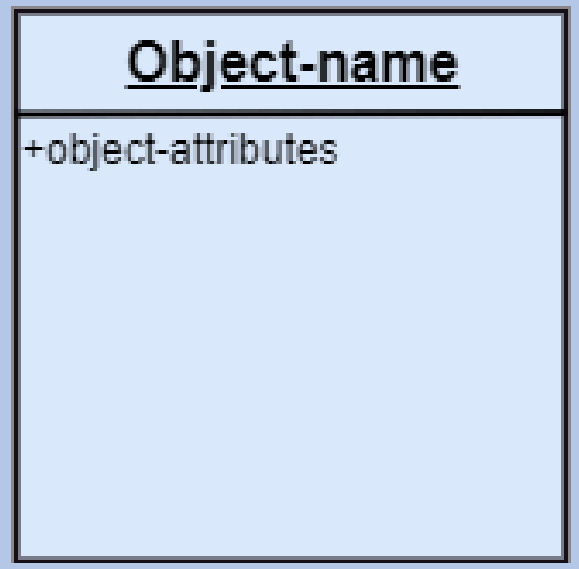
Unified Modelling Language (UML)-II

Ruchita Shah

UML Object Diagram

- Object diagrams are dependent on the class diagram as they are derived from the class diagram
- It represents an instance of a class diagram
- The objects help in portraying a static view of an object-oriented system at a specific instant
- Both the object and class diagram are similar to some extent; the only difference is that the class diagram provides an abstract view of a system
- It helps in visualizing a particular functionality of a system

UML Object Diagram



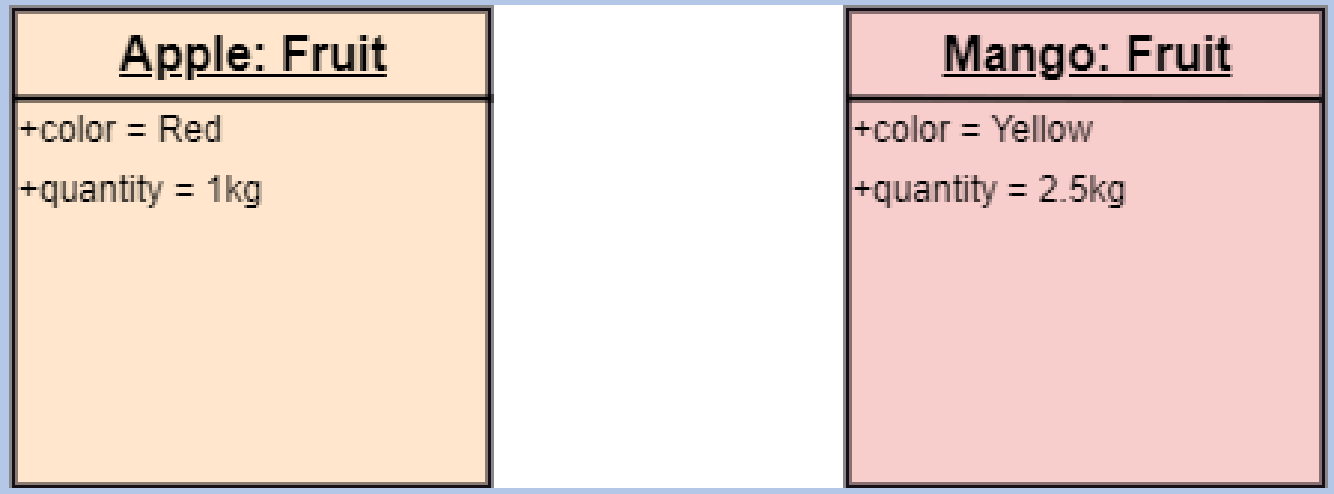
Purpose of Object Diagram

- Same purpose as that of a class diagram
- The class diagram provides an abstract view which comprises of classes and their relationships, whereas the object diagram represents an instance at a particular point of time
- The object diagram is actually similar to the concrete (actual) system behavior
- The main purpose is to depict a static view of a system

Purpose of Object Diagram

- It is used to perform forward and reverse engineering
- It is used to understand object behavior and their relationships practically
- It is used to get a static view of a system
- It is used to represent an instance of a system

Purpose of Object Diagram



How to draw an Object Diagram?

1. All the objects present in the system should be examined before start drawing the object diagram
2. Before creating the object diagram, the relation between the objects must be acknowledged
3. The association relationship among the entities must be cleared already
4. To represent the functionality of an object, a proper meaningful name should be assigned
5. The objects are to be examined to understand its functionality

Applications of Object diagrams

1. To build a prototype of a system
2. To model complex data structures
3. To perceive the system from a practical perspective
4. Reverse engineering

Class vs. Object diagram

Class Diagram	Object Diagram
It depicts the static view of a system.	It portrays the real-time behavior of a system.
Dynamic changes are not included in the class diagram.	Dynamic changes are captured in the object diagram.
The data values and attributes of an instance are not involved here.	It incorporates data values and attributes of an entity.
The object behavior is manipulated in the class diagram.	Objects are the instances of a class

UML Component Diagram

- It is used to break down a large object-oriented system into the smaller components, so as to make them more manageable
- It models the physical view of a system such as executables, files, libraries, etc. that resides within the node
- It visualizes the relationships as well as the organization between the components present in the system
- It helps in forming an executable system

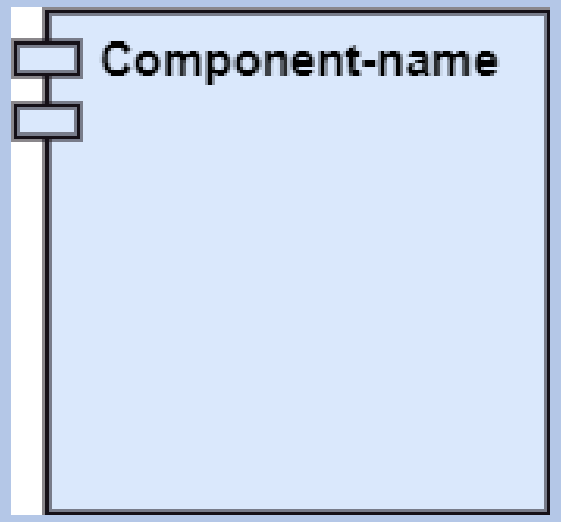
UML Component Diagram

- A component is a single unit of the system, which is replaceable and executable
- The implementation details of a component are hidden, and it necessitates an interface to execute a function
- It is like a black box whose behavior is explained by the provided and required interfaces

UML Component Diagram

- Notation of a Component Diagram

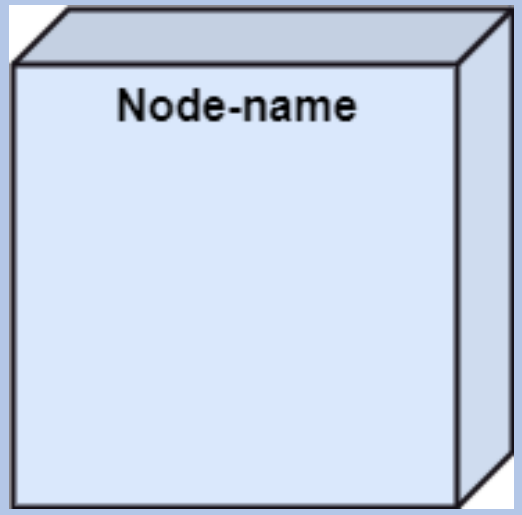
  A Component



UML Component Diagram

- Notation of a Component Diagram

A Node : computational resource upon which UML artifacts may be deployed for execution



Purpose of a Component Diagram

- It describes all the individual components that are used to make the functionalities, but not the functionalities of the system
- It visualizes the physical components inside the system
- The components can be a library, packages, files, etc.
- The component diagram also describes the static view of a system, which includes the organization of components at a particular instant
- The collection of component diagrams represents a whole system

1. It envisions each component of a system
2. It constructs the executable by incorporating forward and reverse engineering
3. It depicts the relationships and organization of components

Why use Component Diagram?

- It is used to depict the functionality and behavior of all the components present in the system, unlike other diagrams that are used to represent the architecture of the system, working of a system, or simply the system itself
- In UML, the component diagram portrays the behavior and organization of components at any instant of time
- The system cannot be visualized by any individual component, but it can be by the collection of components

1. It portrays the components of a system at the runtime
2. It is helpful in testing a system
3. It envisions the links between several connections

When to use a Component Diagram?

- To divide a single system into multiple components according to the functionality
- To represent the component organization of the system

How to Draw a Component Diagram?

- Once the system is designed employing different UML diagrams, and the artifacts are prepared, the component diagram is used to get an idea of implementation
- It plays an essential role in implementing applications efficiently.
- Following are some artifacts that are needed to be identified before drawing a component diagram:

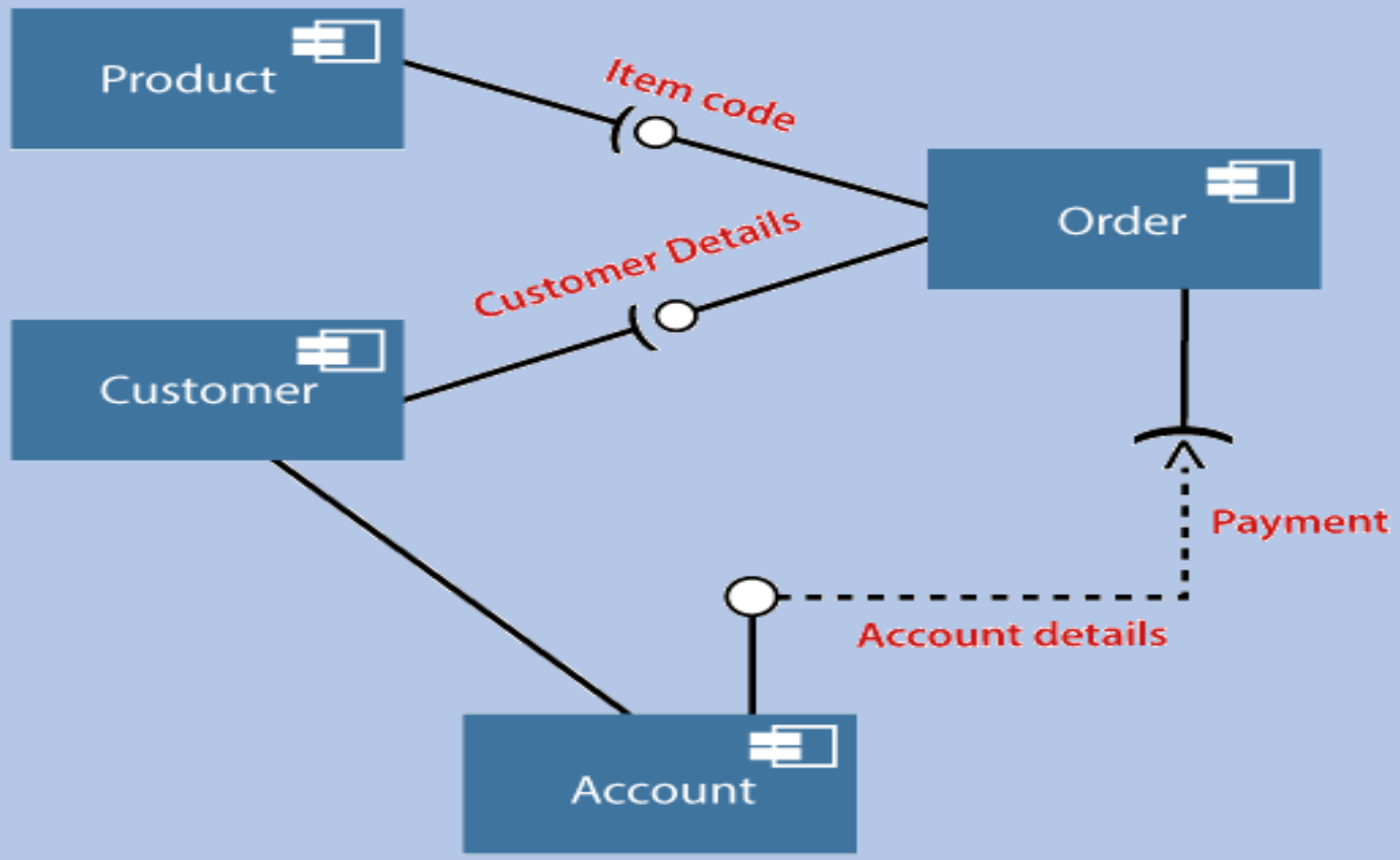
1. What files are used inside the system?
2. What is the application of relevant libraries and artifacts?
3. What is the relationship between the artifacts?

