

# UNIFIED MODELING LANGUAGE



# Objectives

- UML Architecture
- Use case Diagram
- Class Diagram
- Sequence Diagram
- Activity Diagram
- State chart Diagram



# Modeling



A way of thinking about the problems using models organized around the real world ideas.

A modeling method comprises of a language and a procedure for using the language to construct models.

It is the way to visualize your design and check it against requirements before your crew starts to code.

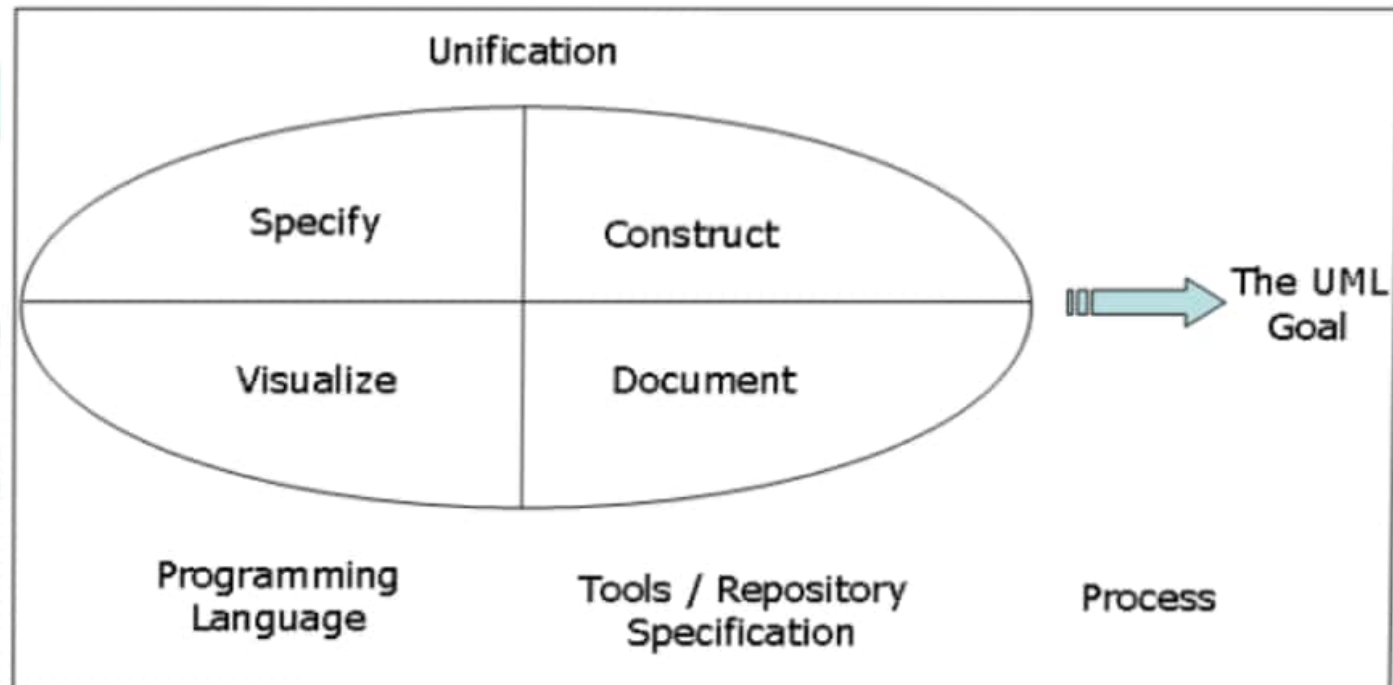
# What is UML?

UML – Unified Modeling Language is a standard language for specifying, visualizing, constructing, and documenting the artifacts of software systems, business modeling and other non-software systems

UML is a pictorial language used in making of software blue prints.

Applies to modeling and systems

# Primary goal of UML



# UML Architecture

## User View

- Models which define a solution to a problem as understood by the client or stakeholders.
- Use Case Diagram

## Structural View

- Models which provide the static, structural dimensions and properties of the modelled system.
- Class diagram, Object diagram

## Behavioural View

- Models describe the behavioural and dynamic features and methods of the modelled system.
- Interaction diagram(sequence diagram, Collaboration diagram), Activity diagram, state chart diagram



