Chapter 6

# OBJECT ORIENTED ANALYSIS AND DESIGN

**CHAPTER OUTLINE**

**After studying this chapter, students will be able to understand the:**

## INTRODUCTION

The term object oriented describes the system as a collection of discrete objects that incorporate both data structure and behavior. It is a way of thinking about problems using models organized around real world concepts. Object oriented analysis and design promote the better understanding of requirements, cleaner design and more maintainable system.

Object-oriented analysis is a method of analysis that examines requirements from the perspective of the classes and objects found in the vocabulary of the problem domain. Object-oriented design is a method of design encompassing the process of object-oriented decomposition and a notation for depicting logical and physical as well as static and dynamic models of the system under design.

## OBJECT ORIENTED DEVELOPMENT LIFE CYCLE

The Object Oriented Methodology of Building Systems takes the objects as the basis. For this, first the system to be developed is observed and analyzed and the requirements are defined as in any other method of system development. Once this is done, the objects in the required system are identified. For example in case of a Banking System, a customer is an object, a chequebook is an object, and even an account is an object. Object oriented development life cycle contains:

1. **System Analysis:** As in any other system development model, system analysis is the first phase of development in case of Object Modeling too. In this phase, the developer interacts with the user of the system to find out the user requirements and analyses the system to understand the functioning. Based on this system study, the analyst prepares a model of the desired system. This model is purely based on what the system is required to do. At this stage the implementation details are not taken care of. Only the model of the system is prepared based on the idea that the system is made up of a set of interacting objects. The important elements of the system are emphasized. While developing systems based on this approach, the analyst makes use of certain models to analyze and depict these objects.

   The methodology supports and uses three basic Models:

   - **Object Model:** This model describes the objects in a system and their interrelationships. This model observes all the objects as static and does not pay any attention to their dynamic nature.
   - **Dynamic Model:** This model depicts the dynamic aspects of the system. It portrays the changes occurring in the states of various objects with the events that might occur in the system.
   - **Functional Model:** This model basically describes the data transformations of the system. This describes the flow of data and the changes that occur to the data throughout the system.

2. **System Design:** System Design is the next development stage where the overall architecture of the desired system is decided. The system is organized as a set of sub systems interacting with each other. While designing the system as a set of interacting subsystems, the analyst takes care of specifications as observed in system analysis as well as what is required out of the new system by the end user. As the basic philosophy of Object-Oriented method of system analysis is to perceive the system as a set of interacting objects, a bigger system may also be seen as a set of interacting smaller subsystems that in turn are composed of a set of interacting objects. While designing the system, the stress lies on the objects comprising the system and not on the processes being carried out in the system as in the case of traditional Waterfall Model where the processes form the important part of the system.
3. **Object Design:** In this phase, the details of the system analysis and system design are implemented. The Objects identified in the system design phase are designed. Here the implementation of these objects is decided as the data structures get defined and also the interrelationships between the objects are defined. Object Oriented Philosophy is very much similar to real world and hence is gaining popularity as the systems here are seen as a set of interacting objects as in the real world. To implement this concept, the process-based structural programming is not used; instead objects are created using data structures. Just as every programming language provides various data types and various variables of that type can be created, similarly, in case of objects certain data types are predefined. For example, we can define a data type called pen and then create and use several objects of this data type. This concept is known as creating a class.
4. **Implementation:** During this phase, the class objects and the interrelationships of these classes are translated and actually coded using the programming language decided upon. The databases are made and the complete system is given a functional shape.

