UML

- UML stands for Unified Modeling Language created by Object Management Group (OMG).
- It is a language for specifying, visualizing, constructing, and documenting the artifacts of software systems.
- UML is a pictorial language used to make software blueprint and it is different from the other common programming languages like C++, Java, COBOL etc.
- UML is described as a general-purpose visual modeling language.
- UML is not a programming language but tools can be used to generate code in various languages using UML diagrams.
- UML concept has a direct relation with object-oriented analysis and design.

# Goals of UML:

- The primary goal of UML is to define some general-purpose simple modeling language so that all modelers can use and understand.
- UML is not a development method rather it accompanies with processes to make a successful system.

Object-oriented concepts:

- UML diagrams are the representation of object-oriented concepts only.
- Thus, before learning UML, it becomes important to understand OO concept in detail.
- UML can be described as the successor of object-oriented analysis and design.

Following are some fundamental concepts of the object-oriented world:

➔ **Class:** A class is a blueprint from which individual objects can be created.

➔ **Objects:** An object represent a particular instance of a class.

➔ **Abstraction:** Abstraction means hiding implementation details from users.

➔ **Encapsulation:** Encapsulation is the mechanism that binds together data and code to save from outsiders.

➔ **Inheritance:** It means taking something that already exists i.e it provides reusability.

➔ **Polymorphism:** It defines the mechanism to exists in different forms.

OO Analysis and Design

- **Step1** is to identify and analyze objects and describing them in a proper way. Objects need to be identified with responsibility i.e the kind of work they will perform.
- **Step 2** is to design. If objects are identified correctly then the design part is easy. In this stage, the objects are collaborated according to their intended association. After the association is complete the design is also complete.
- **Step 3** is implementation in which design is implemented using object-oriented languages like Java, C++ etc.

# Classification of UML diagrams:-

- UML is linked with object-oriented design and analysis.
- UML use Object Oriented elements and forms associations between them to form diagrams.

UML Diagrams can be broadly classified as:

1. **Structural Diagrams** – It captures static aspects or structure of a system. Structure diagrams are not utilized for real-time concepts, because they do not show the details of dynamic behavior. But they may show relationships to the behaviors of the elements in the structure diagrams.
2. **Behavior Diagrams** – It capture dynamic aspects or behavior of the system. It describes a set of actions that any system should or can perform.

# Class Diagram

- Class diagram with the building block of all the object-oriented software applications and it is the most widely used UML diagram
- It is used to show the static structure of a system by showing classes, methods, and attributes.
- It also shows the relationship between different classes, methods, and objects.
- Used for visualizing, describing, and documenting different aspects of a system.
- Along with showing attributes and methods it also shows the kind of operation they will perform and the constraints imposed on these methods and attributes.
- This type of UML diagram is widely used because it can provide direct mapping with object-oriented languages

