book scenario

Search Book : Use Case

- Main scenario -

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

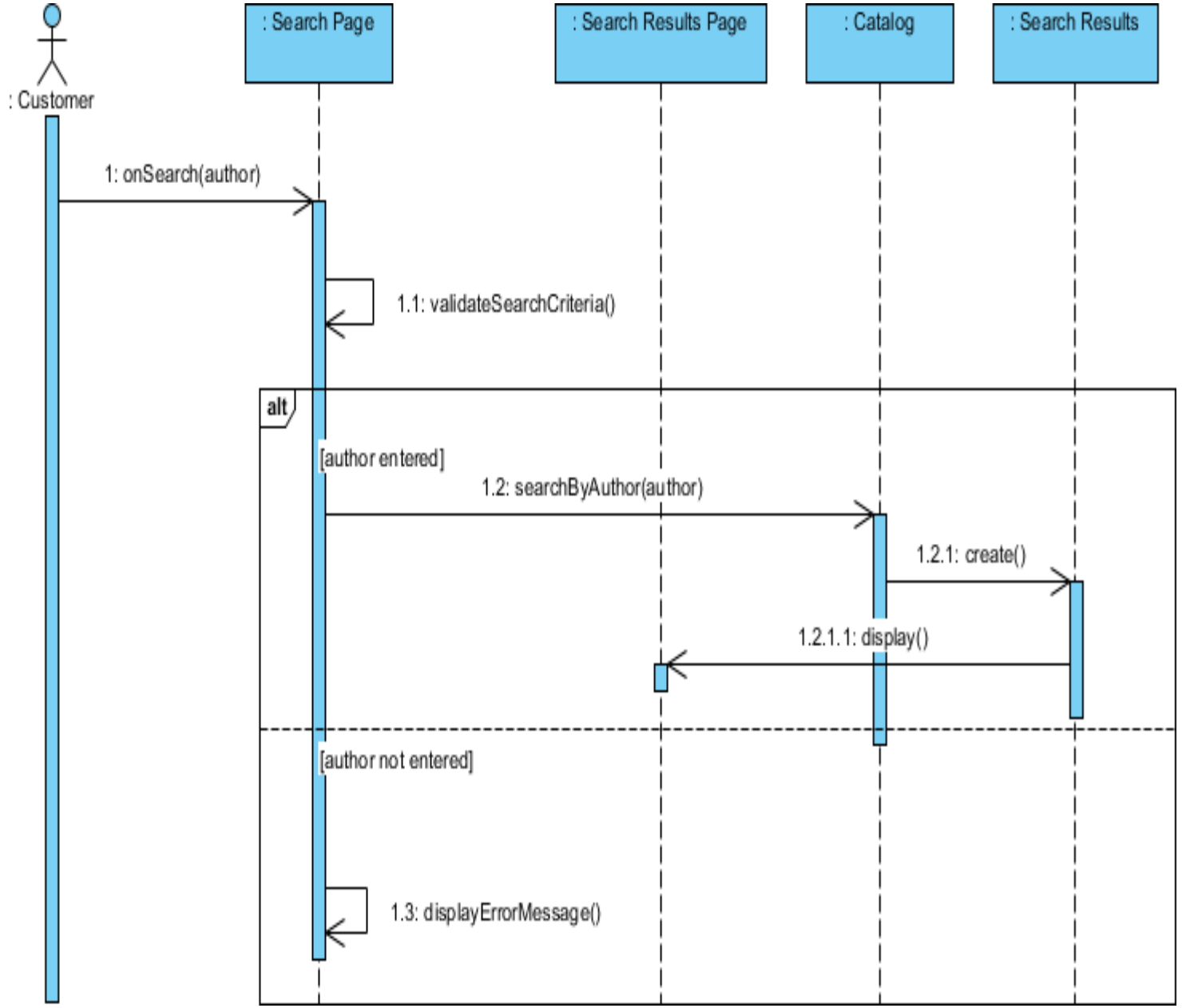
The system validates the Customer's search criteria.

If author is entered, 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 path -

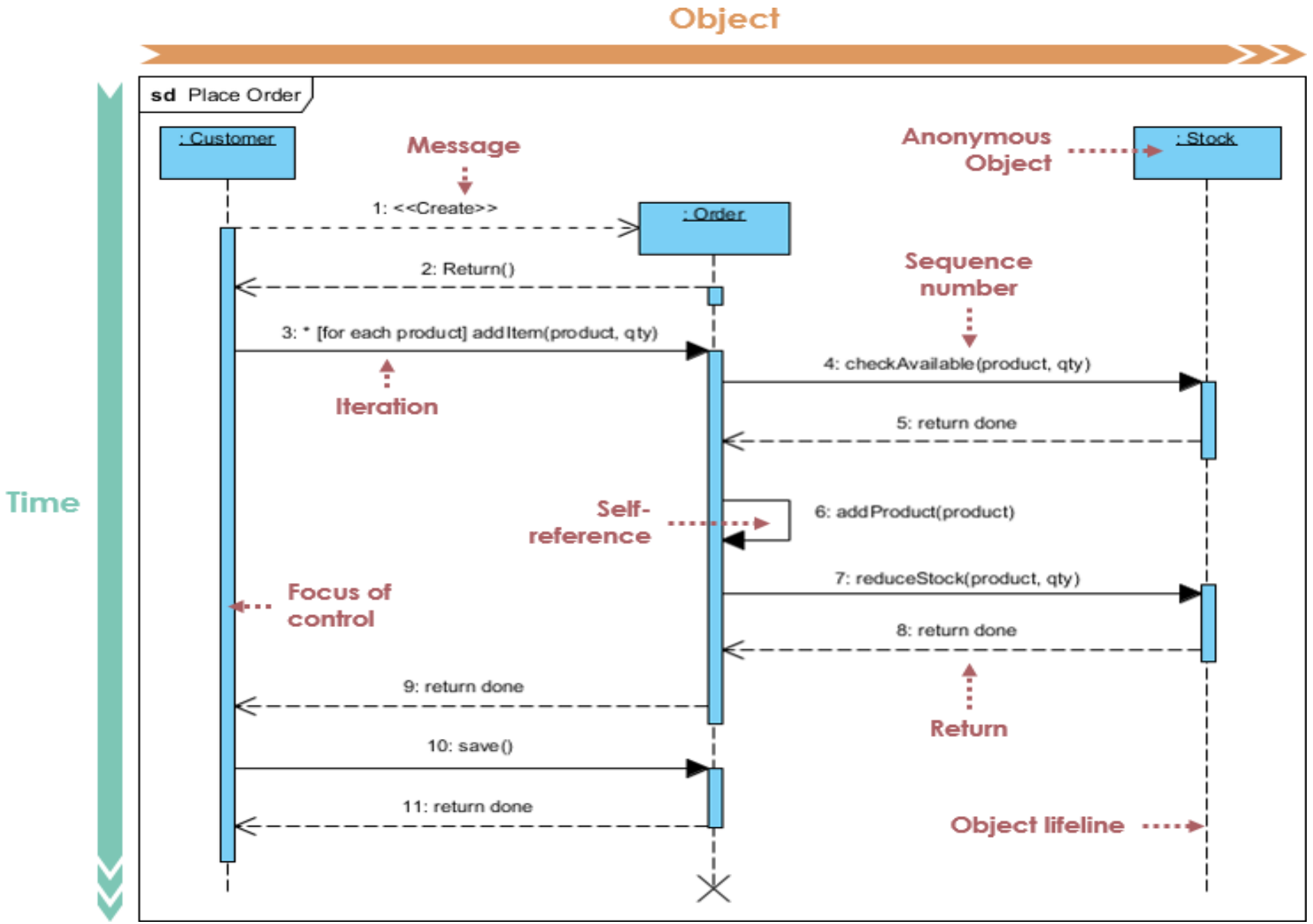
If the Customer did not enter the name of an author before pressing the Search button, the System displays an error message



# Example-2: Place Order scenario

- The example shows a Sequence diagram with three participating objects: **Customer, Order, and the Stock**.
- **Step 1 and 2**: Customer creates an order.
- **Step 3**: Customer add items to the order.
- **Step 4, 5**: Each item is checked for availability in inventory.
- **Step 6, 7, 8** : If the product is available, it is added to the order.
- **Step 9:** return
- **Step 10, 11:** save and destroy order

# …..Cont'd…..Example-2 : Place Order



# Rules of thumb

- Rarely use options, loops, alt/else
  - These constructs complicate a diagram and make them hard to read/interpret
  - Frequently it is better to create multiple simple diagrams
- Create sequence diagrams for use cases when it helps clarify and visualize a complex flow
- **Remember:** The goal of UML is communication and understanding

# State Machine Diagram

# What is a State Machine Diagram?

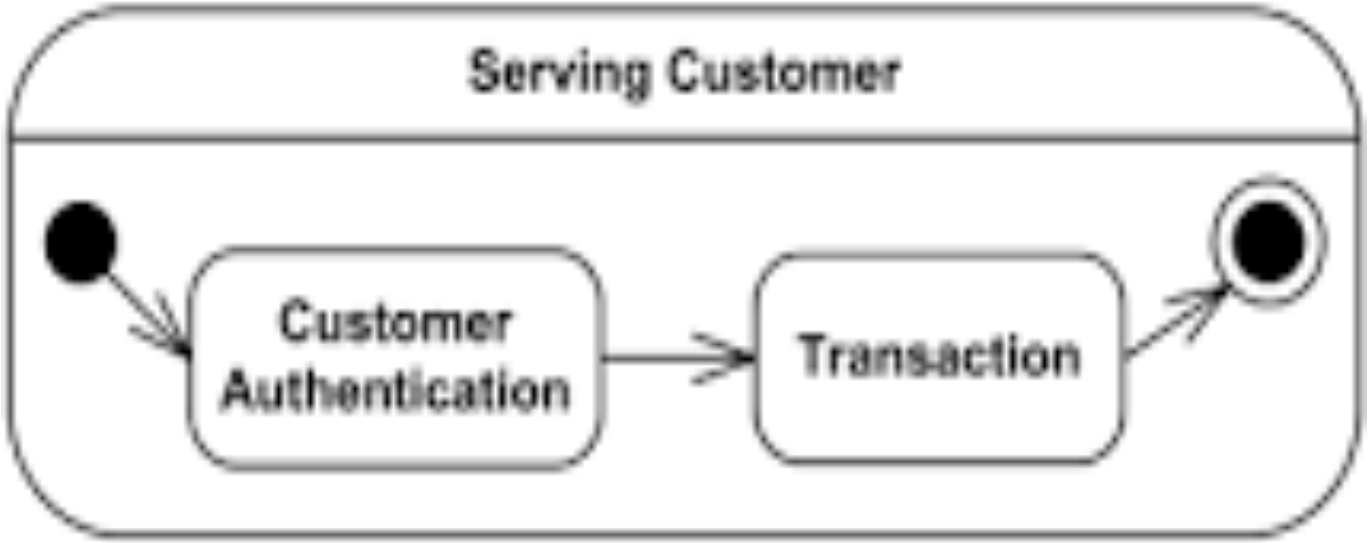
- UML State Diagram/ State Machine/ State Chart
  - Consisting of states, transitions, events and activities of an object
  - Used to **capture the behavior of a software system**.
  - They describe all of the possible states that a particular object can get into and how the object's state changes as a result of events that reach the object.
  - Used to model the ***behavior of a class***, ***a subsystem***, a ***package***, or ***even an entire system***.
  - In most OO techniques, state diagrams are **drawn for a single class to show the lifetime behavior of a single object.**

# …Cont'd

- A state machine is any device that **stores the status of an object at a given time and can change status or cause other actions based on the input it receives**.
- States refer to the different combinations of information that an object can hold, not **how the object behaves**.

