UNIT-6

Introduction to Object-Oriented Development

⊛. Basic Characteristics of Object Oriented System:

→ benefits of object oriented development over structural [Imp]

i) **Class:** Class is a collection of similar objects. Objects with same data structure and behaviour are grouped into a class. 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.

ii) **Abstraction:** Classes are built on the basis of abstraction, where a set of similar objects are observed and their common characteristics are listed. The abstraction of an object varies according to its application. Data abstraction refers to providing only needed information to the outside world and hiding implementation details.

iii) **Inheritance:** Inheritance is the process by which the objects of one class acquire the properties of objects of another class. It helps to share common characteristics with the class from which it is derived.

iv) **Polymorphism:** Polymorphism is the ability to make more than one form. Any operation may show different behaviours in different instances.

v) **Reusability:** The classes once defined can easily be used by other applications. Once a class is defined it can be used again and again by defining object of the class type.

vi) **Data hiding:** The insulation of the data from direct access by the program is called data hiding. It seperates the internal functioning of the object from external functioning.

⊛. Object-Oriented System Analysis and Design (OOSAD):

Object-oriented analysis is a method that examines requirements from the perspective of the classes and objects found in the vocabulary of the problem domain.

Object-oriented design is the process for demonstrating logical and physical as well as static and dynamic models of the system under design.

✓ As compared to conventional system object-oriented (OO) modeling provides many benefits such as: inheritance, reusuability, data hiding, polymorphism etc.

⊛. Unified Modeling Language: [Imp]

The Unified Modeling Language (UML) is a general-purpose modeling language in the field of software engineering, which is designed to provide a standard way to visualize the design of a system. It supports high level development concepts such as frameworks, patterns and collaborations. UML includes a collection of elements such as;

→ Programming language Statements.
→ Actors: specify a role played by a user
→ Logical and Reusable software components. etc.

UML diagrams can be divided into two categories. The first type includes four diagram types representing structural information. The second includes the remaining four representing general types of behaviour.

1> Structural Diagrams:

Structural Diagrams show the things in modeled system. Structural diagrams represent static aspect of the system. Structural diagrams are used in the documenting architecture of software systems and are involved in system being modeled. The four structural diagrams are as follows:

1> **Class diagram:** Class diagrams consists of classes, interfaces, [Imp]// associations and collaboration. Class diagrams basically represent object-oriented view of the system, which is static in nature. It is the building block of all object oriented software systems. Class diagrams help us to identify relationship between different classes or objects.

In UML, a class is represented by a rectangle with three compartments separated by horizontal lines. The class name appears in the top compartment, the list of attributes in the middle compartment, and the list of operations in the bottom compartment of the rectangle.

