1.1 Introduction to Object Oriented Modeling and Development

- Object Oriented Modeling is a programming using object which is a combination of data field and methods operate on data. Methods interact with each other using object to design any computer application.
- Object-Oriented Programming is used to increase the flexibility and maintainability of programs. As programs created with an OO language are modular, they are easier to develop and simpler to understand.
- Object oriented modeling and design is a novel technique of thinking about problems with the help of models planned around real-world concepts.
- The basic constituent is the object. The object merges behaviour and data structure in a solitary unit as a whole. Object oriented models are very helpful for the purpose of understanding of problems, communicating among application specialists, modeling enterprises, preparing documentation and designing databases and programs.
- The object modeling method extends from analysis throughout design to implementation. Initially, an analysis model is constructed in order to abstract crucial aspects of the application domain devoid of regard for ultimate implementation.
- This model comprises objects that are found in the application domain together with a explanation of the properties of the objects and their behavior. Subsequently design decisions are made and details are added to the model to explain and optimize the implementation.
- The application-domain objects outline the framework of the design model, however they are implemented in terms of computer-domain objects.
- In conclusion, the design model is executed in a programming language, database or hardware.

1.1.1 What is Object Orientation ?

GQ. What do you mean by object oriented ? Enlist different types of objects and explain with suitable example.

GQ. Explain in brief the characteristics of object in object oriented modeling.

- Let's start our journey of object oriented programming with the basic concepts. Object is nothing but the entity which is distinguishable amongst number of entities with the help of which data can be quantized. Objects are the basic run-time entities in an object-oriented system.
- Objects may represent a human being, a place, a depository account, a table of information or any other thing that the program has to handle. Objects are also represented with the help of user-defined data such as vectors, time and lists.
- Programming problem is mainly analyzed in terms of objects and the nature of communication between them. Table 1.1.1 depicts some distinctive types of objects.

Table 1.1.1 : Types of Objects

• **Physical Objects :** o Electrical components in a circuit-design program o Automobiles in a traffic-flow simulation o States in an economic model o Aircraft in an air traffic control system.	• **Elements of the computer-user environment :** o Windows o Menus o Graphical objects like rectangles, circles, etc. o Input/output devices like keyboard, printer, etc.
• **Data-storage constructs :** o Customized array o Binary tree o Stack o Linked list	• **Human beings :** o Workers o Students o Consumers o Salespeople
• **Collection of data :** o A personal file o A vocabulary o A table of the latitudes and longitudes of world cities	• **User-defined data types :** o Time o Complex numbers o Points on the plane
• **Components in computer games :** o Cars in an auto race o Challengers in adventure games	

- The match between the real-world objects and the programming objects is the contented and satisfied result of uniting data and functions. The resultant objects proffer a revolution in the program design.
- Such kind of close match in between programming constructs and the items being modelled exists in a procedural language. ***Object Oriented Approach*** is the way of arranging software as a collection of distinct objects which integrate both data structure and behavior.
- Following are the four major characteristics of an object :

1. Identity 2. Classification 3. Polymorphism 4. Inheritance

1.1.1.1 Identity

- All objects have a unique identity.
- Each and every object is distinguishable from other objects with the help of its unique identity.

1.1.1.2 Classification

- Classification is the way of grouping the objects with the same behavior (operations) and data structure (attributes) into an entity which is known as a class.
- In simple manner, class illustrates a collection of objects with similar characteristics, common relationships to other objects and common semantics or meanings.

1.1.1.3 Polymorphism

- The word Polymorphism is derived from the Greek word stems poly, which means many and morph, which means shape. For that reason, polymorphic means 'having many shapes'. We can apply the term discretely to variables as well as to messages.
- A polymorphic variable refers to different types of value at different times and a polymorphic message has more than one method linked with it.
- Therefore, polymorphism indicates that the same operation may perhaps perform in a different way on different classes.

1.1.1.4 Inheritance

- Inheritance is basically the process with the help of which objects of one class accomplishes the properties of objects of another class. In object-oriented programming, Inheritance provides the scheme of reusability.
- Object-oriented development is not a programming technique however it is a new way of thinking.
- Object Oriented Development is essentially a new way of thinking about software based on abstractions that exist in the real world as well as in the program.

1.1.2 Basics of Object Oriented Modeling

- A model is an abstraction of something for the purpose of understanding it before building it. It is easier to manipulate than the original entity. An object model captures the static structure of a system, relationships between the objects and the attributes and operations that characterize each class of objects.
- The modern prominence in the literature is mainly on implementation rather than analysis and design. Object-oriented programming languages are very useful in removing restrictions due to the inflexibility of traditional programming languages.
- Object-oriented programming can be implemented using languages such as C, Java, since OOP is not the right of any particular programming language.
- Design faults that found during implementation are expensive to fix than those faults that are originated before entering into the implementation phase. Focusing on implementation issues too early restricts design choices and often leads to an inferior product.
- An object-oriented development gives confidence to software developers to work and think in terms of the application domain throughout most of the software engineering life cycle. Object-oriented development is a conceptual process that is independent of a programming language until the final stages.

- Object-oriented development is fundamentally a new way of thinking and not a programming technique. Its greatest benefits come from helping specifiers, developers and customers express abstract concepts clearly and communicate them to each other. It can be used as a good platform for the purpose of specification, analysis, design, programming, documentation as well as interfacing.
- It can have a range of goals, together with traditional databases, programming languages as well as object-oriented languages.

1.1.3 Object Oriented (OO) Methodology

GQ. What is object oriented methodology? Enlist and explain phases of object oriented methodology.

- Object Oriented Methodology is a methodology for object oriented development and a graphical notation for representation of objects oriented concepts. It is also known as OMT.
- OO methodology has the following phases :

1. System Conception	2. Analysis	3. System design
4. Class design	5. Implementation	

1.1.3.1 System Conception

In the procedure of the software development, initially the system analysts and end-users (customers) of the proposed software system sits together in order to articulate the tentative requirements of the proposed software system.

1.1.3.2 Analysis

- An analysis model is a brief, exact abstraction of what the desired system should do and not how it will be done. It should not contain any implementation facts and particulars.
- The objects in the model should be application domain concepts. The objects should not be the computer implementation concepts. So, the system analyst should first of all understand the problem.
- The **domain model** and the **application model** are the two variants of the analysis model.

1.1.3.3 System Design

- The designer of the system formulates high level decisions about the overall architecture of the system which is to be designed.
- In system design, the target system is organized as various subsystems based on both the analysis structure and the proposed architecture.

1.1.3.4 Class Design

- The designer builds a design model on the basis of the analysis model but containing implementation details.
- The main intention behind the class design is to finalize the data structures and algorithms required to execute each class.

1.1.3.5 Implementation

- The object, classes and relationships developed throughout the class design are finally translated into a particular programming language, database, or hardware implementation.
- During implementation, it is essential to satisfy good software engineering practice so that the system can remain with the traceability, flexibility and extensibility.

1.1.4 The OMT Methodology Employs Three Types of Models

GQ. Write short note on: Models in object oriented methodology.
GQ. Explain in brief :
(a) Class model (b) State model (c) Interaction model

The OMT methodology employs three types of models which are :

1. Class Model 2. State Model and 3. Interaction Model

1.1.4.1 Class Model

- The class model is the most fundamental model. We make use of the class model for representation of the objects and their relationships involved within the system.
- It defines the static organization of the objects in a system and the relationships amongst different objects.
- This model mostly contains class diagram which shows a collection of classes involved in a particular system along with their respective relationships.