# State diagram symbols and components

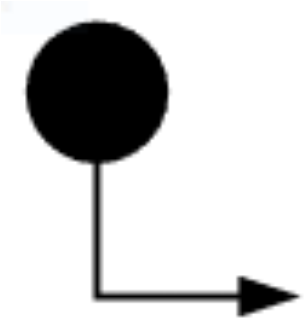
- **Composite state**
  - Generally, **composite state** is defined as **state** that has substates (nested **states**). Substates could be sequential (disjoint) or concurrent (orthogonal).



# State diagram symbols and components…

## First state

- A marker for the first state in the process, shown by a dark circle with a transition arrow.



## Terminator

- A circle with a dot in it that indicates that a process is terminated..

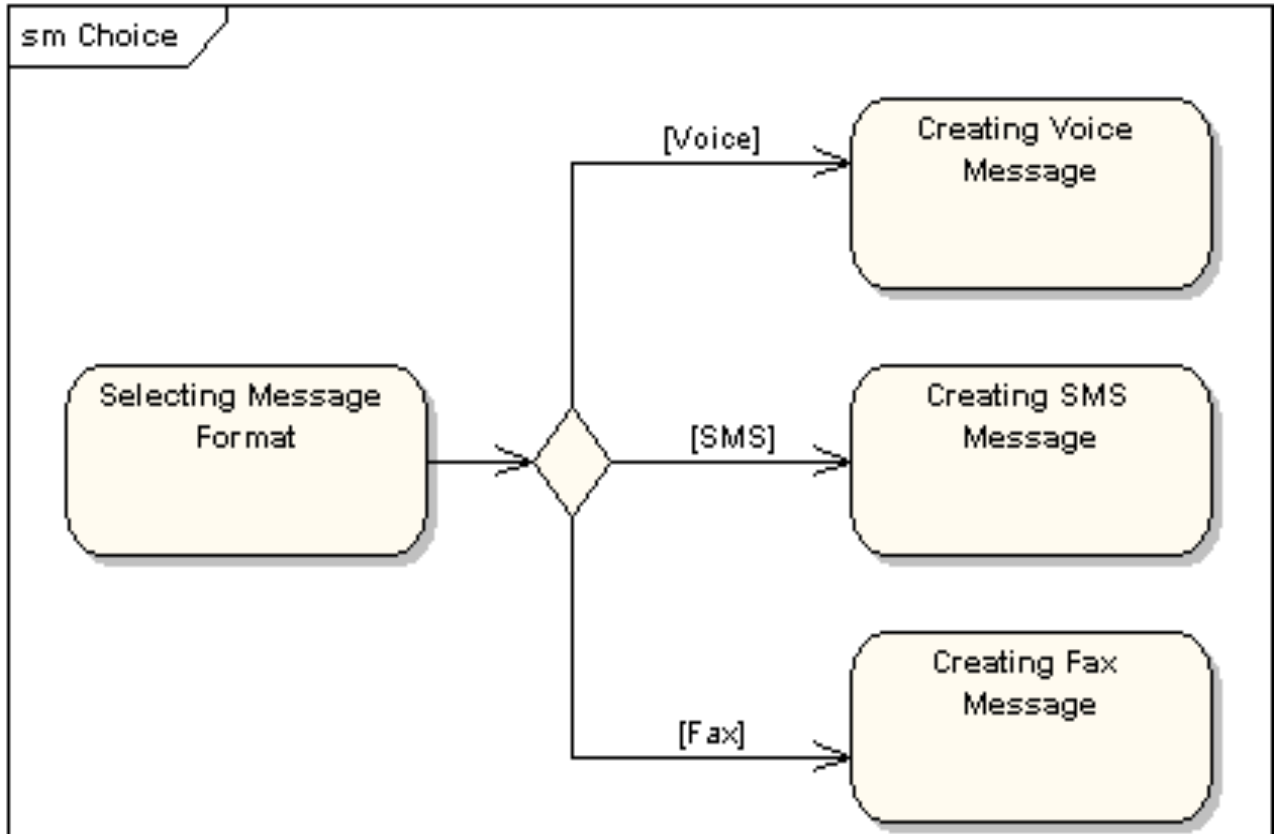
# State diagram symbols and components…

## Exit point

- The point at which an object escapes the composite state or state machine, denoted by a circle with an X through it.
- The exit point is typically used if the process is not completed **but has to be escaped for some error or other issue**.

# State diagram symbols and components…

- **Choice pseudo state**
  - A **diamond symbol** that indicates a **dynamic condition** with branched potential results. It evaluates the guards of the triggers of its outgoing transitions to select only one outgoing transition.



# State diagram symbols and components…

**State** : Represent situations during the life of an object

- A rectangle with rounded corners that indicates the current nature of an object.

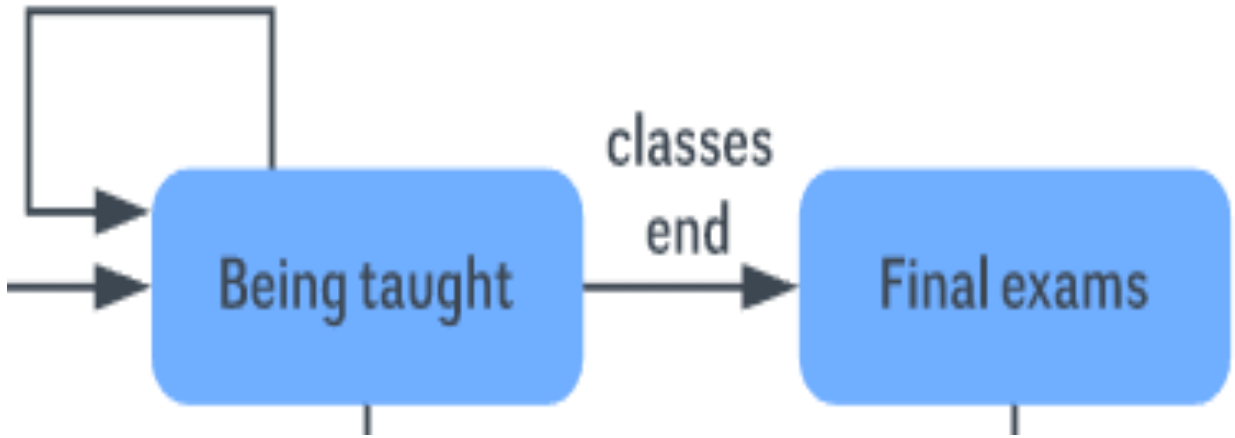

A simple state


A state with internal activities

# State diagram symbols and components…

## Event

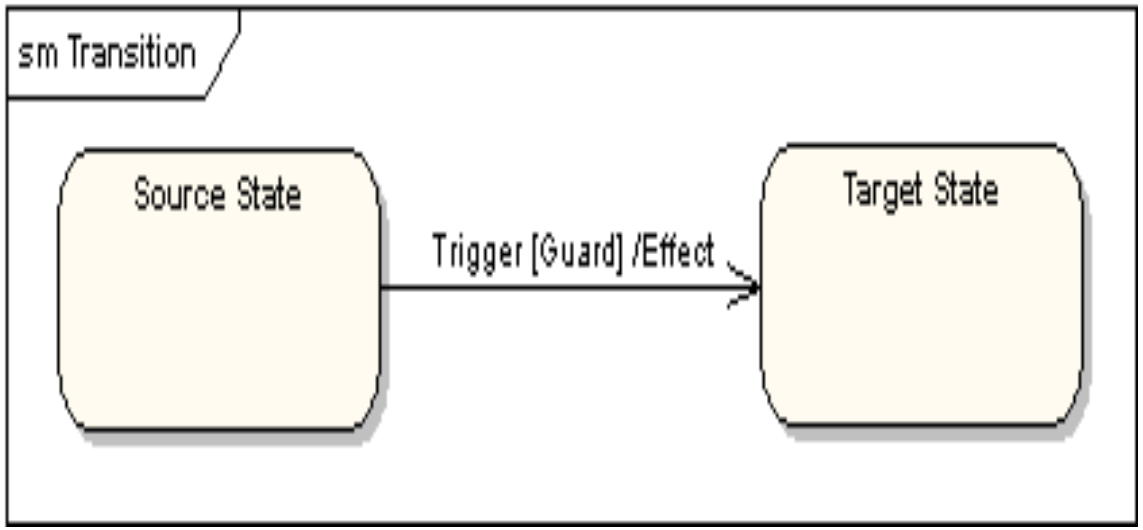
- An instance that **triggers a transition**, labeled above the applicable transition arrow.
- In this case, **“classes end”** is the event that triggers the end of the “Being taught” state and the beginning of the “Final exams” state.



# State diagram symbols and components…

## Transition

- A solid arrow represents the path between different states of an object. Label the transition with the event that triggered it and the action that results from it.
- A state can have a transition that points back to itself.
- An arrow running from one state to another that indicates a changing state.

