Sequence Diagram

INTRODUCTION

- A sequence diagram is a type of interaction diagram because it describes how—and in what order—a group of objects works together.
- These diagrams are used by software developers and business professionals to understand requirements for a new system or to document an existing process.
- Sequence diagrams are sometimes known as event diagrams or event scenarios.
- The purpose of a sequence diagram in UML is to visualize the sequence of a message flow in the system.
- Note that there are two types of sequence diagrams: UML diagrams and code-based diagrams.

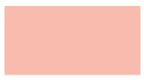
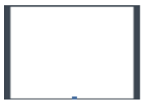
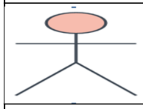
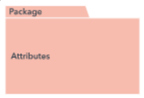
BENEFITS OF SEQUENCE DIAGRAMS

- Sequence diagrams can be useful references for businesses and other organizations.
  1. Represent the details of a UML use case.
  2. Model the logic of a complex procedure, function, or operation.
  3. See how objects and components interact with each other to complete a process.
  4. Plan and understand the detailed functionality of an existing or future scenario.

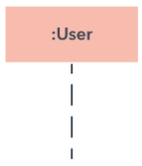
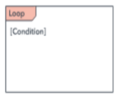
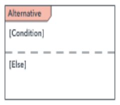
USE CASES FOR SEQUENCE DIAGRAMS

- The following scenarios are ideal for using a sequence diagram:
  - **Usage scenario**: A usage scenario is a diagram of how your system could potentially be used. It's a great way to make sure that you have worked through the logic of every usage scenario for the system.
  - **Method logic**: Just as you might use a UML sequence diagram to explore the logic of a use case, you can use it to explore the logic of any function, procedure, or complex process.
  - **Service logic**: If you consider a service to be a high-level method used by different clients, a sequence diagram is an ideal way to map that out.

BASIC SYMBOLS AND COMPONENTS

Symbol	Name	Description
	Object symbol	Represents a class or object in UML. The object symbol demonstrates how an object will behave in the context of the system. Class attributes should not be listed in this shape.
	Activation box	Represents the time needed for an object to complete a task. The longer the task will take, the longer the activation box becomes.
	Actor symbol	Shows entities that interact with or are external to the system.
	Package symbol	Used in UML 2.0 notation to contain interactive elements of the diagram. Also known as a frame, this rectangular shape has a small inner rectangle for labeling the diagram.

BASIC SYMBOLS AND COMPONENTS

Symbol	Name	Description
	Lifeline symbol	Represents the passage of time as it extends downward. This dashed vertical line shows the sequential events that occur to an object during the charted process. Lifelines may begin with a labeled rectangle shape or an actor symbol.
	Option loop symbol	Used to model if/then scenarios, i.e., a circumstance that will only occur under certain conditions.
	Alternative symbol	Symbolizes a choice (that is usually mutually exclusive) between two or more message sequences. To represent alternatives, use the labeled rectangle shape with a dashed line inside.

LIFELINE

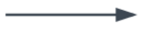
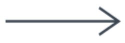
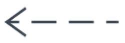
- A lifeline represents a single participant in an interaction. It describes how an instance of a specific classifier participates in the interaction.
- A lifeline represents a role that an instance of the classifier may play in the interaction.
- Following are various attributes of a lifeline,
    - Name : A name of a lifeline is optional. It is used to refer the lifeline within a specific interaction.
    - Type : It is the name of a classifier of which the lifeline represents an instance.
    - Selector : It is a Boolean condition which is used to select a particular instance that satisfies the requirement. Selector attribute is also optional

MESSAGE

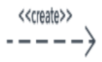
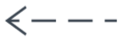
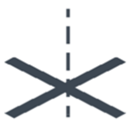
- A message is a specific type of communication between two lifelines in an interaction. A message involves following activities,
    - A call message which is used to call an operation.
    - A message to create an instance.
    - A message to destroy an instance.
    - For sending a signal.
- When a lifeline receives a call message, it acts as a request to invoke an operation that has a similar signature as specified in the message.
- When a lifeline is executing a message, it has a focus of control. As the interaction progresses over time, the focus of control moves between various lifelines. This movement is called a flow of control.