How to Draw a Component Diagram?

- Some points that are needed to be kept in mind after the artifacts are identified:

1. Using a meaningful name to ascertain the component for which the diagram is about to be drawn
2. Before producing the required tools, a mental layout is to be made
3. To clarify the important points, notes can be incorporated

A component diagram for an online shopping system



Where to use Component Diagrams?

- To model the components of the system
- To model the schemas of a database
- To model the applications of an application
- To model the system's source code

UML Deployment Diagram

- Deployment diagram visualizes the physical hardware on which the software will be deployed
- It portrays the static deployment view of a system
- It involves the nodes and their relationships
- It ascertains how software is deployed on the hardware
- It maps the software architecture created in design to the physical system architecture, where the software will be executed as a node
- Since it involves many nodes, the relationship is shown by utilizing communication paths

Purpose of Deployment Diagram

- Main purpose of the deployment diagram is to represent how software is installed on the hardware component
- Depicts in what manner a software interacts with hardware to perform its execution
- Deployment diagram and the component diagram are closely interrelated to each other as they focus on software and hardware components

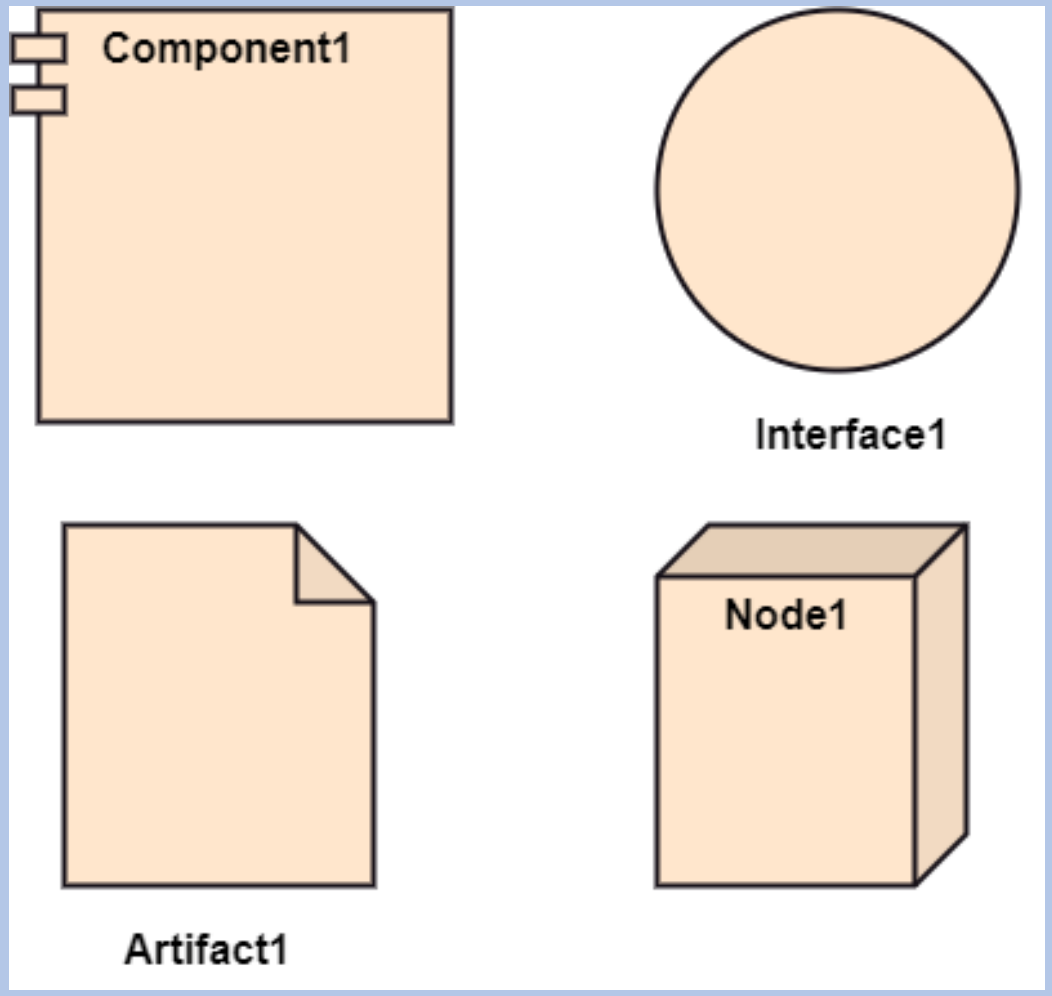
Purpose of Deployment Diagram

- Deployment diagram does not focus on the logical components of the system, but it put its attention on the hardware topology

1. To envision the hardware topology of the system
2. To represent the hardware components on which the software components are installed
3. To describe the processing of nodes at the runtime

Symbol and notation of Deployment diagram

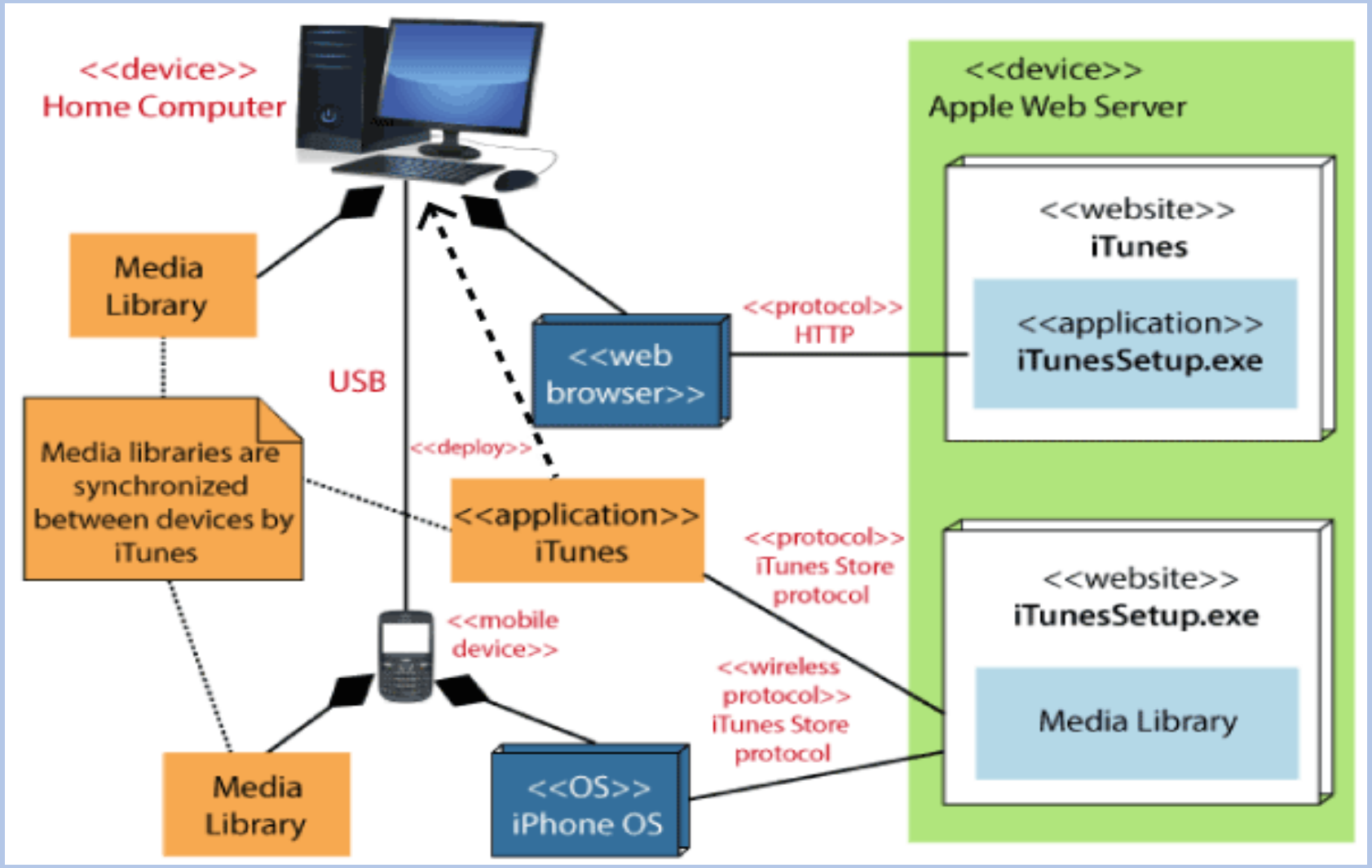
1. A component
2. An artifact
3. An interface
4. A node



Example of a Deployment diagram

- Deployment diagram for the Apple iTunes application
- The iTunes setup can be downloaded from the iTunes website, and also it can be installed on the home computer
- Once the installation and the registration are done, iTunes application can easily interconnect with the Apple iTunes store
- Users can purchase and download music, video, TV serials, etc. and cache it in the media library

Example of a Deployment diagram



When to use a Deployment Diagram?

- To model the network and hardware topology of a system
- To model the distributed networks and systems
- Implement forwarding and reverse engineering processes
- To model the hardware details for a client/server system
- For modeling the embedded system

UML Interaction Diagram

- Portrays the interactions between distinct entities present in the model
- It amalgamates both the activity and sequence diagrams
- The communication is nothing but units of the behavior of a classifier that provides context for interactions
- A set of messages that are interchanged between the entities to achieve certain specified tasks in the system is termed as interaction
- It may incorporate any feature of the classifier of which it has access
- In the interaction diagram, the critical component is the messages and the lifeline