### BASIC CHARACTERISTICS OF OBJECT ORIENTED SYSTEM

1. **Class:** Objects with same data structure and behavior are grouped in to a class. A class is an abstraction that describes properties important to an application and ignores the rest. It is a collection of similar objects. It is a template where certain basic characteristics of a set of objects are defined. The class defines the basic attributes and the operations of the objects of that type. Defining a class does not define any object, but it only creates a template. For objects to be actually created instances of the class are created as per the requirement of the case.
2. **Abstraction:** Classes are built on the basis of abstraction, where a set of similar objects are observed and their common characteristics are listed. Of all these, the characteristics of concern to the system under observation are picked up and the class definition is made. The attributes of no concern to the system are left out. This is known as abstraction. The abstraction of an object varies according to its application. For instance, while defining a pen class for a stationery shop, the attributes of concern might be the pen color, ink color, pen type etc., whereas a pen class for a manufacturing firm would be containing the other dimensions of the pen like its diameter, its shape and size etc.

3. **Inheritance:** It is the sharing of attributes and operations among classes based on hierarchical relationship. A super class ha general information that subclasses refine and elaborate. Each sub class incorporates or inherits all the feature of its super class and adds its own unique features. This concept is used to apply the idea of reusability of the objects. A new type of class can be defined using a similar existing class with a few new features. For instance, a class vehicle can be defined with the basic functionality of any vehicle and a new class called car can be derived out of it with a few modifications. This would save the developers time and effort as the classes already existing are reused without much change. Coming back to our development process, in the Object Designing phase of the Development process, the designer decides onto the classes in the system based on these concepts. The designer also decides on whether the classes need to be created from scratch or any existing classes can be used as it is or new classes can be inherited from them.

4. **Polymorphism:** Polymorphism means the same operation may behave differently for different classes.

5. **Reusability:** The classes once defined can easily be used by other applications. This is achieved by defining classes and putting them into a library of classes where all the classes are maintained for future use. Whenever a new class is needed the programmer looks into the library of classes and if it is available, it can be picked up directly from there.

6. **Data Hiding:** Encapsulation is a technique that allows the programmer to hide the internal functioning of the objects from the users of the objects. Encapsulation separates the internal functioning of the object from the external functioning thus providing the user flexibility to change the external behaviour of the object making the programmer code safe against the changes made by the user. The systems designed using this approach are closer to the real world as the real world functioning of the system is directly mapped into the system designed using this approach .

### Advantages of Object Oriented Analysis and Design

- Object Oriented Methodology closely represents the problem domain. Because of this, it is easier to produce and understand designs.
- The objects in the system are immune to requirement changes. Therefore, allows changes more easily.
- Object Oriented Methodology designs encourage more re-use. New applications can use the existing modules, thereby reduces the development cost and cycle time.
- Object Oriented Methodology approach is more natural. It provides nice structures for thinking and abstracting and leads to modular design.

## INTRODUCTION TO UNIFIED MODELING LANGUAGE

UML short for Unified Modeling Language, is a standardized modeling language consisting of an integrated set of diagrams, developed to help system and software developers for specifying, visualizing, constructing, and documenting the artifacts of software systems, as well as for business modeling and other non-software systems.

It is a general purpose modeling language to define a standard way to visualize the way a system has been designed. It is quite similar to blueprints used in other fields of engineering.

UML is not a programming language, it is rather a visual language. We use UML diagrams to portray the behavior and structure of a system. UML helps software engineers, businessmen and system architects with modelling, design and analysis.

The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems. The UML is a very important part of developing object oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects. Using the UML helps project teams communicate, explore potential designs, and validate the architectural design of the software.

The Object Management Group (OMG) adopted Unified Modelling Language as a standard in 1997. Its been managed by OMG ever since. International Organization for Standardization (ISO) published UML as an approved standard in 2005.

The primary goals of the Object-Oriented Design in UML as follows:

- Provide users with a ready-to-use, expressive visual modeling language so they can develop and exchange meaningful models.
- Provide extensibility and specialization mechanisms to extend the core concepts.
- Be independent of particular programming languages and development processes.
- Provide a formal basis for understanding the modeling language.
- Encourage the growth of the OO tools market.
- Support higher-level development concepts such as collaborations, frameworks, patterns and components.
- Integrate best practices.