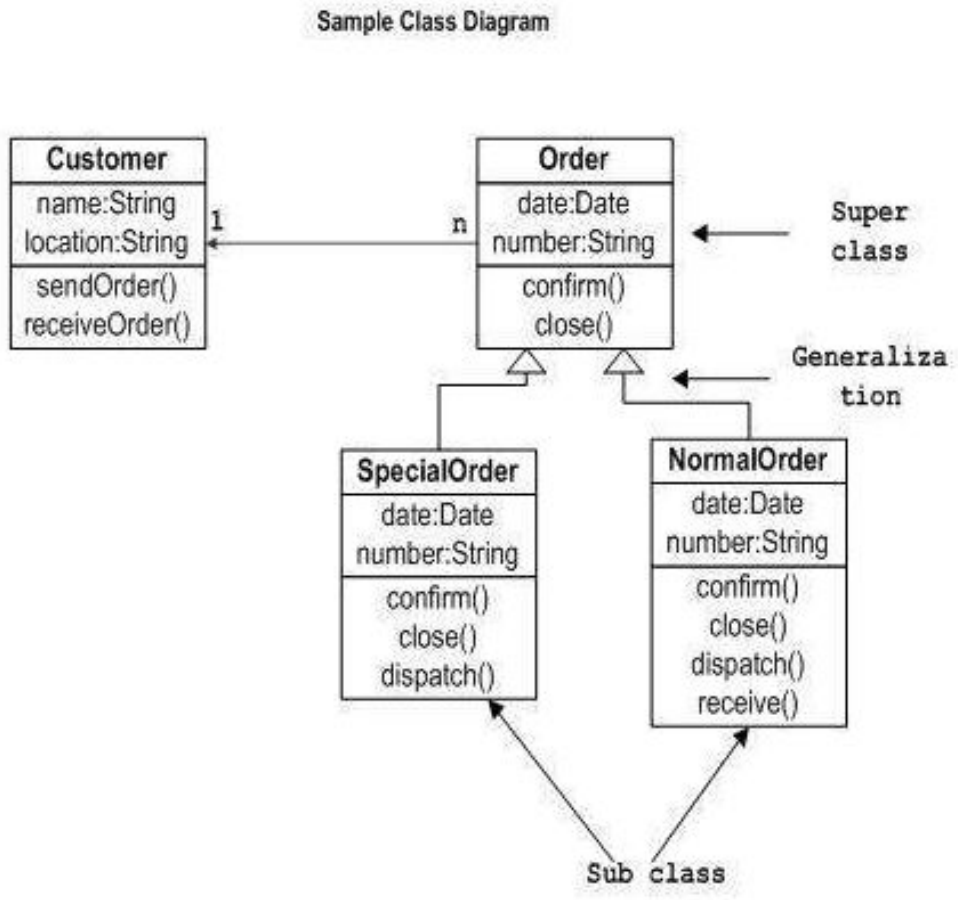
## How to draw the Class Diagram?

The following points should be remembered while drawing a class diagram:

- The name of the Class diagram should be clear and meaningful to describe the system by its own.
- All the relationships between the elements should be identified in advance
- For Class diagram attributes and methods are called as responsibility so the responsibility of each class should be clearly identified
- Only the minimum number of properties should be specified because unnecessary properties can make the diagram complicated
- If needed notes can be used to describe some extra aspects of your diagram
- Finally, the diagram should be drawn on paper or by using any software

## Example

- First of all Order and Customer are identified as the two elements of the system and they have a one to many relationships because a customer can have multiple orders.
- We would keep Order class is an abstract class and it has two concrete classes (inheritance relationship) SpecialOrderand NormalOrder.
- The two inherited classes have all the properties as the Order class. In addition, they have additional functions like dispatch () and receive ().




Relationships

There are three principal kinds of relationships which are used in the object-oriented analysis.

- ➢ Association - represent relationships between instances of types (a person works for a company, a company has a number of offices.
- ➢ Inheritance - the most obvious addition to ER diagrams for use in OO. It has an immediate correspondence to inheritance in OO design.
- ➢ Aggregation - Aggregation, a form of object composition in object-oriented design.

Where to use Class Diagrams?

- Describing the static view of the system.
- Showing the collaboration among the elements of the static view.
- Describing the functionalities performed by the system.
- Construction of software applications using object-oriented languages.

# Object diagrams

- Object diagrams are derived from class diagrams so object diagrams are dependent upon class diagrams.
- Object diagrams represent an instance of a class diagram.
- Object diagrams represent the static view of a system but this static view is a snapshot of the system at a particular moment.
- Object diagrams are used to render a set of objects and their relationships as an instance
- An Object Diagram represents specific instances of classes and relationships between them at a point of time.

## How to draw Object Diagram?

- Object diagram consists of instances of things used in a class diagram.
- In object diagrams, we can have an unlimited number of instances which are unique in nature. So only those instances are considered which are having an impact on the system.
- First, analyze the system and decide which instances are having important data and association.
- Second, consider only those instances which will cover the functionality.
- Third, make some optimization as the numbers of instances are unlimited.