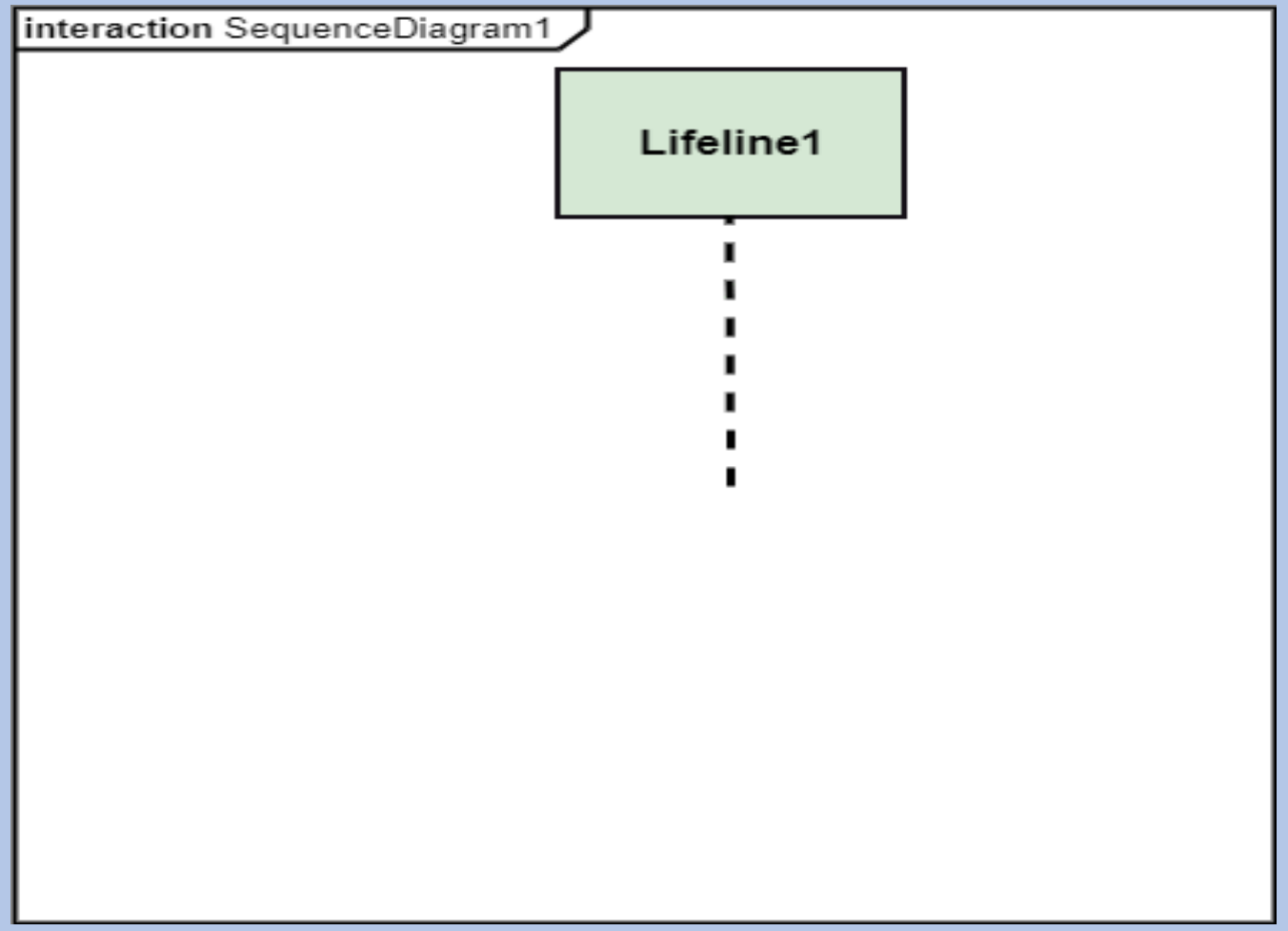
UML Interaction Diagram

- In UML, the interaction overview diagram initiates the interaction between the objects utilizing message passing
- While drawing an interaction diagram, the entire focus is to represent the relationship among different objects which are available within the system boundary and the message exchanged by them to communicate with each other
- The message exchanged among objects is either to pass some information or to request some information

UML Interaction Diagram

- And based on the information, the interaction diagram is categorized into the sequence diagram, collaboration diagram, and timing diagram
- The sequence diagram envisions the order of the flow of messages inside the system by depicting the communication between two lifelines, just like a time-ordered sequence of events
- The collaboration diagram, which is also known as the communication diagram, represents how lifelines connect within the system
- whereas the timing diagram focuses on that instant when a message is passed from one element to the other

Notation of an Interaction Diagram



Purpose of an Interaction Diagram

- To visualize the dynamic behavior of the system.
- To envision the interaction and the message flow in the system
- To portray the structural aspects of the entities within the system
- To represent the order of the sequenced interaction in the system
- To visualize the real-time data and represent the architecture of an object-oriented system

How to draw an Interaction Diagram?

- First step is to discover the scenario for which the diagram will be made
- Next, we will identify various lifelines that will be invoked in the communication, and then we will classify each lifeline
- After that, the connections are investigated and how the lifelines are interrelated to each other

How to draw an Interaction Diagram?

1. A total no of lifeline which will take part in the communication
2. The sequence of the message flow among several entities within the system
3. No operators used to ease out the functionality of the diagram
4. Several distinct messages that depict the interactions in a precise and clear way
5. The organization and structure of a system
6. The order of the sequence of the flow of messages
7. Total no of time constructs of an object

Use of an Interaction Diagram

1. The sequence diagram is employed to investigate a new application
2. The interaction diagram explores and compares the use of the collaboration diagram sequence diagram and the timing diagram
3. The interaction diagram represents the interactive (dynamic) behavior of the system
4. The sequence diagram portrays the order of control flow from one element to the other elements inside the system, whereas the collaboration diagrams are employed to get an overview of the object architecture of the system

Use of an Interaction Diagram

5. The interaction diagram models the system as a time-ordered sequence of a system
6. The interaction diagram models the system as a time-ordered sequence of a system
7. The interaction diagram systemizes the structure of the interactive elements

UML Use Case Diagram

- To represent the dynamic behavior of a system
- It encapsulates the system's functionality by incorporating use cases, actors, and their relationships
- It models the tasks, services, and functions required by a system/subsystem of an application
- It depicts the high-level functionality of a system and also tells how the user handles a system

Purpose of Use Case Diagrams

- Main purpose is to portray the dynamic aspect of a system
- It accumulates the system's requirement, which includes both internal as well as external influences
- It invokes persons, use cases, and several things that invoke the actors and elements accountable for the implementation of use case diagrams
- It represents how an entity from the external environment can interact with a part of the system

Purpose of Use Case Diagrams

1. It gathers the system's needs
2. It depicts the external view of the system
3. It recognizes the internal as well as external factors that influence the system
4. It represents the interaction between the actors

How to draw a Use Case diagram?

- Analyze the whole system before starting with drawing a use case diagram, and then the system's functionalities are found
- Once every single functionality is identified, they are then transformed into the use cases to be used in the use case diagram
- After that, enlist the actors that will interact with the system
- The actors are the person or a thing that invokes the functionality of a system

How to draw a Use Case diagram?

- It may be a system or a private entity, such that it requires an entity to be pertinent to the functionalities of the system to which it is going to interact
- Once both the actors and use cases are enlisted, the relation between the actor and use case/ system is inspected
- It identifies the no of times an actor communicates with the system
- Basically, an actor can interact multiple times with a use case or system at a particular instance of time

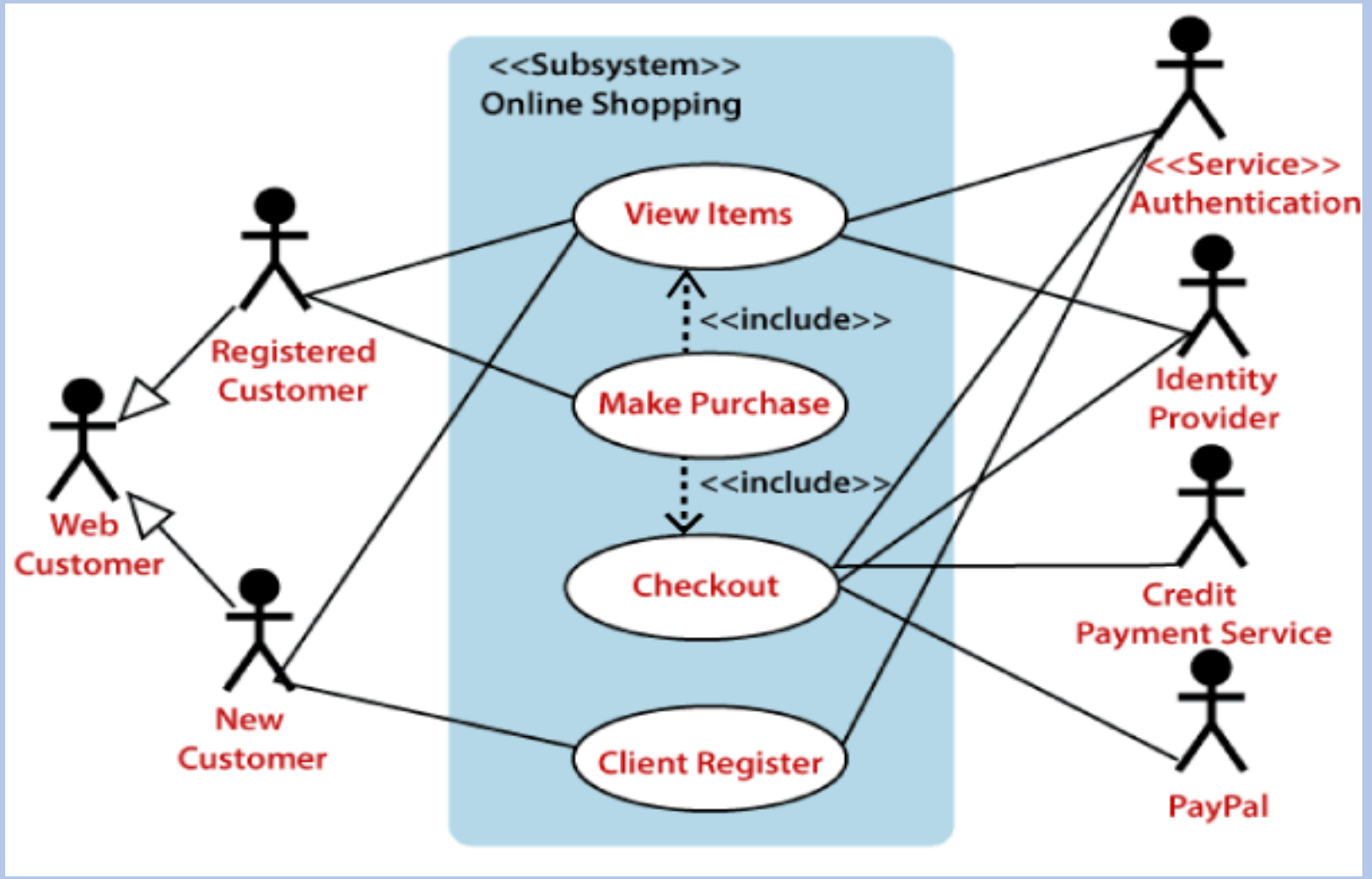
How to draw a Use Case diagram?

1. A pertinent and meaningful name should be assigned to the actor or a use case of a system
2. The communication of an actor with a use case must be defined in an understandable way
3. Specified notations to be used as and when required
4. The most significant interactions should be represented among the multiple no of interactions between the use case and actors

Example of a Use Case Diagram

- Use case diagram depicting the Online Shopping website
- Web Customer actor makes use of any online shopping website to purchase online
- The top-level uses are as follows; View Items, Make Purchase, Checkout, Client Register
- The View Items use case is utilized by the customer who searches and view products
- The Client Register use case allows the customer to register itself with the website for availing gift vouchers, coupons, or getting a private sale invitation
- It is to be noted that the Checkout is an included use case, which is part of Making Purchase, and it is not available by itself

Example of a Use Case Diagram



Example of a Use Case Diagram

- The **View Items** is further extended by several use cases such as; Search Items, Browse Items, View Recommended Items, Add to Shopping Cart, Add to Wish list
- All of these extended use cases provide some functions to customers, which allows them to search for an item
- The View Items is further extended by several use cases such as; Search Items, Browse Items, View Recommended Items, Add to Shopping Cart, Add to Wish list

