Topic 5: Object-oriented Design

❖Analysis to Design

- Object-oriented design is the process of planning a system of interacting objects for the purpose of solving a software problem. It is one approach to software design.

Input

- The output of object -oriented analysis is provided as input to object -oriented design.
- Analysis and design may occur in parallel with incremental and iterative process.
- Conceptual model, use case, system sequence diagram, UI documentation and relational data model are the input artifacts.

Design

With the help of the input artifacts, we can conclude the following:

- Define object and class diagram from conceptual model.
- Attributes are identified.
- A description of solution to a common problem (design patterns) are used.
- Application framework is defined.
- The persistent objects or data are identified.
- The remote objects are identified and defined.

Design

With the help of the input artifacts, we can conclude the following:

- ➢Define object and class diagram from conceptual model.
- ➢Attributes are identified.
- ➢A description of solution to a common problem (design patterns) are used.
- ➢Application framework is defined.
- ➢The persistent objects or data are identified.
- ➢The remote objects are identified and defined.

Output

- Sequence diagram and design class diagram are the typical output artifacts of object-oriented design.

❖ Describing and Elaborating Use cases

Class Responsibility Collaboration (CRC) Cards

- It is a text-oriented modelling techniques or tools used in design of object-oriented software.
- It is a paper index cards in which the responsibilities and collaborators of classes are written.
- Each card represents a single class.

Responsibility

- A Responsibility is anything that the class knows or does.

Collaborators

- A Collaborator is another class that is used to get information for or perform actions for the class at hand.

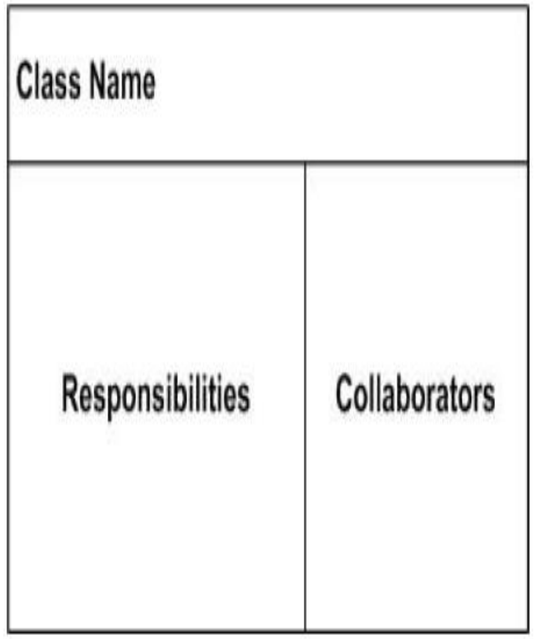


Fig: CRC card Format

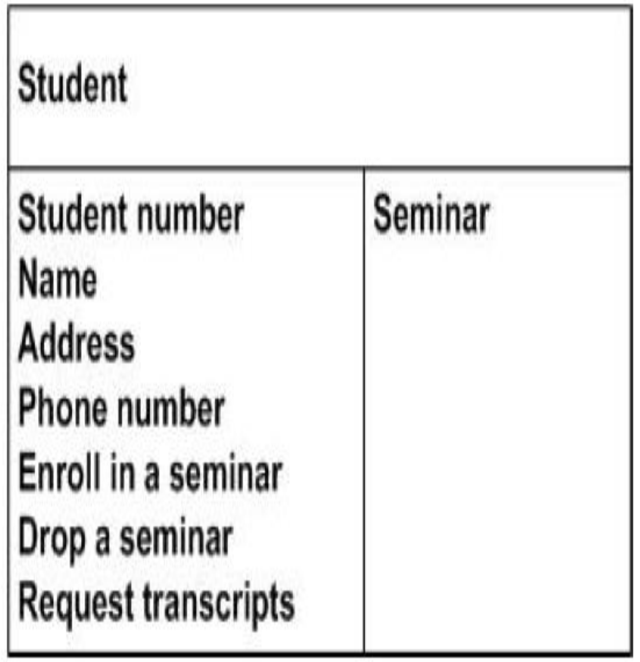


Fig: CRC card Example

Realization of Use case

• It represents the implementation of use case in terms of collaborating objects.