COMMON MESSAGE SYMBOLS

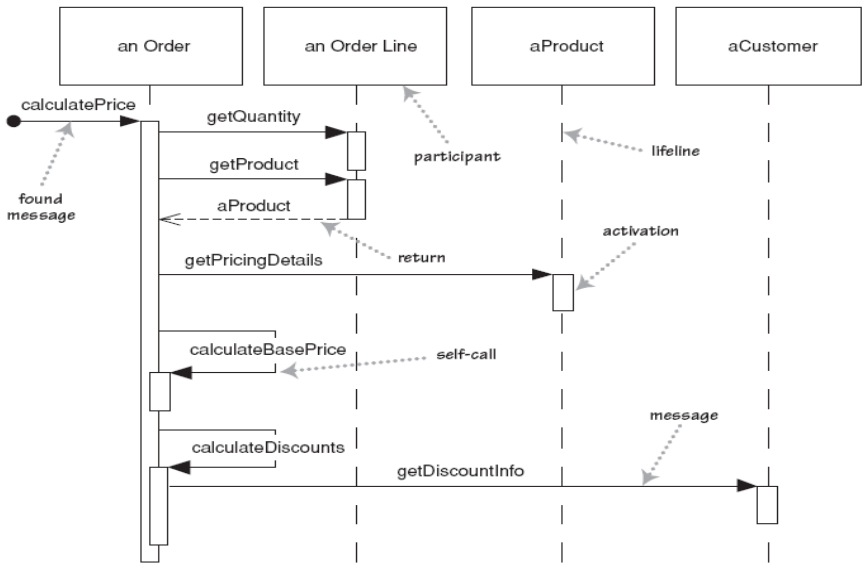
- Use the following arrows and message symbols to show how information is transmitted between objects. These symbols may reflect the start and execution of an operation or the sending and reception of a signal.

Symbol	Name	Description
	Synchronous message symbol	Represented by a solid line with a solid arrowhead. This symbol is used when a sender must wait for a response to a message before it continues. The diagram should show both the call and the reply.
	Asynchronous message symbol	Represented by a solid line with a lined arrowhead. Asynchronous messages don't require a response before the sender continues. Only the call should be included in the diagram.
	Asynchronous return message symbol	Represented by a dashed line with a lined arrowhead.

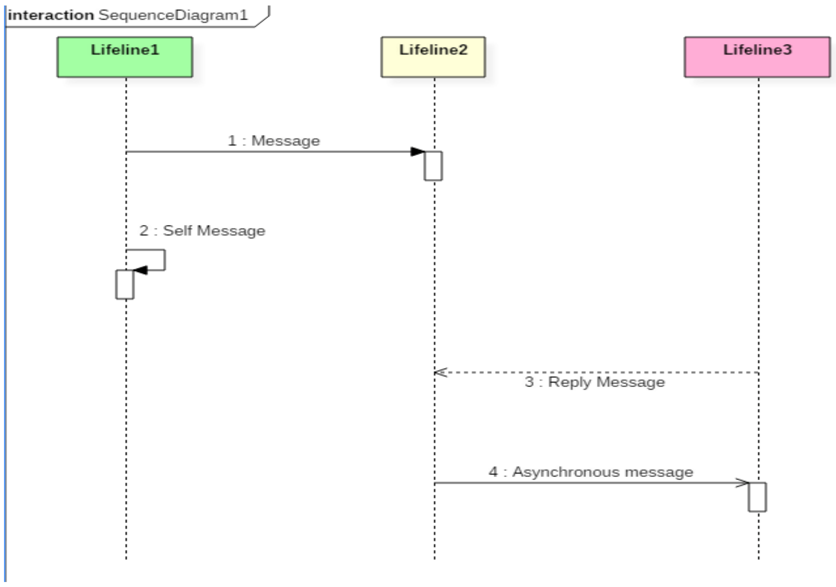
COMMON MESSAGE SYMBOLS

Symbol	Name	Description
	Asynchronous create message symbol	Represented by a dashed line with a lined arrowhead. This message creates a new object.
	Reply message symbol	Represented by a dashed line with a lined arrowhead, these messages are replies to calls.
	Delete message symbol	Represented by a solid line with a solid arrowhead, followed by an X. This message destroys an object.