# UML Architecture

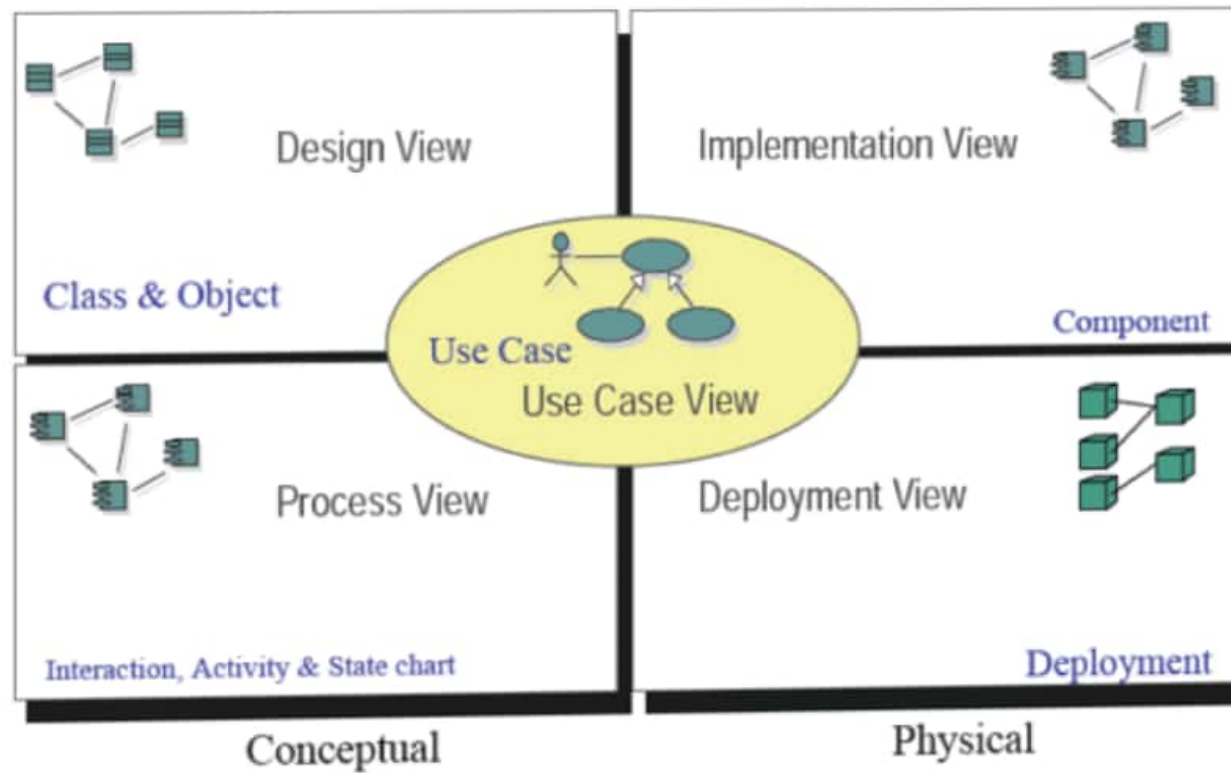
## Implementation View

- View combines the structural and behavioral dimensions of the solution realization or implementation.
- Component diagram, package diagram

## Environment view

- Models describe both structural and behavioral dimensions of the domain or environment in which the solution is implemented.
- Deployment diagram

# UML





# UML Diagrams

UML Diagram is classified as

- Use case Diagram
- Class Diagram
- Interaction Diagram
- Activity Diagram
- StateChart Diagram

# Use Case Diagram

---

Depicts different users and the functions of the system.

---

Exposes the requirements of the system.

---

Actors interact with a use case and the functions are called as use cases.

---

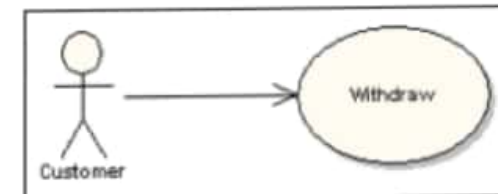
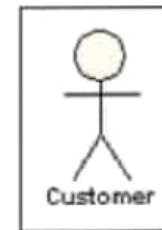
It describes the interaction between the actor and the use case.

# Use Case Diagram Components

**Actor** : which represent users of a system, including human users and other systems.

**Usecase** : which represent functionality or services provided by a system to users.

**Association** :Associations between actors and use cases are indicated by solid lines. An association exists whenever an actor is involved with an interaction described by a use case.

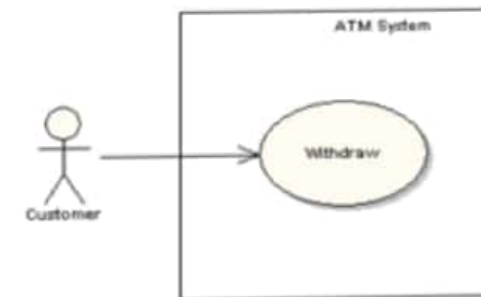
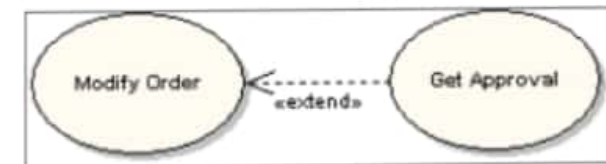
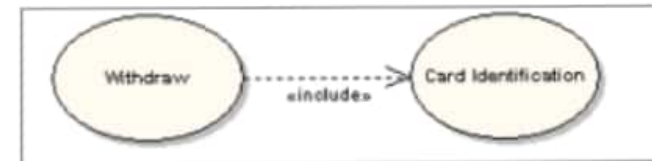


# Use Case Diagram

**Include:** Use cases may contain the functionality of another use case as part of their normal processing. In general it is assumed that any included use case will be called every time the basic path is run.

**Extend :** One use case may be used to extend the behavior of another; this is typically used in exceptional circumstances.

**System Boundary :** It is usual to display use cases as being inside the system and actors as being outside the system.



# Use Case Diagram Specification

Use case name - States the use case name

Use case actor - Actor interacting with the use case

Preconditions - A state of the system that must be present before a use case is performed

Basic flow of events - Describes the ideal, primary behavior of the system

Alternative flow of events - Describes exceptions or deviations from the basic flow

Post conditions - A list of possible states for the system immediately after a use case is finished

Extension points - A point in the use-case flow of events at which another use case is referenced



# Class Diagram

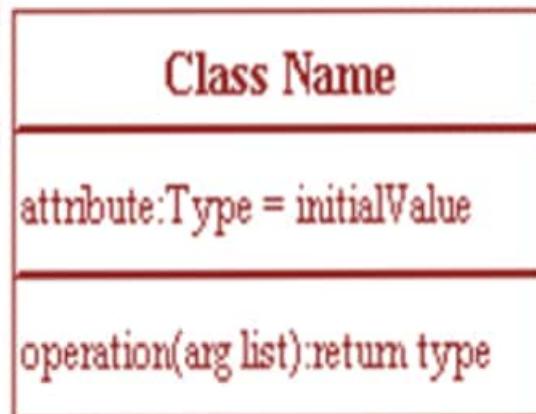
Class Diagrams describe the static structure of a system, or how it is structured rather than how it behaves.

These diagrams contain the following elements.

- Classes - represent entities with common characteristics or features.
- Features include attributes, operations and associations.
- Associations - represent relationships that relate two or more classes



# Class Diagram Notation



## Order

- orderId : int
- productList : List
- orderDate : String
- price : long
- delivered : boolean = false

- + placeOrder(list,price) : void
- + shipProduct(orderid) : boolean

# Class Diagram



Aggregation

## Relationship

- Relationships connect two or more classes
- Relationships among classes can be
  - Association
  - Inheritance or Generalization
  - Aggregation
  - Composition
  - Dependency
  - Realization

Composition

Inheritance

Realization

# Class Diagram

- The association is a simple relationship.
- It connects two classes, denoting the structural relationship between them.
- An association is shown as a line connecting the related classes.



