Object-oriented modeling (**OOM**) is an approach to modeling an application that is used at the beginning of the software life cycle when using an object-oriented approach to software development.

The software life cycle is typically divided up into stages going from abstract descriptions of the problem to designs then to code and testing and finally to deployment. Modeling is done at the beginning of the process. The reasons to model a system before writing the code are:

- Communication. Users typically cannot understand programming language or code. Model diagrams can be more understandable and can allow users to give developers feedback on the appropriate structure of the system. A key goal of the Object-Oriented approach is to decrease the "semantic gap" between the system and the real world by using terminology that is the same as the functions that users perform. Modeling is an essential tool to facilitate achieving this goal .
- Abstraction. A goal of most software methodologies is to first address "what" questions and then address "how" questions. I.e., first determine the functionality the system is to provide without consideration of implementation constraints and then consider how to take this abstract description and refine it into an implementable design and code given constraints such as technology and budget. Modeling enables this by allowing abstract descriptions of processes and objects that define their essential structure and behavior.

Object-oriented modeling is typically done via use cases and abstract definitions of the most important objects. The most common language used to do object-oriented modeling is the Object Management Group's Unified Modeling Language (UML).[1]

Object diagrams are derived from class diagrams so object diagrams are dependent upon class diagrams.

Object diagrams represent an instance of a class diagram. The basic concepts are similar for class diagrams and object diagrams. Object diagrams also 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.

Purpose of Object Diagrams

The purpose of a diagram should be understood clearly to implement it practically. The purposes of object diagrams are similar to class diagrams.

The difference is that a class diagram represents an abstract model consisting of classes and their relationships. However, an object diagram represents an instance at a particular moment, which is concrete in nature.

It means the object diagram is closer to the actual system behavior. The purpose is to capture the static view of a system at a particular moment.

The purpose of the object diagram can be summarized as −

- Forward and reverse engineering.
- Object relationships of a system
- Static view of an interaction.
- Understand object behaviour and their relationship from practical perspective

How to Draw an Object Diagram?

We have already discussed that an object diagram is an instance of a class diagram. It implies that an object diagram consists of instances of things used in a class diagram.

So both diagrams are made of same basic elements but in different form. In class diagram elements are in abstract form to represent the blue print and in object diagram the elements are in concrete form to represent the real world object.

To capture a particular system, numbers of class diagrams are limited. However, if we consider object diagrams then we can have unlimited number of instances, which are unique in nature. Only those instances are considered, which have an impact on the system.

From the above discussion, it is clear that a single object diagram cannot capture all the necessary instances or rather cannot specify all the objects of a system. Hence, the solution is −

- First, analyze the system and decide which instances have important data and association.
- Second, consider only those instances, which will cover the functionality.
- Third, make some optimization as the number of instances are unlimited.

Before drawing an object diagram, the following things should be remembered and understood clearly −

- Object diagrams consist of objects.
- The link in object diagram is used to connect objects.
- Objects and links are the two elements used to construct an object diagram.

After this, the following things are to be decided before starting the construction of the diagram −

- The object diagram should have a meaningful name to indicate its purpose.
- The most important elements are to be identified.
- 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.

The following diagram is an example of an object diagram. It represents the Order management system which we have discussed in the chapter Class Diagram. The following diagram is an instance of the system at a particular time of purchase. It has the following objects.

- Customer
- Order
- SpecialOrder
- NormalOrder

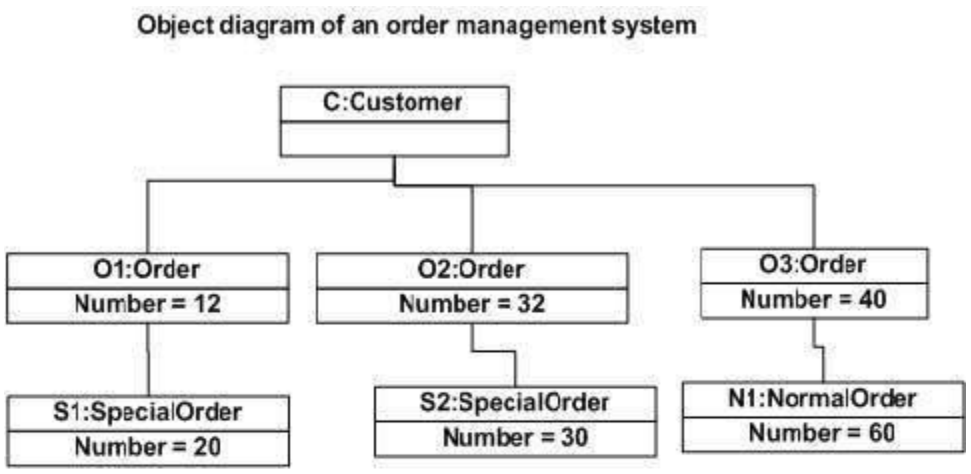
Now the customer object (C) is associated with three order objects (O1, O2, and O3). These order objects are associated with special order and normal order objects (S1, S2, and N1). The customer has the following three orders with different numbers (12, 32 and 40) for the particular time considered.