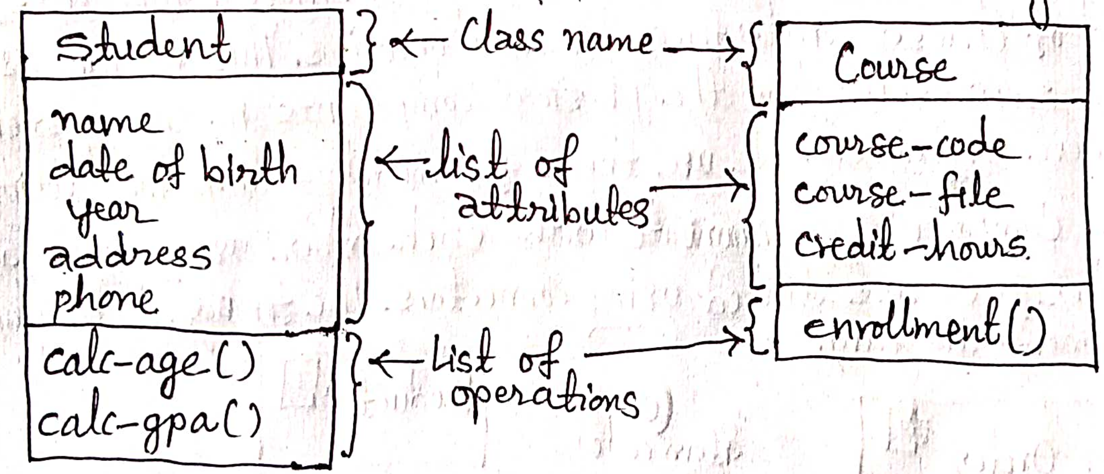
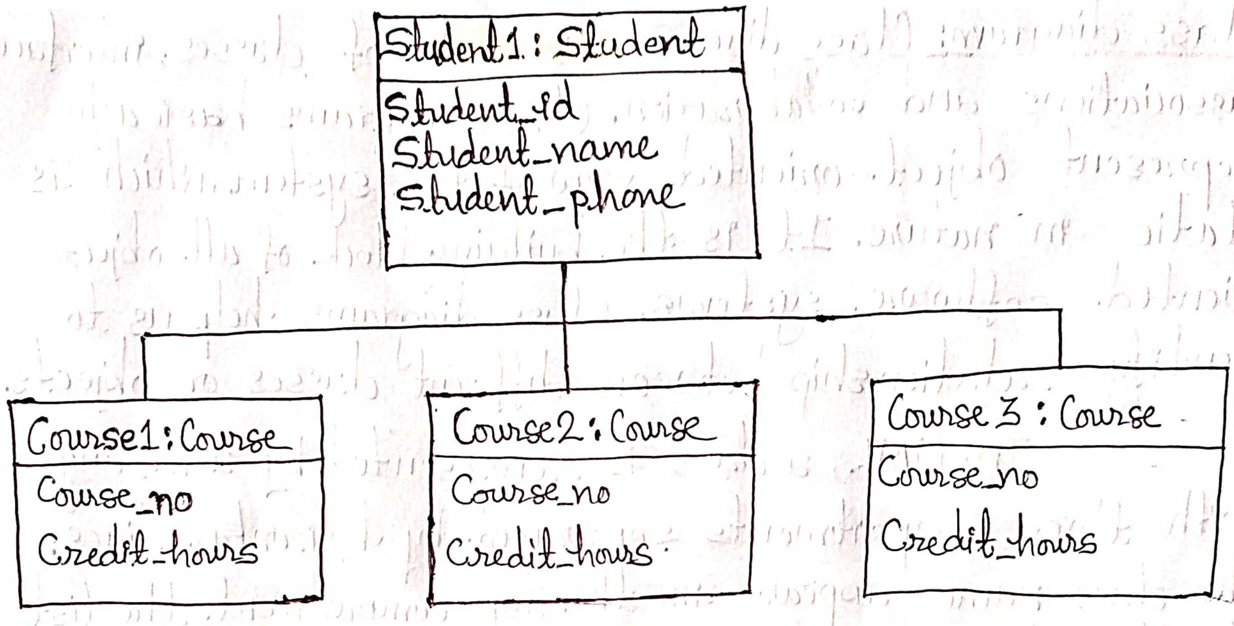