**NOTE**

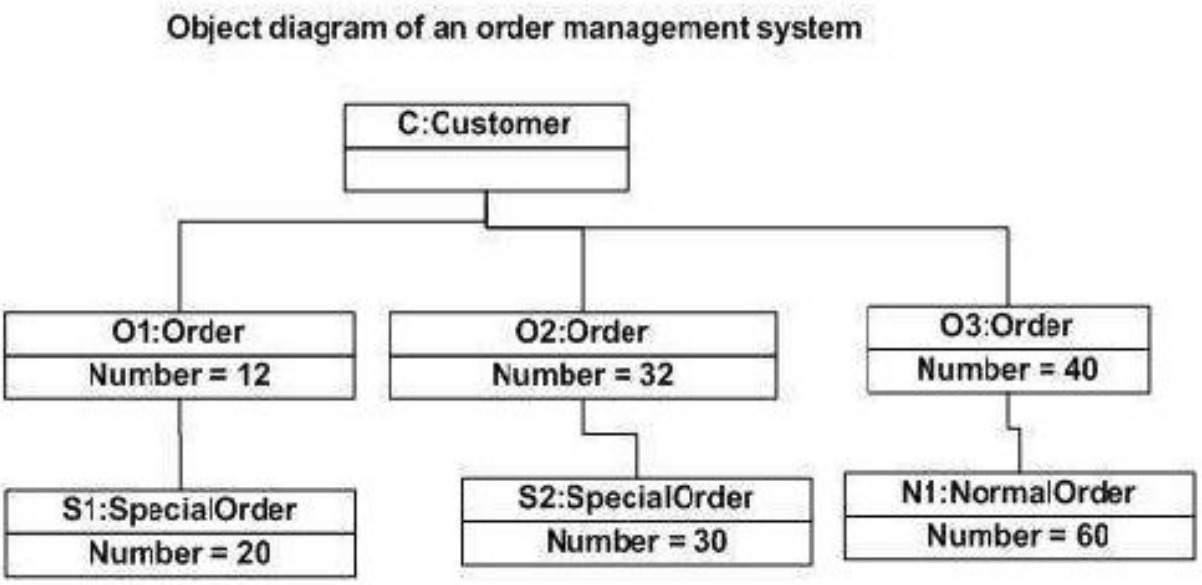
- Object diagrams are made up of objects.
- The link in the object diagram is used to connect objects.
- Objects and links are the two elements used to construct an object diagram.
- The object diagram should have a meaningful name to indicate its purpose.
- The most important elements are to be identified first.
- The association among objects should be clarified.
- Values of different elements need to be captured to include in the object diagram.
- Add proper notes at points where more clarity is required.

## Example

Objects for the Order *management system* (which we have discussed in *Class Diagram)*.

1. Customer
2. Order
3. SpecialOrder
4. NormalOrder

- The customer object (C) is associated with three order objects (O1, O2, and O3).
- These order objects are associated with the special order and normal order objects (S1, S2, and N1).
- The customer is having the following three orders with different numbers (12, 32 and 40) for the particular time considered.




- Now the customer can increase the number of orders in the future and in that scenario, the object diagram will reflect that.
- For orders, the values are 12, 32, and 40 which implies that the objects are having these values for the particular moment (here the particular time when the purchase is made is considered as the moment) when the instance is captured.
- The same is for special order and normal order objects which are having the number of orders as 20, 30 and 60. If a different time of purchase is considered then these values will change accordingly.

## Where to use Object Diagrams?

object diagrams are used for:

- Making the prototype of a system.
- Reverse engineering.
- Modeling complex data structures.
- Understanding the system from a practical perspective.

# Component diagram

- Component diagrams are quite different in nature and they actually represent how the physical components in a system have been organized?.
- Component Diagrams depict the structural relationship between software system components and help us in understanding whether all functional requirements have been covered by planned development.
- Component Diagrams become essential to use when we design and build complex systems.
- Component diagrams are used to visualize the organization and relationships among components in a system. These diagrams are also used to make executable systems.
- It does not describe the functionality of the system but it describes the components used to make those functionalities.
- A single component diagram cannot represent the entire system but a collection of diagrams are used to represent the whole.

## purpose

- Visualize the components of a system.
- Construct executables by using forward and reverse engineering.
- Describe the organization and relationships of the components.

## How to draw a Component Diagram?

- Component diagrams are used to describe the physical artifacts of a system. This artifact includes files, executables, libraries etc.
- Initially, the system is designed using different UML diagrams and then when the artifacts are ready component diagrams are used to get an idea of the implementation.