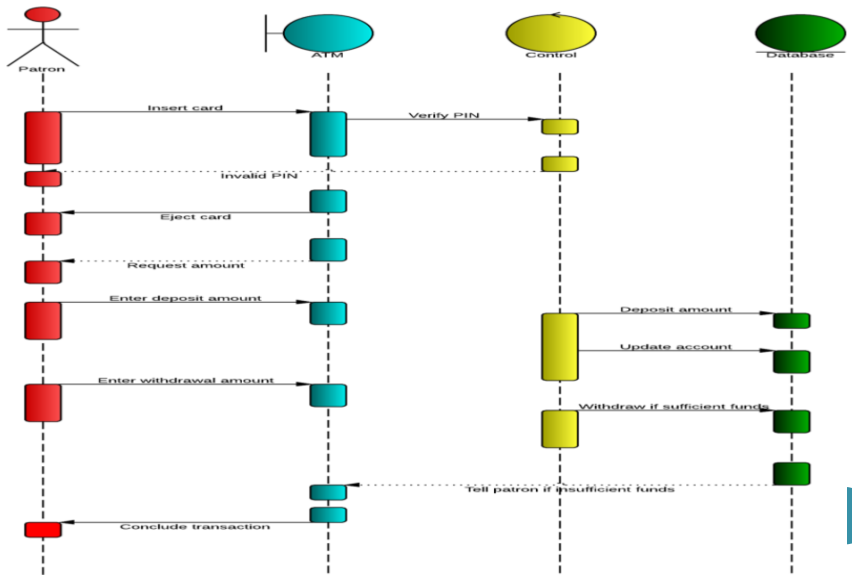
SEQUENCE DIAGRAM



SEQUENCE DIAGRAM



SEQUENCE DIAGRAM FOR ATM SYSTEMS

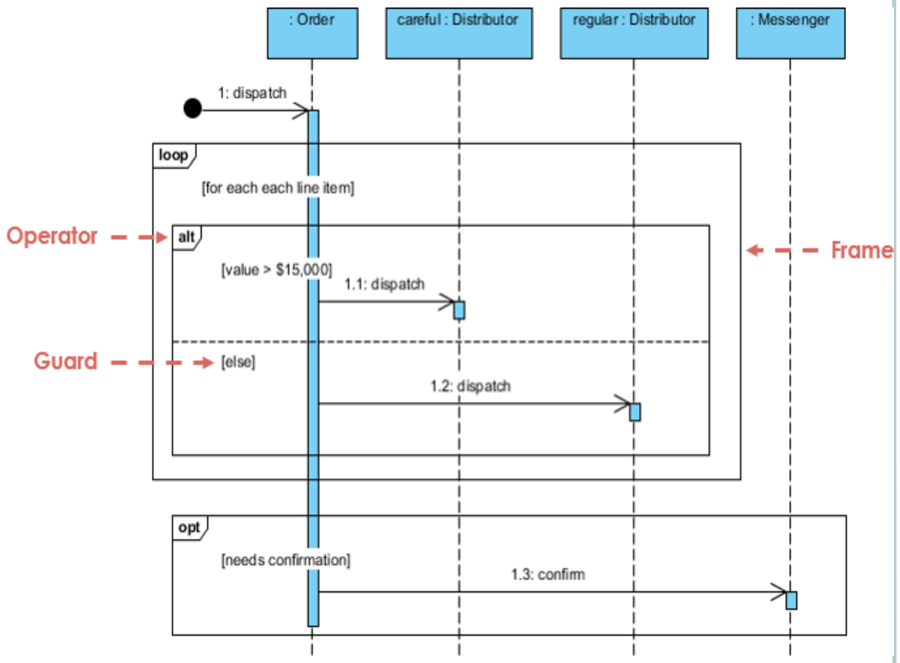


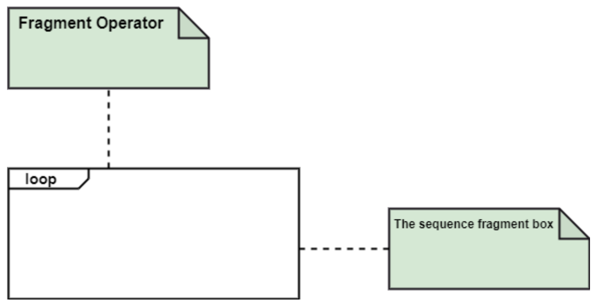
BENEFITS OF A SEQUENCE DIAGRAMS

- Sequence diagrams are used to explore any real application or a system.
- Sequence diagrams are used to represent message flow from one object to another object.
- Sequence diagrams are easier to maintain.
- Sequence diagrams are easier to generate.
- Sequence diagrams can be easily updated according to the changes within a system.
- Sequence diagram allows reverse as well as forward engineering.

DRAWBACK OF A SEQUENCE DIAGRAMS