• It may include textual document, class diagrams of participating classes and interaction diagrams.

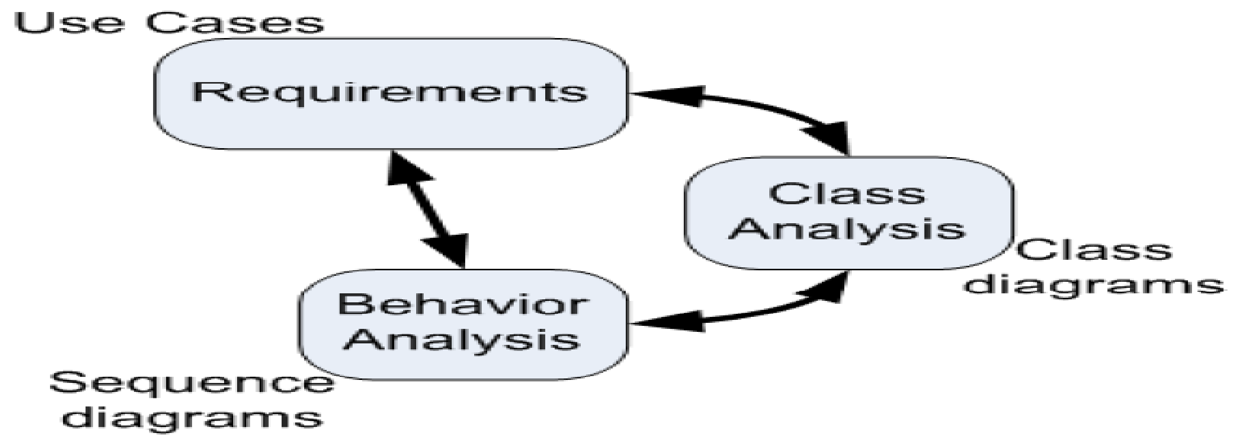
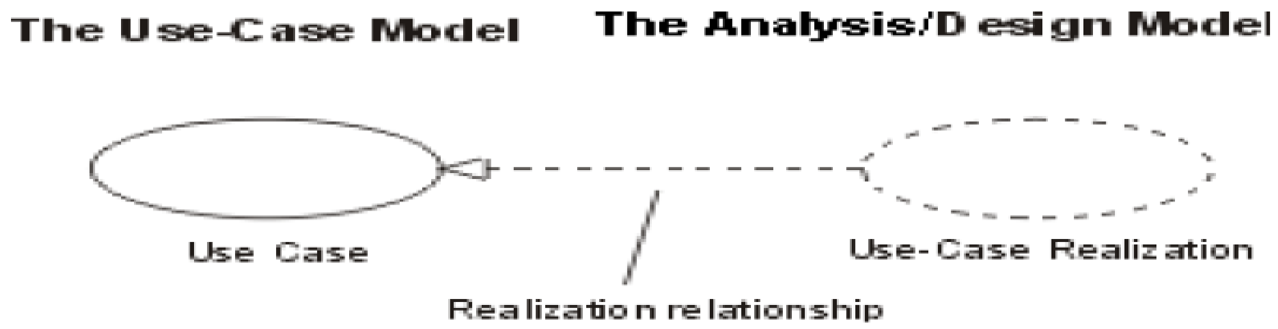