1.1.4.2 State Model

It is the model which describes the aspects and characteristics of a system that change over time. This model mostly contains state diagrams.

1.1.4.3 Interaction Model

- This model depicts how the objects involved of a particular system collaborate with each other in order to achieve more significant output.
- At its initial stage, the interaction model starts with use case diagram which is then followed by sequence and activity diagrams. A use case diagram fundamentally deals with the functionality of the system.
- A sequence diagram represents the objects that co-operate with each other along with the time sequence of their interactions. An activity diagram ultimately models the dynamic aspects of the system.

1.1.5 Object Oriented (OO) Themes

GQ. Explain the following terms with respect to object oriented methodology :
(a) Abstraction (b) Encapsulation (c) Sharing.

- In an object oriented technology, there exist numerous themes.
- These themes are not unique to object oriented systems.

- Let's have a look at some of the essential themes in short :

1. Abstraction 2. Encapsulation 3. Sharing

1.1.5.1 Abstraction

- Abstraction consists of focusing on the essential, inherent aspects of an entity and ignoring its accidental properties. It is the basic standard of modeling.
- Hence, abstraction is about looking only at the information that is appropriate and correct at the respective time and hiding overall details.
- In system development, it focuses only on what an object is and does, before deciding how it should be implemented.

1.1.5.2 Encapsulation

- The wrapping up of functions and data into a single entity which is simply nothing but the class is called as an encapsulation. It can also be known as information hiding.
- It consists of separating the external aspects of an object, which are accessible to other objects, from the internal implementation details of the object, which are hidden from other objects. It is not unique to object oriented languages.
- The data encapsulation is one of the most outstanding characteristic of a class. The data is not accessible to the exterior world. It is accessible by only those functions that are enveloped in the class.

1.1.5.3 Sharing

- Inheritance of both data structure and behavior permits an ordinary structure and behavior permits ordinary and widespread structure to be shared amongst numerous similar subclasses without redundancy.
- The sharing of code by using inheritance is one of the core advantages of object oriented languages.
- To end with the basic of object oriented modeling, the difference between procedure oriented language and object oriented language is given in Table 1.1.2.

Table 1.1.2 : Difference between Procedure Oriented Language and Object Oriented Language

Sr. No.	Procedure Oriented Language	Object Oriented Language
1.	Programs are made up of procedures or functions.	Programs are made up of modules.
2.	Procedures cannot be separately coded or tested.	Modules can be separately coded or tested.
3.	The design method in procedure oriented language is top-down. Start with the problem and systematically divide the problem into sub problems.	It follows bottom-up approach. The object-oriented approach lets you create classes and objects that model real world objects.

Sr. No.	Procedure Oriented Language	Object Oriented Language
4.	Software maintenance can be difficult and time consuming.	Software maintenance can be easy and fast.
5.	Emphasis on procedure or function.	Emphasis on data rather than procedure.
6.	Data can be accessed by any function in a program.	Data is hidden cannot be accessed by external entity.
7.	New data and functions cannot be easily added.	New data and functions can be easily added.
8.	Example : C language.	Example : C++ language.

1.2 Object Oriented Modeling History

1.2.1 Object Oriented Design By Booch

- The object oriented design by Booch is a commonly used methodology which helps software developers in designing the software systems by means of object oriented standard.
- It covers the system analysis and the system design phases of an object oriented system.
- The Booch method comprises of following diagrams :
  (1) Object Diagram (2) Class Diagram
  (3) State Transition Diagram (4) Module Diagram
  (5) Process Diagram (6) Interaction Diagram
- Moreover, the Booch methodology recommends a macro development process and a micro development process.

The Macro Development Process

- The key concern of the macro process is technical management of the system.
- The macro development process comprises of the following steps :
  (1) **Conceptualization :** Establishing the core requirements of the system.
  (2) **Analysis and development of the model :** In this stage, the class diagram is used in order to describe the individual roles and responsibilities of the objects involved within the system. The object diagram or alternatively the interaction diagram is used to describe the desired behavior of the system in terms of scenarios.
  (3) **Design the system architecture :** In this stage, the class diagram is used to decide the classes to be used and to define the interrelationships amongst these classes. The object diagram is used to decide about the mechanisms used in order to control the collaborations amongst objects involved within the system. The module diagram is used to plan out where each class and object should be declared. At the end, the process diagram is used for deciding the allocation of processes to respective processors involved within a system scenario.

(4) **Implementation :** Generate a stream of software modules implementations.

(5) **Maintenance :** Localized refinements to the system are allowed in order to remove bugs and to add new requirements.

☞ **The Micro Development Process**

- The micro development process comprise of following steps :
  (1) Identify classes and objects
  (2) Identify class and object semantics
  (3) Identify class and object relationships
  (4) Identify class and object interfaces and implementation

1.2.2 Object Modeling Techniques by Rumbaugh

- The Object Modeling Technique (OMT) is the methodology given by Rumbaugh and his co-workers which describes a method for the analysis, design and implementation of a system with the help of an object-oriented technique.
- The OMT is used to describe the dynamic behavior of objects involved within a system.
- The dynamic model lets us specify detailed state transitions and their explanations within a system.
- At the end, the functional model of the OMT states a process description and consumer-producer relationships.

☞ **OMT encompasses four phases**

(1) System Analysis (The results are objects and dynamic and functional models)

(2) System Design (The results are a structure of the basic architecture of the system along with high-level strategy decisions)

(3) Object Design (The result is a design document containing detailed objects static, dynamic and functional models)

(4) Implementation (The result is reusable, extendible and robust code)

☞ **In OMT, modeling is divided into three phases**

(1) An object Model (object model and data dictionary)

(2) A dynamic model (state diagram and event flow diagram)

(3) A functional model (data flow and constraints)

(1) The object model

- The object model describes the overall structure of objects in a system, i.e., identity of objects, attributes, operations and relations to other objects. Graphically, the object model is represented using an object diagram.
- The object diagram consists of classes interlinked by association lines. Each class from the object diagram signifies a set of individual objects involved within a system scenario. The association lines shows interrelationships amongst the classes.

- Each association line represents a set of links from the objects of one class to the objects of another class.

(2) The dynamic model

- In OMT dynamic model, the stat transition diagram is a network of states and events.
- Each state receives one or m ore events and the next state depends on the current state as well as the events.

(3) The functional model

- In OMT functional model, the data flow diagram (DFD) shows the flow of data between several processes involved in a business.
- Data flow diagram consists of four symbols :
  (1) The process (For e.g. Login into ATM System)
  (2) The data flow (depicts the direction of data element movement, For e.g. PIN code)
  (3) The data store (For e.g. Account, where the customers information is stored)
  (4) An external entity (For e.g. ATM Card Reader)

1.2.3 Object Oriented Analysis by Coad and Yourdon

- Object oriented analysis methodology is the object modeling at the enterprise level. It is the prototype oriented approach to methods.
- Use cases are the basic building blocks in this model that offers traceability throughout the software engineering processes.
- Basically, there are three phases :

(1) Analysis phase	(2) Implementation phase	(3) Testing phase

(1) **Analysis phase :** In analysis phase, the proposed software system scenario is defined in terms of the problem-domain object model, the requirements model, and the analysis model.

(2) **Implementation phase :** The system implementation environment should be identified for the design model which contains elements like database management system, component libraries and graphical user interface tools. The analysis objects are translated into design objects that fit the existing system implementation environment.