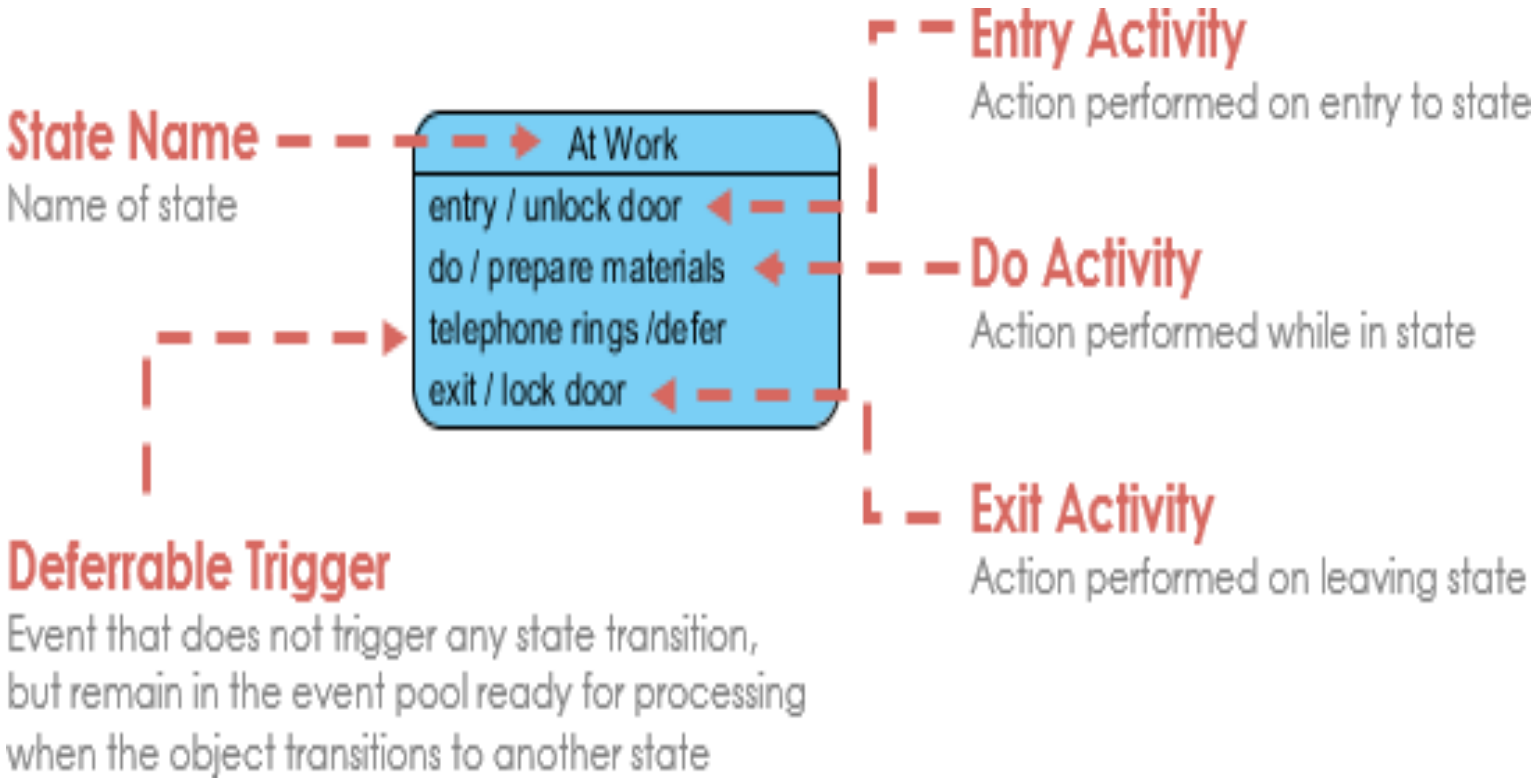
# State diagram symbols and components…

- **Guard**
  - A Boolean condition that allows or stops a transition, written above the transition arrow.

# State diagram symbols and components…

## State Actions

- We can associate the effect with the target state rather than the transitions. This can be done by defining an entry action for the state. The diagram below shows a state with an entry action and an exit action.



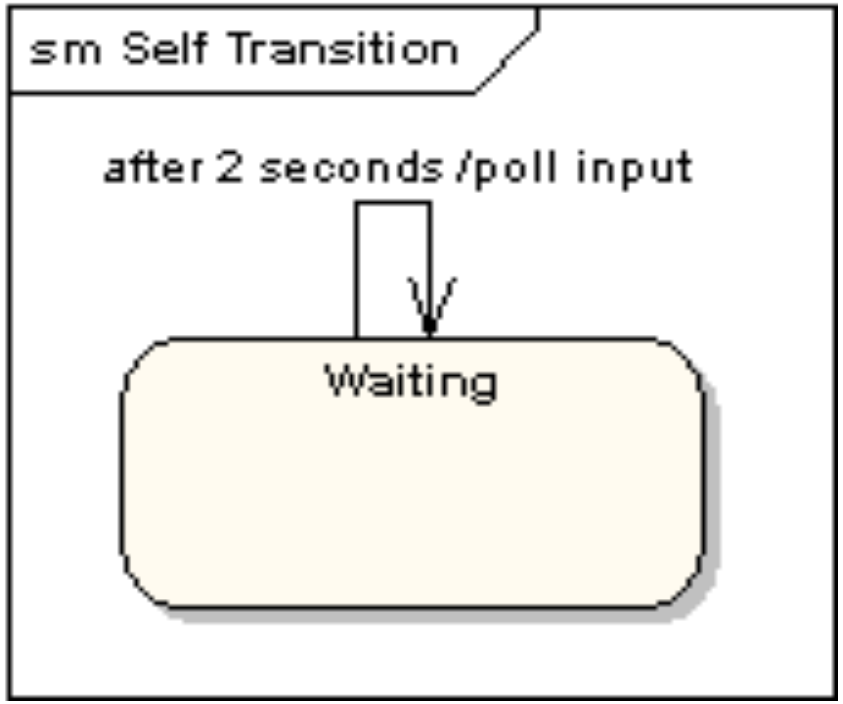
# State diagram symbols and components…

## Entry and Exit Actions

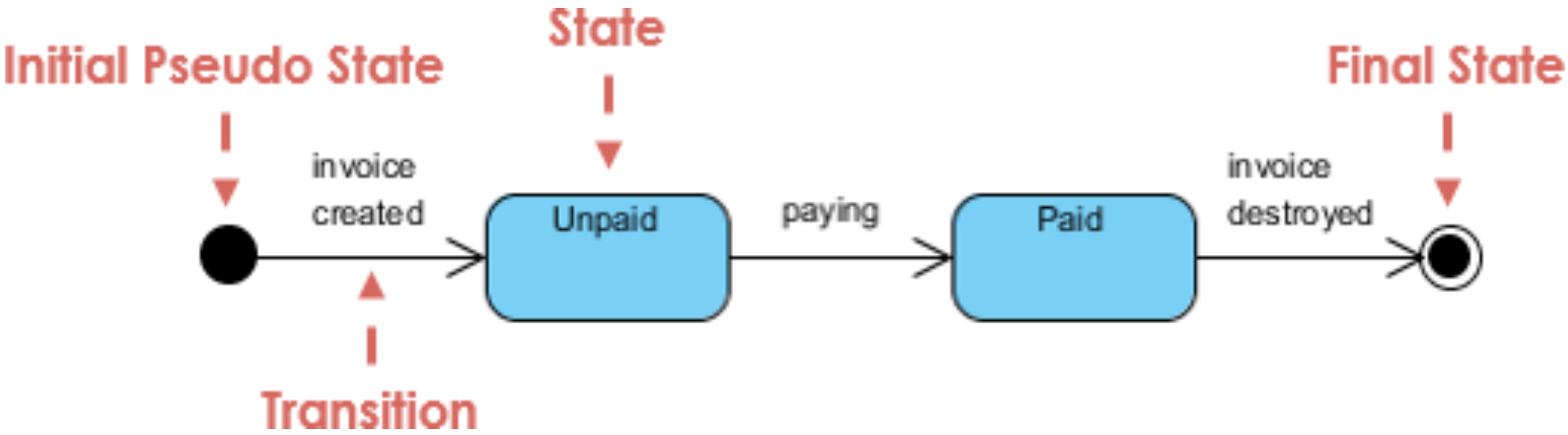
- Entry and Exit actions specified in the state. It must be true for every entry / exit occurrence. If not, then you must use actions on the individual transition arcs
- **Entry Action** executed on entry into state with the
  - **notation:** Entry / action
- **Exit Action** executed on exit from state with the
  - **notation:** Exit / action

# State diagram symbols and components…

- **Self-Transitions**
  - We use a solid arrow pointing back to the state itself to represent a self transition. There might be scenarios when the state of the object does not change upon the occurrence of an event.