We prepare UML diagrams to understand the system in a better and simple way. A single diagram is not enough to cover all the aspects of the system. UML defines various kinds of diagrams to cover most of the aspects of a system.

There are two broad categories of diagrams and they are again divided into subcategories –

- Structural Diagrams
- Behavioral Diagrams

## Structural Diagrams

The structural diagrams represent the static aspect of the system. These static aspects represent those parts of a diagram, which forms the main structure and are therefore stable. Structural diagram assist in understanding and communicating the elements that make up a system and the functionality the system provides.

These static parts are represented by classes, interfaces, objects, components, and nodes. The four structural diagrams are –

- Class diagram
- Object diagram
- Component diagram
- Deployment diagram

### Class Diagram

Class diagrams are the most common diagrams used in UML. Class diagram consists of classes, interfaces, associations, and collaboration. Class diagrams basically represent the object-oriented view of a system, which is static in nature. It is the building block of all object oriented software systems. We use class diagrams to depict the static structure of a system by showing system's classes,their methods and attributes. Class diagrams also help us identify relationship between different classes or objects.

Class diagram represents the object orientation of a system. Hence, it is generally used for development purpose. This is the most widely used diagram at the time of system construction.

Class diagram is a static diagram. It represents the static view of an application. Class diagram is not only used for visualizing, describing, and documenting different aspects of a system but also for constructing executable code of the software application.

Class diagram describes the attributes and operations of a class and also the constraints imposed on the system. The class diagrams are widely used in the modeling of objectoriented systems because they are the only UML diagrams, which can be mapped directly with object-oriented languages. Class diagram shows a collection of classes, interfaces, associations, collaborations, and constraints. It is also known as a structural diagram.

The purpose of class diagram is to model the static view of an application. Class diagrams are the only diagrams which can be directly mapped with object-oriented languages and thus widely used at the time of construction.

The purpose of the class diagram can be summarized as –

- Analysis and design of the static view of an application.
- Describe responsibilities of a system.
- Base for component and deployment diagrams.
- Forward and reverse engineering.

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

- The name of the class diagram should be meaningful to describe the aspect of the system.
- Each element and their relationships should be identified in advance.
- Responsibility (attributes and methods) of each class should be clearly identified
- For each class, minimum number of properties should be specified, as unnecessary properties will make the diagram complicated.
- Use notes whenever required to describe some aspect of the diagram. At the end of the drawing it should be understandable to the developer/coder.
- Finally, before making the final version, the diagram should be drawn on plain paper and reworked as many times as possible to make it correct.

The following diagram is an example of an Order System of an application. It describes a particular aspect of the entire application.

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

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

### Object Diagram

Object diagrams can be described as an instance of class diagram. It depict the structure of a system at a particular point in time. Thus, these diagrams are more close to real-life scenarios where we implement a system. It can be referred to as a screenshot of the instances in a system and the relationship that exists between them. Since object diagrams depict behaviour when objects have been instantiated, we are able to study the behaviour of the system at a particular instant. An object diagram is similar to a class diagram except it shows the instances of classes in the system. We depict actual classifiers and their relationships making the use of class diagrams. On the other hand, an Object Diagram represents specific instances of classes and relationships between them at a point of time.

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

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

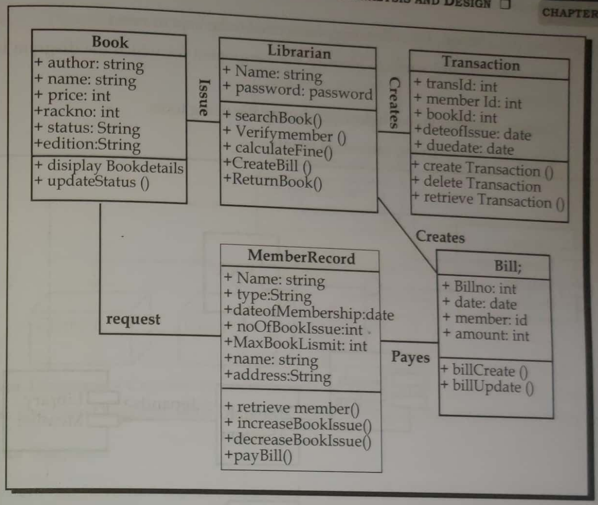
- First, analyze the system and decide which instances have important data and association.
- Second, consider only those instances, which will cover the functionality.
- Third, make some optimization as the number of instances are unlimited.