- **Multiplicity**
  - This association relationship indicates that (at least) one of the two related classes make reference to the other.
  - Indicates whether or not an association is mandatory.
  - Provides a lower and upper bound on the number of instances.



# Class Diagram



## Multiplicity Indicators

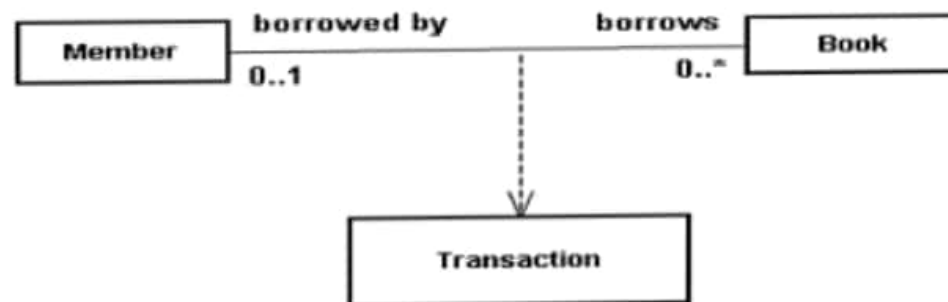
Exactly one	1
Zero or more (unlimited)	* (0..*)
One or more	1..*
Zero or one (optional association)	0..1
Specified range	2..4
Multiple, disjoint ranges	2, 4..6, 8



# Class Diagram

## Association class

- We can also show an association class in an association relationship.
- An association class is one that makes it possible for the association to exist between two other classes.

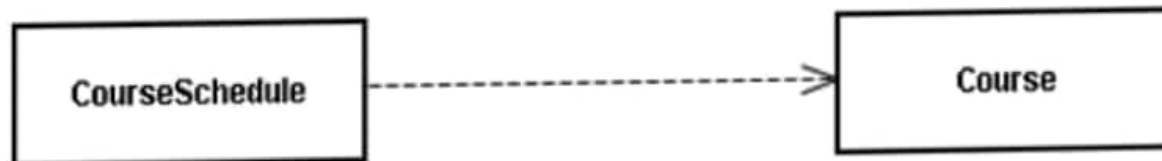




# Class Diagram

## Dependency

- Is the “using” or “uses” relationship where one element is using another element to get its task done.
- A change in one element affects another element that uses it. i.e. if one changes, then it affects the other element. The reverse need not be true.
- The dependency from *CourseSchedule* to *Course* exists because *Course* is used in both the **add** and **remove** operations of *CourseSchedule*.





# Class Diagram

## Aggregation

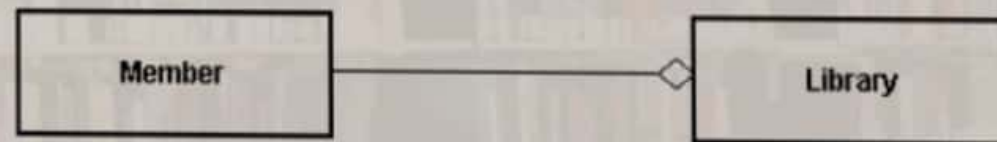
- Aggregation is a variant of the "has a" association relationship, it is more specific than association
- It is an association that represents a part-whole or part-of relationship
- Examples:
  - A room has a door.
  - Building has rooms.
  - Vehicle has an engine.
  - Company has departments
  - Book has chapters.

## Two forms of Aggregation:

- Simple Aggregation
- Composition

# Class Diagram

- **Aggregation – “HAS-A” relationship. The part remains even after the whole is destroyed**



- **Composition – Stronger form of aggregation. The Whole is solely responsible for the part**



# Class Diagram

## Generalization

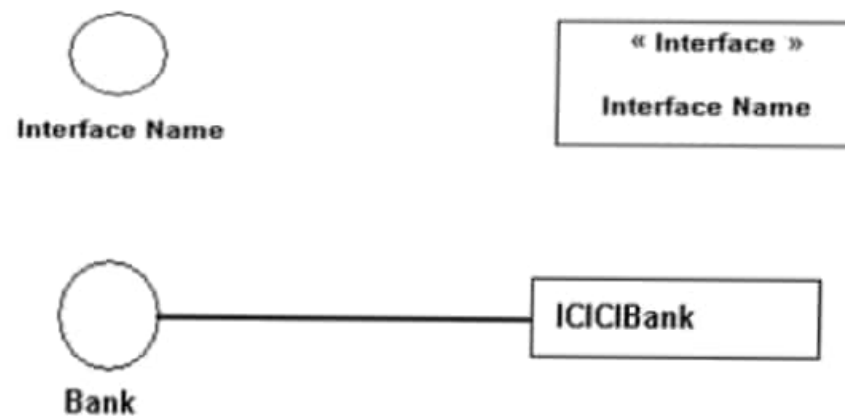
- Generalization consists of the super class and sub-class relationship.
- The super class represents a general concept while the sub class represents a more specific concept.
- Also called as “is a type of”.
- Examples:
  - Rose is a type of flower.
  - Windows is a kind of operating system.



# Class Diagram

## Realization

- The relationship between the class and the interface.
- An interface is a contract which can be implemented by classes which may /may not be related
- All methods inside the interface are public and abstract



Which UML model depicts the interactions between the objects by passing series of messages

- ☐ class diagram
- ☐ usecase diagram
- ☐ state chart diagram
- ☒ sequence diagram

**Correct**

That's right! You chose the correct response.

There are two use cases, viewEmployeeID and validateEmployeeID. What is the relationship between these 2 usecases.




- ☒ include
- ☐ mandatory
- ☐ extends
- ☐ excludes

**Correct**

That's right! You chose the correct response.