So before drawing a component diagram, the following artifacts are to be identified clearly:

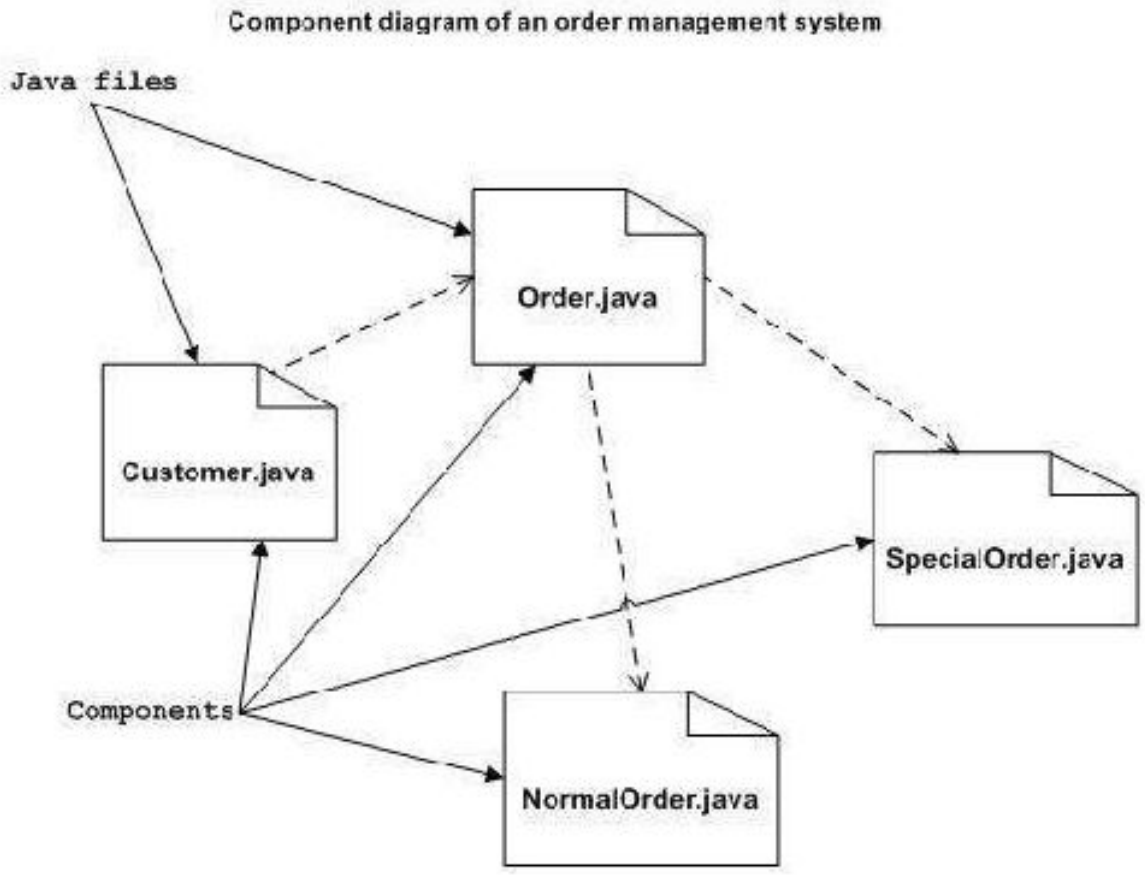
1. Files used in the system.
2. Libraries and other artifacts relevant to the application.
3. Relationships among the artifacts.

## NOTE

- Use a meaningful name to identify the component for which the diagram is to be drawn.
- Prepare a mental layout before producing using tools.
- Use notes for clarifying important points.

## Example

- The diagram shows the files in the application and their relationships.
- In actual, the component diagram also contains dlls, libraries, folders etc.




## Where to use Component Diagrams?

Component Diagram can be used to:-

- Model the components of a system.
- Model database schema.
- Model executables of an application.
- Model system's source code.

# Deployment diagrams

- The word *Deployment* itself describes the purpose of the diagram
- It's close to the component diagram and it shows the complete deployment
- Deployment Diagrams are used to represent hardware of the system and its software.
- It gives brief information about what hardware components exist and what software components run on them.
- Deployment diagrams are used to describe the static deployment view of a system. Deployment diagrams consist of nodes and their relationships.