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.



### Component Diagram

Component diagrams represent a set of components and their relationships. These components consist of classes, interfaces, or collaborations. This diagram is used to represent the how the physical components in a system have been organized. We use them for modeling implementation details. Component Diagrams depict the structural relationship between software system elements and help us in understanding if functional requirements have been covered by planned development. Component Diagrams become essential to use when we design and build complex systems. Interfaces are used by components of the system to communicate with each other. Component diagrams represent the implementation view of a system. Component diagram also known as implementation diagrams, depict the implementation of a system.

The purpose of the component diagram can be summarized as –

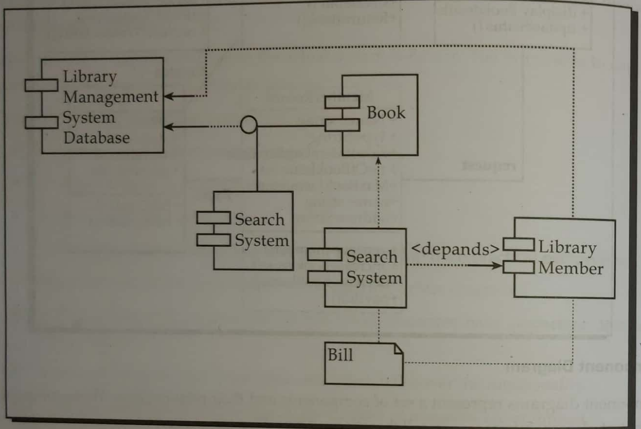
- Visualize the components of a system.
- Construct executables by using forward and reverse engineering.
- Describe the organization and relationships of the components.

Before drawing a component diagram, the following artifacts are to be identified clearly –

- Files used in the system.
- Libraries and other artifacts relevant to the application.
- Relationships among the artifacts.

After identifying the artifacts, the following points need to be kept in mind.

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



### Deployment Diagram

Deployment Diagrams, also known as the implementation diagram, depict the implementation environment of system are used to represent system hardware and its software. It tells us what hardware components exist and what software components run on them. They are primarily used when a software is being used, distributed or deployed over multiple machines with different configurations. Deployment diagrams has set of nodes and their relationships. These nodes are physical entities where the components are deployed. Deployment diagrams are used for visualizing the deployment view of a system. This is generally used by the deployment team.

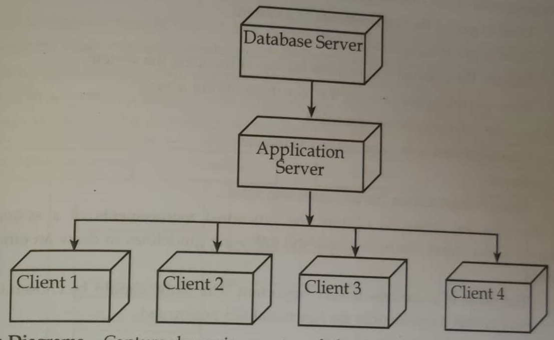
Deployment diagrams are used to visualize the topology of the physical components of a system, where the software components are deployed.

Deployment diagrams are used to describe the static deployment view of a system. Deployment diagrams consist of nodes and their relationships.

The purpose of deployment diagrams can be described as:

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

- Describe the runtime processing nodes.



**Behavior Diagrams –** Capture dynamic aspects or behavior of the system. Behavior diagrams include: Use Case Diagrams, State Diagrams, Activity Diagrams and Interaction Diagrams.