## Who is an actor

-  ☒ An actor can be a human, device or an external system
- ☐ An actor only gives but does not receive the information
- ☐ An actor only receives information

**Correct** 

That's right! You chose the correct response.

## Which of the following are true about the class diagram

- ☐ depicts the behaviour of the system
- ☐ depicts transition of the objects of the classes
- ☒ depicts the relationship between the classes
- ☐ depicts the life cycle of a class

**Correct**

That's right! You chose the correct response.

Which of the following multiplicity indicator is used to denote the optional association?



☐ 0..1

☒ \*(0..\*)

☐ \*0..1

☐ 1..\*

**Incorrect**

You did not choose the correct response.

# Interaction Diagram

Visualize the interactive behavior of the system

Purpose of interaction diagram

- To capture dynamic behavior of a system.
- To describe the message flow in the system.
- To describe structural organization of the objects.
- To describe interaction among objects.

Interactive behaviour is represented in UML by two diagrams known as Sequence diagram and Collaboration diagram.

# Sequence Diagram

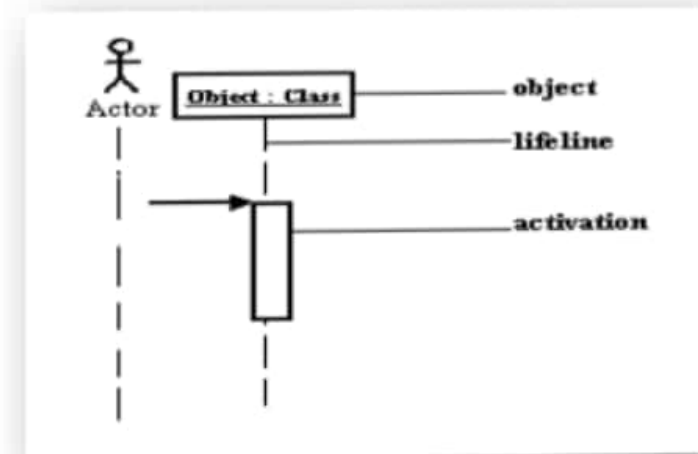
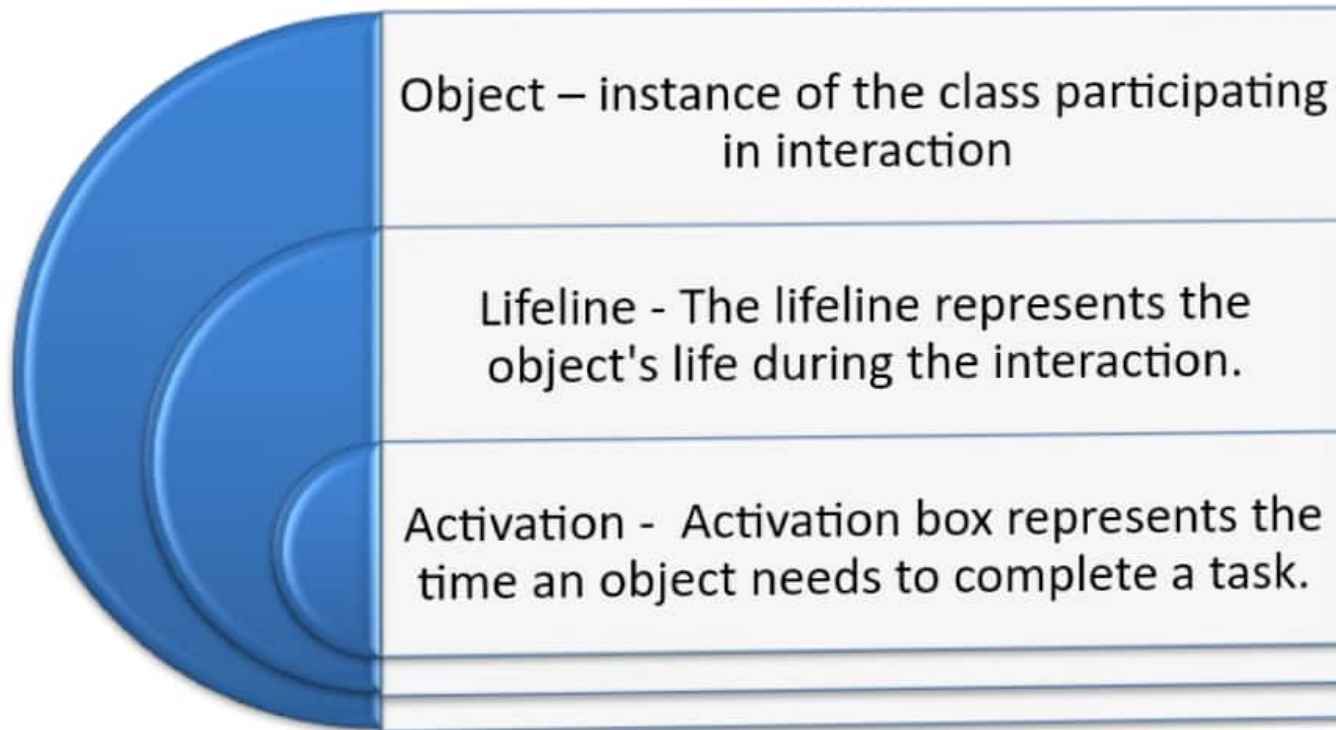
A sequence diagram shows object interactions arranged in time sequence

It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality

Graphically a sequence diagram is a table that shows objects arranged along X axis and messages, ordered in increasing time along the Y axis.

It can model simple sequential flow, branching, iteration and concurrency.