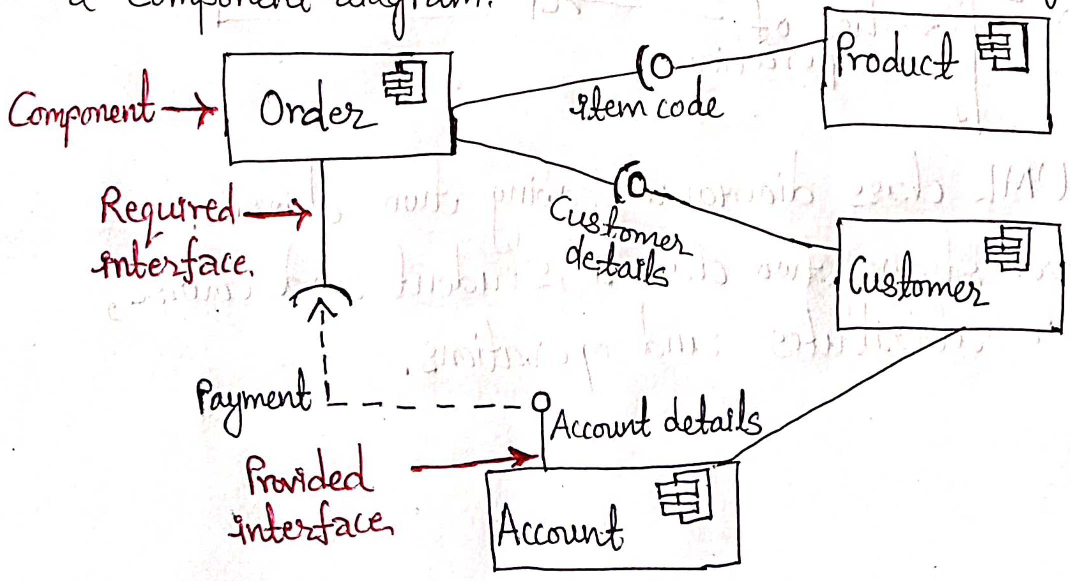
Fig: UML class diagram showing two classes.

The fig. above shows two classes, Student and Course, along with their attributes and operations.

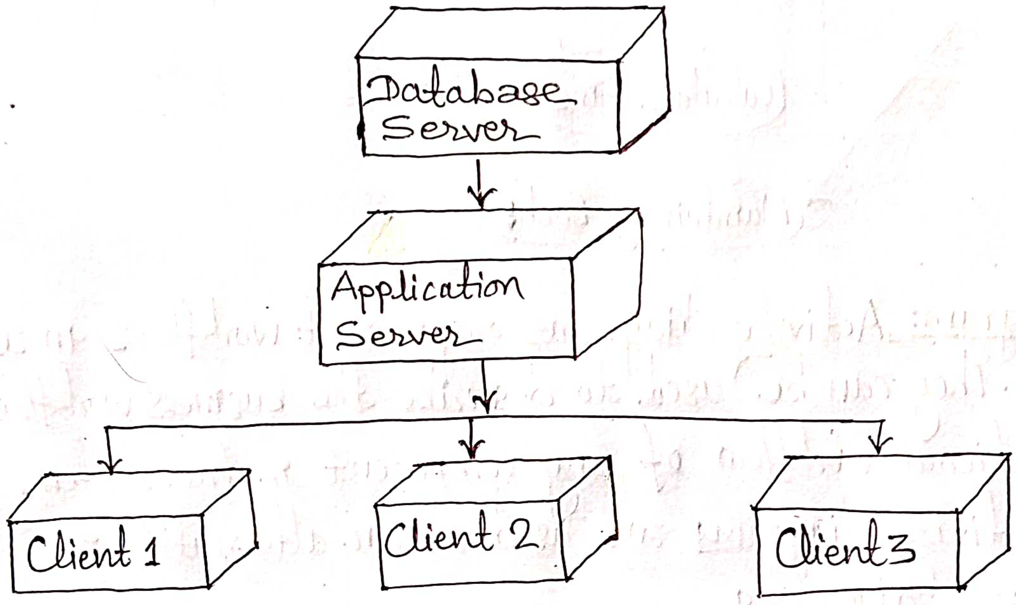
ii) Object diagram: Object diagrams consist of specific instances [Imp] of classes and relationships between them at a point of time. An object diagram is similar to object diagram except that it shows the instances of classes in the system. Object diagrams are derived from class diagrams so object diagrams are dependent on class diagrams.



iii) Component diagram: Component diagrams represent a set of components and their relationships. These components consists of classes, interfaces or collaborations. This diagram is used to represent how the physical components in the system have been organized. We use them for modeling implementation details. Components communicate with each other using interfaces. The interfaces are linked using connectors. The figure below shows a component diagram.