(3) **Testing phase :** This phase comprises unit testing, integration testing and system testing.

- Coad and Yourdon also recommend that we cope with complex diagrams by using a CASE tool that allows us to select which parts, or ``layers" of the diagram we wish to see.
- In particular, they introduce the notion of an "attribute layer" of a diagram, which includes the attributes and instance connections shown on the diagram, and a "service layer" which shows the services and message connections (as well as a "class and object layer" which draws in all the classes, showing only their names, and a "subject layer" showing the boundaries of the subjects), and suggest that we use several diagrams that show various layers separately (perhaps, one with the attribute layer but without the service layer, and another with the service layer but without the attribute layer), in order to avoid dealing with a single diagram that would be too cluttered to be understandable.

1.2.4 Object Oriented Software Engineering by Ivar Jacobson

- Object oriented software engineering methodology is also known as Objectory. It is the method of object oriented development with the ultimate goal of development of large, real-time systems. This methodology is based on a use case driven design.
- Following models are involved in the objectory methodology :
  (1) Use case model (2) Domain object model
  (3) Analysis object model (4) Implementation model
  (5) Test model
- The use case model defines the actors and use cases involved in the software system scenario and hence depicts the overall system behavior.
- In domain object model, the objects from the real world scenario are represented.
- The analysis object model depicts how the source code should be written and how the software system implementation should be carried out.
- The implementation of a software system as a hole is represented with the help of the implementation model.
- The test model provides the test plans, detailed test specifications and test reports.

1.3 Modeling as a Technique

- Modeling is a fundamental part of all of the activities which are dedicated for deployment of the quality software system.
- Software professionals ordinarily build models to discuss the overall behavior of the system in brief. Also, the model is useful to control and regulate the performance of the system.
- So, model is prepared in order to get better idea about the proposed system before building it and to manage risk.

1.3.1 Importance of Modeling

(1) A model of the software system is an interpretation of genuineness of the proposed software system.
(2) We can better understand the system that we are developing by means of building the model.
(3) Model depicts the overall behavior of the system.
(4) Decisions made by the developer's team are acknowledged via models.
(5) Model acts as a prototype during the development of the proposed system.
(6) With the help of model, we can imagine the system as it is or as we want it to be.

1.3.2 Purpose of Modeling

GQ. What is modeling ? Describe its purpose.

(1) Communication with End-User (Customer)

System designers come up with the elementary model of the proposed system and discuss the same with end-users (customers) of the proposed system.

(2) Design of a system

A software design specification document represents an archetype of the proposed system. This allows novelty and creativity at little cost.

(3) Decreasing complexity

Essentially, models are designed for better explanation of too complex systems. Models decreases the complexity of the proposed system by making distinction amongst the contents (elements) of the system.

(4) Imagining

End-users can imagine the proposed system along with the help of model in order to get better idea of the proposed system.

Analyzing and testing a physical entity before constructing

(1) Analysis and testing of a physical entity in association with the simulation environment really matters, since a physical entity is low-priced in comparison to the act of building a complete system.

(2) Also, it facilitates early correction of faults and errors involved within the model.

To produce functional work products

(1) A model is considered as a prototype for concrete implementation of the system.

(2) A model of a proposed software system can be used to explain databases, user interfaces, and other related data about the system.

1.4 Abstraction

- Abstraction is the fundamental way in which the definite facets of a particular problem are scrutinized selectively. The main objective of an abstraction is to segregate those facets of a particular problem which are essential for some purpose and destroy those features which are irrelevant.
- Abstraction should be purpose specific, for the reason that the purpose defines what is important, and what is not important. An abstraction signifies the crucial features of an object that differentiate it from all other types of objects.
- All abstractions are partial (incomplete) and erroneous (incorrect). An abstraction mainly concentrates on the exterior view of an object and assists for separation of an object's important behavior from its implementation.
- The main motive behind an abstraction is to limit the universe so we can recognize and realize.

- **Abstraction can be categorized into four types :**
  (1) **Entity abstraction :** An object which denotes a useful model of a problem-domain entity or solution-domain entity.
  (2) **Coincidental abstraction :** An object that bundles a set of operations which are not interrelated with each other.
  (3) **Virtual machine abstraction :** An object that makes clusters of operations which are used by some superior level of control or operations that make use of secondary level set of operations.
  (4) **Action abstraction :** An object that offers a general set of operations, all of which implement the similar type of function.
- All abstraction have static as well as dynamic properties.

1.5 The Three Models

- A system is divided into discrete views by means of three types of models which are class model, state model and interaction model.
- All of these models are mutually supporting each other and are having open and restricted interconnections. Throughout the development of a specific system, all of these three models are involved.
- As far as development of the software system is concerned, it is the responsibility of the system analyst to build an archetype of the proposed system. Software designers design the software design specification of the projected system.
- Developers are the personalities that are involved in implementation of an archetype. Let us have a brief discussion on these three models from object oriented programming point of view.

1.5.1 Class Model

- The organization and detailed structure of objects involved in a system is depicted by means of the model known as class model. The detailed structure of an object comprise of their characteristics, their interactions to other objects in the scenario, and their individual tasks or services.
- The class model offers a framework for the state model and the interaction model. Objects are instances of classes. Objects are simply the building elements of the class model.
- The main motivation behind building a class model is to apply the theories, principles and conceptions from the real world that are significant and vital to the proposed software system.
- The class model must comprise of the things familiar to end users of the proposed system. Basically, the class model is articulated by means of class diagram.

We shall discuss the basics of class concept in subsequent sections of this chapter.

1.5.2 State Model

- Objects from the class model are described with respect to sequencing and timing of operations in case of the state model. The state model describes those aspects of objects concerned with time and the sequencing of operations.

- The state model simply organizes and systemizes the states and the event involved in the proposed software system.
- The state model is dedicated for detailed description of sequence of tasks and services involved within the system operations and do not deals with the contents like details of operations and services, how different services and operations are executed during implementation and execution of the proposed system, etc.
- The state model is basically articulated by means of a state diagram. The state diagram represents the sequences of states and events that are tolerable in a proposed system for individual class of objects. The state diagram talks about the other models too.
- Tasks or jobs of a particular object from the class model are equivalent to the events and activities from the state diagram. References between state diagrams become interactions in the interaction model.

1.5.3 Interaction Model

- The communications amongst objects from the class model is indicated by means of the interaction model. Essentially, the interaction model represents that, how distinct objects from the class model cooperate with each other in order to manage the overall behavior of the proposed system.
- The interaction model comes with three components that are : use cases, sequence diagrams, and activity diagrams.
- Use cases are a superior practice for capturing the functional requirements of a system. Use cases provide a description of how the proposed system will be used and also discuss the responsibilities of the actors involved within a system.
- Sequence diagram usually captures the flow of activities involved within a system. It represents the detailed interaction amongst objects and talks about the time sequence of their interactions. Activity diagrams illustrate the exact work flow of the system.

1.6 Objects

GQ. Define object. Explain component of object diagram in brief.

- An object is an entity and it can be anything that we can imagine which is having its individual identity. An object typically denotes the real world entity having data and attributes (characteristics) along with it.
- All objects are distinguishable and are having identity for their unique identification. Two mangos with the same shape and color are reasonably distinct mangos since a person can eat them separately.
- A person, a book, a college, a vehicle, a laptop, a mobile, an animal are some of the typical objects in real world. Fig. 1.6.1 represents some real world objects.
- An object itself is an instance of a class which outlines the set of characteristics or features that are unitedly used by all of the instances of the respective class.
- We can think of an object as a unified pack of data and function. The data is nothing but the information about an object and functions are known as ***operations***.