## Purpose:

- Deployment diagrams are used for describing the hardware components where software components are deployed. Component diagrams and deployment diagrams are closely related.
- Component diagrams are used to describe the components and deployment diagrams shows how they are deployed in hardware.
- UML is mainly designed to focus on software artifacts of a system. But these two diagrams are special diagrams used to focus on software components and hardware components.
- So most of the UML diagrams are used to handle logical components but deployment diagrams are made to focus on hardware topology of a system. Deployment diagrams are used by system engineers.

The purpose of deployment diagrams can be described as:

- Visualize hardware topology of a system.
- Describe the hardware components used to deploy software components.
- Describe runtime processing nodes.

## How to draw the Deployment Diagram?

Deployment diagram represents the deployment view of a system. It is related to the component diagram. Because the components are deployed using the deployment diagrams. A deployment diagram consists of nodes. Nodes are nothing but physical hardware used to deploy the application.
Deployment diagrams are useful for system engineers. An efficient deployment diagram is very important because it controls the following parameters

- Performance

- Scalability
- Maintainability
- Portability

So before drawing a deployment diagram, the following artifacts should be identified:

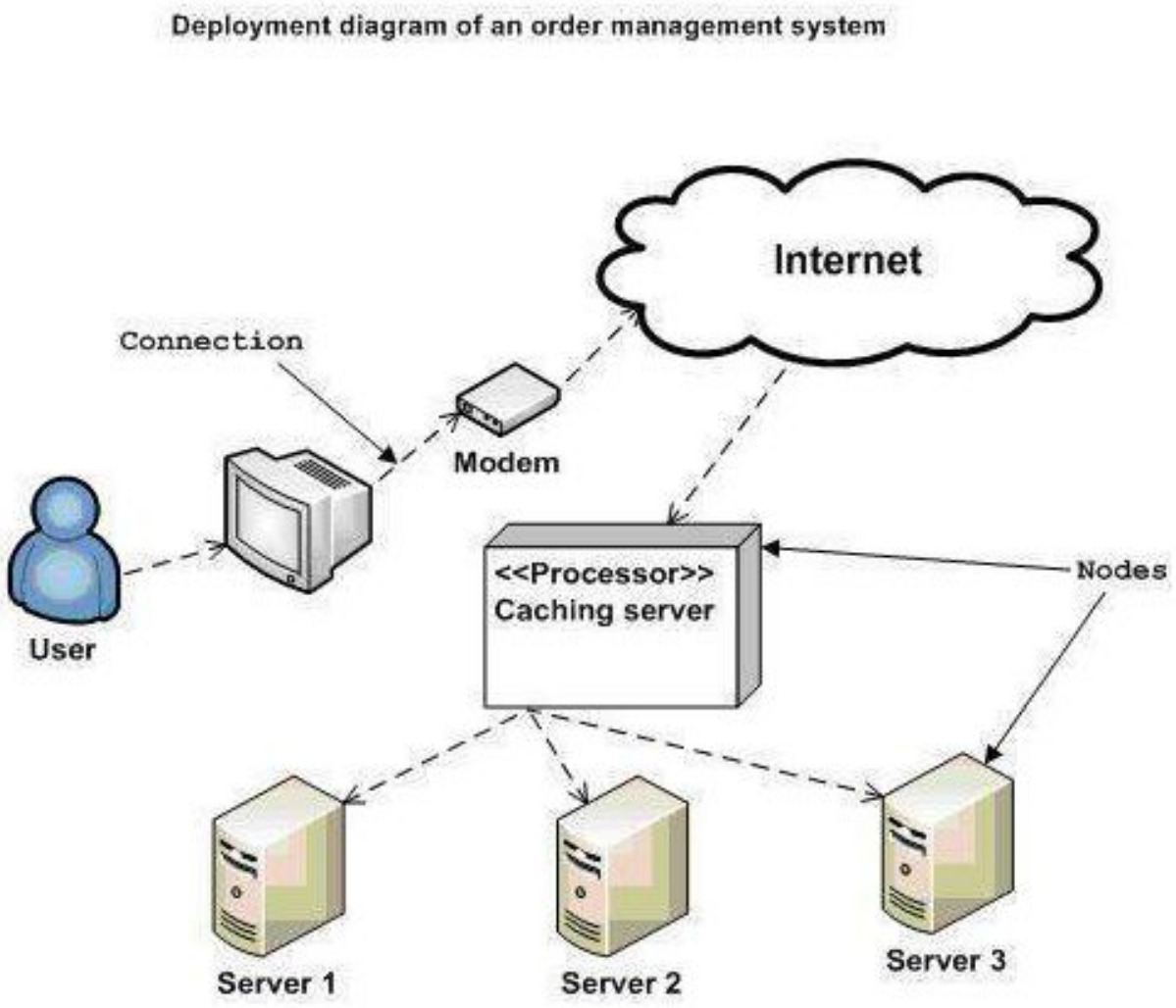
- Nodes
- Relationships among nodes

The following deployment diagram is a sample to give an idea of the deployment view of the order management system. Here we have shown nodes as:

- Monitor
- Modem
- Caching server
- Server

The application is assumed to be a web based application which is deployed in a clustered environment using server 1, server 2 and server 3. The user is connecting to the application using the internet. The control is flowing from the caching server to the clustered environment.

So the following deployment diagram has been drawn considering all the points mentioned above:




## Where to use Deployment Diagrams?

So the usage of deployment diagrams can be described as follows:

- To model the hardware topology of a system.
- To model embedded system.
- To model hardware details for a client/server system.
- To model hardware details of a distributed application.
- Forward and reverse engineering

# Interaction Diagram

- Interaction diagrams focus on the overview of the flow of control where the nodes are interactions.