Fig: Use case realization process



UML Interaction Diagram

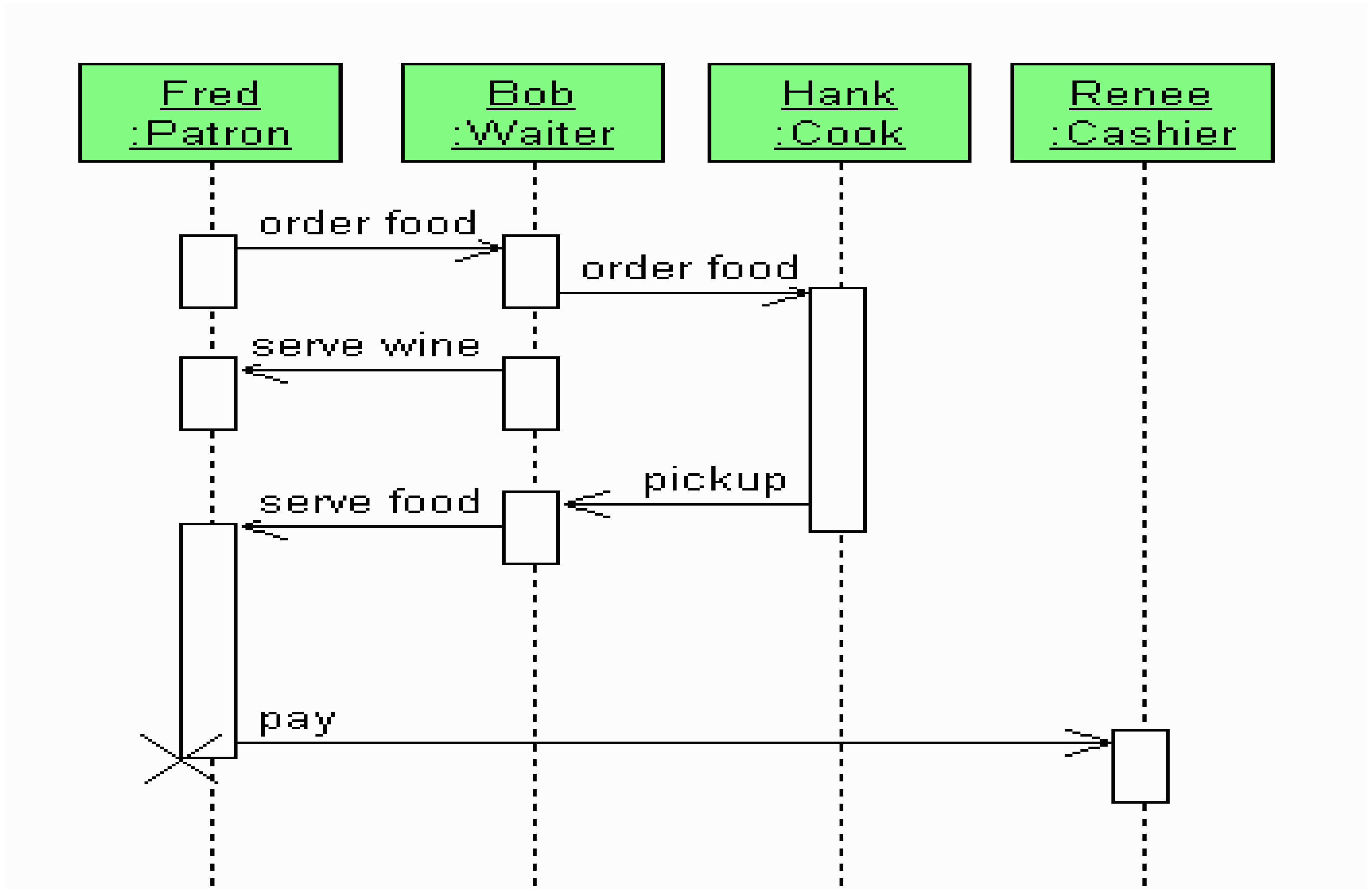
- Interaction diagram shows how objects interact via messages.
- It is used for dynamic object modelling.
- It is of two types. They are as follows:

1. Sequence diagram
2. Communication or collaboration diagram

Sequence Diagram

- A sequence diagram shows the interaction among objects as a two-dimensional chart, in a fence format, in which each new object is added to the right.
- The objects participating are shown at the top of the chart as boxes attached to a vertical dashed line.