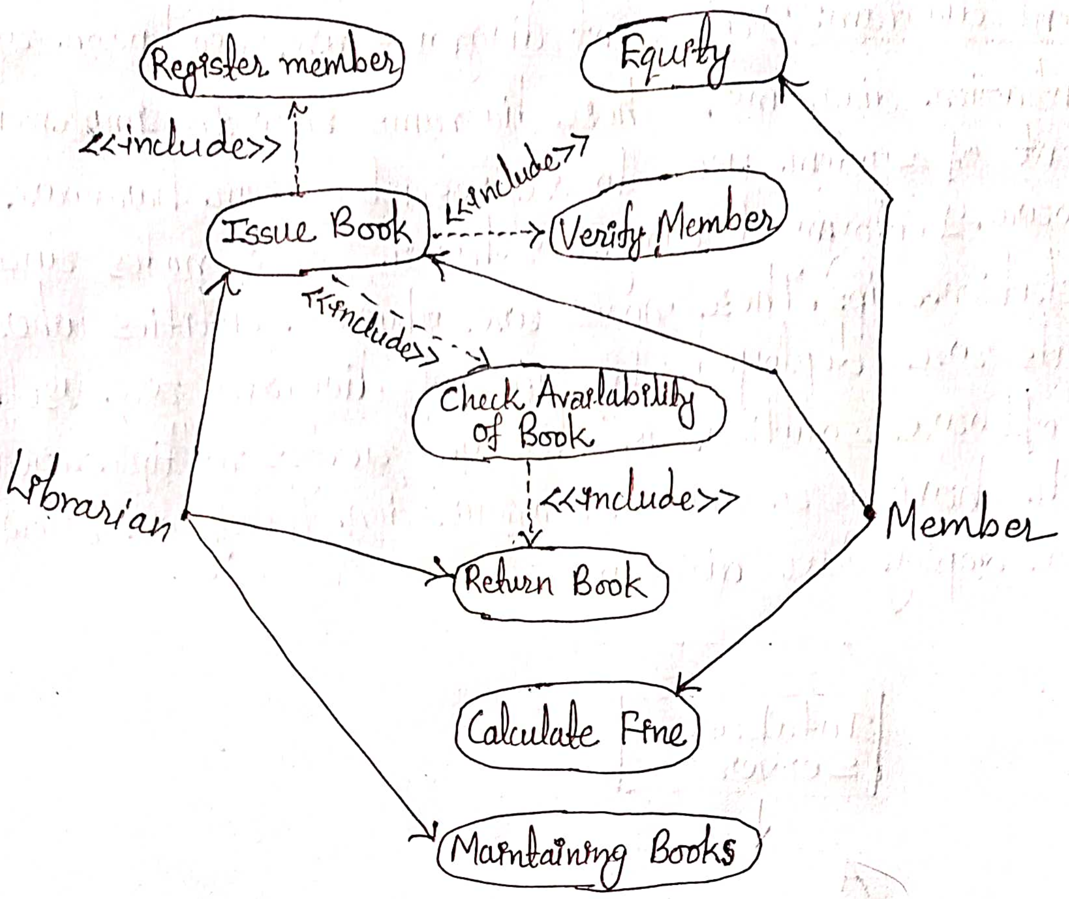


iv) **Deployment diagram:** Deployment diagrams are also known as implementation diagrams. These diagrams show the implementation environment of system used to represent system hardware and it's software. Deployment diagrams has a set of nodes and their relationships. These nodes are physical entities where the components are deployed. Deployment diagrams are useful when software solution is deployed across multiple machines with each having a unique configuration. The figure below shows the deployment diagram.