### Behavior Diagrams

Any system can have two aspects, static and dynamic. So, a model is considered as complete when both the aspects are fully covered. Behavioral diagrams basically capture the dynamic aspect of a system. Dynamic aspect can be further described as the changing/moving parts of a system. Behavioral diagram assist in understanding and communicating how elements interact and collaborate to provide the functionality of a system.

UML has the following five types of behavioral diagrams –

- Use case diagram
- Sequence diagram
- Collaboration diagram
- Statechart diagram
- Activity diagram

### Use Case Diagram

Use case diagrams are a set of use cases, actors, and their relationships. They represent the use case view of a system. A use case represents a particular functionality of a system. Hence, use case diagram is used to describe the relationships among the functionalities and their internal/external controllers. Use Case Diagrams are used to depict the functionality of a system or a part of a system. They are widely used to illustrate the functional requirements of the system and its interaction with external agents (actors). It gives us a high level view of what the system or a part of the system does without going into implementation details.

In brief, the purposes of use case diagrams can be said to be as follows –

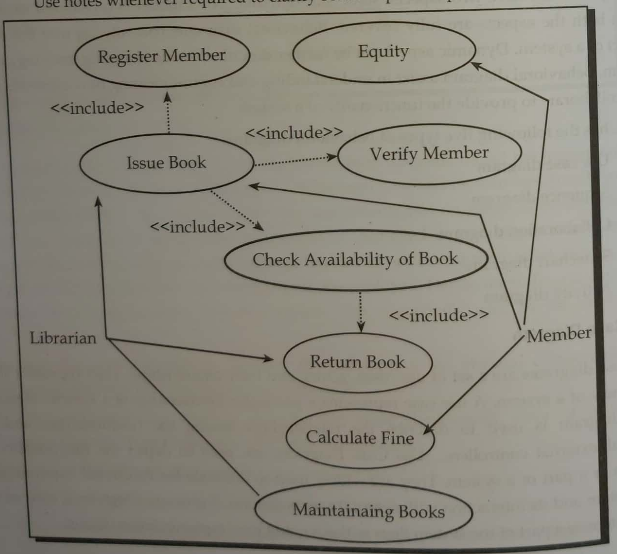
- Used to gather the requirements of a system.
- Used to get an outside view of a system.
- Identify the external and internal factors influencing the system.
- Show the interaction among the requirements are actors.

We should have the following items identified.

- Functionalities to be represented as use case
- Actors
- Relationships among the use cases and actors.

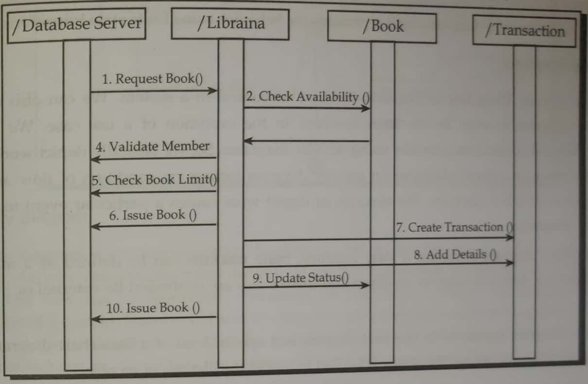
Use case diagrams are drawn to capture the functional requirements of a system. After identifying the above items, we have to use the following guidelines to draw an efficient use case diagram

- The name of a use case is very important. The name should be chosen in such a way so that it can identify the functionalities performed.
- Give a suitable name for actors.
- Show relationships and dependencies clearly in the diagram.
- Do not try to include all types of relationships, as the main purpose of the diagram is to identify the requirements.
- Use notes whenever required to clarify some important points.