Example of a Use Case Diagram

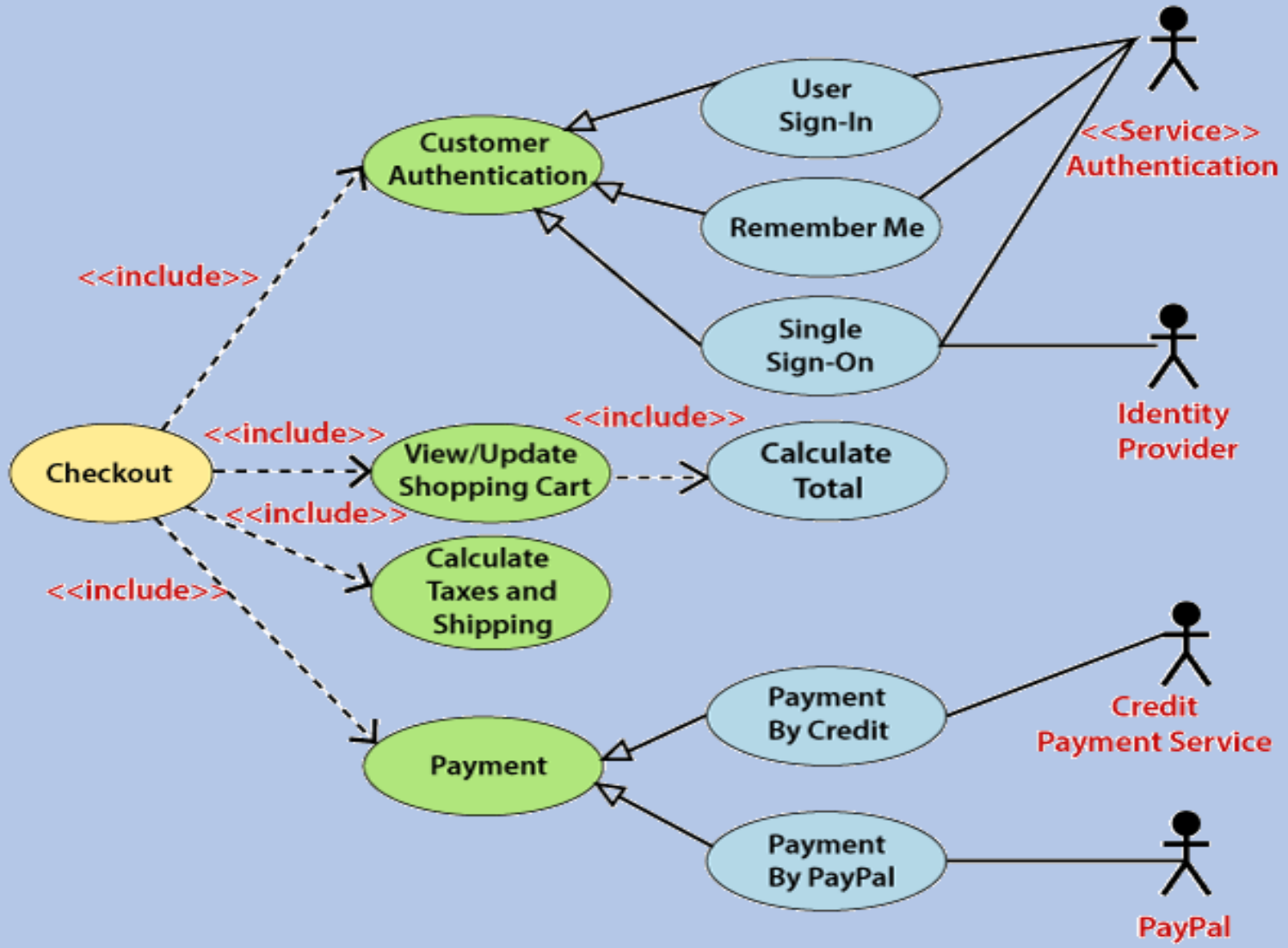
- All of these extended use cases provide some functions to customers, which allows them to search for an item
- Both View Recommended Item and Add to Wish List include the Customer Authentication use case, as they necessitate authenticated customers, and simultaneously item can be added to the shopping cart without any user authentication

Example of a Use Case Diagram

Example of a Use Case Diagram

- The Checkout use case
- It requires an authenticated Web Customer, which can be done by login page, user authentication cookie ("Remember me"), or Single Sign-On (SSO)
- SSO needs an external identity provider's participation, while Web site authentication service is utilized in all these use cases
- The Checkout use case involves Payment use case that can be done either by the credit card and external credit payment services or with PayPal

Example of a Use Case Diagram



Important tips for drawing a Use Case diagram

- A simple and complete use case diagram should be articulated
- A use case diagram should represent the most significant interaction among the multiple interactions
- At least one module of a system should be represented by the use case diagram
- If the use case diagram is large and more complex, then it should be drawn more generalized

Sequence Diagram

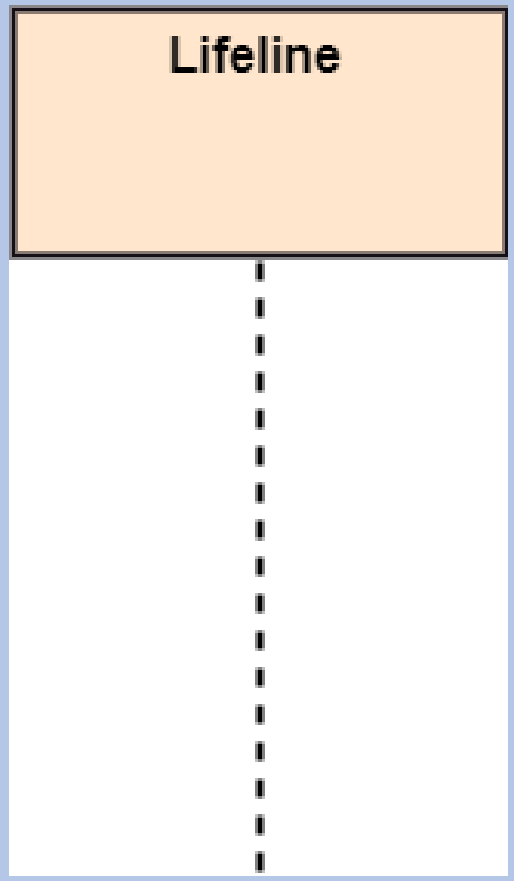
- Represents the flow of messages in the system and is also termed as an event diagram
- It helps in envisioning several dynamic scenarios
- It portrays the communication between any two lifelines as a time-ordered sequence of events, such that these lifelines took part at the run time
- In UML, the lifeline is represented by a vertical bar, whereas the message flow is represented by a vertical dotted line that extends across the bottom of the page
- It incorporates the iterations as well as branching

Purpose of a Sequence Diagram

1. To model high-level interaction among active objects within a system
2. To model interaction among objects inside a collaboration realizing a use case
3. It either models generic interactions or some certain instances of interaction

Notations of a Sequence Diagram

- Lifeline : An individual participant in the sequence diagram is represented by a lifeline. It is positioned at the top of the diagram

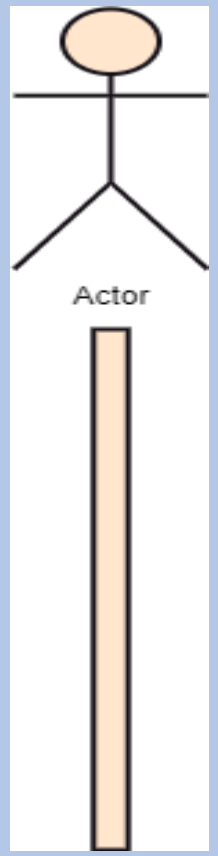


Notations of a Sequence Diagram

- Actor : A role played by an entity that interacts with the subject is called as an actor
- It is out of the scope of the system
- It represents the role, which involves human users and external hardware or subjects
- An actor may or may not represent a physical entity, but it purely depicts the role of an entity
- Several distinct roles can be played by an actor or vice versa

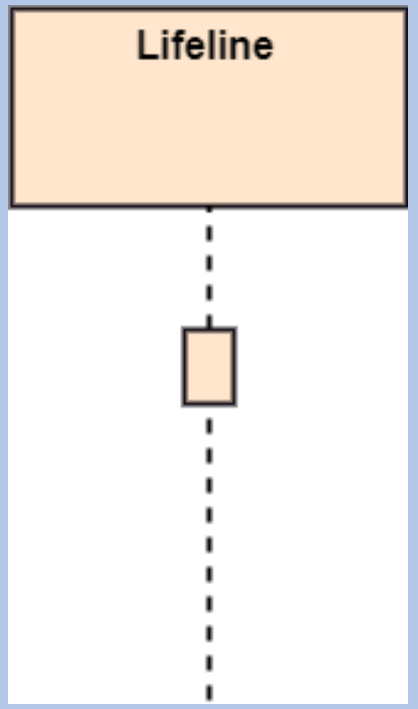
Notations of a Sequence Diagram

- Actor :



Notations of a Sequence Diagram

- Activation : It is represented by a thin rectangle on the lifeline. It describes that time period in which an operation is performed by an element, such that the top and the bottom of the rectangle is associated with the initiation and the completion time, each respectively

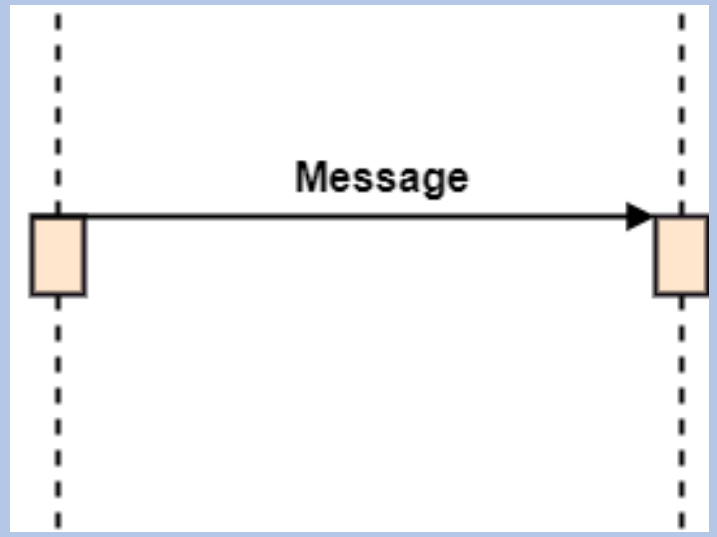


Notations of a Sequence Diagram

- Messages : The messages depict the interaction between the objects and are represented by arrows. They are in the sequential order on the lifeline. The core of the sequence diagram is formed by messages and lifelines.

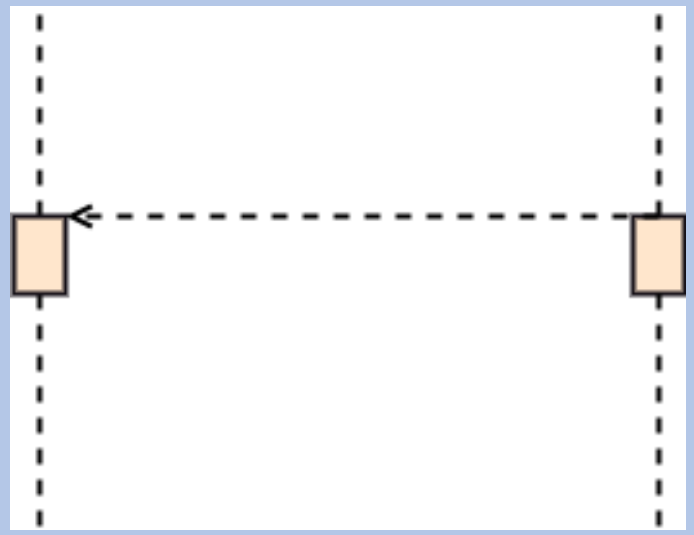
Notations of a Sequence Diagram

- Call Message: It defines a particular communication between the lifelines of an interaction, which represents that the target lifeline has invoked an operation



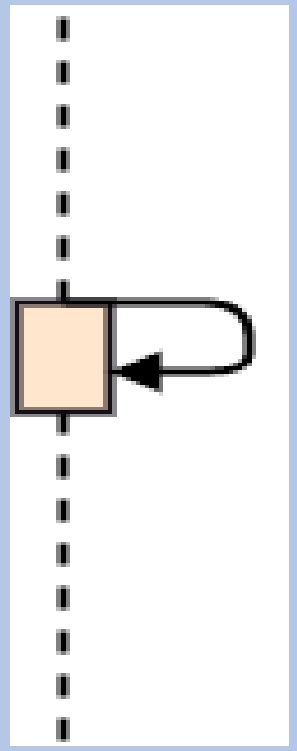
Notations of a Sequence Diagram

- Return Message: It defines a particular communication between the lifelines of interaction that represent the flow of information from the receiver of the corresponding caller message.