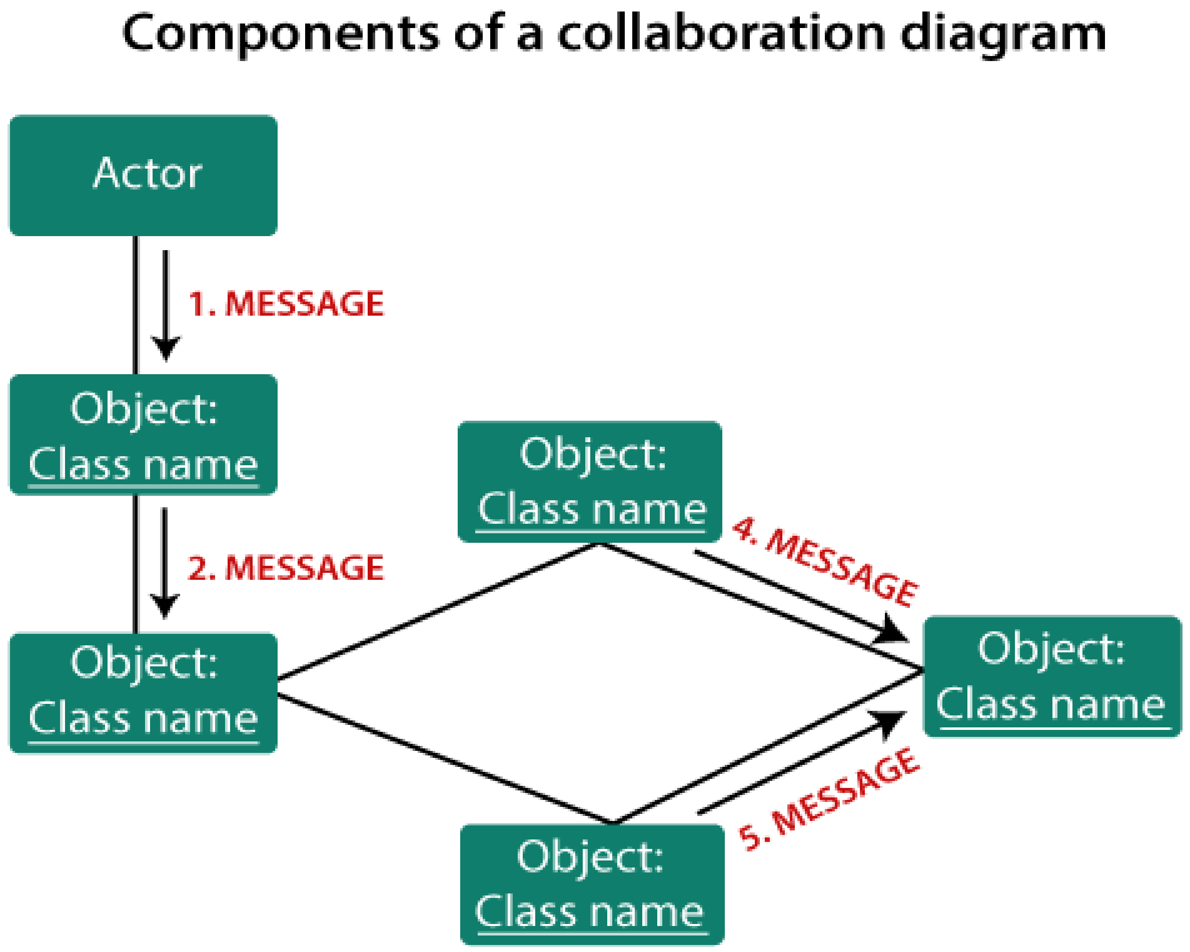
- The name of object with a semicolon separating it from the name of the class is written inside the box.
- The name of object and the class is underlined.
- The vertical dashed line indicates the lifeline of objects.
- The rectangle drawn on the lifeline is called activation symbol, which indicates that the object is active as long the rectangle exists.
- Each message is indicated as an arrow between the lifeline of two objects.
- The messages are shown in chronological order.
- Each message is labeled with the message name.
- **It shows the time sequence of messages.**
- **It has large set of detailed notation options.**
- **It is forced to extend to right when new objects are added, which consumes horizontal space, making it difficult to view by the user.**