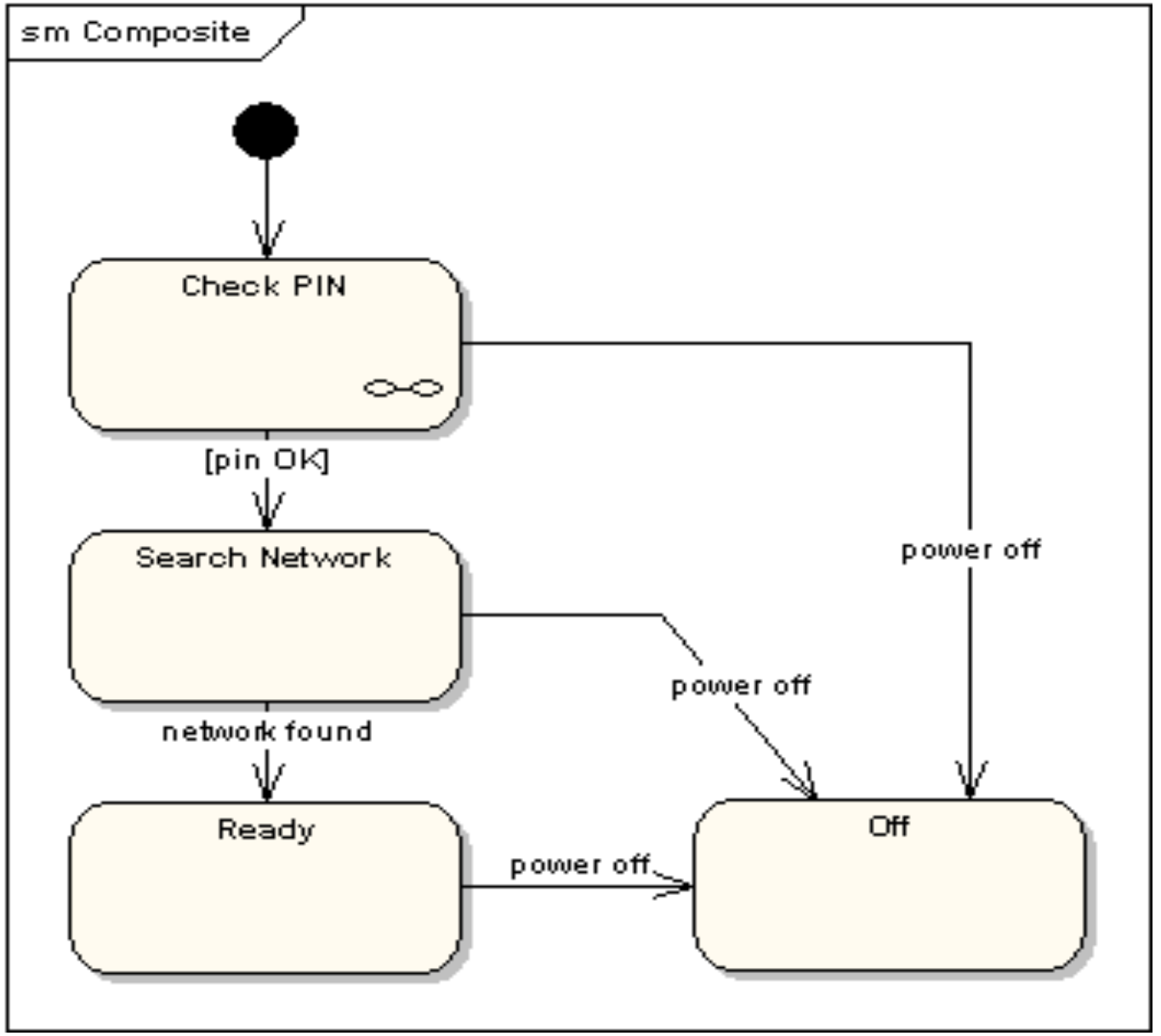


# Simple State Machine Diagram Notation

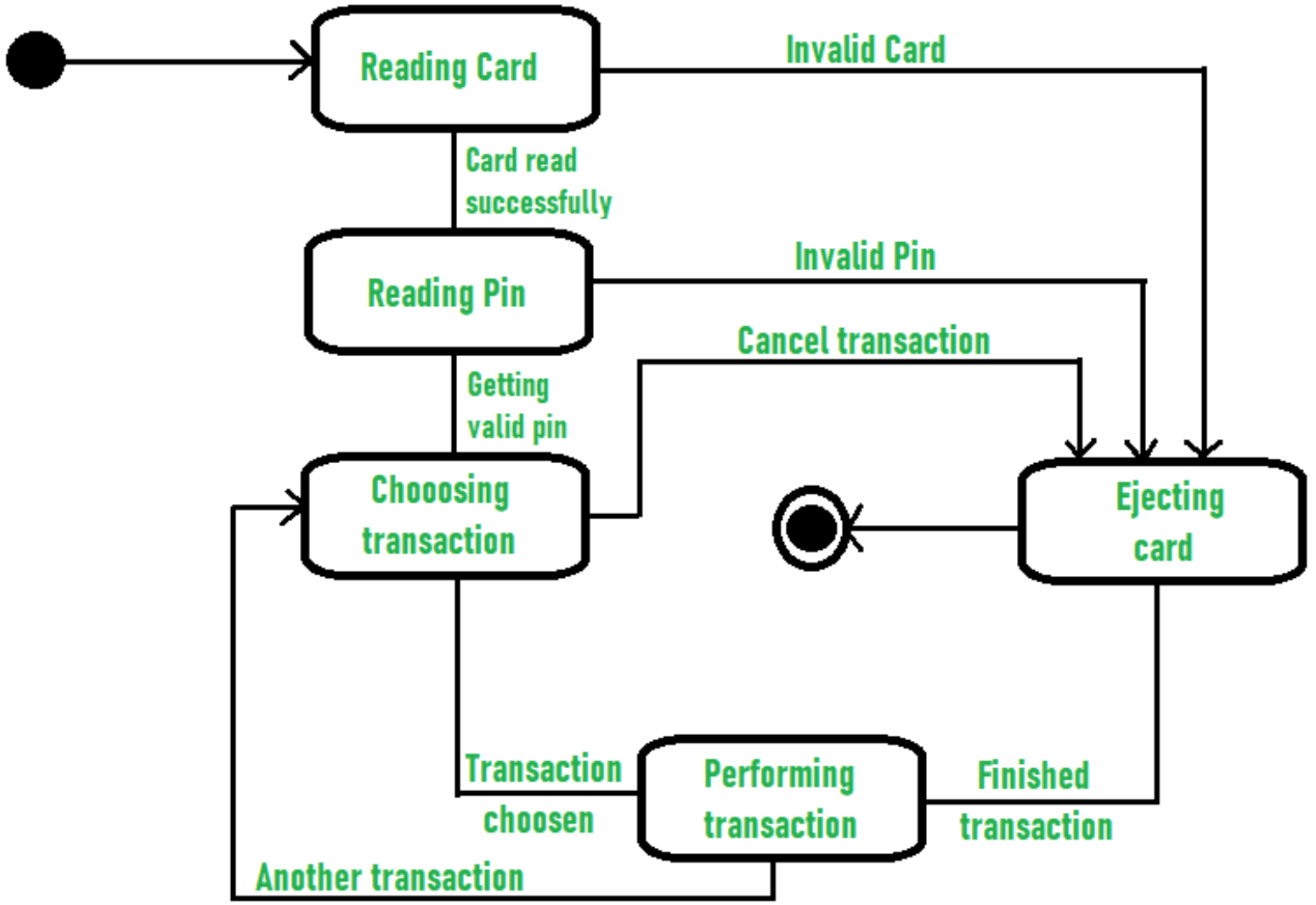


# Example-1

- A state machine diagram may include sub-machine diagrams



# Example-2- State Diagram for ATM System




State Transition Diagram for ATM System

# How to draw a State chart diagram?

- To draw a state diagram, one must identify all the possible states of any particular entity.
  1. Identify entities that have complex behavior or identify a class participating in behavior whose lifecycle is to be specified
  2. Model states – **Determine the initial and final states of the entity**
  3. Model transitions
  4. Model events – **Identify the events that affect the entity**
  5. Working from the initial state, trace the impact of events and identify intermediate states
  6. Identify any **entry and exit actions on the states**
  7. Expand states using substates where necessary
  8. If the entity is a class, check that the action in the state are supported by the operations and relationships of the class, and if not extend the class
  9. Refine and elaborate as required

# Rules

- Following rules must be considered while drawing a state chart diagram:

1. The name of a state transaction must be unique.
2. The name of a state must be easily understandable and describe the behavior of a state.
3. If there are multiple objects, then only essential objects should be implemented.
4. Proper names for each transition and an event must be given.

# Activity Diagram

# What is an Activity Diagram?

- It is an extended version of flowchart that models the transition form one activity to another.
- Emphasize and show flow of control among objects
- Visually **presents a series of actions or flow of control in a system similar to a flowchart or a data flow diagram.**
- Often used in **business process modeling**.
- They can also describe the steps in a use case diagram.
- Activities modeled can be sequential and concurrent.

# Basic Activity Diagram Notations and Symbols

- **Initial State or Start Point**
  - A small filled circle followed by an arrow represents the initial action state or the start point for any activity diagram.


  - For activity diagram using swim lanes, make sure the start point is placed in the top left corner of the first column.

- **Final State or End Point**
  - An arrow pointing to a filled circle nested inside another circle represents the final action state.

# Basic Activity Diagram Notations and Symbols…

- **Activity or Action State**
  - An action state represents the non-interruptible action of objects.

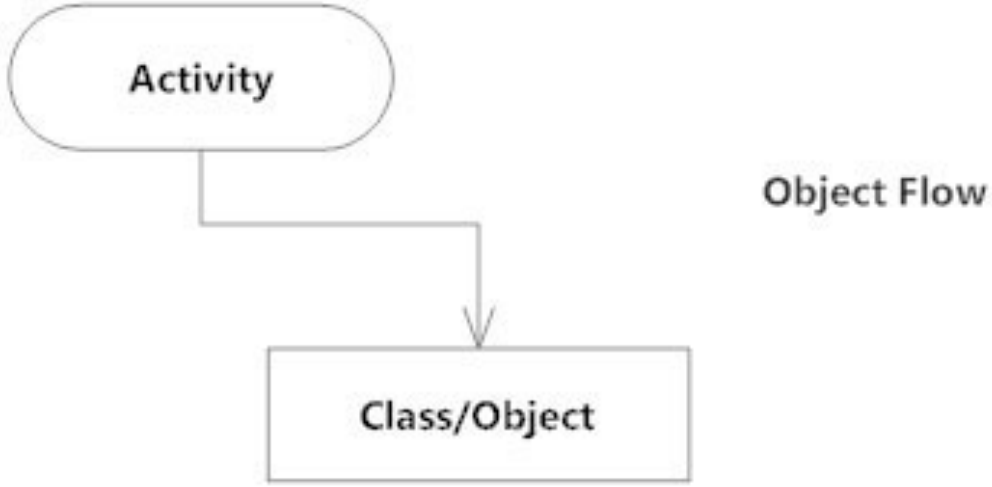

Activity

- **Action Flow**
  - Action flows, also called edges and paths**, illustrate the transitions from one action state to another**. They are usually drawn with an arrowed line.

Action Flow

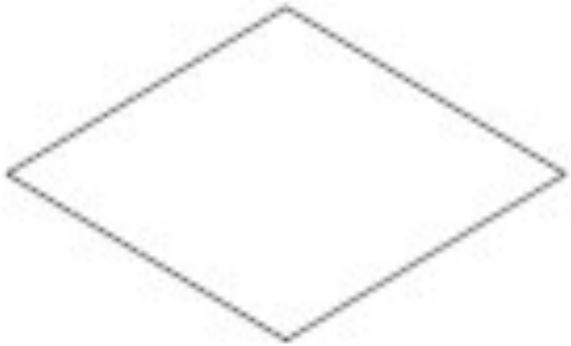
# Basic Activity Diagram Notations and Symbols…

- **Object Flow**
  - Object flow refers to the creation and modification of objects by activities.
  - An **object flow arrow from an action to an object** means that the *action creates or influences the object*.
  - An **object flow arrow from an object to an action** indicates that *the action state uses the object*.



# Basic Activity Diagram Notations and Symbols…

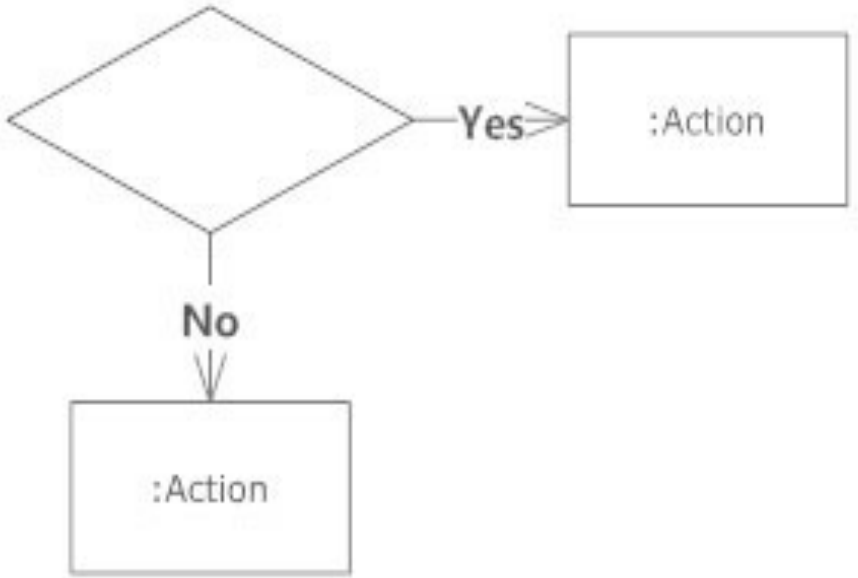
- **Decisions and Branching**
  - A **diamond** represents a decision **with alternate paths.**
  - When an activity requires a decision prior to moving on to the next activity, add a diamond between the two activities.
  - The outgoing alternates should be labeled with a condition or guard expression.
  - You can also label one of the paths "else."