# Sequence Diagram - Notation



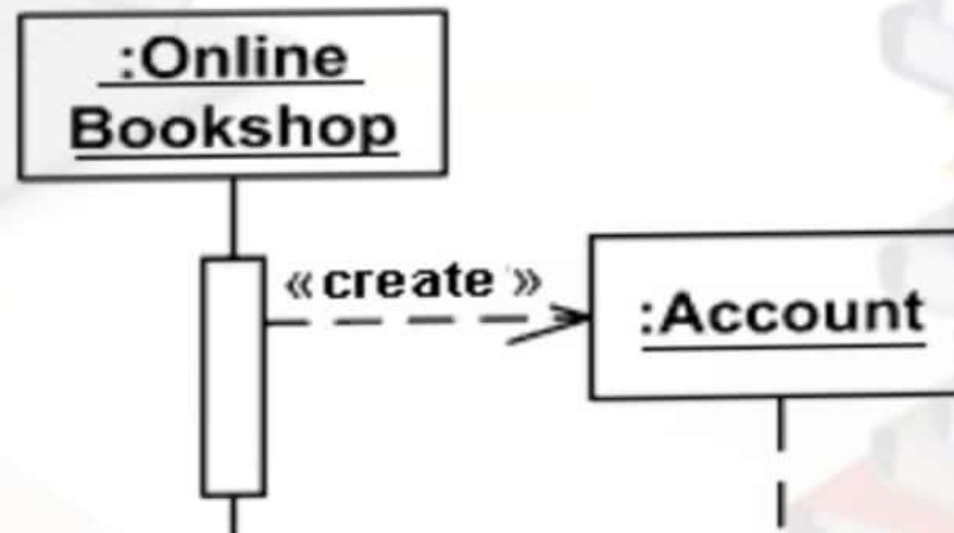


# Sequence Diagram - Notation

Messages

Create

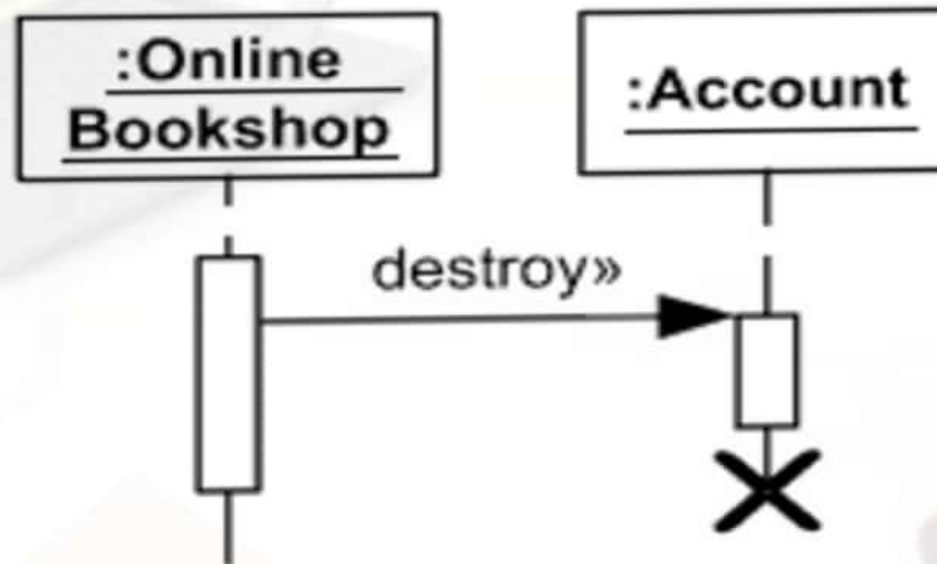
- Create message is a kind of message that represents the instantiation of (target) lifeline.



# Sequence Diagram - Notation

## Destroy

- Destroy message is a kind of message that represents the request of destroying the life cycle of target lifeline.



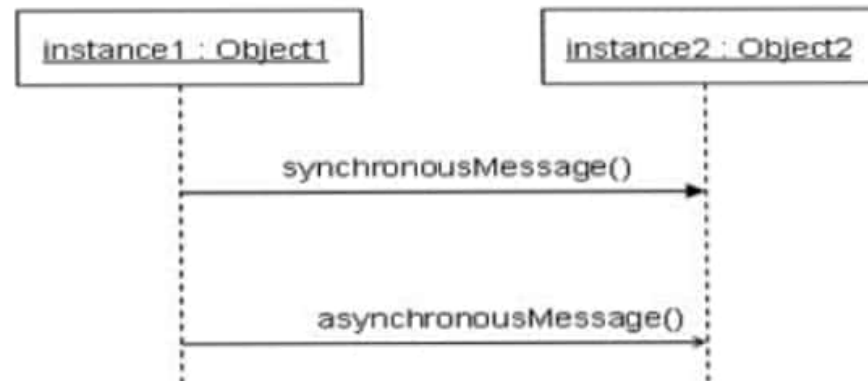
# Sequence Diagram - Notation

## Synchronous call

- It must wait until the response is received , such as invoking a subroutine.

## Asynchronous call

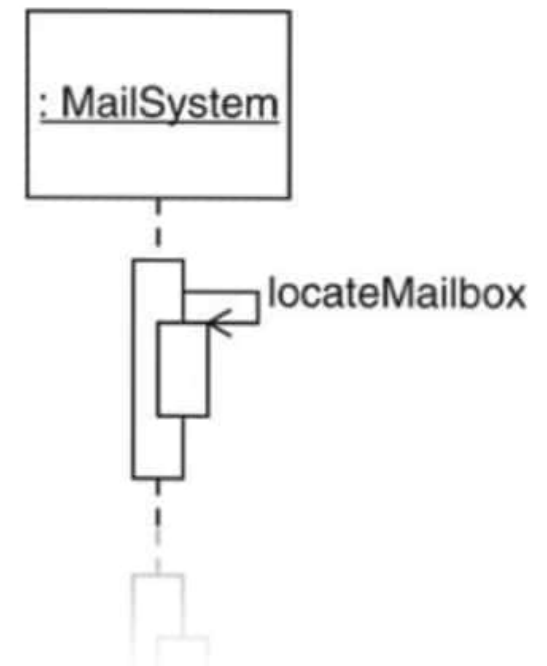
- It can continue processing and doesn't have to wait for a response.