- An object have its own qualities and characteristics known as ***attributes***. For instance, a person has a name, a qualification, a color, an age and hobbies; a vehicle has a number, a manufacturer, a color, a shape, an owner and a price. The attribute values hold an object's information.
- Objects also can have a behavior. For instance, a vehicle can move from one location to other.
- Also, objects have their specific self-determining attributes. These attributes plays a vital role in unique identification of an object in a particular scenario.
- Attributes of an object depicts the state of an object. For example, the properties of a bicycle can be manufacturer, color, cost, number of gears, etc., and are the attributes of a bicycle object (Fig. 1.6.2).

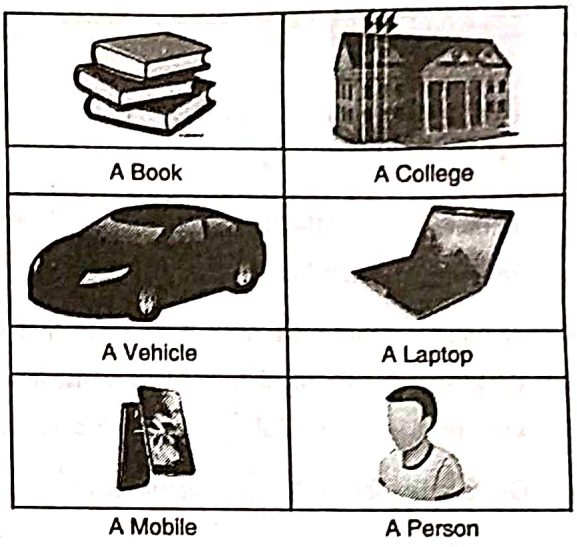

Fig. 1.6.1 : Some real world objects

- Each attribute of an object can be represented in different ways in a programming language. For instance, manufacturer of a bicycle can be signified by a name of a manufacturer, unique commercial tax identification number or a reference to a manufacturer object.
- So, in summary; an object is a physical or visible thing that exhibits definite behavior.

Fig. 1.6.2 : The attributes of a bicycle object

1.6.1 Representing Objects

- After confirmation of the objects in a class, it's time to represent an object in a UML diagram.
- The dedicated UML diagram for presentation of objects involved in an object-oriented system is known as an **Object Diagram**.
- In next section, we will discuss an object diagram in brief. For now, just have a look at Fig. 1.6.3 that represents a notation used for an object.

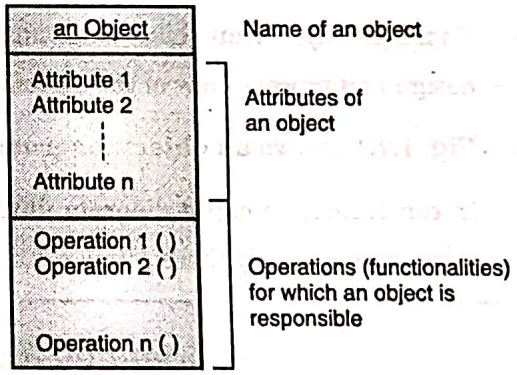

Fig. 1.6.3 : Notation of an Object

- The three portions of the box comprises of the name of an object, its attributes (properties) and its operations (functionalities or behavior).
- Let us study the ATM machine object. The basic operations of ATM machine are : account balance enquiry, money withdrawal, etc.

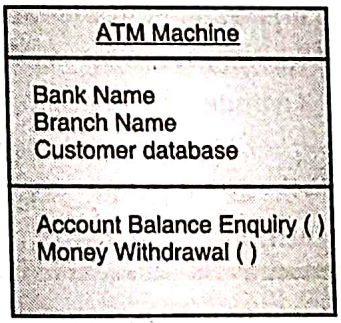

Fig. 1.6.4 : ATM Machine Object

1.7 Object Diagram

- A special type of diagram which depicts a group of objects and their interrelationships with each other at a particular instance of time, is known as an ***object diagram***.
- It is simply a collection of objects in a system at a specific point of time. Since object diagram represents instances instead of classes, it is also called as an **instance diagram**.
- Basically, instances or objects contained in a class are modeled by means of an object diagram. Object diagram can be used to design a static archetype of a system. Object diagram is the best solution to interpret a group of objects, their states and their interrelationships within a scenario.
- Generally, object diagram consists of :

  (1) Objects

  (2) Links between objects
- The components of an object diagram are objects' specifications instead of exact objects. The main motive behind this is to leave required attributes vacant in order to add new attributes in future. Typically, Object diagram shows classical objects involved during execution of a system.
- In conclusion, an object diagram is a graphical representation of the static process view of a system and always, it is not possible to identify and state the objects involved in the system as a whole.
- Consequently, for an object diagram; it is not at all necessary to consider each and every thing about a design and process view of the system.
- **Fig. 1.7.1 shows an object diagram for ATM machine.**
- In conclusion, an object diagram illustrates objects involved in a system along with their relationships and consists of only those components that are crucial for better understanding of a system.

Guidelines for designing an object diagram

- Appropriate name should be given to every object involved in the scenario, for better understanding the purpose of a system.
- Identify function and behavior of each object involved within a system.
- Arrange the objects in a systematic manner in order to reduce the cross lines (that shows relationships) in the diagram.
- For each object, its value, role and state should be mentioned clearly. Expose the attribute values and operations of each object to realize the situation.