- An interaction describes an array of messages which are exchanged by a selected number of participants in a chronologically-limited circumstance.
- An interaction diagram is used to show the interactive behavior of a system.
- It can control flow with nodes that can contain interaction diagrams which show how a set of fragments might be initiated in various scenarios.

The other notation elements are

- ❖ Initial
- ❖ Final
- ❖ Decision
- ❖ Merge
- ❖ Fork
- ❖ Join nodes

From the name Interaction it is clear that the diagram is used to describe some type of interactions among the different elements in the model. So this interaction is a part of dynamic behavior of the system. Dynamic aspect can be defined as the snapshot of the running system at a particular moment.
This interactive behavior is represented in UML by two diagrams known as Sequence diagram and Collaboration diagram. The basic purposes of both the diagrams are similar. Sequence diagram emphasizes on time sequence of messages and collaboration diagram emphasizes on the structural organization of the objects that send and receive messages.

So the purposes of interaction diagram can be describes as:

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

# The Sequence Diagram:

- A Sequence Diagram is predominantly concerned with the chronological progression of messages.
- The messaging sequence  concur with horizontal position in the diagram. When an object is created, and what object information is sent, are all determined in sequence diagram.
- The participating objects are represented by a rectangle and a dashed vertical line.
- Both together are known as Lifeline.
- Messages are shown using arrows between the Lifelines.
- Time progresses is shown from top to bottom. The chronological progress of messages is thereby highlighted.

## Symbols:-

### System Border

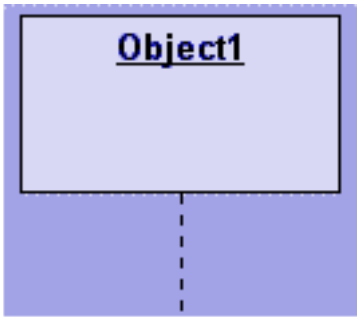
- The System Border isolates the concerned part of the program from the rest of the program.
- It usually serves as the start point of the method call.

### Object

- An object is shown by a rectangular box containing the name.


### Lifeline

- Every object is on a vertical line - that is considered as the Lifeline.
- An object's lifeline grows in the downward direction.
- It shows the time till which object is active.



## Object Activity

- If an object is involved in a method call, it is said to be active and the Lifeline thickens.
- If an object calls its own method, the Lifeline thickens again.