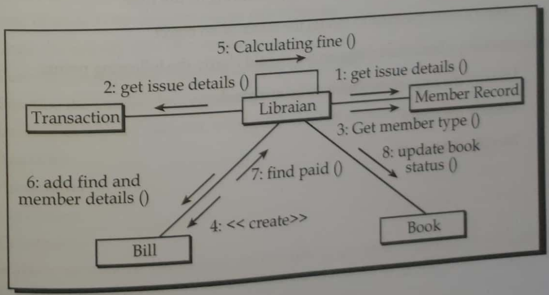
### Sequence Diagram

A sequence diagram simply depicts interaction between objects in a sequential order i.e. the order in which these interactions take place. We can also use the terms event diagrams or event scenarios to refer to a sequence diagram. Sequence diagrams describe how and in what order the objects in a system function. These diagrams are widely used by businessmen and software developers to document and understand requirements for new and existing systems.



### Collaboration Diagram

Collaboration diagram is another form of interaction diagram. It represents the structural organization of a system and the messages sent/received. Structural organization consists of objects and links. The purpose of collaboration diagram is similar to sequence diagram. However, the specific purpose of collaboration diagram is to visualize the organization of objects and their interaction.



## State Diagrams

Any real-time system is expected to be reacted by some kind of internal/external events. These events are responsible for state change of the system. Statechart diagram is used to represent the event driven state change of a system. It basically describes the state change of a class, interface, etc. State chart diagram is used to visualize the reaction of a system by internal/external factors.

It is used to represent the condition of the system or part of the system at finite instances of time. It's a behavioral diagram and it represents the behavior using finite state transitions.

## Activity Diagrams

We use Activity Diagrams to illustrate the flow of control in a system. We can also use an activity diagram to refer to the steps involved in the execution of a use case. We model sequential and concurrent activities using activity diagrams. So, we basically depict workflows visually using an activity diagram.An activity diagram focuses on condition of flow and the sequence in which it happens. We describe or depict what causes a particular event using an activity diagram.

A Statechart diagram describes a state machine. State machine can be defined as a machine which defines different states of an object and these states are controlled by external or internal events.

Activity diagram explained in the next chapter, is a special kind of a Statechart diagram. As Statechart diagram defines the states, it is used to model the lifetime of an object.

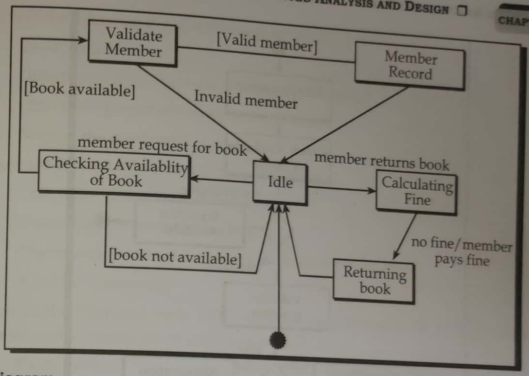
Statechart diagrams are used for forward and reverse engineering of a system. However, the main purpose is to model the reactive system.

Following are the main purposes of using Statechart diagrams –

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

Before drawing a Statechart diagram we should clarify the following points:

- Identify the important objects to be analyzed.
- Identify the states.
- Identify the events.



## Activity Diagram

Activity diagram is another important diagram in UML to describe the dynamic aspects of the system. Activity diagram describes the flow of control in a system. It consists of activities and links. The flow can be sequential, concurrent, or branched. Activities are nothing but the functions of a system. Numbers of activity diagrams are prepared to capture the entire flow in a system. Activity diagrams are used to visualize the flow of controls in a system. This is prepared to have an idea of how the system will work when executed. It is basically a flowchart to represent the flow from one activity to another activity.

Activity is a particular operation of the system. Activity diagrams are not only used for visualizing the dynamic nature of a system, but they are also used to construct the executable system by using forward and reverse engineering techniques. The only missing thing in the activity diagram is the message part.

It does not show any message flow from one activity to another. Activity diagram is sometimes considered as the flowchart. Although the diagrams look like a flowchart, they are not. It shows different flows such as parallel, branched, concurrent, and single.