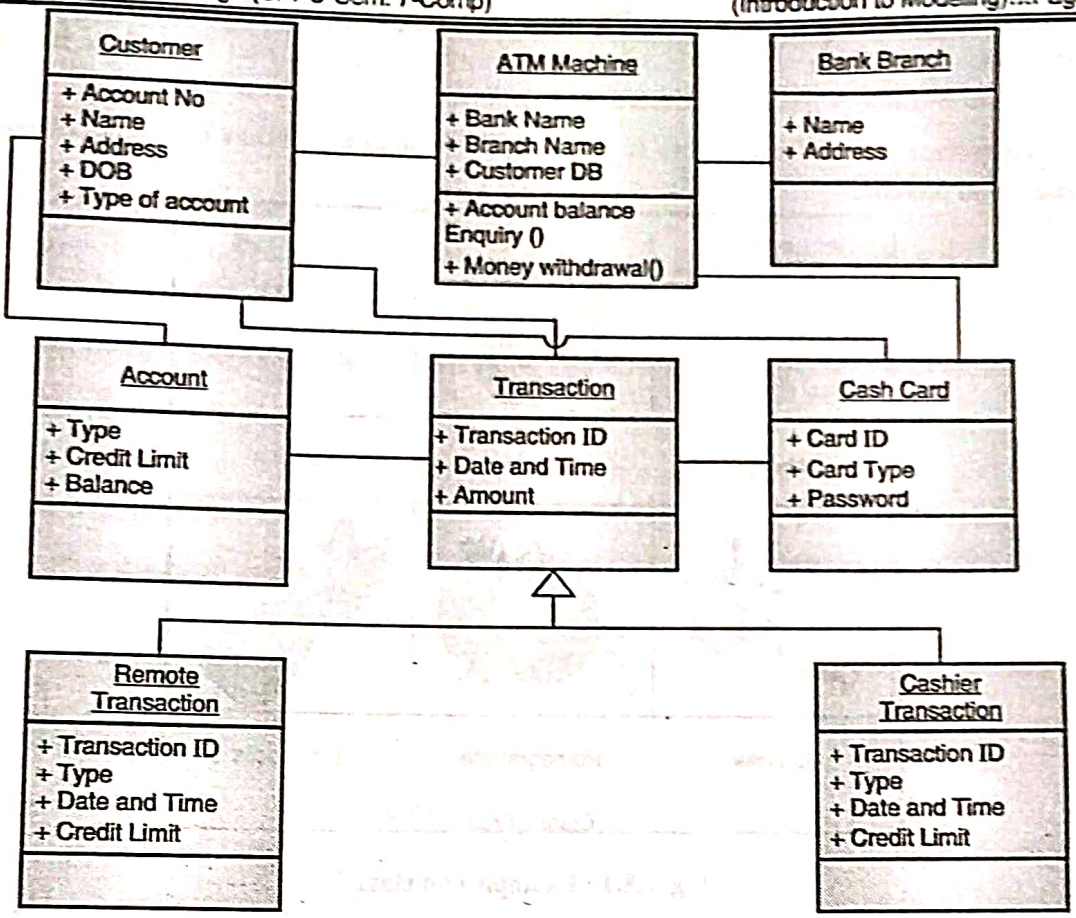

Fig. 1.7.1 : Object diagram showing working of an ATM machine

1.8 Class

GQ. What is class ? Draw and explain the UML notation of a class.

- A class is a collection of objects those are having common characteristics.
- For any object oriented system, classes are the essential building block. A class depicts a collection of objects that have common state and behavior.
- In an object oriented system, a class is considered as a blueprint of the objects. The behavior is explained by means of the probable messages that the class is capable of understanding and accompanied by operations or functions.
- A class can be defined as a detailed explanation of a group of objects that share the equivalent attributes, relationships and operations. A class itself is not a specific object however, it is a complete set of objects.
- An object is characteristically an occurrence of a class. A class depicts a set of objects having similar attributes (properties), meanings, operations (functionality) and types of relationships.
- For example; employee, bicycle, fruit, college are the classes. Every employee working in a particular organization has employee-id, name, department, post, qualification, etc. as attributes that can be used for unique identification of a particular employee in an organization. As discussed in Section 1.5, bicycle

has characteristics like manufacturer, color, cost, number of gears, etc. Each fruit has name, color, shape and flavour. College has its own attributes as name, affiliation, type etc.

- Fig. 1.8.1 represents a class fruit having no. of fruits as the objects that are having attributes like name, color, shape and flavour.

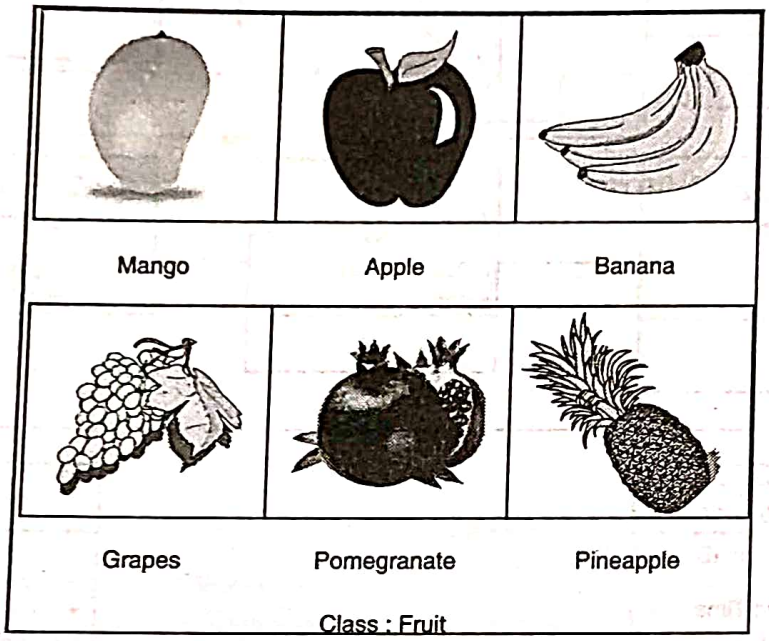

Fig. 1.8.1 : Example of a class

- As shown in the Fig. 1.8.1, all fruits are the objects of a class fruit and all of them are having similar attributes - name, color, shape, flavor and forms of behavior.
- Typically, objects of a class are distinctive in nature and they are uniquely identified by means of variances in their attribute values and definite association to other objects of the same class.
- On the other hand, we can have objects with equal attribute values and relationships inside a particular class. For instance, let us consider a class of two pencils. Imagine, you are holding one pencil in your left hand and one pencil in your right hand.
- Now, in this case; both pencils are having similar attributes like manufacturer, color, shape and size. As far as attributes of pencils are concerned, you can interchange the pencils in your hand.
- Furthermore, if you marked something on a piece of a paper; no one can tell which pencil you had used unless and till people saw you do it. Hence, pencils are same in nature but are not identical. This is only the difference between real life world and imaginary things and it really matters during the implementation and execution of an object oriented system as a software.
- A class of a particular object is a fundamental feature of that object. For each object, it is possible to identify and distinguish its own class. Object oriented programming languages are capable of describing a class of an object at run time.

1.9 Class Diagram

GQ. Draw and explain the UML notation of a class.

- Class diagram is the furthermost common diagram in object oriented modeling and design. Class diagram represents a group of classes, interfaces, their interrelationships and collaborations of classes.
- Class diagram effectively shows how things are organized collectively in the given system scenario. The class diagram is used for modeling the static design archetype of a system which is the extra ordinary version of the object diagram.
- Class diagrams are very useful and therefore suitable for specification, complete execution and documentation of a system.
- Basically, a class diagram can be considered as a set of arcs and vertices in terms of a graph.
- Class diagram mainly consists of :
  - Classes
  - Relationships
    - (1) Association (2) Aggregation
    - (3) Generalization (4) Dependency
  - Collaborations
- A class diagram is normally used to model simple relationships between classes, vocabulary of a system and a logical database schema.
- A decent class diagram comprises of only the things which are crucial for better understanding of a system and is dedicated for designing a static archetype of a system.

Let us start our journey towards a class diagram with notation of a class.

1.9.1 Representing a Class

A well-organized class consists of three blocks: Name of a class, Attributes of a class and Operations of a class. Typically, a class can be represented as follows :

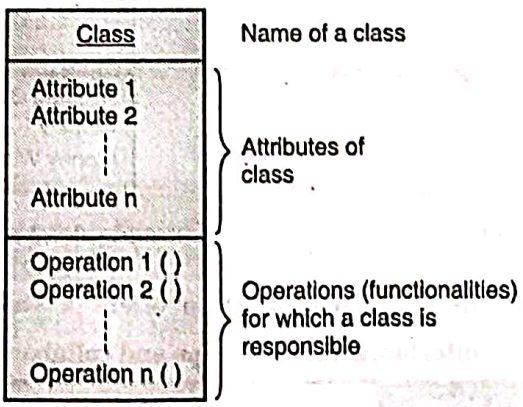

Fig. 1.9.1 : Notation of a class

1.9.1.1 Name of a Class

- Every single class inside a system must be given a name for unique identification of that class in a system.
- A name of a class is a textual string. A name of a class may comprise of numbers, letters and some punctuation marks.
- Practically, name of a class is a small noun or noun phrase drawn from the terminology of a system that we are designing.