# Sequence Diagram - Notation

## Self call

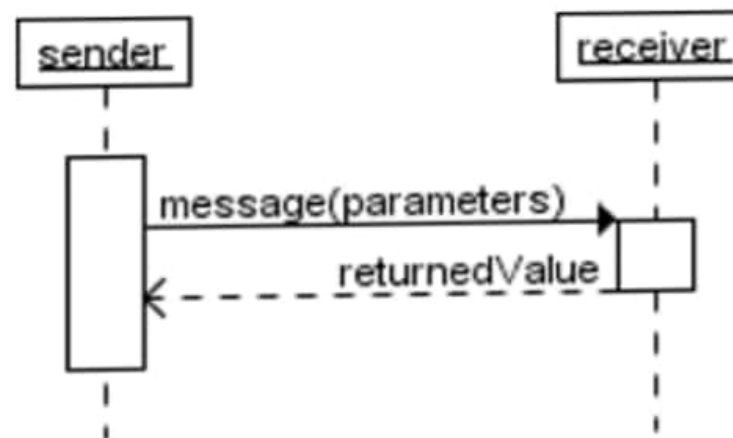
- A self message can represent a recursive call of an operation, or one method calling another method belonging to the same object.



# Sequence Diagram - Notation

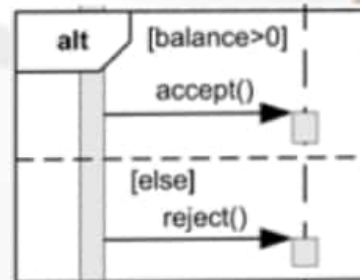
## Return message

- Return message is a kind of message that represents the passing of information back to the caller to a corresponding former message.

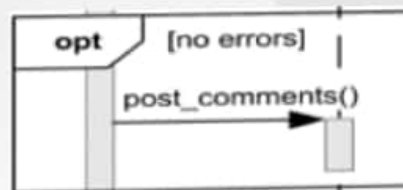


# Sequence Diagram - Notation

- **Alt** – shows the true and the false logic flow



**Opt** - The interaction operator opt means that the combined fragment represents a choice of behavior where either the (sole) operand happens or nothing happens.

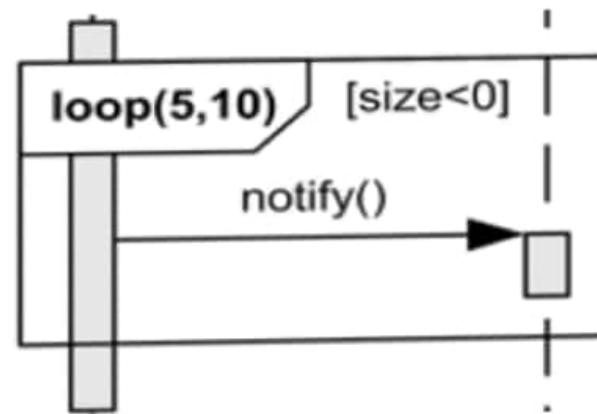




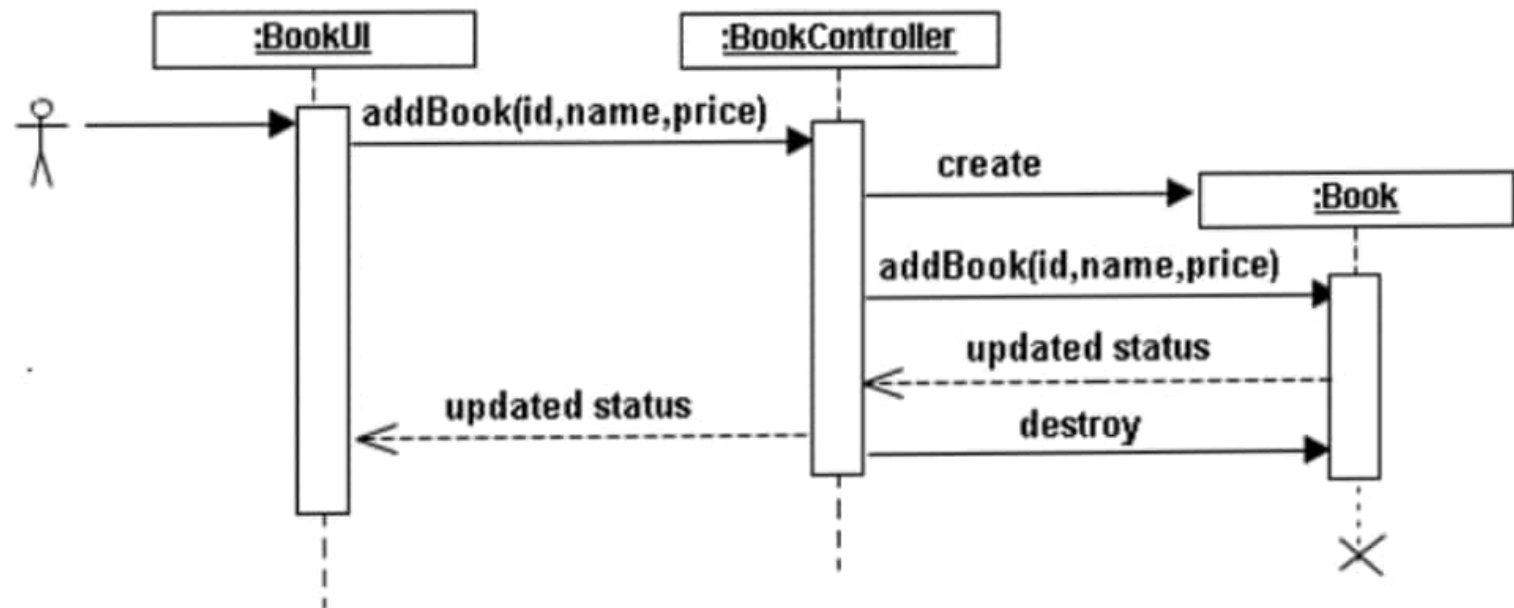
# Sequence Diagram - Notation

The loop operand will be repeated a number of times.