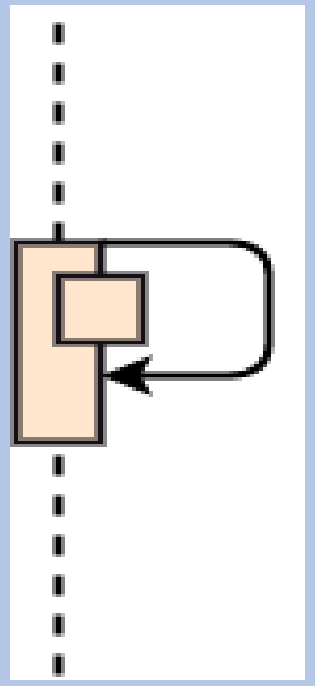
Notations of a Sequence Diagram

- Self Message: It describes a communication, particularly between the lifelines of an interaction that represents a message of the same lifeline, has been invoked



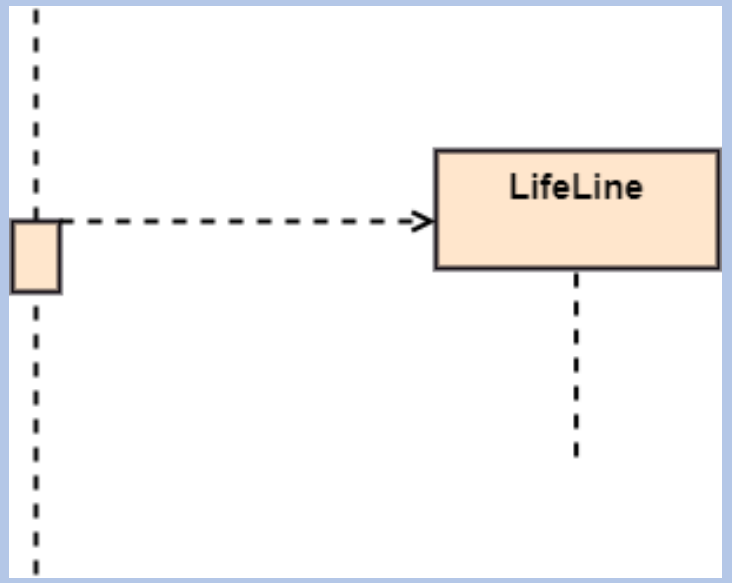
Notations of a Sequence Diagram

- Recursive Message: A self message sent for recursive purpose is called a recursive message. In other words, it can be said that the recursive message is a special case of the self message as it represents the recursive calls



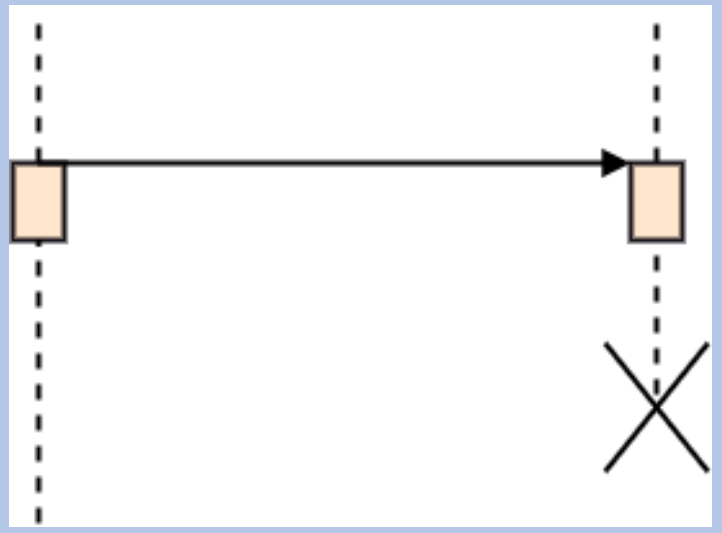
Notations of a Sequence Diagram

- Create Message: It describes a communication, particularly between the lifelines of an interaction describing that the target (lifeline) has been instantiated



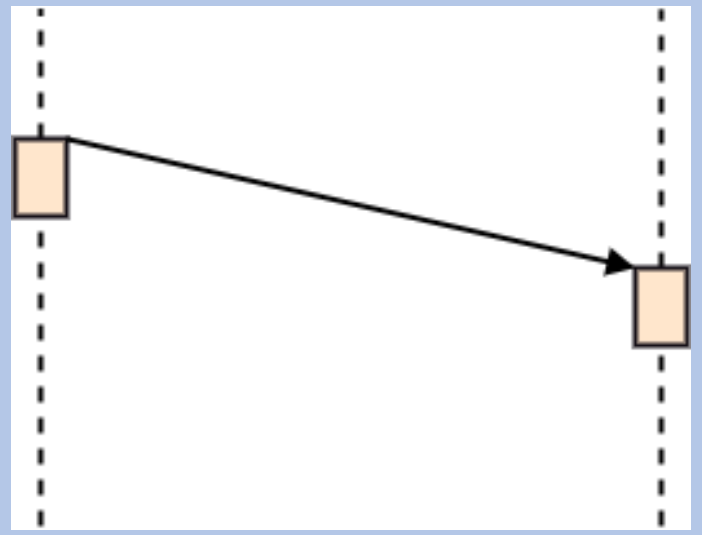
Notations of a Sequence Diagram

- Destroy Message: It describes a communication, particularly between the lifelines of an interaction that depicts a request to destroy the lifecycle of the target



Notations of a Sequence Diagram

- Duration Message: It describes a communication particularly between the lifelines of an interaction, which portrays the time passage of the message while modeling a system



Notations of a Sequence Diagram

- Note : A note is the capability of attaching several remarks to the element. It basically carries useful information for the modelers

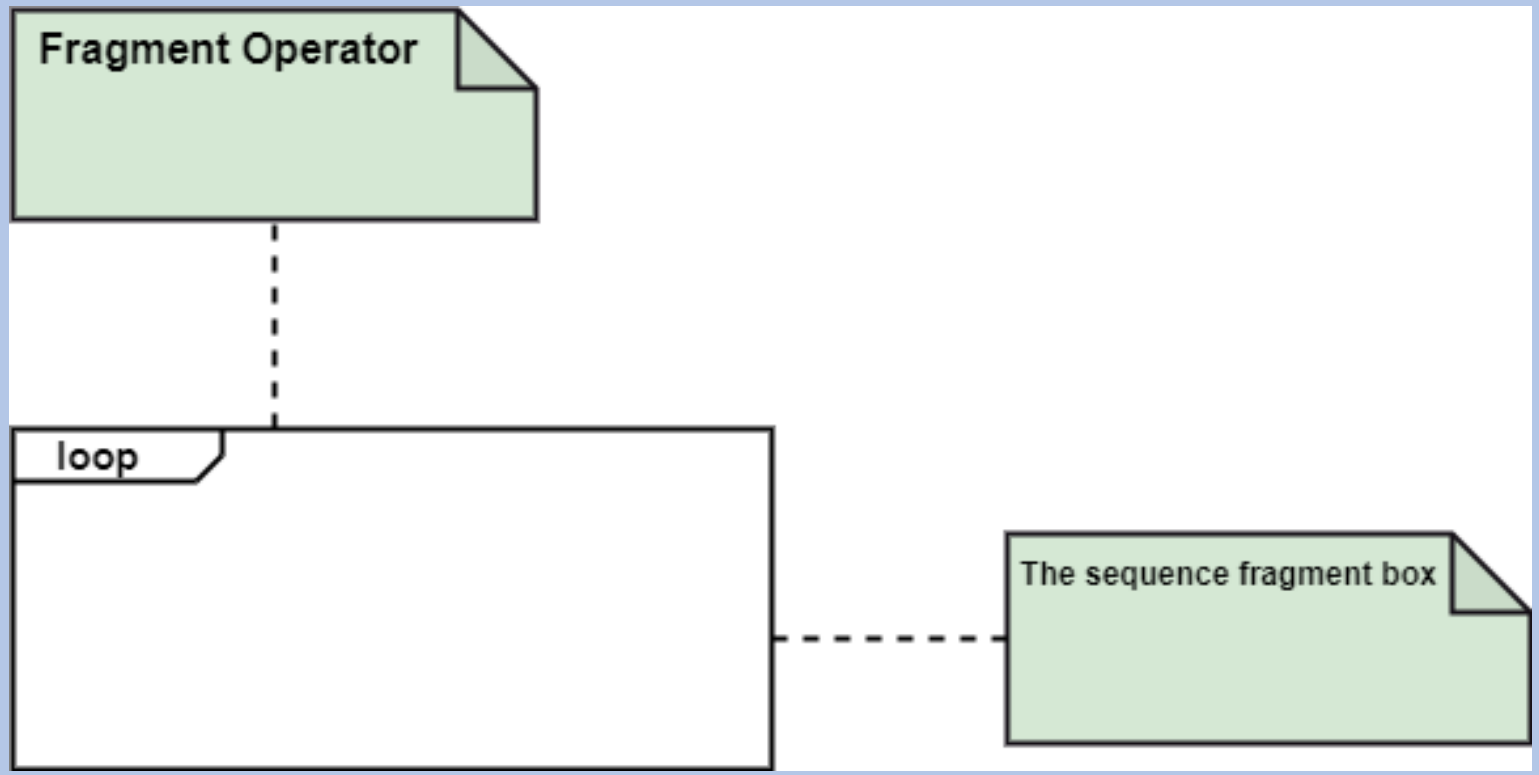
Notations of a Sequence Diagram

- Sequence Fragments :

1. Sequence fragments have been introduced by UML 2.0, which makes it quite easy for the creation and maintenance of an accurate sequence diagram
2. It is represented by a box called a combined fragment, encloses a part of interaction inside a sequence diagram
3. The type of fragment is shown by a fragment operator

Notations of a Sequence Diagram

- Sequence Fragments :



Notations of a Sequence Diagram

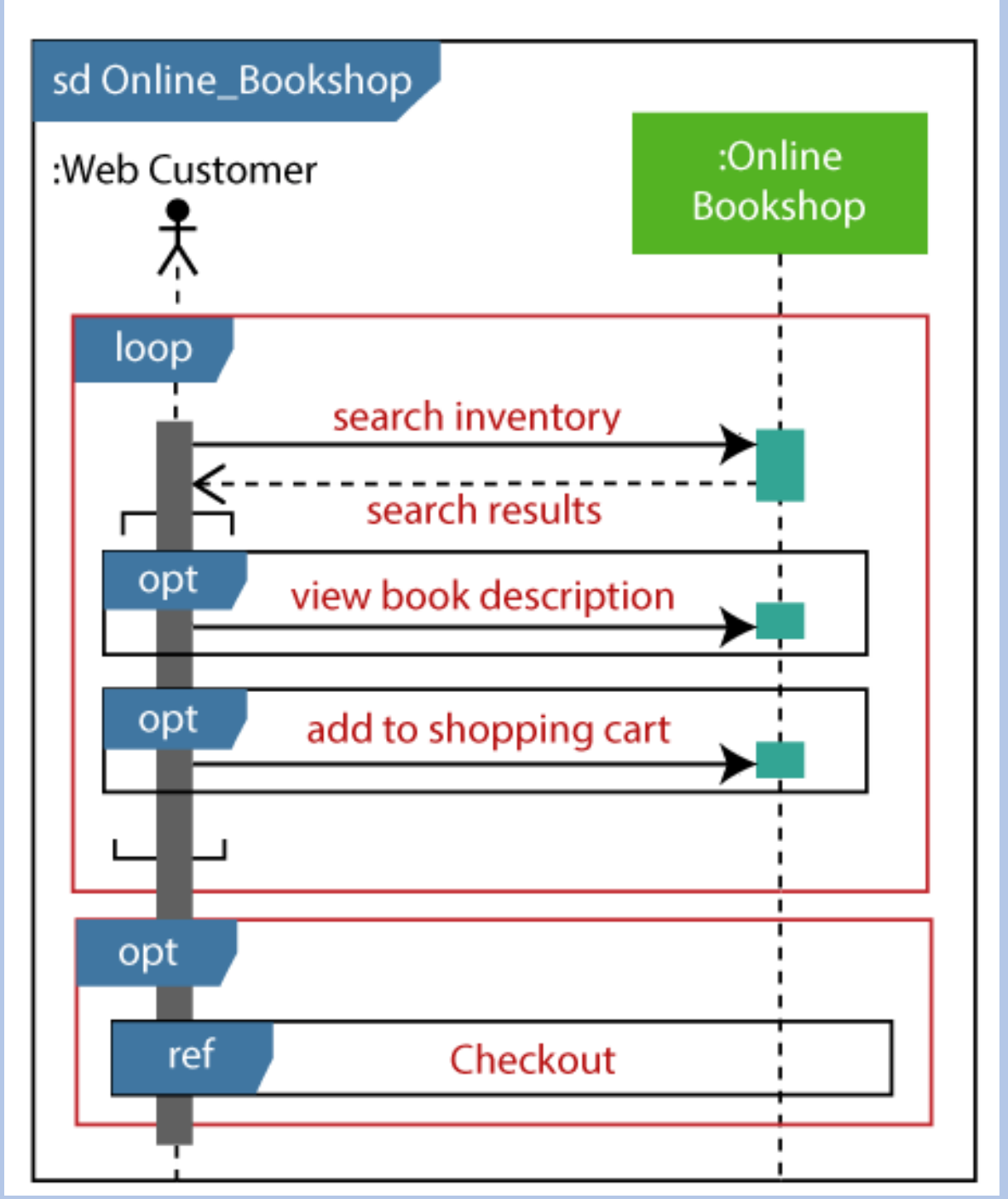
- Types of fragments:

Operator	Fragment Type
alt	Alternative multiple fragments: The only fragment for which the condition is true, will execute.
opt	Optional: If the supplied condition is true, only then the fragments will execute. It is similar to alt with only one trace.
par	Parallel: Parallel executes fragments.
loop	Loop: Fragments are run multiple times, and the basis of interaction is shown by the guard.
region	Critical region: Only one thread can execute a fragment at once.
neg	Negative: A worthless communication is shown by the fragment.
ref	Reference: An interaction portrayed in another diagram. In this, a frame is drawn so as to cover the lifelines involved in the communication. The parameter and return value can be explained.
sd	Sequence Diagram: It is used to surround the whole sequence diagram.