1.9.1.2 Attributes

- An attribute is simply a characteristics or property of a class that defines a set of instances and range of values involved in a system.
- The name of an attribute roughly corresponds to the name of a particular field in a programming language.
- Practically, an attribute shows some property of the thing that you are modeling and is jointly used by all of the objects of that class.
- A name of an attribute can be a textual string containing numbers and letters. Furthermore, attribute can be specified in detail along with its data type and default initial value. Fig. 1.9.2 shows bicycle class and its attributes.

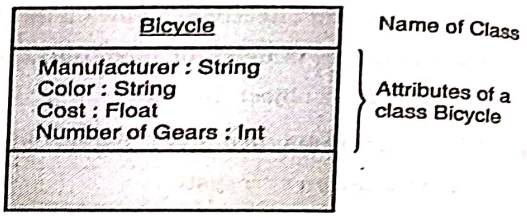

Fig. 1.9.2 : Bicycle class and its attributes

1.9.1.3 Operations

- An operation is simply nothing but a procedure or a function that can be applied to or by objects in a class. The similar operations are jointly shared by all objects of a certain class.
- Operations are features of a class that identify how to invoke a specific behaviour. An operation might have a number of constraints accompanying with it. Constraints supports for getting a detailed idea about how the operations cooperates with the rest of the system.
- A class might have any number of operations within it or no operations at all. The similar operation can be applicable to several different classes in a system.
- The operations are represented in the third block of the UML notation of a class as shown in Fig. 1.9.3.

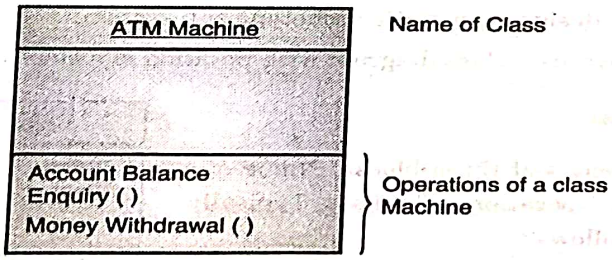

Fig. 1.9.3 : ATM Machine class and its operations

- In conclusion, a class offers a well-structured archetype that represents a collection of classes, interfaces, relationships and collaborations which is comprehensible.

NOTES

Guidelines for designing a class

- For better understanding of the system, show only the important properties of the class.
- Related classes should be shown in the same class diagram.
- Lists of attributes and operations should be arranged in systematic manner by category wise grouping them.
- A well-structured class depicts the balanced archetype in association with attributes and operations.

1.9.2 Relationships

GQ. Enlist and explain different kinds of relationships used in class diagram.
GQ. Explain following types of relationships along with their notations:
(a) Association (b) Aggregation
(c) Generalization (d) Dependency
(e) Multiplicity

Basically, three types of relationships are quite important in object oriented modeling which are : association, aggregation, generalization, dependency and multiplicity. Let us have a look at each of this relationship in this section.

1.9.2.1 Association

- ***Association*** is a structural relationship amongst two classes and is static in nature. It shows fixed relationships between classes involved in a system.
- Association is shown by a solid line between two classes and it can be called as a binary association.
- We can give a name to an association which can be used for better understanding of the nature of the relationship as shown in Fig. 1.9.4.

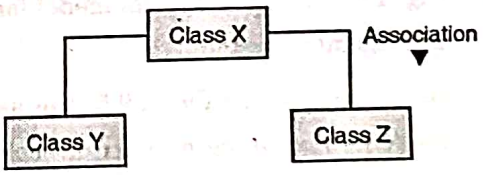

Fig. 1.9.4 : Association between two classes

- We can represent an association relationship without giving a name. In Fig. 1.9.4, Class X and Class Y are linked by association which is nameless association but, link between Class X and Class Z is of the type named association relationship. A small black arrowhead shown in between Class X and Class Z is optional and is dedicated for showing the direction of an association relationship.
- The direction of navigability can be expressed with a stick arrow which is again not compulsory.
- Association is also known as "**---has a ---** "relationship.
- In case of an association relation, the similar objects can form a group but all of the objects in a particular group are not totally dependent on each other. Therefore, association is considered as a weak form connection in object oriented programming.

- For example, consider a bus, number of passengers and a driver. We can say that, the passengers inside the bus and the driver are associated when they are in the bus because all of them occupy some space inside the bus and all of them move in same direction.

Links vs Association

GQ. What is the difference between link and association in class modeling ? Explain with appropriate example.

- A link is defined as a semantic connectivity amongst two objects and it supports message passing mechanism for transfer of data from one object to other object since, objects requires communication between each other in an object oriented program.
- A link is a physical connectivity between two objects while an association is a physical connectivity between two classes. So, we can say that; link in object diagram is replaced by an association in class diagram and both are having same motive as far as their functionality is concerned.
- Moreover, a link is an instance of an association.
- Normally, most of the links are used for representing connectivity between two objects but it is quite possible to relate three or more objects with a single link.
- An association is a detailing of a group of links with similar characteristics and functionalities.
- The links of an association relationship syndicates objects from the similar classes.
- Fig. 1.9.5 shows difference between link in object diagram and association in class diagram. The top of Fig. 1.9.5 shows a class diagram and the bottom shows an object diagram.
- Fig. 1.9.5 depicts a typical model for book issuing in library management system.

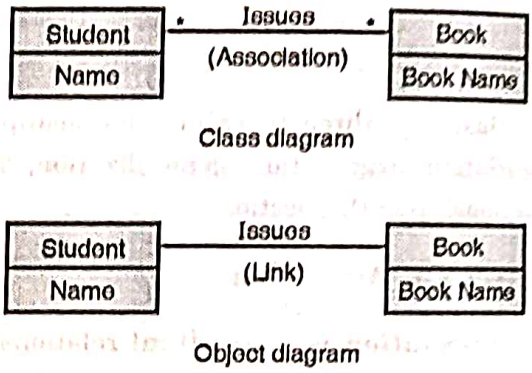

Fig. 1.9.5 : Link vs Association

- As it is shown in Fig. 1.9.5, the association relationship in a class diagram ensures that number of students may issue number of books at the same time (many to many association) while, links in the object diagram shows one to one association. Also, we can show one to many association in this scenario.

1.9.2.2 Aggregation

- ***Aggregation*** is an extra-ordinary version of Association relationship.
- Fundamentally, Aggregation indicates ownership along with a relationship amongst lifelines.
- Aggregation is also called as "--- **owns a** ---"relationship or "**part/whole**" relationship. Fig. 1.9.6 shows an example aggregation.

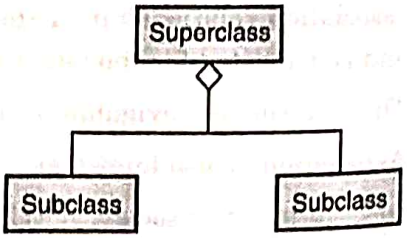

Fig. 1.9.6 : Aggregation between two classes

- ***Composition*** relationship is also same as that of the aggregation relationship. Only the difference is that, aggregation is shown by a hollow diamond as shown in Fig. 1.9.6 while composition is shown by a black diamond as shown in Fig. 1.9.7.