The loop is expected to execute minimum 5 times and no more than 10 times. If guard condition [size<0] becomes false, loop terminates regardless of the minimum number of iterations specified.



# Sample Sequence Diagram



# Activity Diagram

An activity diagram illustrates the dynamic nature of a system by modeling the flow of control from activity to activity.

An activity represents an operation on some class in the system that results in a change in the state of the system.

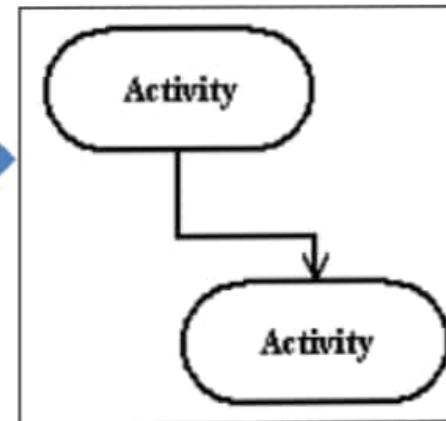
Activity diagrams are used to model workflow or business processes and internal operation

## Notations

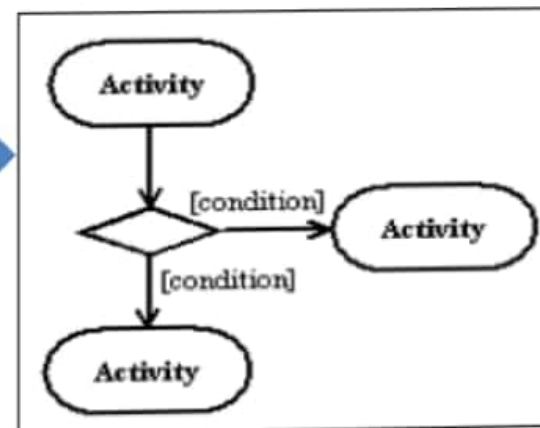
- Initial state
- Final state
- Action state

# Activity Diagram

Action  
flow



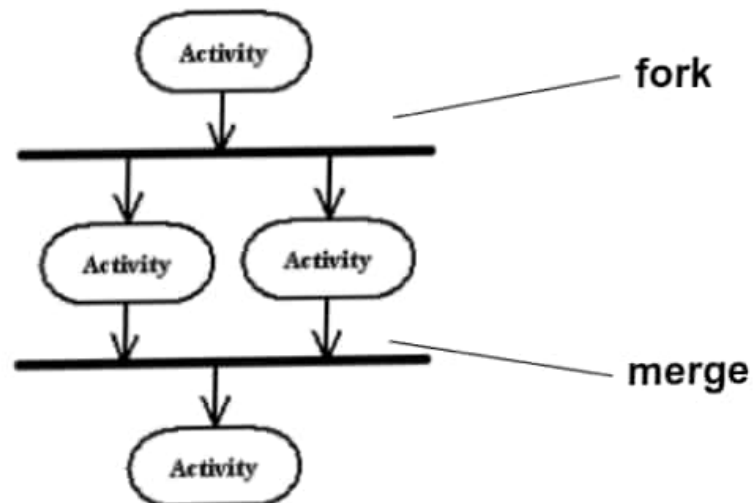
Branching



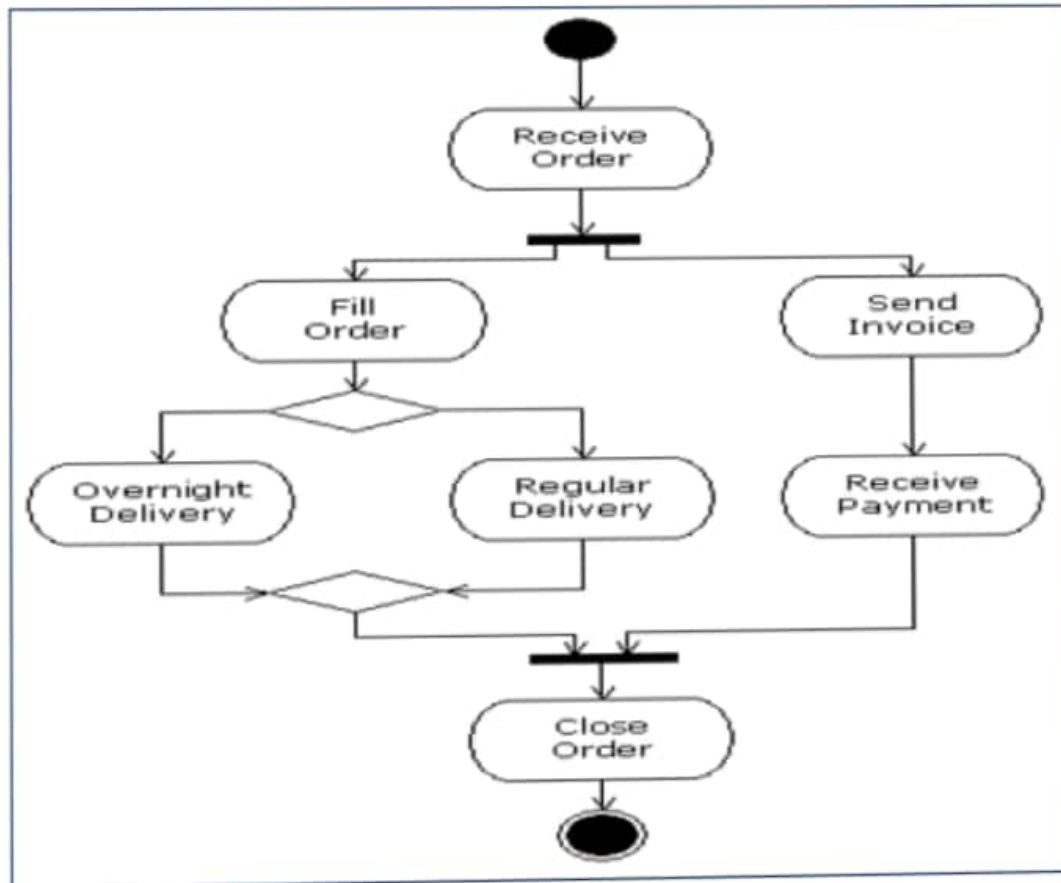
# Activity Diagram

## Synchronization

- A synchronization bar helps illustrate parallel transitions. Synchronization is also called forking and joining.