Decision Symbol

# Basic Activity Diagram Notations and Symbols…

- **Guards**
  - In UML, guards are a **statement written** next to a decision diamond that must be true before moving next to the next activity.
  - These are not essential, but are useful when a specific answer, such as "Yes, three labels are printed," is needed before moving forward.

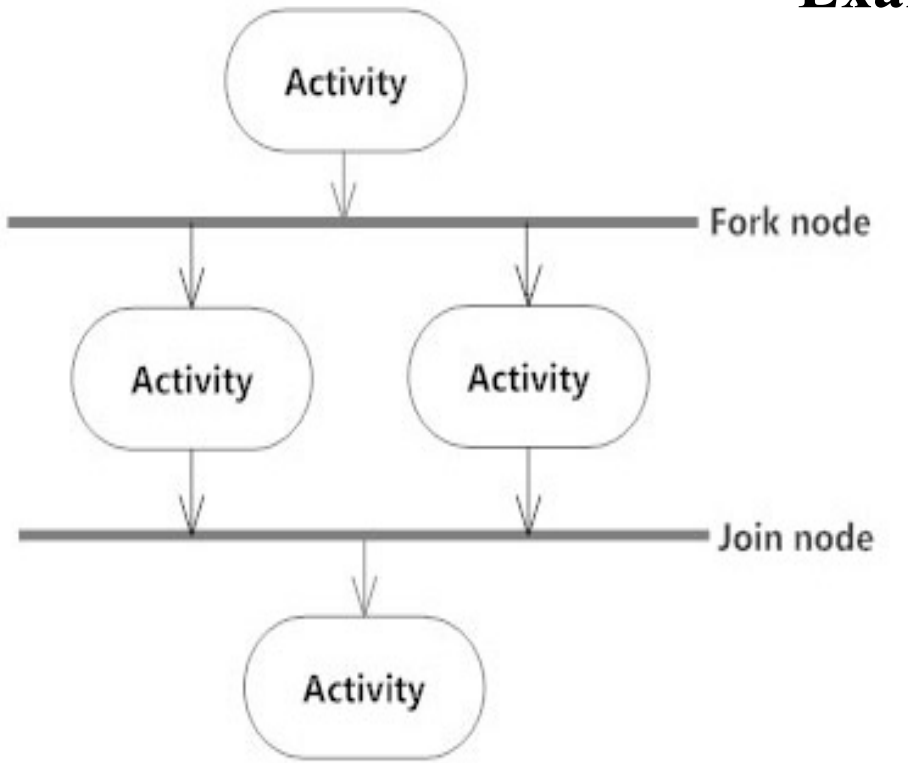


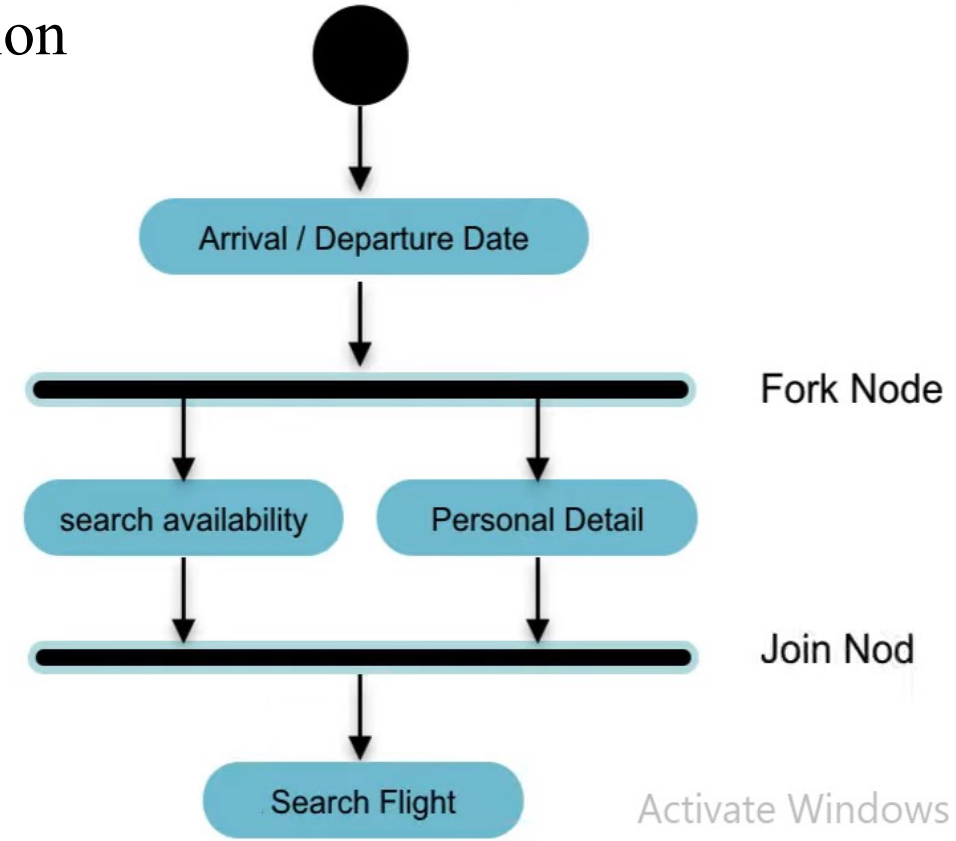

Guard Symbols

# Basic Activity Diagram Notations and Symbols…

- **Synchronization**
  - **A fork node:** is used to split a **single** incoming flow **into multiple concurrent flows**.
    - It is represented as a straight, slightly thicker line in an activity diagram.
  - **A join node:** joins **multiple** concurrent flows back **into** a **single** outgoing flow.
    - A fork and join node used together are often referred to as **synchronization**.

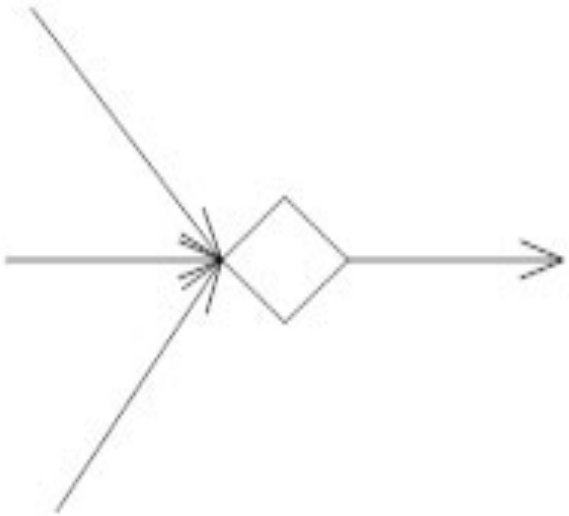
**Example** :- Online reservation system



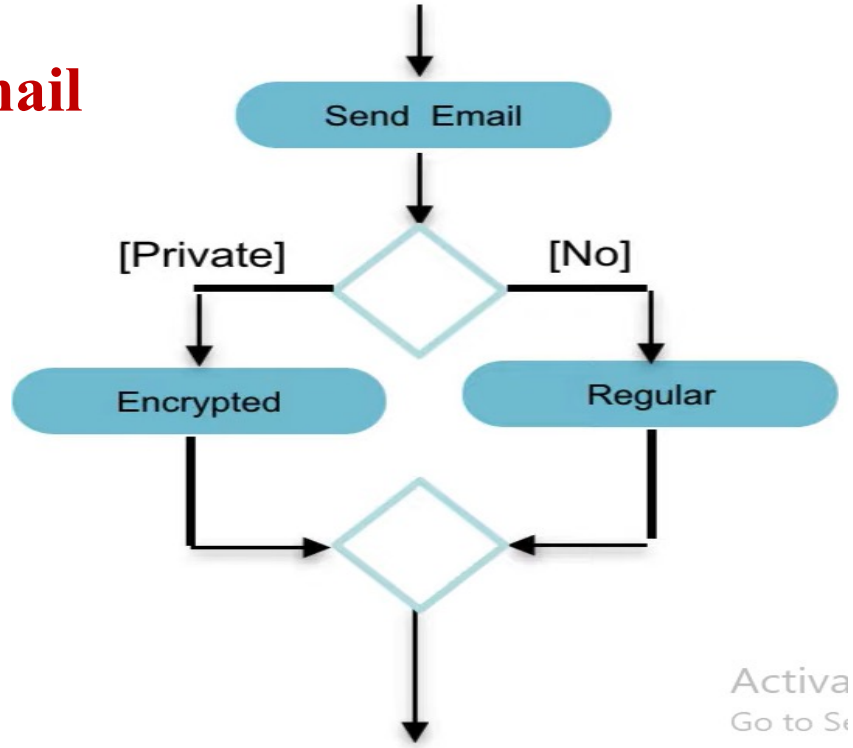
# Basic Activity Diagram Notations and Symbols…

- **Merge Event**
  - A merge event brings together multiple flows that are not concurrent.