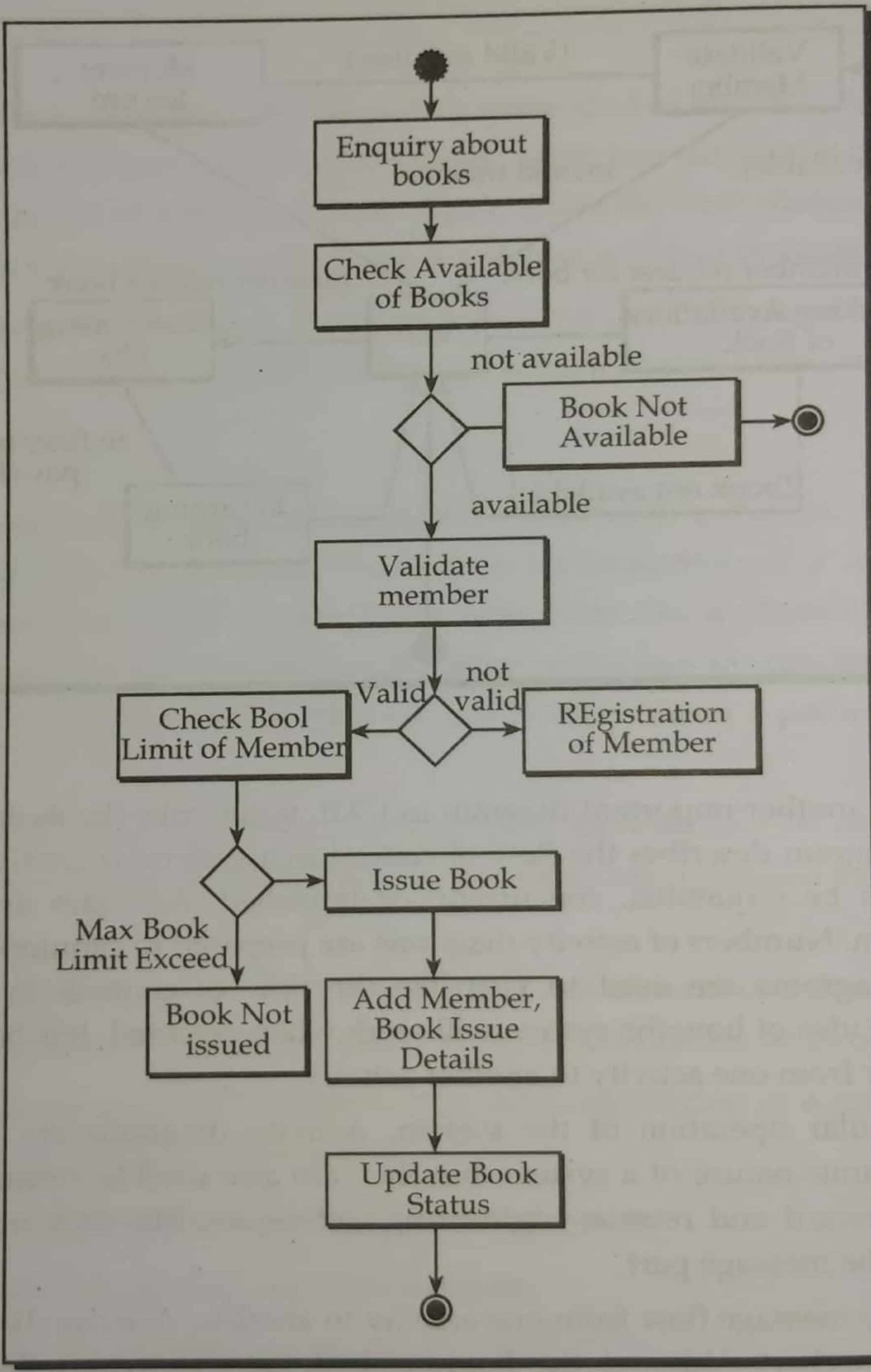
The purpose of an activity diagram can be described as:

- Draw the activity flow of a system.
- Describe the sequence from one activity to another.
- Describe the parallel, branched and concurrent flow of the system.