Fred :Patron
Bob :Waiter
Hank :Cook
Renee :Cashier
order food
order food
serve wine
pickup
serve food
pay

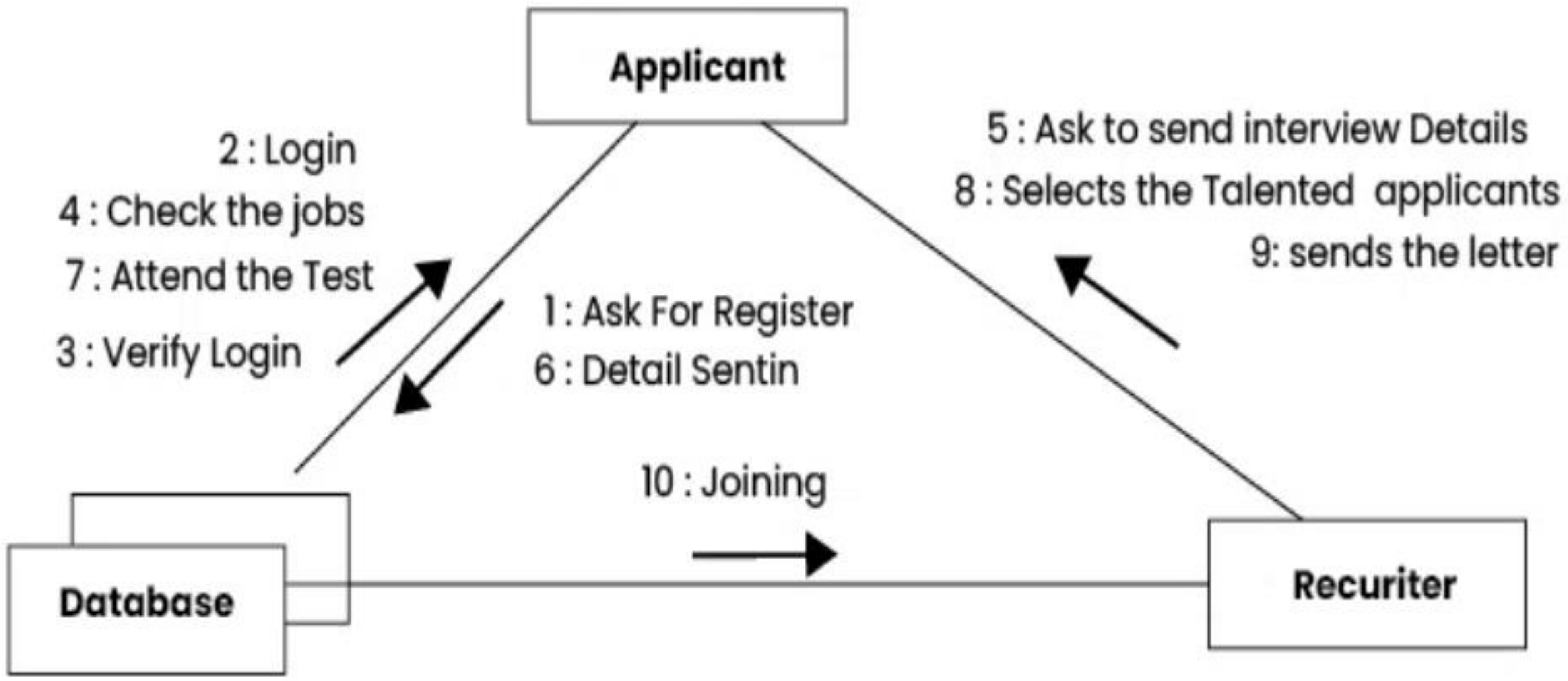
❖ Collaboration Diagram

- Is a behavioral diagram which is also referred to as a communication diagram, It illustrates how objects or components interact with each other to achieve specific tasks or scenarios within a system.
- Collaboration diagram shows both structural and behavioral aspects.
- Structural aspect includes objects and link between them.
- It illustrates the object interactions in a graph format in which objects can be placed anywhere.
- The link between objects is shown as a solid line.
- The messages are labelled with arrow and prefixed with sequence number.