Example of a Sequence Diagram

- Example of a Sequence Diagram: online bookshop
- Any online customer can search for a book catalog, view a description of a particular book, add a book to its shopping cart, and do checkout

Example of Sequence Diagram



Benefits of a Sequence Diagram

- It explores the real-time application
- It depicts the message flow between the different objects
- It has easy maintenance
- It is easy to generate
- Implement both forward and reverse engineering
- It can easily update as per the new change in the system

Drawbacks of a Sequence Diagram

- In the case of too many lifelines, the sequence diagram can get more complex
- The incorrect result may be produced, if the order of the flow of messages changes
- Since each sequence needs distinct notations for its representation, it may make the diagram more complex
- The type of sequence is decided by the type of message

UML Collaboration Diagram

- The collaboration diagram is used to show the relationship between the objects in a system
- Both the sequence and the collaboration diagrams represent the same information but differently
- Instead of showing the flow of messages, it depicts the architecture of the object residing in the system as it is based on object-oriented programming

UML Collaboration Diagram

- An object consists of several features
- Multiple objects present in the system are connected to each other
- The collaboration diagram, which is also known as a communication diagram, is used to portray the object's architecture in the system

Notations of a Collaboration Diagram

1. Objects: The representation of an object is done by an object symbol with its name and class underlined, separated by a colon

- In the collaboration diagram, objects are utilized in the following ways:
  - The object is represented by specifying their name and class
  - It is not mandatory for every class to appear
  - A class may constitute more than one object
  - In the collaboration diagram, firstly, the object is created, and then its class is specified
  - To differentiate one object from another object, it is necessary to name them

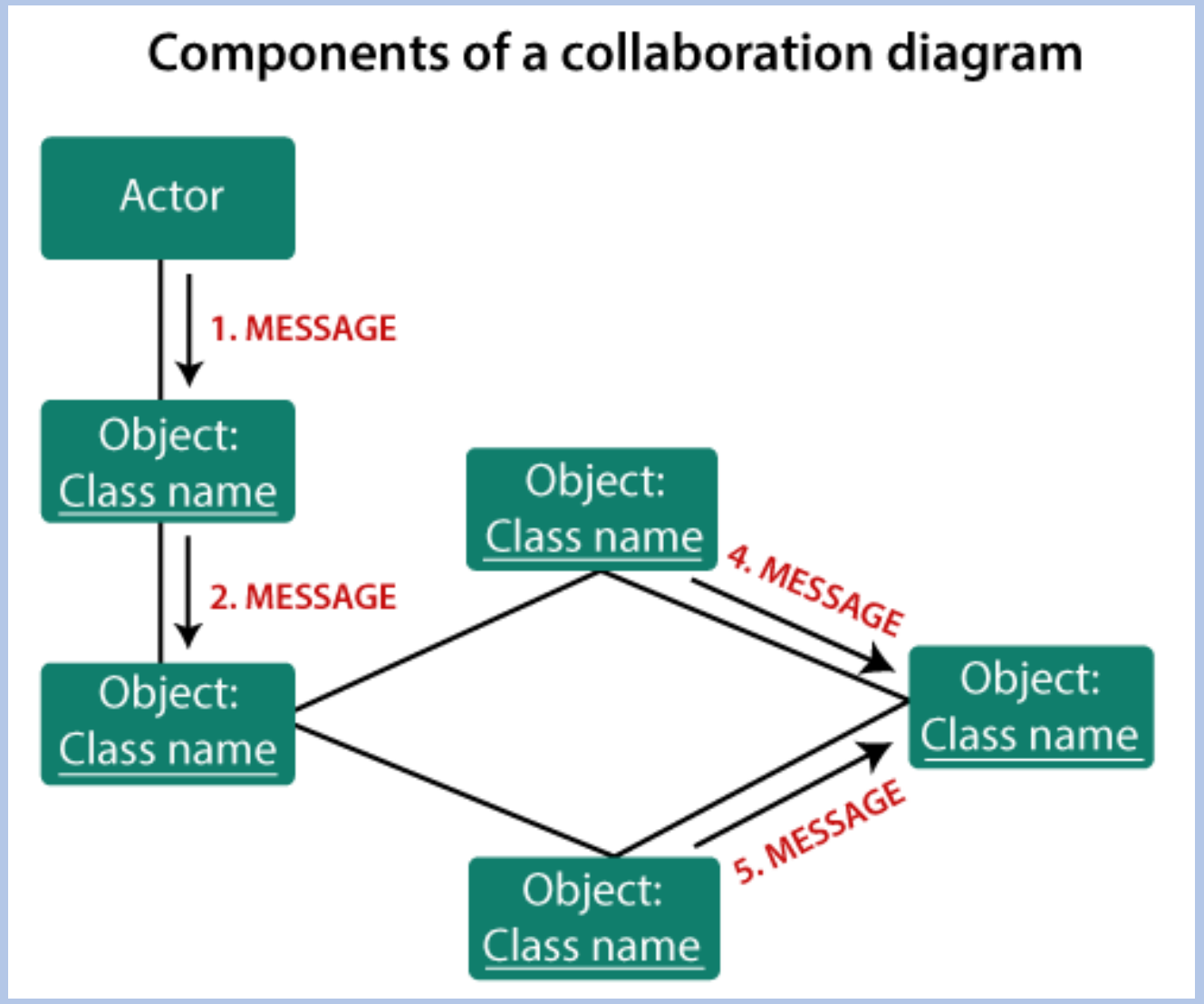
Notations of a Collaboration Diagram

2. **Actors**: In the collaboration diagram, the actor plays the main role as it invokes the interaction. Each actor has its respective role and name. In this, one actor initiates the use case.
3. **Links**: The link is an instance of association, which associates the objects and actors. It portrays a relationship between the objects through which the messages are sent. It is represented by a solid line. The link helps an object to connect with or navigate to another object, such that the message flows are attached to links

Notations of a Collaboration Diagram

4. **Messages:** It is a communication between objects which carries information and includes a sequence number, so that the activity may take place. It is represented by a labeled arrow, which is placed near a link. The messages are sent from the sender to the receiver, and the direction must be navigable in that particular direction. The receiver must understand the message.

Notations of a Collaboration Diagram



When to use a Collaboration Diagram?

- When it is essential to depict the relationship between the object
- Both the sequence and collaboration diagrams represent the same information, but the way of portraying it quite different
- The collaboration diagrams are best suited for analyzing use cases
- To model collaboration among the objects or roles that carry the functionalities of use cases and operations
- To model the mechanism inside the architectural design of the system
- To capture the interactions that represent the flow of messages between the objects and the roles inside the collaboration

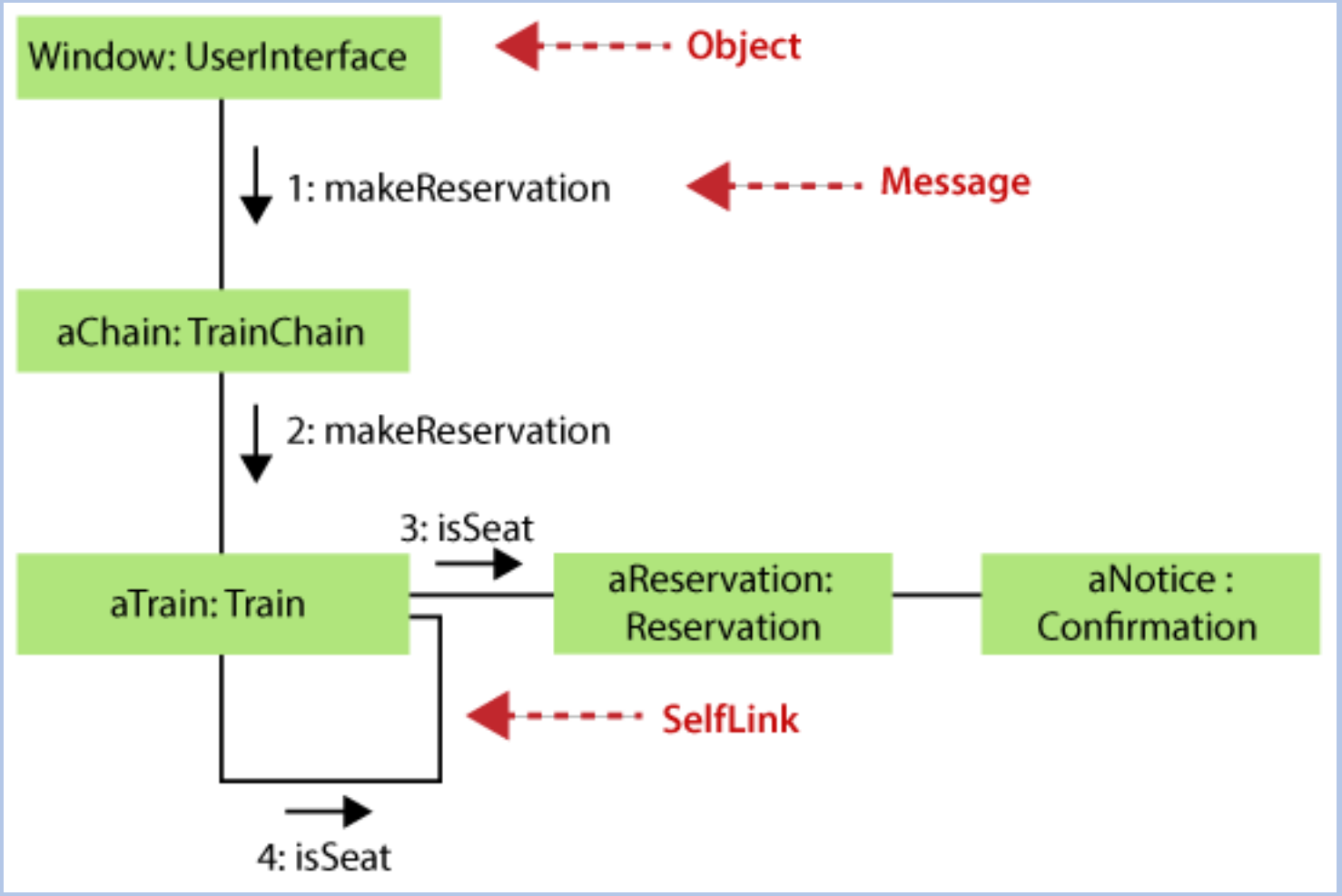
When to use a Collaboration Diagram?