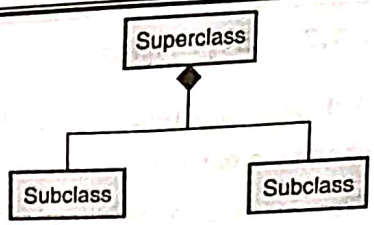

Fig. 1.9.7 : Composition between two classes

- Composition relationship is of the type "part/whole" relationship.

1.9.2.3 Generalization and Inheritance

- A generalization is a kind of relationship between the superclass and the subclass.
- A superclass can be any general thing known as parent and a subclass is a more precise type of that thing called as child. Sometimes, generalization relationship is also known as **"is-a-kind-of"** relationship.
- Generalization classifies classes by means of their similarities and differences, constituting the detailed explanation of objects.
- In case of generalization, objects of the subclass (child class) might be used anywhere in the superclass (parent class). Because, the superclass is having common attributes and operations and the subclasses enhance specific attributes and operations. In short, child (subclass) is substitutable to the parent (superclass).
- Graphically, generalization is shown as a solid directed line with a large open arrowhead that is pointing to the parent class.
- For example, in Fig. 1.9.8, class X is a parent class having two child classes Y and Z. we can say that, class Y and class Z are a part of superclass X.

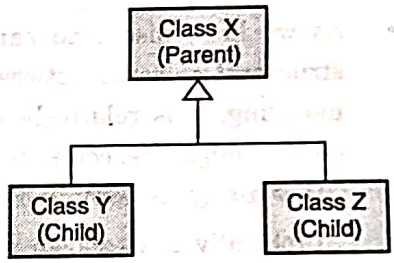

Fig. 1.9.8 : Generalization

- Generalization increases the flexibility of software; i.e., we can add new subclass and that new subclass can automatically inherit attributes and operations of the superclass. Generalization supports for structuring the objects involved in the scenario.
- Also, we can inherit code within the application since, generalization enables reuse of code during implementation and execution of the software system. Adjustment within a lines of code in order to acquire the particular behavior is also supported generalization relationship.
- Inheritance is basically the process with the help of which objects of one class accomplishes the properties of objects of another class. In object-oriented programming, Inheritance provides the scheme of reusability.
- Object-oriented development is not a programming technique however it is a new way of thinking.
- Object Oriented Development is essentially a new way of thinking about software based on abstractions that exist in the real world as well as in the program.
- Inheritance of both data structure and behavior permits an ordinary structure and behavior permits ordinary and widespread structure to be shared amongst numerous similar subclasses without redundancy. The sharing of code by using inheritance is one of the core advantages of object oriented languages.

1.9.2.4 Dependency

- A dependency relationship shows a relationship or connection amongst two or more elements where a change in one element may affect the other element.
- That is, using dependency relationship is a change in description of a particular element may affect the change in other element that makes use of the first element.
- Graphically, dependency relationship is depicted as a directed dashed line that is directed to the element being dependent on. In short, we should make use of dependency when we need to show one element with the help of the other.
- Furthermore, dependencies are used to model relations amongst classifiers where a particular classifier is dependent on the other classifier; but the connection is not of the type generalization or association.

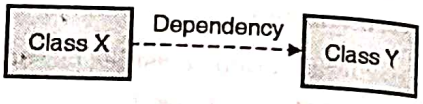

Fig. 1.9.9 : Dependency

- Fig. 1.9.9 shows dependency between two classes.

1.9.2.5 Multiplicity

- As we have discussed earlier, association illustrates a structural relation between objects. In object oriented modeling, it is relatively important to state how many objects might be connected across an association. This **"how many"** is nothing but the multiplicity.
- Graphically it is shown as a solid line along with the expression that estimates to a range of values. We can show a multiplicity of exactly one (1), zero or one (0...1), one or more (1...*) or many (0...*). Moreover, we can state an exact number; for example five (5).
- Fig. 1.9.10 depicts one to many multiplicity between class X and class Y.

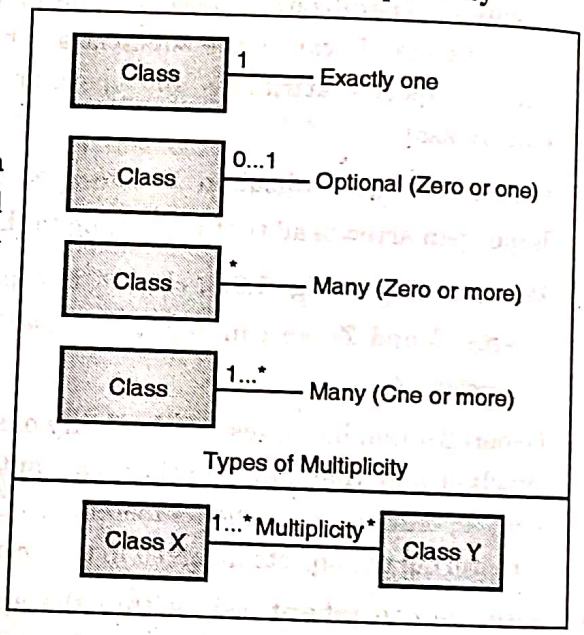

Fig. 1.9.10 : Multiplicity

1.9.3 Navigation in Class Model

- Along with the representation of an archetype of a particular software system application, a class model also refers to the behavior of a software system and hence articulates software system behavior.
- The mechanism of navigation is quite helpful in the process of identification and documentation of hidden faults and defects involved within the software system scenario.
- After successful detection of defects from software system archetype, software developers can make suitable alterations in further versions of a respective software system application.

1.9.4 Class Diagram for ATM System

GQ. Draw and explain a class diagram for ATM system scenario.