# Activity Diagram





# State Chart Diagram

Used to model dynamic nature of a system.

They define different states of an object during its lifetime.

Purpose of State chart diagram:

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

# State Chart Diagram Notation

- **State** - State represents situations during the life of an object.
- **Transition** - A solid arrow represents the path between different states of an object



# State Chart Diagram

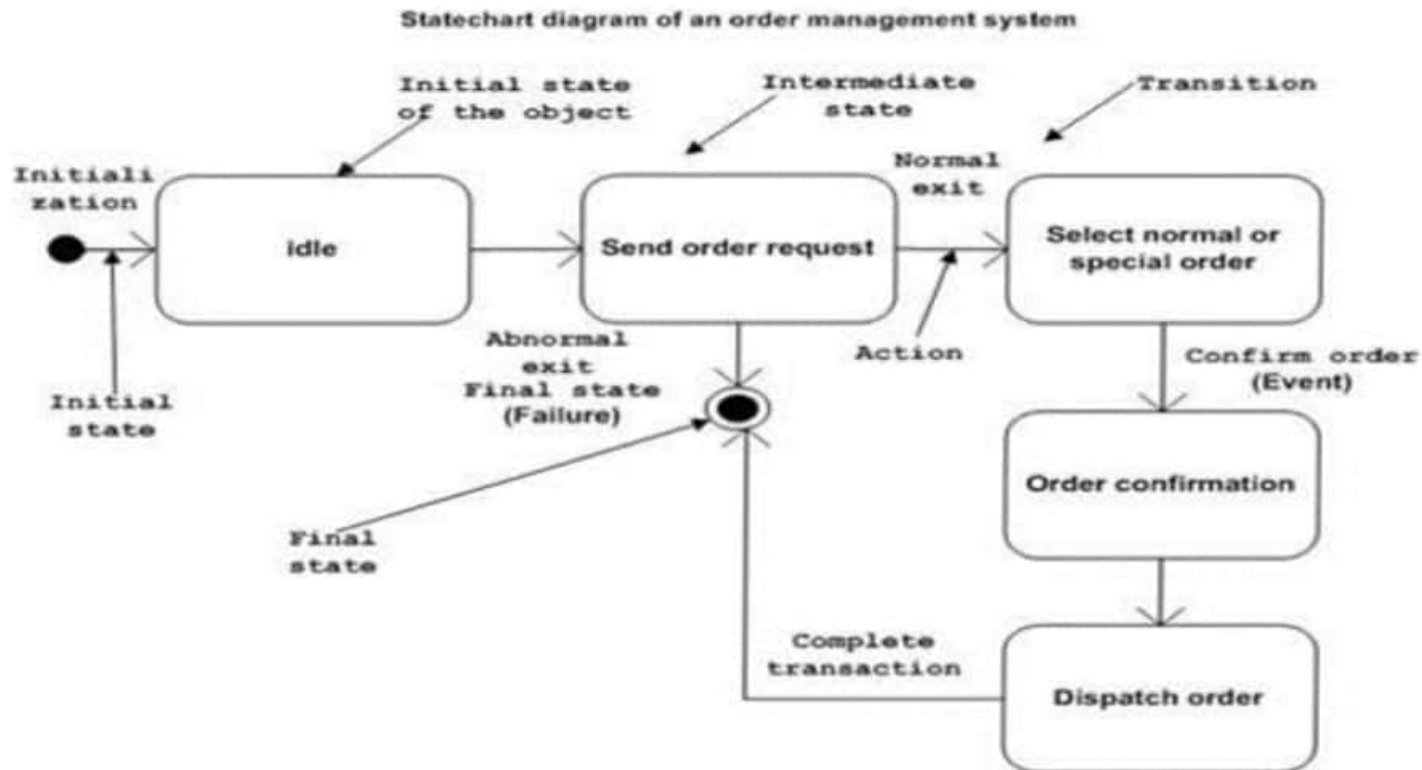
Initial State -A filled circle followed by an arrow represents the object's initial state.



Final State - An arrow pointing to a filled circle nested inside another circle represents the object's final state.



# State Chart Diagram



## Sequence diagram is intended

- ☐ all the options
- ☐ to represent the static aspect of the system
- ☒ to show the logical execution flow of the business process
- ☐ to represent the temporal aspect of the system

**Incorrect**

You did not choose the correct response.

Activity diagram does not contain

- ☐ action
- ☐ Activity state
- ☒ attributes
- ☐ transition

**Correct**

That's right! You chose the correct response.



In statechart diagram the objects state is changed because of

- ☐ actor
- ☐ none of the option
- ☒ event
- ☐ class

**Correct**

That's right! You chose the correct response.

## Activity diagram

- ☐ none of the options
- ☒ represents the flow of control from activity to activity
- ☐ represents the behaviour of the individual objects
- ☐ represents normal and alternate flow of business

**Correct**

That's right! You chose the correct response.

Fork is used for representing

- ☐ Iterations
- ☐ All the operations
- ☐ Sequential flow
- ☒ Concurrent flow

**Correct**

That's right! You chose the correct response.

Notation in sequence diagram which is used to represent the period of time the object is active

- ☐ Life line
- ☐ Object
- ☒ Focus of control
- ☐ synchronous message

**Correct**

That's right! You chose the correct response.

# Summary

- UML Architecture
- Use case Diagram
- Class Diagram
- Sequence Diagram
- Activity Diagram
- State chart Diagram





**THANK YOU**