- To model different scenarios within the use case or operation, involving a collaboration of several objects and interactions
- To support the identification of objects participating in the use case
- In the collaboration diagram, each message constitutes a sequence number, such that the top-level message is marked as one and so on. The messages sent during the same call are denoted with the same decimal prefix, but with different suffixes of 1, 2, etc. as per their occurrence

Steps for creating a Collaboration Diagram

1. Determine the behavior for which the realization and implementation are specified
2. Discover the structural elements that are class roles, objects, and subsystems for performing the functionality of collaboration
   - Choose the context of an interaction: system, subsystem, use case, and operation
3. Think through alternative situations that may be involved
   - Implementation of a collaboration diagram at an instance level, if needed
   - A specification level diagram may be made in the instance level sequence diagram for summarizing alternative situations

Steps for creating a Collaboration Diagram



Benefits of a Collaboration Diagram

1. The collaboration diagram is also known as Communication Diagram
2. It mainly puts emphasis on the structural aspect of an interaction diagram, i.e., how lifelines are connected
3. The syntax of a collaboration diagram is similar to the sequence diagram; just the difference is that the lifeline does not consist of tails
4. The messages transmitted over sequencing is represented by numbering each individual message
5. The collaboration diagram is semantically weak in comparison to the sequence diagram

Benefits of a Collaboration Diagram

6. The special case of a collaboration diagram is the object diagram
7. It focuses on the elements and not the message flow, like sequence diagrams
8. Since the collaboration diagrams are not that expensive, the sequence diagram can be directly converted to the collaboration diagram
9. There may be a chance of losing some amount of information while implementing a collaboration diagram with respect to the sequence diagram

Drawbacks of a Collaboration Diagram

1. Multiple objects residing in the system can make a complex collaboration diagram, as it becomes quite hard to explore the objects
2. It is a time-consuming diagram
3. After the program terminates, the object is destroyed
4. As the object state changes momentarily, it becomes difficult to keep an eye on every single that has occurred inside the object of a system

UML State Machine Diagram

- Also called the Statechart or State Transition diagram, which shows the order of states underwent by an object within the system
- It captures the software system's behavior
- It models the behavior of a class, a subsystem, a package, and a complete system
- It tends out to be an efficient way of modeling the interactions and collaborations in the external entities and the system

UML State Machine Diagram

- It models event-based systems to handle the state of an object
- It also defines several distinct states of a component within the system
- Each object/component has a specific state
- **Behavioral state machine :** The behavioral state machine diagram records the behavior of an object within the system. It depicts an implementation of a particular entity. It models the behavior of the system

UML State Machine Diagram

- It models event-based systems to handle the state of an object
- It also defines several distinct states of a component within the system
- Each object/component has a specific state

UML State Machine Diagram

- **Protocol state machine :** It captures the behavior of the protocol. The protocol state machine depicts the change in the state of the protocol and parallel changes within the system. But it does not portray the implementation of a particular component.

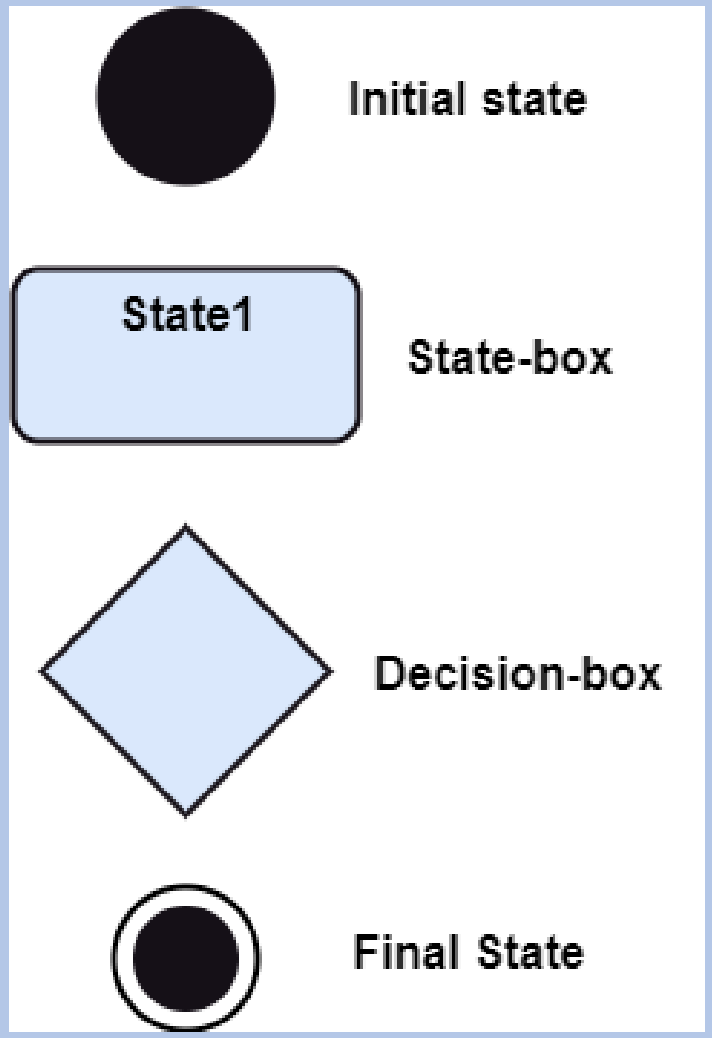
Why State Machine Diagram?

- Since it records the dynamic view of a system, it portrays the behavior of a software application
- During a lifespan, an object underwent several states, such that the lifespan exist until the program is executing
- Each state depicts some useful information about the object
- It blueprints an interactive system that response back to either the internal events or the external ones

Why State Machine Diagram?

- The execution flow from one state to another is represented by a state machine diagram
- It visualizes an object state from its creation to its termination.
- The main purpose is to depict each state of an individual object
- It represents an interactive system and the entities inside the system
- It records the dynamic behavior of the system

Notation of a State Machine Diagram



Notation of a State Machine Diagram

- **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.

Notation of a State Machine Diagram

- **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

How to Draw a State Machine Diagram?

- A unique and understandable name should be assigned to the state transition that describes the behavior of the system
- Out of multiple objects, only the essential objects are implemented
- A proper name should be given to the events and the transitions

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

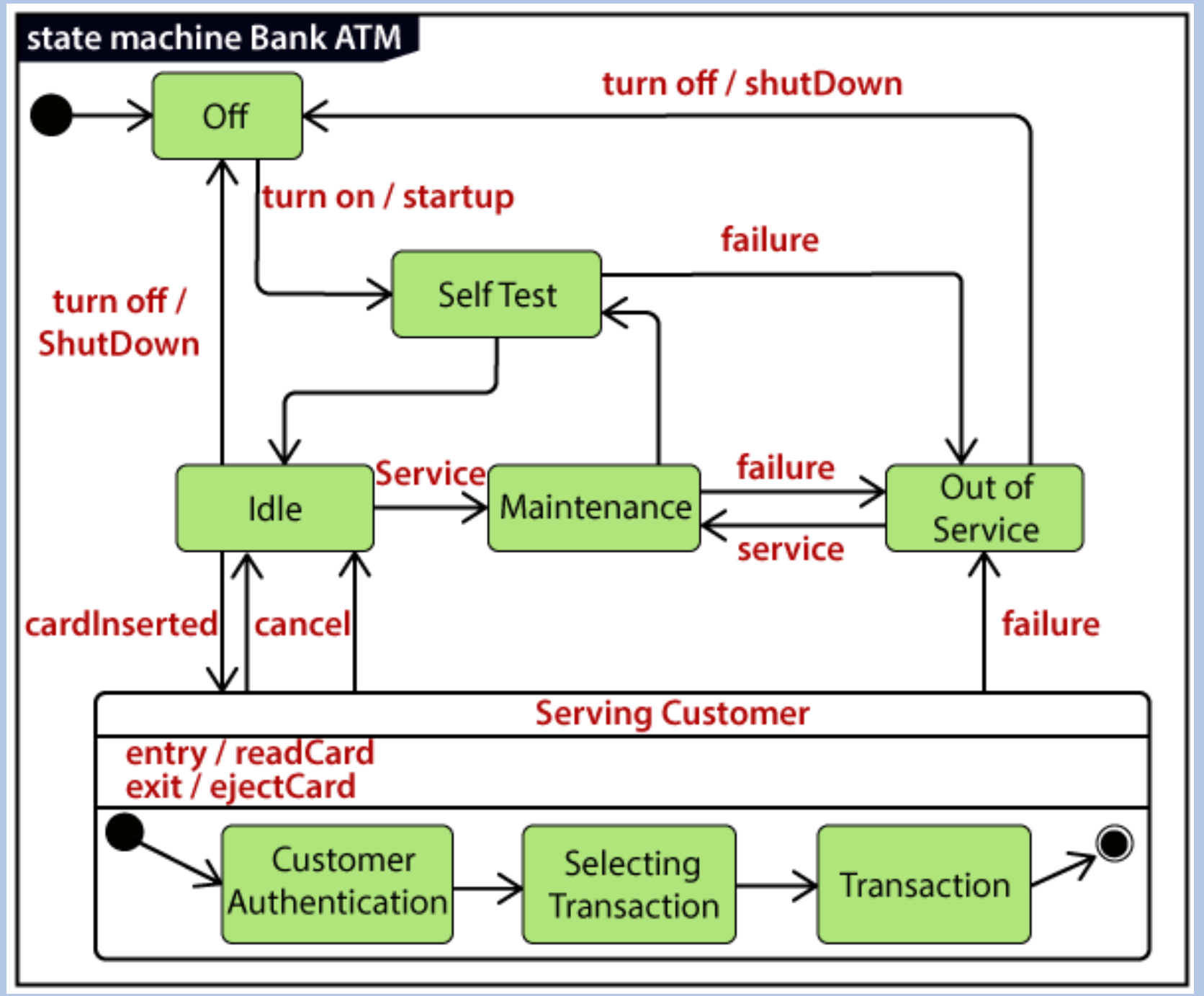
Example of a State Machine Diagram

- Bank Automated Teller Machine (ATM)
- Initially, the ATM is turned off.
- After the power supply is turned on, the ATM starts performing the startup action and enters into the Self Test state
- If the test fails, the ATM will enter into the Out Of Service state, or it will undergo a triggerless transition to the Idle state
- This is the state where the customer waits for the interaction

Example of a State Machine Diagram

- Bank Automated Teller Machine (ATM)
- Whenever the customer inserts the bank or credit card in the ATM's card reader, the ATM state changes from Idle to Serving Customer, the entry action readCard is performed after entering into Serving Customer state
- Since the customer can cancel the transaction at any instant, so the transition from Serving Customer state back to the Idle state could be triggered by cancel event

Example of a State Machine Diagram



Example of a State Machine Diagram

- Bank Automated Teller Machine (ATM)
- Here the **Serving Customer** is a composite state with sequential substates that are **Customer Authentication, Selecting Transaction,** and **Transaction**
- **Customer Authentication** and **Transaction** are the composite states itself is displayed by a hidden decomposition indication icon. After the transaction is finished, the **Serving Customer** encompasses a triggerless transition back to the **Idle** state. On leaving the state, it undergoes the exit action **ejectCard** that discharges the customer card.

State Machine vs. Flowchart

State Machine	Flowchart
It portrays several states of a system.	It demonstrates the execution flow of a program.
It encompasses the concept of WAIT, i.e., wait for an event or an action.	It does not constitute the concept of WAIT.
It is for real-world modeling systems.	It envisions the branching sequence of a system.
It is a modeling diagram.	It is a data flow diagram (DFD)
It is concerned with several states of a system.	It focuses on control flow and path.