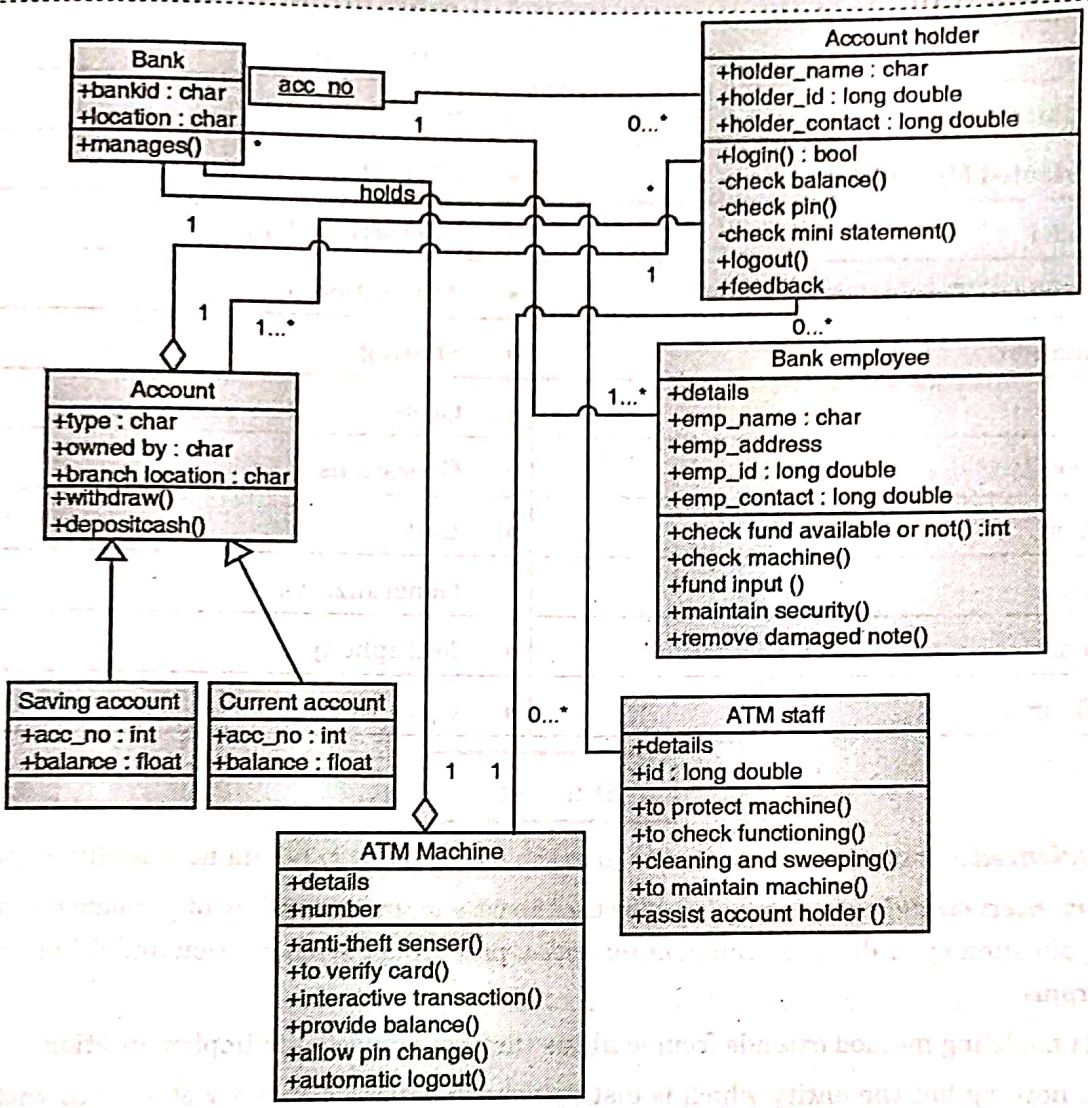

Fig. 1.9.11 : Class Diagram for ATM System

Guidelines for designing a class diagram

- Appropriate name should be given to every class involved in the scenario, for better understanding the purpose of a system.
- Identify function and behavior of each class involved within a system.
- Arrange the classes in a systematic manner in order to reduce the cross lines (that shows relationships) in the diagram.
- For each class, its value, role and state should be mentioned clearly. Expose the attribute values and operations of each class to realize the situation.
- Color and notes should be used in order to give more attention to vital features of a class diagram.
- Don't try to show too many types of relationships amongst the classes involved in the system.

Key Concepts

• Object Oriented (OO)	• Object Oriented Modeling
• Identity	• Classification
• Polymorphism	• Inheritance
• Object Oriented Methodology	• Class Model
• State Model	• Interaction Model
• Object Oriented Themes	• Abstraction
• Encapsulation	• Sharing
• Object	• Class
• Attributes	• Operations
• Association	• Link
• Aggregation	• Generalization
• Dependency	• Multiplicity
• Object Diagram	• Class Diagram

Summary

- ***Object-Oriented Programming*** is used to increase the flexibility and maintainability of programs.
- ***Object oriented models*** are very helpful for the purpose of understanding of problems, communicating among application specialists, modeling enterprises, preparing documentation and designing databases and programs.
- The object modeling method extends from analysis throughout design to implementation.
- ***Object*** is nothing but the entity which is distinguishable amongst number of entities with the help of which data can be quantized.
- Objects are the basic run-time entities in an object-oriented system.
- Objects may represent a human being, a place, a depository account, a table of information or any other thing that the program has to handle.
- ***Object Oriented Approach*** is the way of arranging software as a collection of distinct objects which integrate both data structure and behavior.
- Following are the four major characteristics of an object :
  - Identity
  - Classification
  - Polymorphism and
  - Inheritance
- Each and every object is distinguishable from other objects with the help of its unique ***identity***.

- ***Classification*** is the way of grouping the objects with the same behavior (operations) and data structure (attributes) into an entity which is known as a class.
- The word ***Polymorphism*** is derived from the Greek word stems poly, which means many, and morph, which means shape. For that reason, polymorphic means 'having many shapes'.
- ***Inheritance*** is basically the process with the help of which objects of one class accomplishes the properties of objects of another class.
- A ***model*** is an abstraction of something for the purpose of understanding it before building it. It is easier to manipulate than the original entity.
- Object-oriented development is a conceptual process that is independent of a programming language until the final stages.
- ***Object Oriented Methodology*** is a methodology for object oriented development and a graphical notation for representation of objects oriented concepts.
- Phases of Object Oriented Methodology are: System Conception, Analysis, System Design, Class Design & Implementation.
- The OMT methodology employs three types of models which are :
  - Class Model
  - State Model
  - Interaction Model
- We make use of the ***class model*** for representation of the objects and their relationships involved within the system.
- ***State model*** describes the aspects and characteristics of a system that change over time.
- ***Interaction model*** depicts how the objects involved of a particular system collaborate with each other in order to achieve more significant output.
- ***Abstraction*** consists of focusing on the essential, inherent aspects of an entity and ignoring its accidental properties. It is the basic standard of modeling.
- The wrapping up of functions and data into a single entity which is simply nothing but the class is called as an ***encapsulation***.
- Inheritance of both data structure and behavior permits an ordinary structure and behavior permits ordinary and widespread structure to be shared amongst numerous similar subclasses without redundancy.
- ***Abstraction*** is the fundamental way in which the definite facets of a particular problem are scrutinized selectively.
- An ***object*** is an entity and it can be anything that we can imagine which is having its individual identity.
- A special type of diagram which depicts a group of objects and their interrelationships with each other at a particular instance of time, is known as an ***object diagram***.
- Since ***object diagram*** represents instances instead of classes, it is also called as an ***instance diagram***.

- A ***class*** is a collection of objects those are having common characteristics.
- ***Class diagram*** represents a group of classes, interfaces, their interrelationships and collaborations of classes.
- ***Class diagram*** effectively shows how things are organized collectively in the given system scenario.
- An attribute is simply a characteristics or property of a class that defines a set of instances and range of values involved in a system.
- An operation is a procedure or a function that can be applied to or by objects in a class.
- ***Association*** is a structural relationship amongst two classes and is static in nature. It shows fixed relationships between classes involved in a system.
- ***Association*** is also known as "**--- has a ---** "relationship.
- A ***link*** is defined as a semantic connectivity amongst two objects and it supports message passing mechanism for transfer of data from one object to other object since, objects requires communication between each other in an object oriented program.
- ***Aggregation*** is an extra-ordinary version of Association relationship.
- ***Aggregation*** is also called as "**--- owns a ---**"relationship or "**part/whole**" relationship.
- ***Generalization*** classifies classes by means of their similarities and differences, constituting the detailed explanation of objects and it is also known as "**is-a-kind-of**" relationship.
- A ***dependency*** relationship shows a relationship or connection amongst two or more elements where a change in one element may affect the other element.
- In object oriented modeling, it is relatively important to state how many objects might be connected across an association. This "**how many**" is nothing but the ***multiplicity***.
- ***Control objects*** are the objects which act as a controller in overall collection of objects. They are also known as coordinator objects.
- ***Entity objects*** envelopes information and offers an access to the stored information.
- The objects that are dedicated for communication and coordination with external environment or outside world are known as ***Boundary objects***.
- The detailed application logic of a system is comprised by ***Application logic object***.

Chapter Ends...

□□□