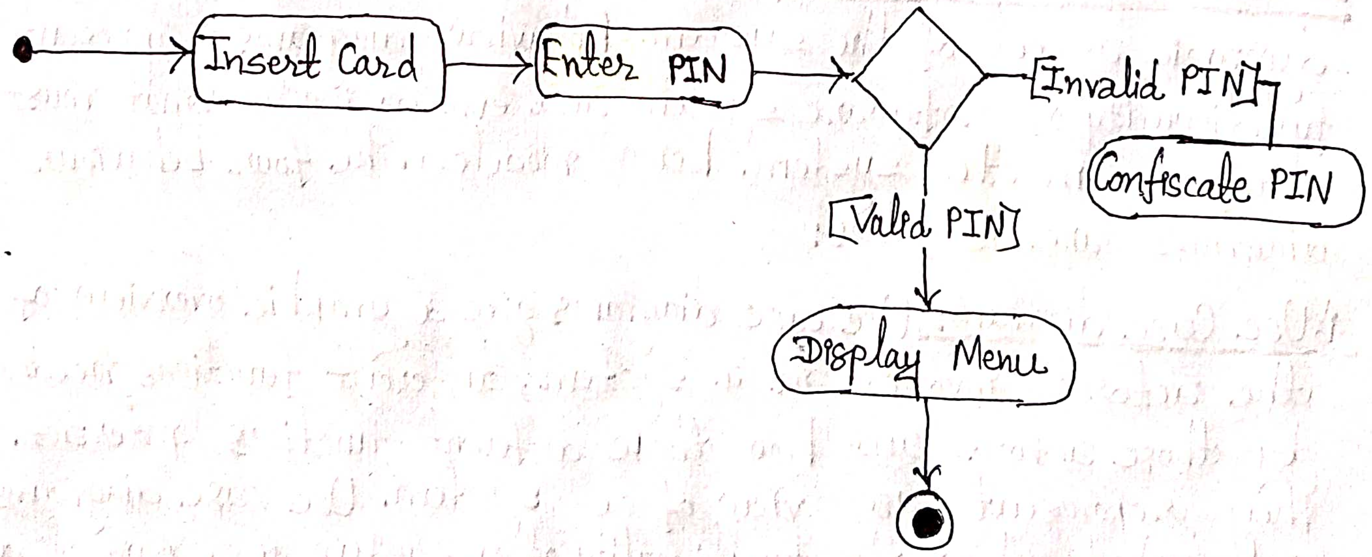


2) **Behaviour Diagrams:** Behaviour diagrams capture the dynamic aspect of the system. Behaviour diagrams represent functionality of software system and emphasize on what must happen in the system being modeled. The four behaviour diagrams are as follows:

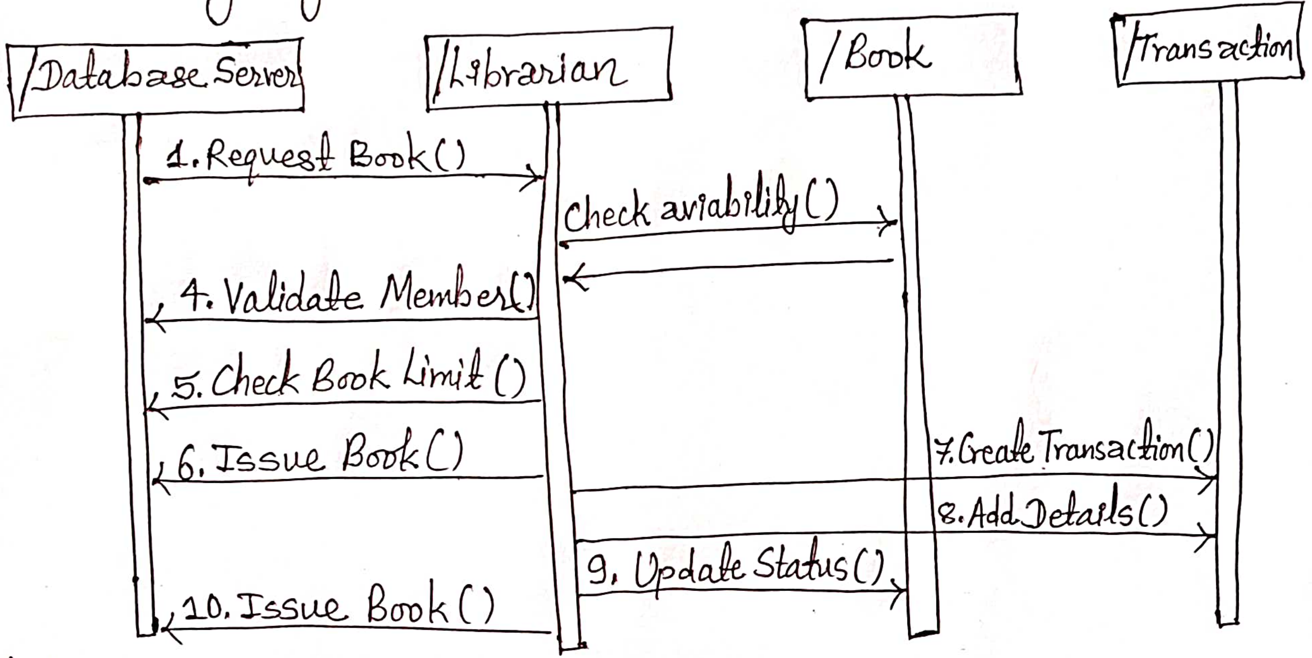
i) **Use Case diagram:** Use case diagrams give a graphic overview of the actors involved in a system, different functions needed by those actors and how these different functions interact. They represent case view of a system. Use case diagrams are used to show the functionality of a system or a part of a system. It gives us high level of view of what the system or a part of system does without going into implementation details.



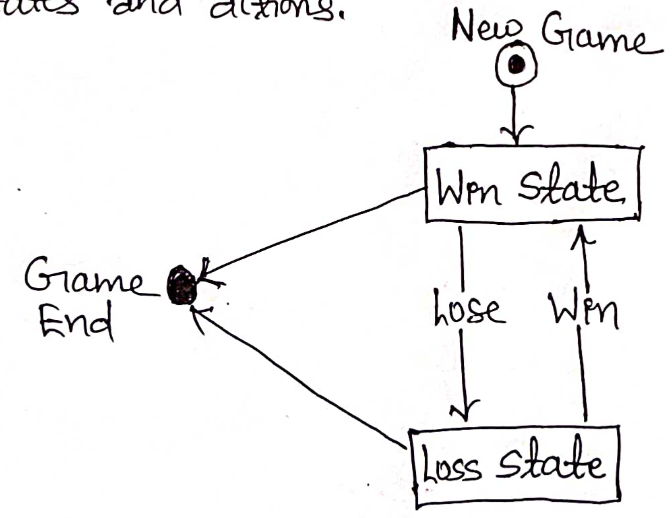
ii) <u>Activity diagram:</u> Activity diagrams represent workflows in a graphical way. They can be used to describe the business workflow or the operational workflow of any component in the system. Sometimes activity diagrams are used as an alternative to state machine diagrams.



iii) **Sequence diagram:** Sequence diagram shows how objects interact with each other and the order those interactions occur. They show the interactions for a particular scenario. The processes are represented vertically and interactions are shown as arrows. These diagrams are widely used by businessman and software developers to document and understand requirements for new and existing systems.



iv) **State diagram:** State diagrams are similar to activity diagrams, although notations and usage change a bit. They are sometimes known as state machine diagrams or state chart diagrams as well. These diagrams are very useful to describe the behaviour of objects that act differently according to the state they are in at the moment. The state diagram below shows the basic states and actions.



❤️

If my notes really helped you,then you can support me on esewa for my hardwork.

Esewa ID: 9806470952