UML Activity Diagram

- Is used to demonstrate the flow of control within the system rather than the implementation
- It models the concurrent and sequential activities
- The activity diagram helps in envisioning the workflow from one activity to another
- It put emphasis on the condition of flow and the order in which it occurs
- The flow can be sequential, branched, or concurrent, and to deal with such kinds of flows, the activity diagram has come up with a fork, join, etc
- It is also termed as an object-oriented flowchart
- It encompasses activities composed of a set of actions or operations that are applied to model the behavioral diagram

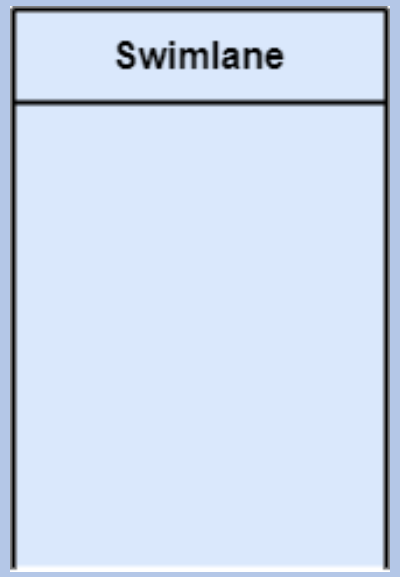
Components of an Activity Diagram

Activities

- The categorization of behavior into one or more actions is termed as an activity
- In other words, it can be said that an activity is a network of nodes that are connected by edges
- The edges depict the flow of execution

- It may contain action nodes, control nodes, or object nodes
- The control flow of activity is represented by control nodes and object nodes that illustrates the objects used within an activity
- The activities are initiated at the initial node and are terminated at the final node

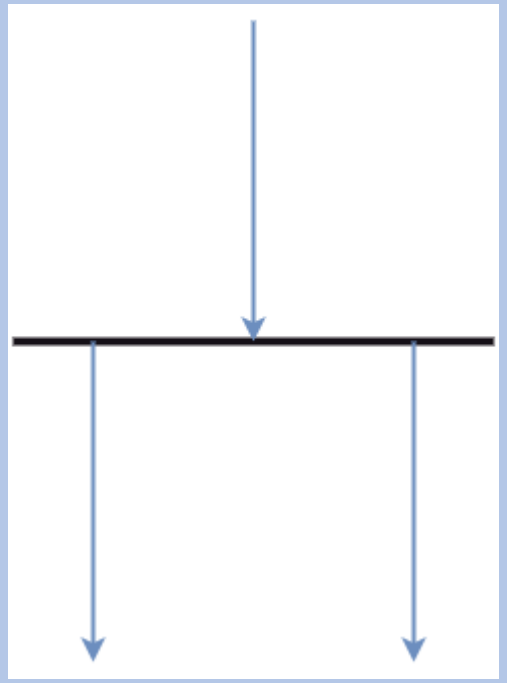
Components of an Activity Diagram



Activity partition /swimlane

- The swimlane is used to cluster all the related activities in one column or one row
- It can be either vertical or horizontal
- It used to add modularity to the activity diagram
- It is not necessary to incorporate swimlane in the activity diagram
- But it is used to add more transparency to the activity diagram.

Components of an Activity Diagram



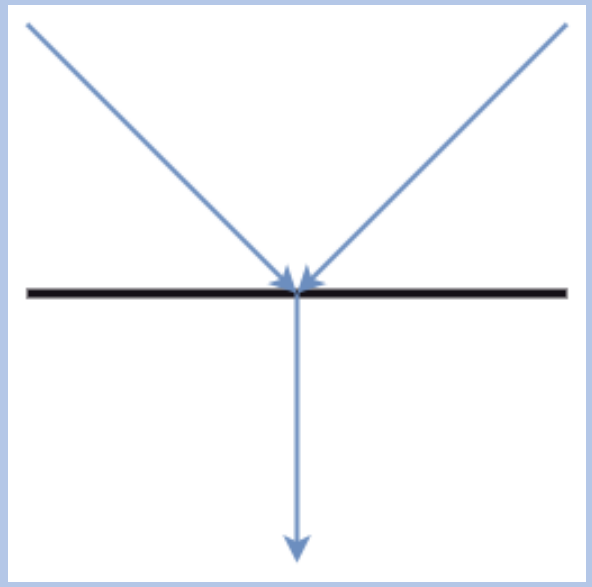
Forks

- Forks and join nodes generate the concurrent flow inside the activity
- A fork node consists of one inward edge and several outward edges
- It is the same as that of various decision parameters
- Whenever a data is received at an inward edge, it gets copied and split crossways various outward edges
- It split a single inward flow into multiple parallel flows

Components of an Activity Diagram

Join Nodes

- Join nodes are the opposite of fork nodes
- A Logical AND operation is performed on all of the inward edges as it synchronizes the flow of input across one single output (outward) edge



Components of an Activity Diagram

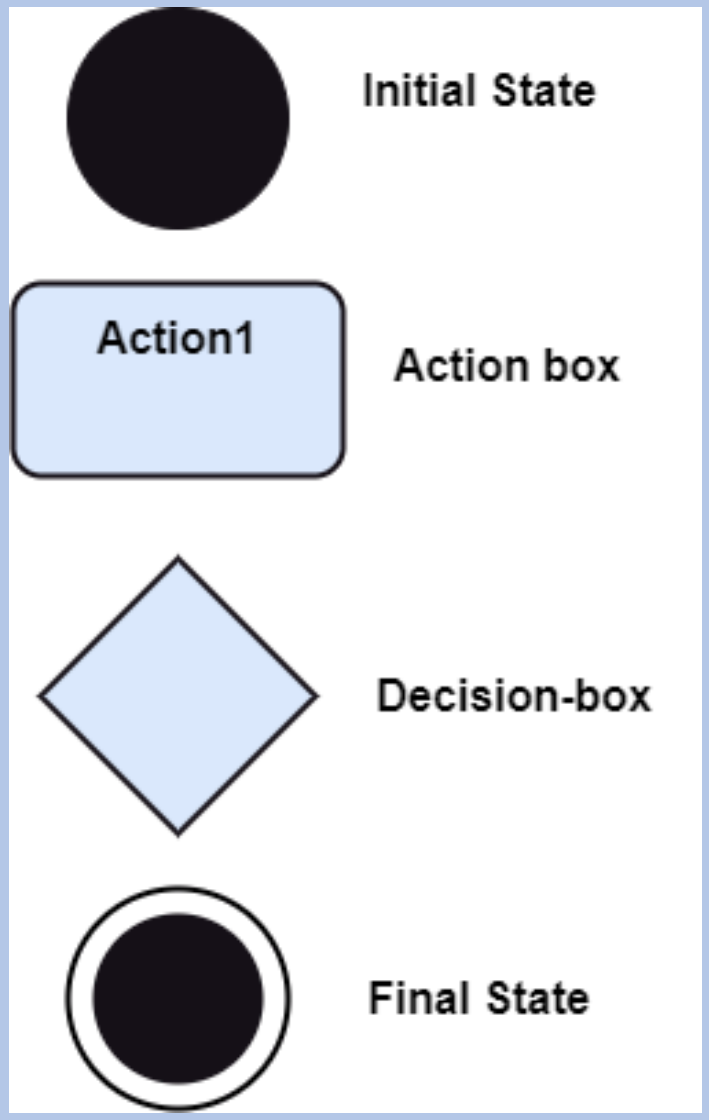
Pins

- It is a small rectangle, which is attached to the action rectangle
- It clears out all the messy and complicated thing to manage the execution flow of activities
- It is an object node that precisely represents one input to or output from the action

Notation of an Activity diagram

- Initial State: It depicts the initial stage or beginning of the set of actions
- Final State: It is the stage where all the control flows and object flows end
- Decision Box: It makes sure that the control flow or object flow will follow only one path
- Action Box: It represents the set of actions that are to be performed

Notation of an Activity diagram



Why use Activity Diagram?

- An event is created as an activity diagram encompassing a group of nodes associated with edges
- To model the behavior of activities, they can be attached to any modeling element
- It can model use cases, classes, interfaces, components, and collaborations
- It mainly models processes and workflows
- It envisions the dynamic behavior of the system as well as constructs a runnable system that incorporates forward and reverse engineering
- It does not include the message part, which means message flow is not represented in an activity diagram
- It is the same as that of a flowchart but not exactly a flowchart itself
- It is used to depict the flow between several activities

How to draw an Activity Diagram?

- After gathering all the essential information, an abstract or a prototype is built, which is then transformed into the actual diagram
- Following are the rules that are to be followed for drawing an activity diagram:

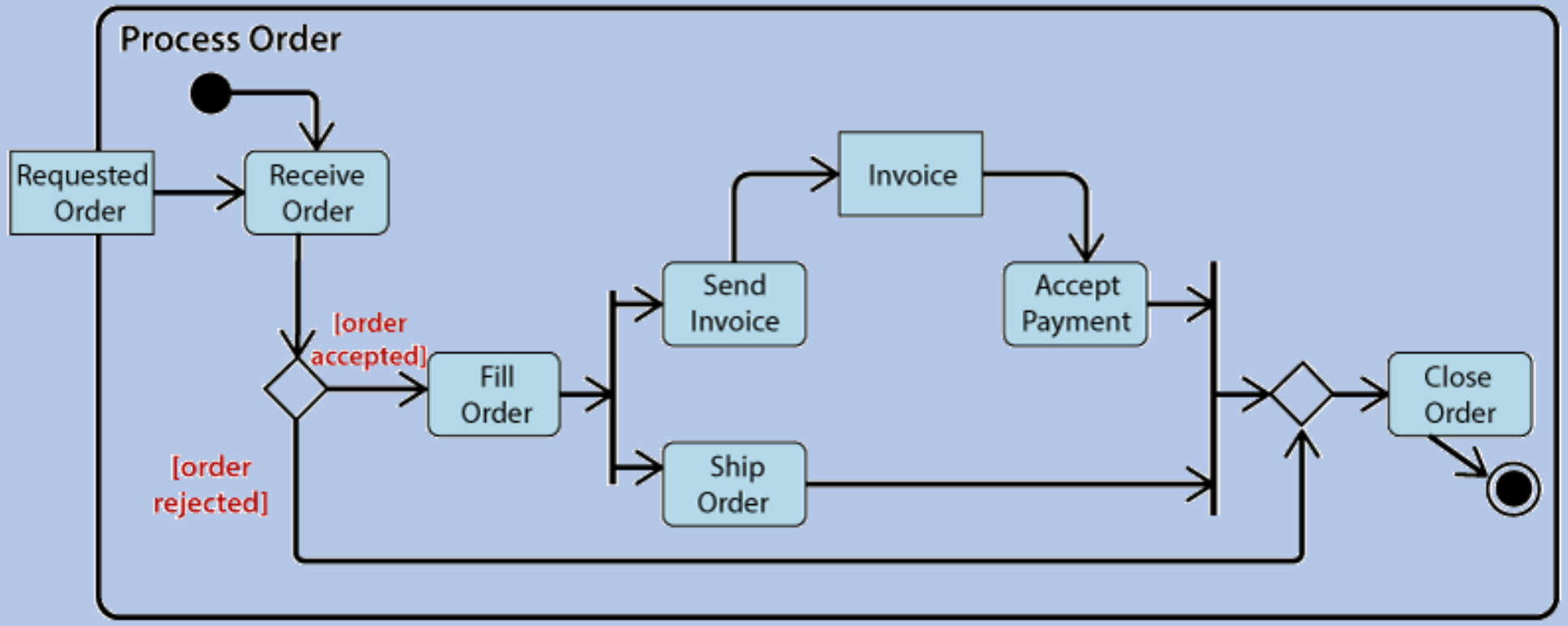
1. A meaningful name should be given to each and every activity
2. Identify all of the constraints
3. Acknowledge the activity associations

Example of an Activity Diagram

- Business flow activity of order processing
- Here the input parameter is the Requested order, and once the order is accepted, all of the required information is then filled, payment is also accepted, and then the order is shipped. It permits order shipment before an invoice is sent or payment is completed

Example of an Activity Diagram

- Business flow activity of order processing



When to use an Activity Diagram?

- To graphically model the workflow in an easier and understandable way
- To model the execution flow among several activities
- To model comprehensive information of a function or an algorithm employed within the system
- To model the business process and its workflow
- To envision the dynamic aspect of a system
- To generate the top-level flowcharts for representing the workflow of an application
- To represent a high-level view of a distributed or an object-oriented system