Merge

**E.g. Sending Email**

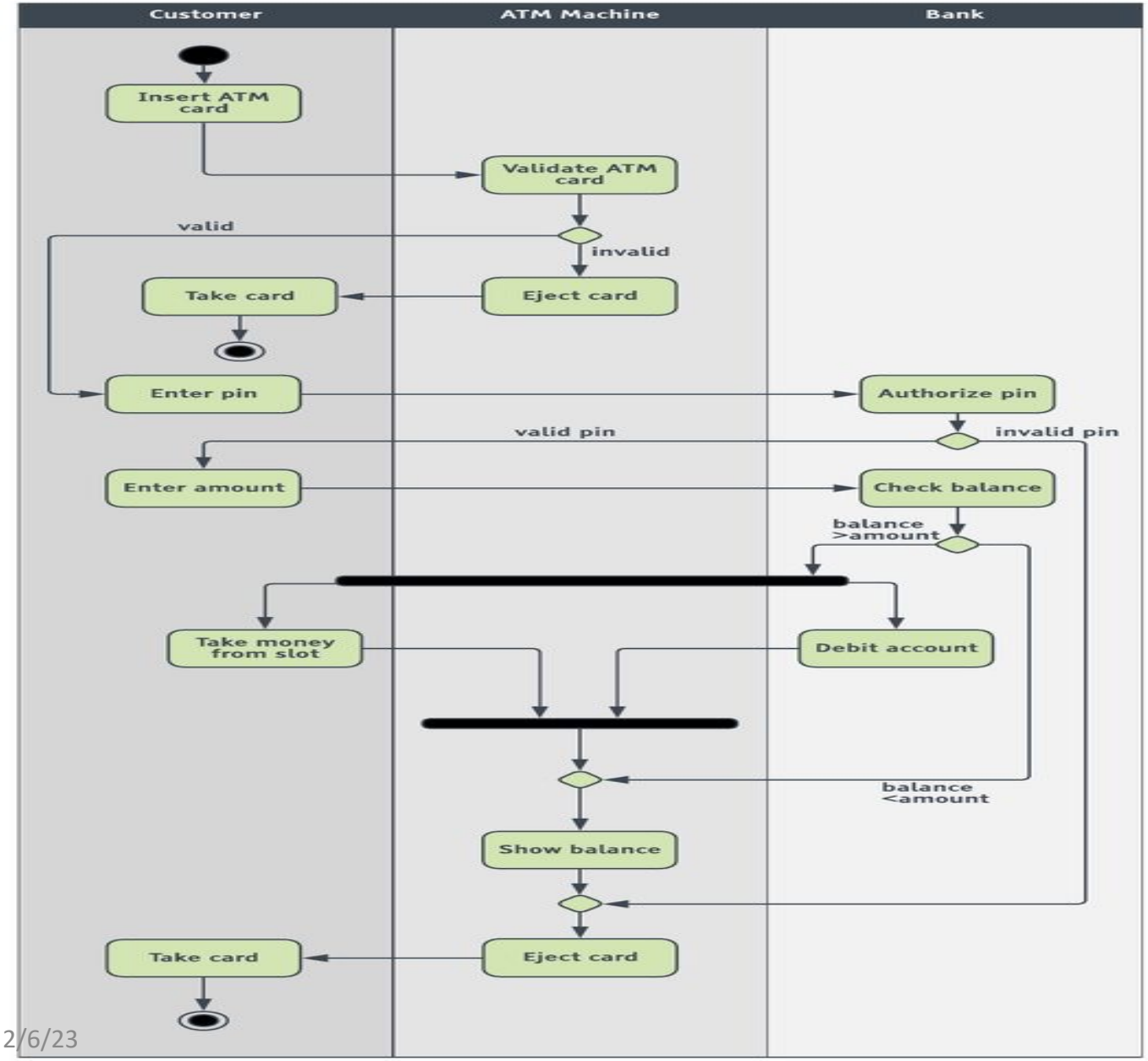



- **Swim lanes**
  - A swim lane is a way to group activities performed by the same actor on an activity diagram or to group activities in a single thread.

# Steps to model Activity Diagram

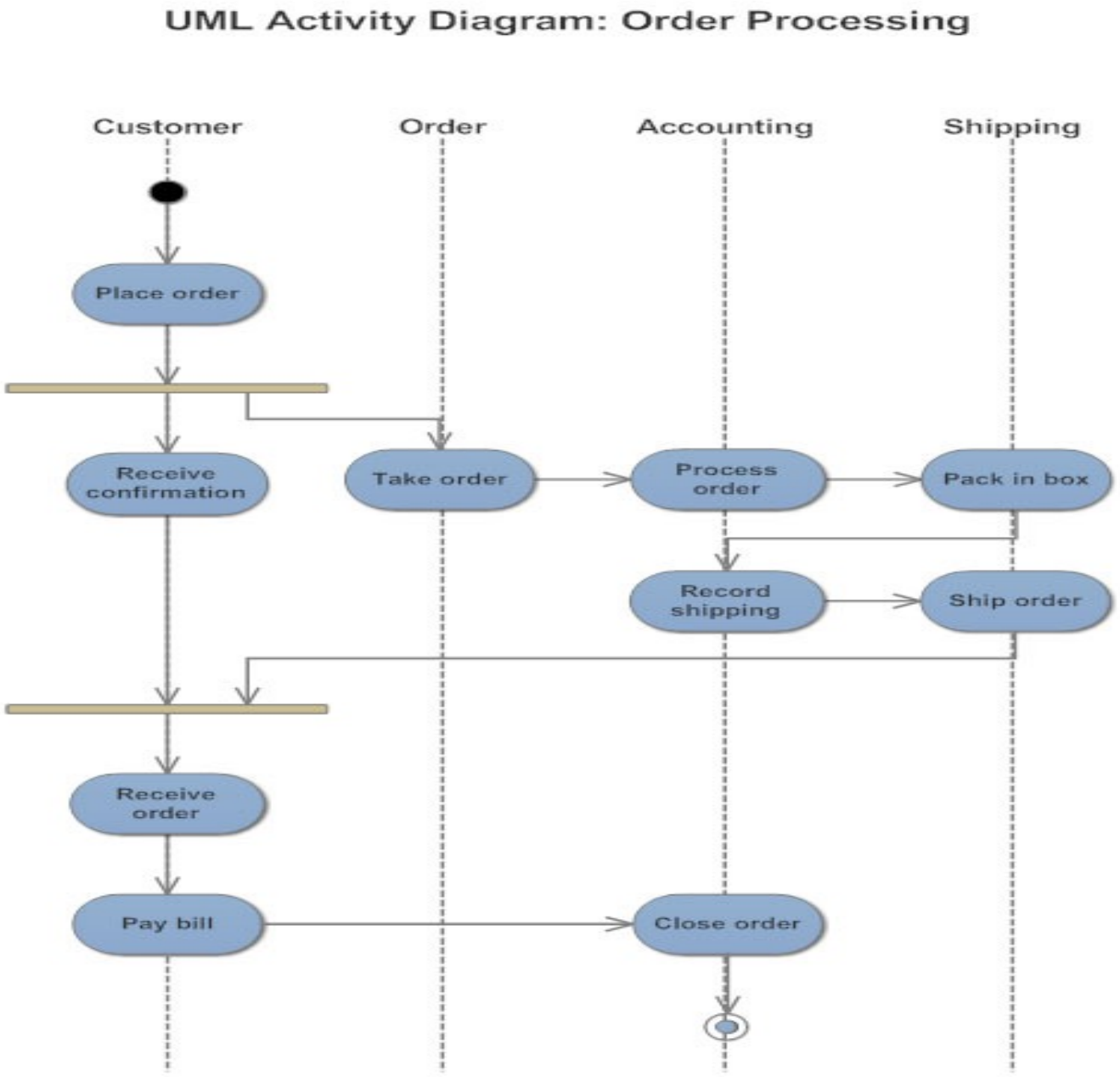
1. Identify candidate use cases, through the examination of business workflows
2. Identify pre- and post-conditions (the context) for use cases
3. Model workflows between/within use cases
4. Model complex workflows in operations on objects
5. Model in detail complex activities in a high level activity Diagram

# Example-1




- Activity diagram for an ATM System **withdraw money** use case

# Example -2


UML Activity Diagram: Order Processing
Customer
Order
Accounting
Shipping
Place order
Receive confirmation
Take order
Process order
Pack in box
Record shipping
Ship order
Receive order
Pay bill
Close order

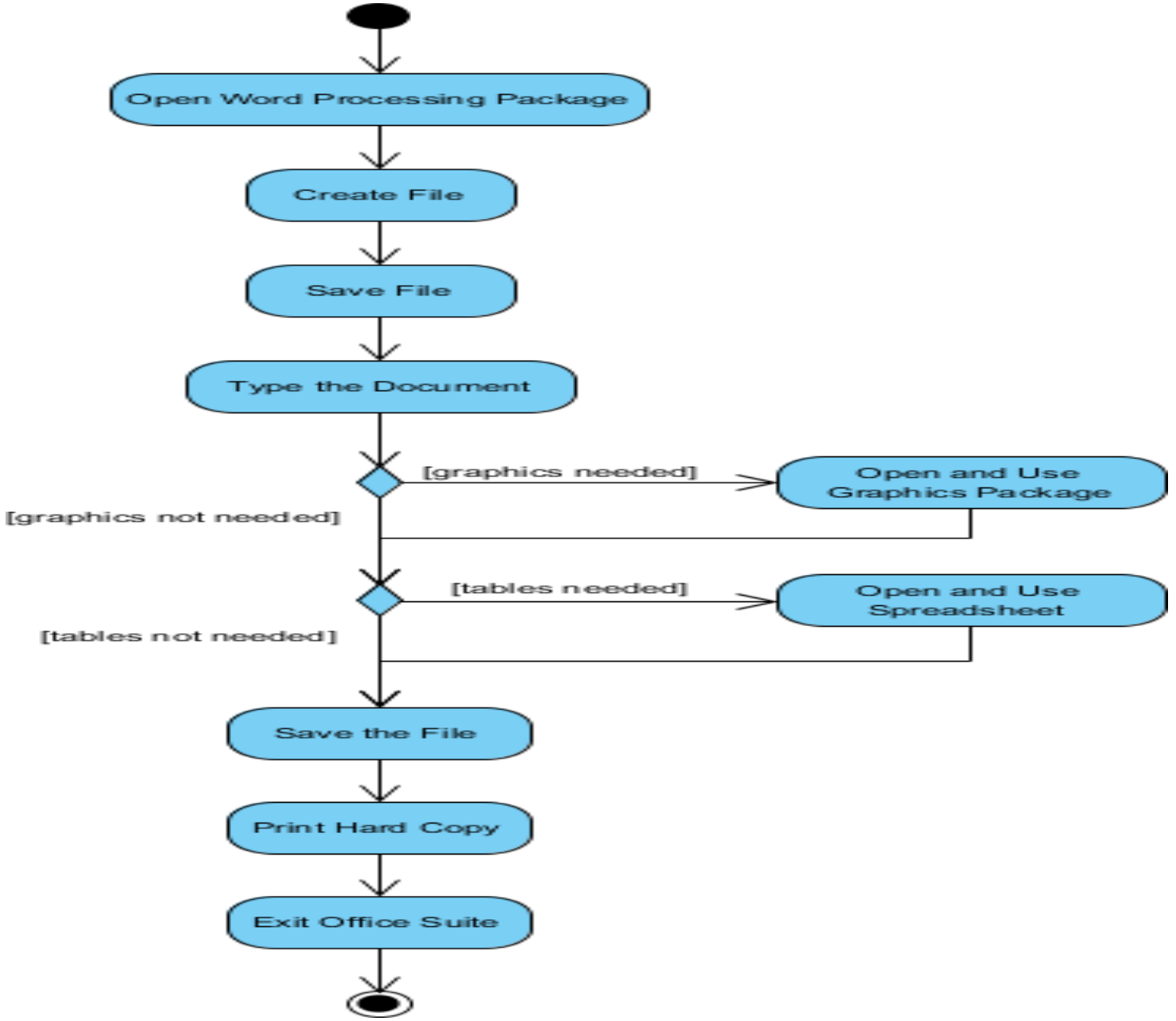
# Exercise

Activity Diagram - Modeling a Word Processor

The activity diagram example below describes the workflow for a word process to create a document through the following steps:

1. Open the word processing package.
2. Create a file.
3. Save the file under a unique name within its directory.
4. Type the document.
5. If graphics are necessary, open the graphics package, create the graphics, and paste the graphics into the document.
6. If a spreadsheet is necessary, open the spreadsheet package, create the spreadsheet, and paste the spreadsheet into the document.
7. Save the file.
8. Print a hard copy of the document.
9. Exit the word processing package.