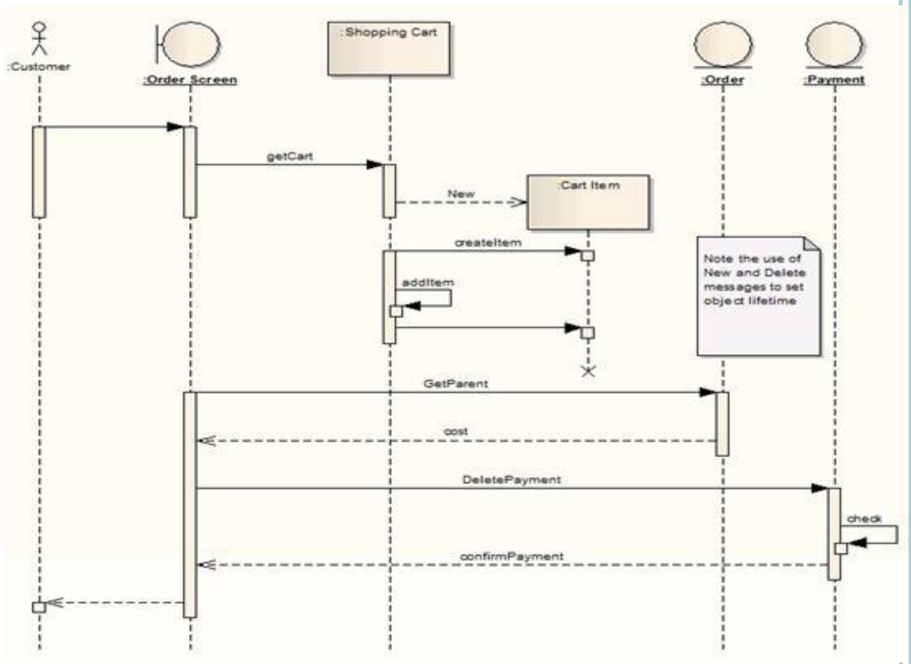
- Sequence diagrams can become complex when too many lifelines are involved in the system.
- If the order of message sequence is changed, then incorrect results are produced.
- Each sequence needs to be represented using different message notation, which can be a little complex.
- The type of message decides the type of sequence inside the 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.


:Customer
:Order Screen
:Shopping Cart
:Order
:Payment
getCart
New
:Cart Item
createItem
addItem
Note the use of New and Delete messages to set object lifetime
GetParent
cost
DeletePayment
check
confirmPayment