## Method Call

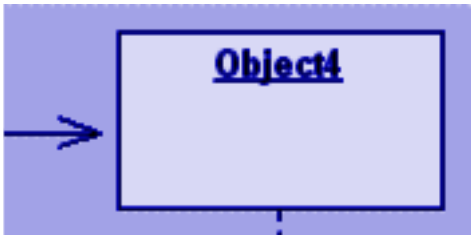
- When an object calls a method of another object, this is symbolized by a continuous arrow which points to the object from which the method was called.
- The method name of called method is placed on this symbol


## Method Return

- Method returns can be done with an arrow and a dashed line.
- It is not generally required.

## Object Creation

- If a method creates an object, the method's arrow ends on the object's rectangular symbol. The Lifeline begins at this symbol.




## Object Destruction

- If an object is destroyed when a method is called, the object's Lifeline ends with a cross below the method call symbol.

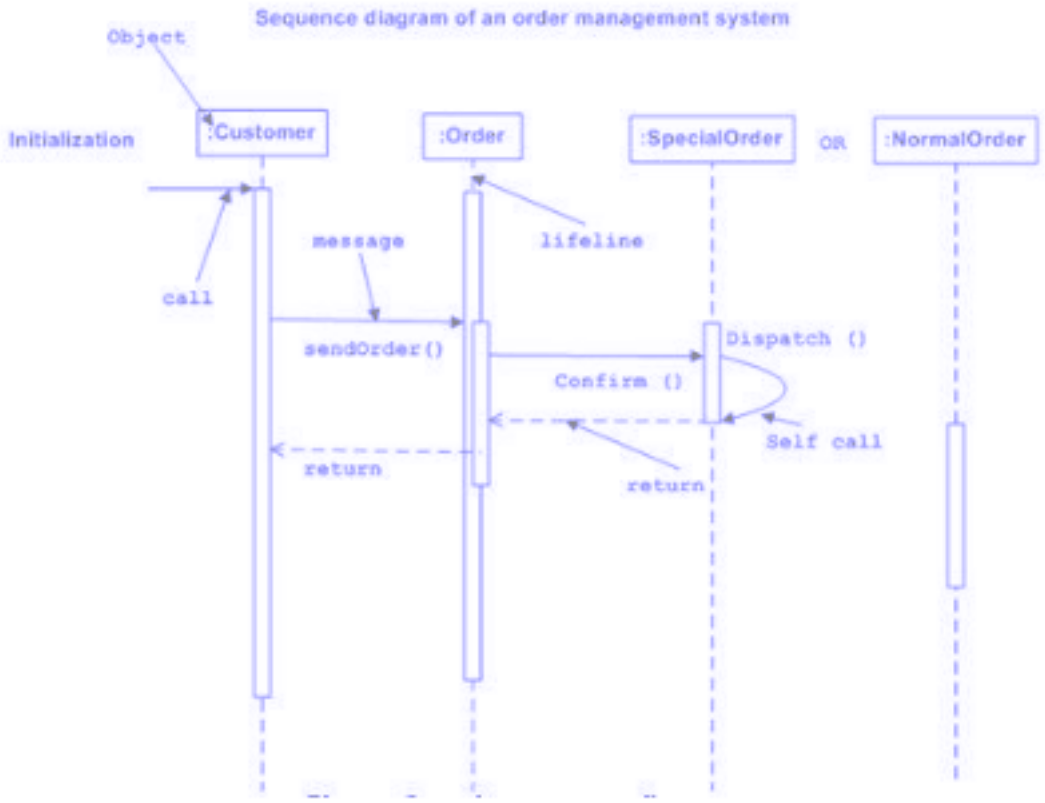
## Example

This sequence diagram is having four objects

1. Customer
2. Order
3. SpecialOrder
4. NormalOrder

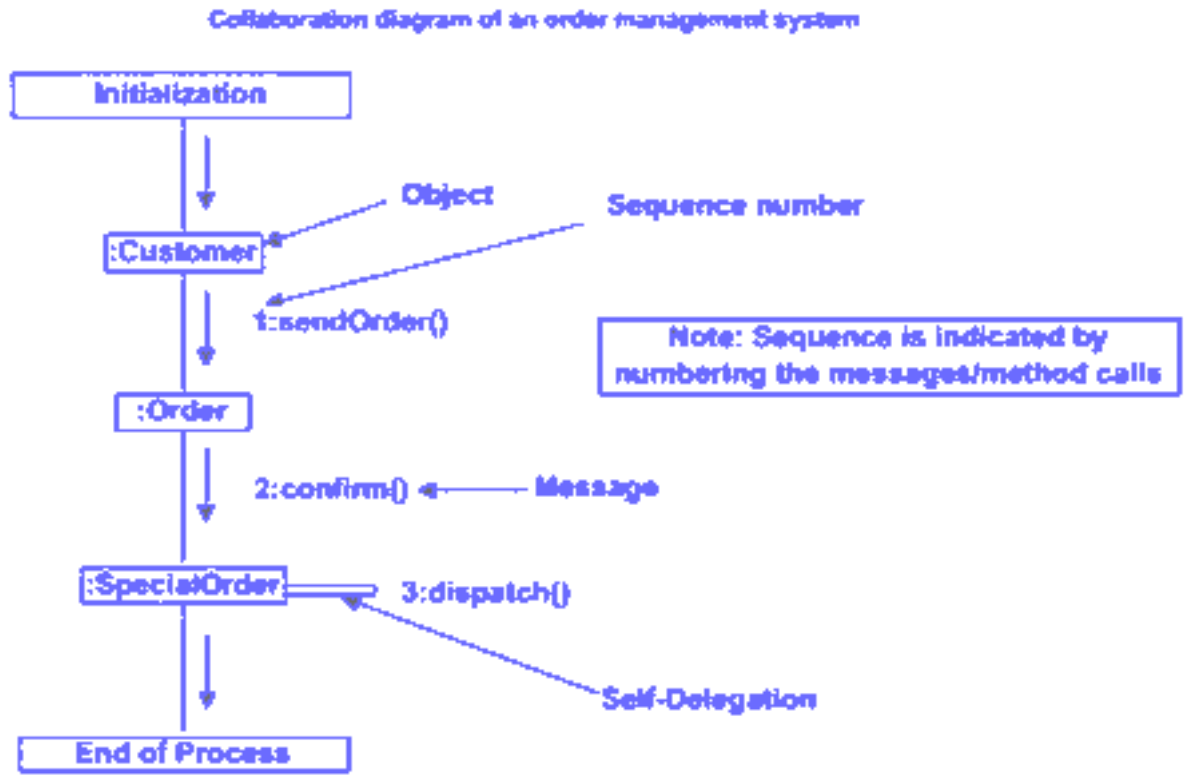
**Working :-**

- The first call is sendOrder () which is a method of Order object.
- The next call isconfirm () which is a method of SpecialOrder object.
- The last call is Dispatch () which is a method of SpecialOrder object



# The Collaboration Diagram:

- A Collaboration diagram is a collection of all objects and actors with links connecting them which collaborate in performing some task.
- A Collaboration between objects working together provides desirable functionalities in Object-Oriented systems
- The difference between  sequence diagram and collaboration diagram is that sequence diagram does not describe the object organization where as the collaboration diagram shows the object organization.




## Symbols:-

### Object

- This rectangle is the symbol of an object and contains the object name. No confusion with classes may occur, so the name must not be underlined.


### Communication

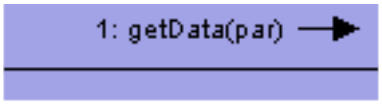
- When two objects communicate with one another via a method call, this link is shown with a solid line connecting both objects.

## Communication Direction

- In addition to the connecting line, after the method name an arrow shows to which object the method belongs. The arrow points to this object.


## Label

- This line is labeled with the method name and the parameter list.
- Numbers placed in front of the method name indicate the chronological order of method calls, separated by a colon.
- You can enter the method numbers in a condition which are, as a precondition, already being processed.
- The * symbol in front of the method name indicates a repeat of that method. The condition for repetition can be entered after the * symbol.




## Where to use Interaction Diagrams?

Interaction diagrams can be used :

- To model flow of control by time sequence.
- To model flow of control by structural organizations.
- For forward engineering.
- For reverse engineering.