# Possible answer for the previous exercise



# END of the Course!

# Collaboration (Communication Diagrams)

# What is Collaboration Diagram?

- Collaboration diagrams (**known as Communication Diagram in UML 2.x**) are used to show how objects interact to perform the behavior of a particular use case, or a part of a use case.
- Collaboration diagrams are extensions of object diagram which **illustrate interactions between objects**
- The collaboration diagram illustrates **messages being sent between classes and objects (instances).**
- These diagrams can be used to portray the dynamic behavior of a particular use case and define the role of each object.
- An object consists of several features. Multiple objects present in the system are connected to each other.
- The collaboration diagrams are best suited for analyzing use cases.

# What is a Collaboration Diagram

- A Collaboration is a collection of named objects and actors with links connecting them. They collaborate in performing some task.
- A Collaboration between objects working together provides emergent desirable functionalities in Object-Oriented systems
- Objects collaborate by communicating (passing messages) with one another in order to work together
- Unlike a sequence diagram, a collaboration diagram shows the relationships among the objects.

# Why Collaboration Diagram?

- They are very useful for visualizing the relationship between objects collaborating to perform a particular task
- They provide a good view – although static - view of interaction between objects which may be difficult to see at the class level
- Model collaborations between objects or roles that deliver the functionalities of use cases and operations
- Model mechanisms within the architectural design of the system
- Capture interactions that show the messages passing between objects and roles within the collaboration
- Model alternative scenarios within use cases or operations that involve the collaboration of different objects and interactions
- Support the identification of objects (hence classes) that participate in use cases
- Each message in a collaboration diagram has a sequence number.
- The top-level message is numbered 1. **Messages sent during the same call have the same decimal prefix** but suffixes of 1, 2, etc. according to when they occur.

# Primary elements in Collaboration Diagram

## 1) Objects

- Objects in collaboration come in two flavors- Supplier and client
- Supplier objects supply methods that is being called and therefore receive message
- Client objects call methods on supplier objects, and therefore send message
- An object is represented by an object symbol showing the name of the object and its class underlined, separated by a colon:
- Each object in the collaboration is named and has its class specified
- Normally we create a collaboration diagram with objects first and specify their classes later.

**Object_name : class_name**

# Primary elements in Collaboration Diagram

## b) Actors

- Normally an actor instance occurs in the collaboration diagram, as the invoker of the interaction..
- Each Actor is named and has a role
- One actor will be the initiator of the use case

## c) Links

- Links connect objects and actors and are instances of associations and each link corresponds to an association in the class diagram
- A link is a relationship among objects across which messages can be sent.
- In collaboration diagrams, a link is shown as a solid line between two objects.

# Primary elements in Collaboration Diagram

## d) Messages

- A message is a communication between objects that conveys information with the expectation that activity will ensue.
- In collaboration diagrams, a message is shown as a labeled arrow placed near a link.
- The message is directed from sender to receiver
- The receiver must understand the message
- The association must be navigable in that direction

# Primary elements in Collaboration Diagram

## ❑Messages

### ❑ Iterating Messages

- Collaboration diagrams use syntax similar to sequence diagrams to indicate that either a message iterates (is run multiple times) or is run conditionally
    - An asterisk (*) indicates that a message runs more than once
    - Or the number of times a message is repeated can be shown by numbers (for example, 1..5)

### ❑ Conditional Messages

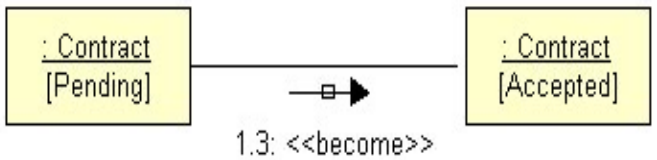
- To indicate that a message is run conditionally, prefix the message sequence number with a conditional [guard] clause in brackets [ x = true ]:

  [IsMediaOverdue ]

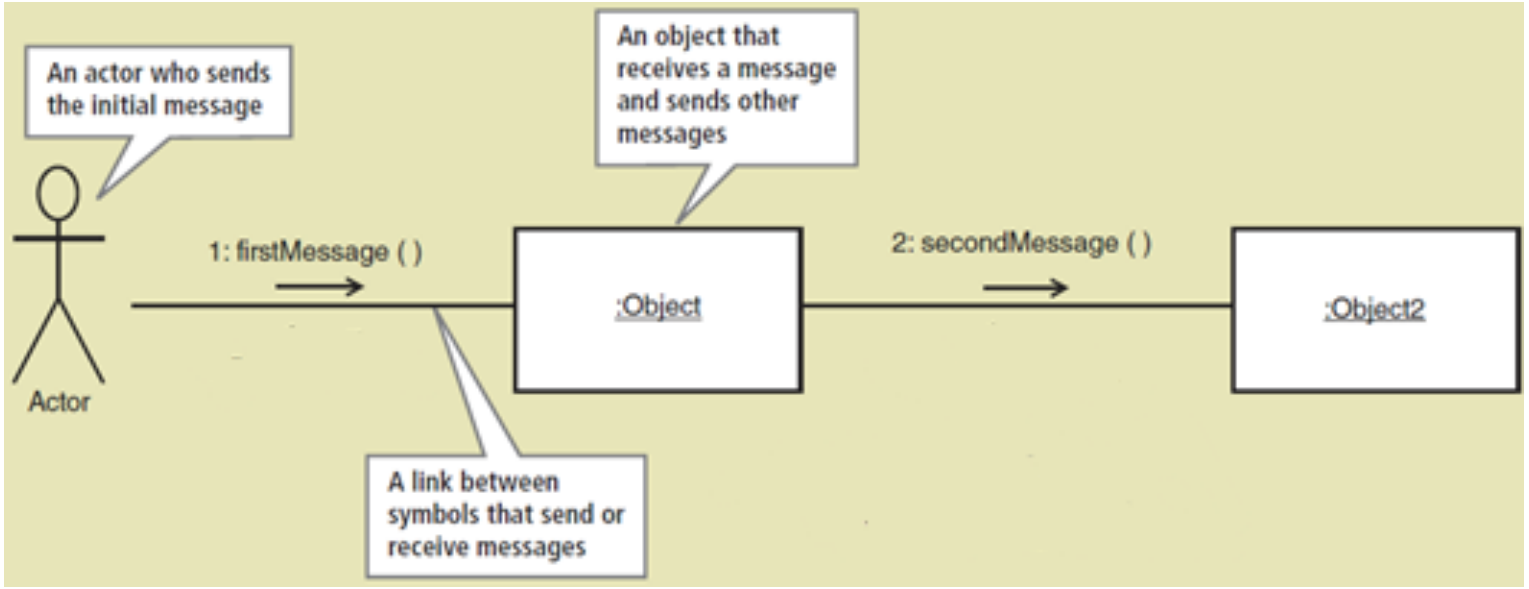
- This indicates that the message is sent only if the condition is met

# Object state

- **Creation and Deletion**
- Unlike sequence diagrams, you don't show an object's lifeline in a collaboration diagram
- If you want to indicate the lifespan of an object in a collaboration diagram, you can use create and destroy messages to show when an object is instantiated and destroyed
- **Objects Changing State**
- State of on object can be indicated
- Initial state is indicated with <<create>>
- If an object changes significantly during an interaction, you can add a new instance of the object to the diagram, draw a link between them and add a message with the stereotype <<become>>
-