Before drawing an activity diagram, we should identify the following elements –

- Activities
- Association
- Conditions
- Constraints

Once the above-mentioned parameters are identified, we need to make a mental layout of the entire flow. This mental layout is then transformed into an activity diagram.



## MULTIPLE CHOICE QUESTIONS

## EXERCISE

1. Explain five UML diagrams.
2. Object oriented is a software engineering concept in which concepts are represented as objects. Explain the benefits that come with it.
3. Using diagrams, explain the following concepts as applied in Object Oriented Analysis and Design:
   a. Object  b. Class
   c. Polymorphism  d. Abstraction
4. What do you mean by analysis and design?
5. What are the main underlying concepts of object orientation?
6. Differentiate Persistent & Non-persistent Objects?
7. What is the meaning of encapsulation from the viewpoint of structured system analysis and design? Explain how does encapsulation and abstraction concepts works together in object orientation?
8. What is the use of sequence diagram? Explain with suitable example.
9. Explain object oriented system with reference to class, object, encapsulation, abstraction, message passing, inheritance, interface and polymorphism with suitable example.
10. A web based online store has Buy a product scenario as follow:
11. The customer browses the catalog and adds desired items to the shopping basket. When the customer wishes to pay, the customer describes the shipping and credit card information and confirms the sale. The system checks the authorization on the credit card and confirms the sale both immediately and with a follow-up e-mail. Now construct conceptual model for this scenario.
12. Draw a class diagram for point of sale system with association and multiplicity.
13. What is the use of class diagram? Explain use of class diagram with suitable example.
14. What is framework? How design pattern is useful? Explain any one design pattern in detail with suitable example.
15. In OOAD, there are various types of models, like conceptual, structural, behavioral etc. what is the significance of these many different types of model? Explain with illustrative example.
16. How does the requirement elicitation process happen in object oriented analysis? Explain with reference to system behavior analysis of any exemplary system case.
17. What is external agent in use case diagram? explain
18. What is the use of use case diagram? Explain with suitable example.
19. Prepare a list of dynamic model available in UML with brief detail.
20. Describe the activities involves in object oriented system development.

❑❑❑

**Tribhuvan University**
**Institute of Science and Technology**
**Model Question**

Level: B.Sc. CSIT
Course Title: System Analysis Design
Code No: CSC 252
Semester: V

Full Marks: 60
Pass Marks: 24
Time: 3 hours

*Candidates are required to answer the questions in their own words as far as possible.*

**Group 'A'**

**Long Answer questions: Answer any TWO questions:** [2x10=20]

1. What is Information system analysis and Design? Explain the stages of SDLC
2. Draw a DFD diagram of College library system up to level 2.
3. Explain the process of system implementation and Maintenance

**Group 'B'**

**Short Answer questions: Attempt any EIGHT questions:** [8x5=40]

4. Who is system analyst? List and explain the skills of system analyst.
5. A bank has the following policy on deposit. On deposits of Rs. 50,000 and above for five year or above the interest rate is 15%. On the same deposit for a period less than 5 years it is 12%. On deposit below Rs. 50,000 the interest is 10% regardless of period of deposit. Write the above process using:
   i. Structured English
   ii. A decision table
6. A system costs Rs. 2,00,000 to install and Rs. 10,000 per month as recurring expenses. The benefit per year is 1,50,000. Assuming an interest rate is 15%, what is the payback period of the investment?
7. Briefly explain the steps in feasibility analysis.
8. What are the main principles used in designing Forms and Reports.
9. What is file? Explain any technique for implementing records in file.
7. What is software testing? Explain different types of testing.
10. Briefly Explain the CASE tools.
11. State the activities involved Object-Oriented Development Life Cycle.
12. What is Class Diagram? Draw a class diagram for Payment and Purchase of Customer.

❑❑❑