The customer can increase the number of orders in future and in that scenario the object diagram will reflect that. If order, special order, and normal order objects are observed then you will find that they have some values.

For orders, the values are 12, 32, and 40 which implies that the objects have these values for a particular moment (here the particular time when the purchase is made is considered as the moment) when the instance is captured

The same is true for special order and normal order objects which have number of orders as 20, 30, and 60. If a different time of purchase is considered, then these values will change accordingly.

The following object diagram has been drawn considering all the points mentioned above



Where to Use Object Diagrams?

Object diagrams can be imagined as the snapshot of a running system at a particular moment. Let us consider an example of a running train

Now, if you take a snap of the running train then you will find a static picture of it having the following −

- A particular state which is running.
- A particular number of passengers. which will change if the snap is taken in a different time

Here, we can imagine the snap of the running train is an object having the above values. And this is true for any real-life simple or complex system.

In a nutshell, it can be said that object diagrams are used for −

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

Question: What do you mean by object modeling technique?

Answer:

The object modeling techniques is an methodology of object oriented analysis, design and implementation that focuses on creating a model of objects from the real world and then to use this model to develop object–oriented software. object modeling technique, OMT was developed by James Rambaugh. Now-adays, OMT is one of the most popular object oriented development techniques. It is primarily used by system and software developers to support full life cycle development while targeting object oriented implementations.

OMT has proven itself easy to understand, to draw and to use. It is very successful in many application domains: telecommunication, transportation, compilers etc. The popular object modeling technique are used in many real world problems. The object-oriented paradigm using the OMT spans the entire development cycle, so there is no need to transform one type of model to another.

Phase of OMT

The OMT methodology covers the full software development life cycle. The methodology has the following phase.

1. **Analysis** - Analysis is the first phase of OMT methodology. The aim of analysis phase is to build a model of the real world situation to show its important properties and domain. This phase is concerned with preparation of precise and correct modelling of the real world. The analysis phase starts with defining a problem statement which includes a set of goals. This problem statement is then expanded into three models; an object model, a dynamic model and a functional model. The object model shows the static data structure or skeleton of the real world system and divides the whole application into objects. In others words, this model represents the artifacts of the system. The dynamic model represents the interaction between artifacts above designed represented as events, states and transitions. The functional model represents the methods of the system from the data flow perspective. The analysis phase generates object model diagrams, state diagrams, event flow diagrams and data flow diagrams.
2. **System design** - The system design phase comes after the analysis phase. System design phase determines the overall system architecture using subsystems, concurrent tasks and data storage. During system design, the high level structure of the system is designed. The decisions made during system design are:
    - The system is organized in to sub-systems which are then allocated to processes and tasks, taking into account concurrency and collaboration.
    - Persistent data storage is established along with a strategy to manage shared or global information.
    - Boundary situations are checked to help guide trade off priorities.
3. **Object design** - The object design phase comes after the system design phase is over. Here the implementation plan is developed. Object design is concerned with fully classifying the existing and remaining classes, associations, attributes and operations necessary for implementing a solution to the problem. In object design:
    - Operations and data structures are fully defined along with any internal objects needed for implementation.
    - Class level associations are determined.
    - Issues of inheritance, aggregation, association and default values are checked.
4. **Implementation** - Implementation pahse of the OMT is a matter of translating the design in to a programming language constructs. It is important to have good software engineering practice so that the design phase is smoothly translated in to the implementation phase. Thus while selecting programming language all constructs should be kept in mind for following noteworthy points.

- To increase flexibility.
- To make amendments easily.
- For the design traceability.
- To increase efficiency.

Object-Oriented Modelling and Object-Oriented Programming

Object-Oriented Modeling refers to the process where you are designing how the code will look like. You will use a modeling language like UML to do Object-Oriented Modeling. Object-Oriented Programming refers to a programming paradigm where you use objects. These objects have been designed during the desing phase using Object-Oriented Modeling techniques, and they are implemented during the construction (programming phase) using a language that supports Object-Oriented programming and based on the model.