# …….. Primary elements in Collaboration Diagram

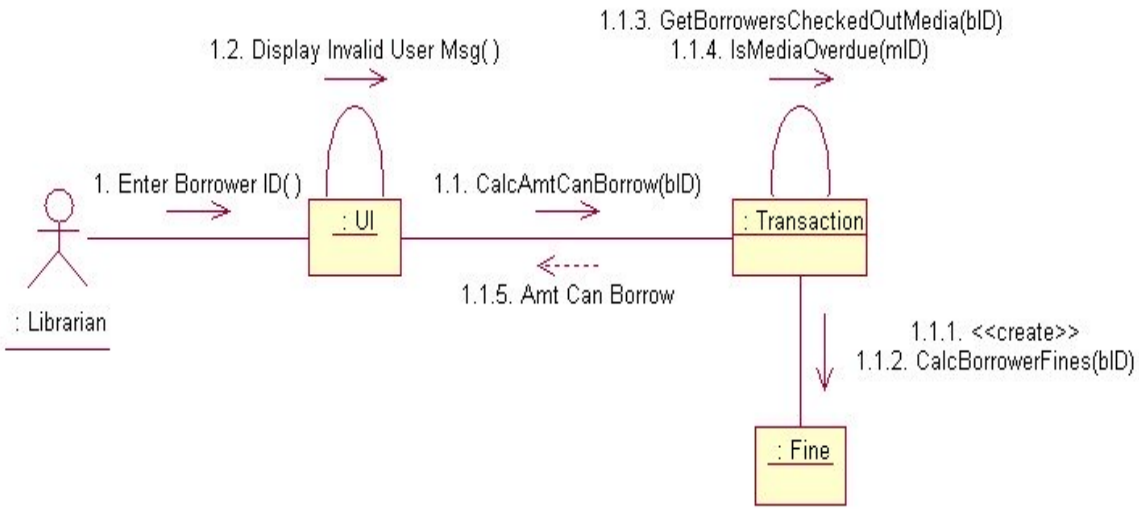


# Steps for Creating Collaboration Diagrams

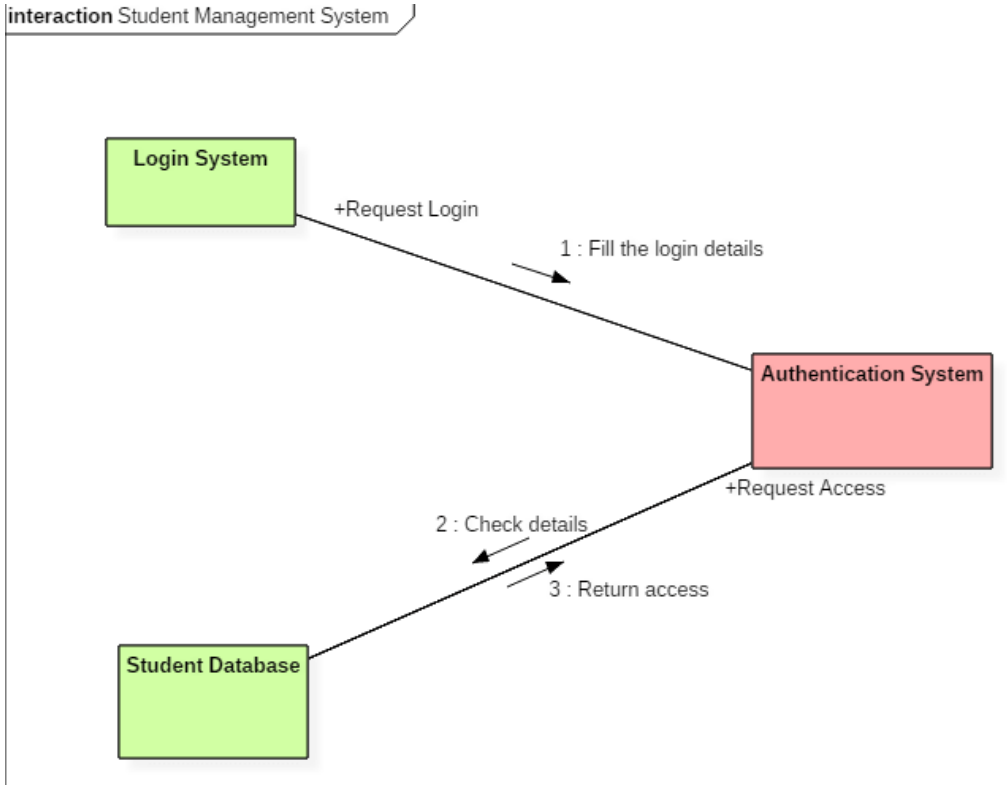
- Step-1 Find the domain Class
  - ➢ Use the high level use case description to find the noun such as person or thing
- Step-2 Draw an object symbol for each of the domain class
- Step-3 Draw Boundary objects
- Step-4 Draw Actor
- Step-5 Add Associations
- Step -6 Adding message

# Collaboration Diagram:- Example

- the figure below shown documents the interaction that occurs between business objects when determining how many items a borrower can check out of the library.
- it's easier to see the relationships between objects by looking at the collaboration diagram.
- transaction object acts as a supplier to the UI (user interface) client object.
  in turn, the fine object is a supplier to the transaction client object.



# Collaboration Diagram:- Example



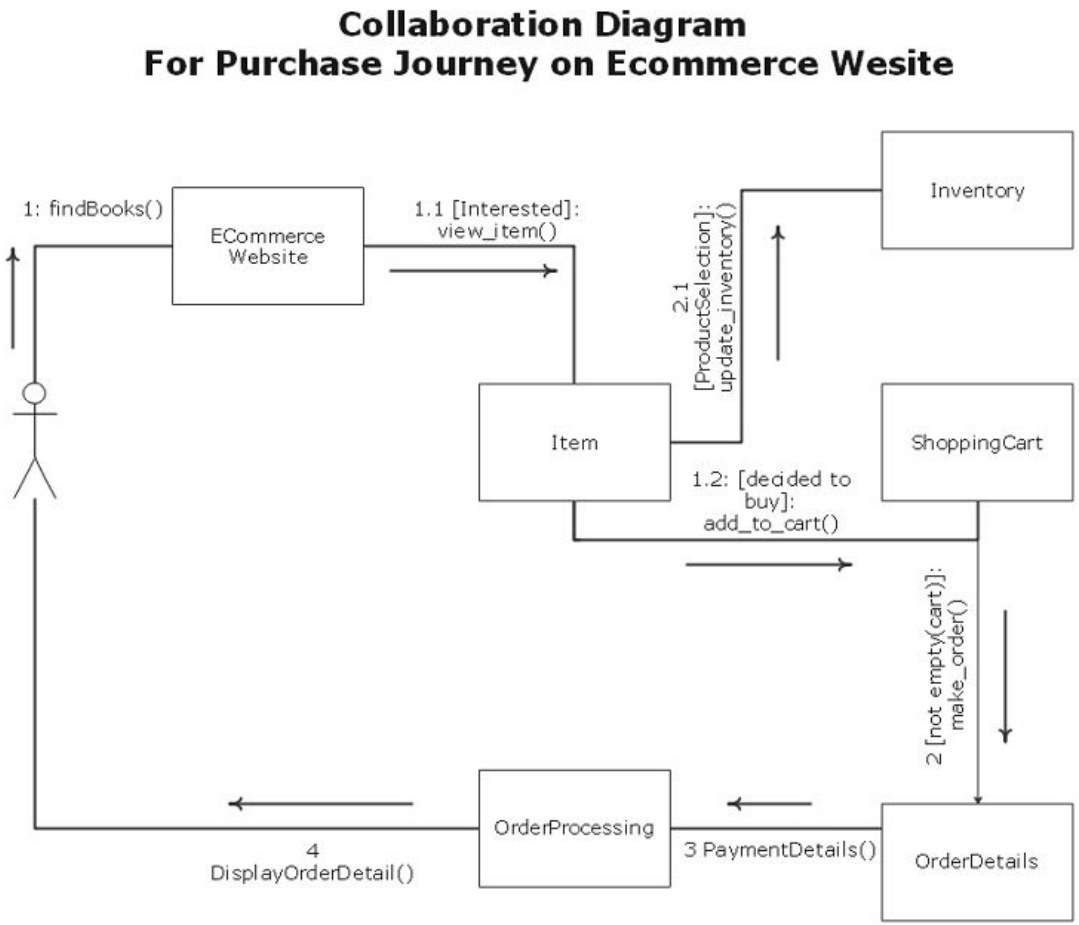

**Collaboration diagram Example**

- Following diagram represents the sequencing over student management system:

➢ The above collaboration diagram represents a student information management system. The flow of communication in the above diagram is given by,

1. A student requests a login through the login system.
2. An authentication mechanism of software checks the request.
3. If a student entry exists in the database, then the access is allowed; otherwise, an error is returned.

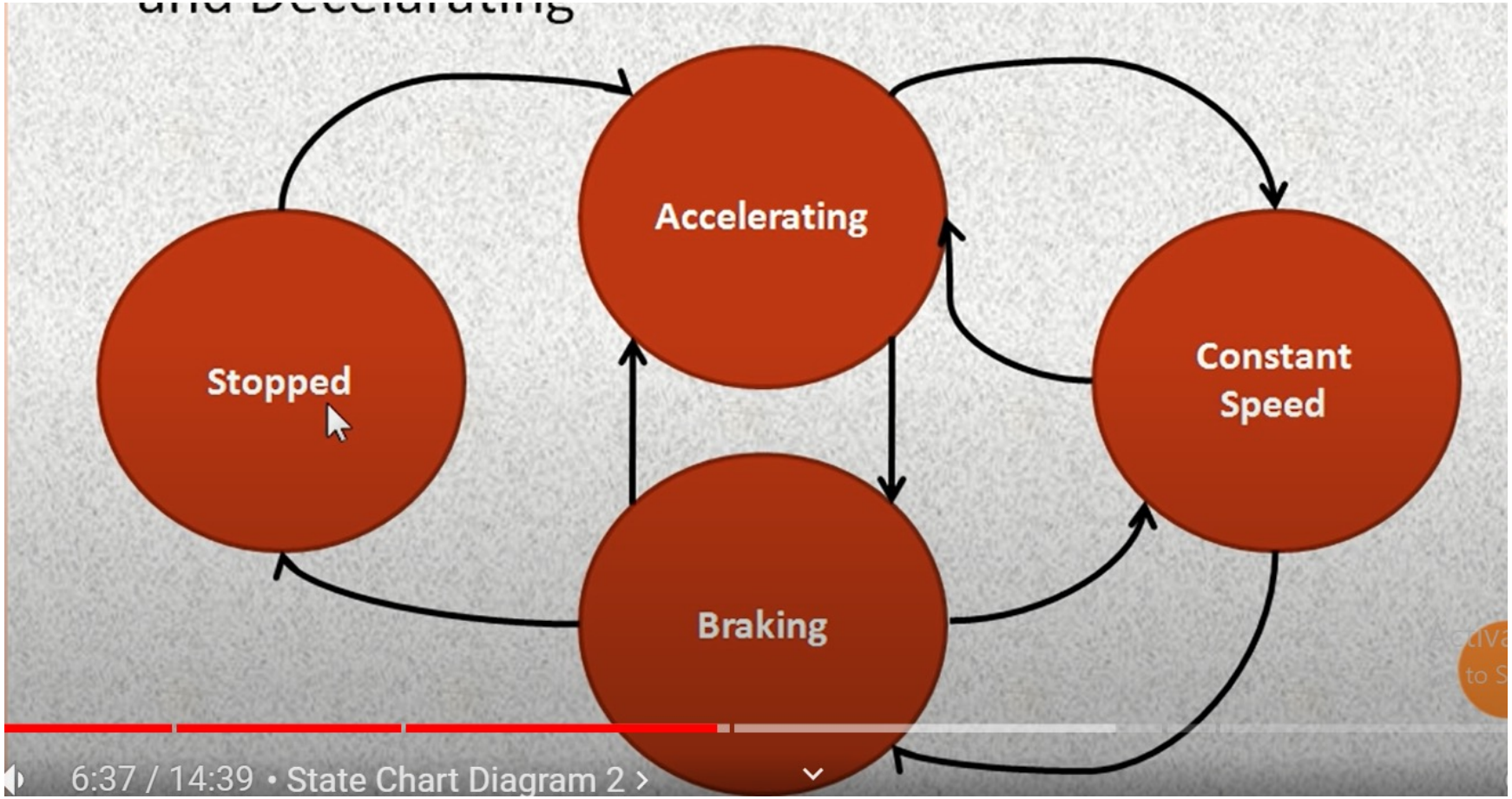
# …..Collaboration Diagram Example




This illustration depicts the order of purchase from the ‘Item’ object to the message of ‘order details’ being displayed

# Collaboration vs Sequence Diagram

- In reality, sequence diagrams and collaboration diagrams show the **same information**, but just present it differently
- We can turn communication diagram into sequence diagrams and vice versa
- Sequence diagram emphasize in  time ordering
- collaboration diagrams emphasize in structural ordering
- Communication diagram are very useful for visualizing relationship between objects that collaborate with each other to perform a specific task. Which is difficult to determine from sequence diagram
- collaboration diagrams  is not used as often as sequence diagrams but are closely related


Accelerating
Stopped
Constant Speed
Braking
6:37 / 14:39 • State Chart Diagram 2