Symbols in Collaboration Diagram



Job Recruiter System



❖ Objects and Patterns

Patterns

- Patterns is a way of doing something.
- Design pattern is a category of patterns that deals with object-oriented software.
- Design pattern is a general repeatable solution to a commonly occurring problem in software design.
- It is a description for how to solve a problem that can be used in many different situations.

- OOAD design patterns that are frequently used include the following:
  - **Creational Patterns:** These patterns focus on the techniques involved in the creation of objects, helping in their appropriate creation. Examples include the Factory Method pattern, Builder pattern, and Singleton pattern.
  - **Structural Patterns:** Structural patterns deal with object composition and class relationships, aiming to simplify the structure of classes and objects. Examples include the Adapter pattern, Composite pattern, and Decorator pattern.
  - **Behavioral Patterns:** Behavioral patterns address how objects interact and communicate with each other, focusing on the delegation of responsibilities between objects. Examples include the Observer pattern, Strategy pattern, and Command pattern.
  - **Architectural Patterns:** These patterns provide high-level templates for organizing a software system’s general structure. Examples include the Model-View-Controller (MVC) pattern Layered Architecture pattern, and Microservices pattern.

Commonly used Design patterns

1. **Singleton Pattern**

- Ensures that a class has only one instance and provides a global point of access to it. Useful for managing global resources or maintaining a single configuration throughout an application.

2. **Factory Method Pattern**

- Defines an interface for creating an object but allows subclasses to alter the type of objects that will be created. Useful for decoupling the creation of objects from the client code.

3. **Abstract Factory Pattern**

- Provides an interface for creating families of related or dependent objects without specifying their concrete classes. Useful for creating objects with varying implementations but ensuring they work together seamlessly.

4. Builder Pattern

- Separates the construction of a complex object from its representation, allowing the same construction process to create different representations. Useful for creating objects with large number of configuration options or parameters.

5. Prototype Pattern

- Creates new objects by copying an existing object, known as the prototype, rather than creating new instances from scratch. Useful for improving performance and reducing the overhead of object creation.

6.Adapter Pattern

- Allows incompatible interfaces to work together by providing a wrapper or intermediary that converts the interface of one class into another interface expected by the client. Useful for integrating legacy code or third-party libraries into new systems.

7. Observer Pattern

- Defines a one-to-many dependency between objects so that when one object changes state, all its dependents are notified and updated automatically. Useful for implementing event handling systems or maintaining consistency between related objects.

8. Strategy Pattern

- Defines a family of algorithms, encapsulates each one, and makes them interchangeable. It allows the algorithm to vary independently from the clients that use it. Useful for selecting algorithms at runtime or providing different implementations of the same behavior.
- These design patterns provide solutions to common design problems encountered during software development and promote principles such as code reuse, modularity, and flexibility in OOAD.

Frameworks

- Framework is a group of concrete classes which can be directly implemented on an existing platform.
- They are written in programming languages.
- They are concerned with specific application domain.
- Frameworks in Object-Oriented Analysis and Design (OOAD) are reusable, customizable structures that provide a foundation for developing software applications.
- typically consist of pre-defined classes, interfaces, and design patterns that encapsulate common functionalities and architectural decisions.
- They streamline the development process by offering a set of conventions, guidelines, and tools that developers can leverage to build applications more efficiently.

Patterns vs Frameworks

- Pattern is a concept while framework is code to be used.
- Pattern supports reuse of software architecture and design while framework supports reuse of detailed design and code.

❖ Determining Visibility

- It is the ability of an object to have a reference to another object.
- It indicates the scope of the objects.
- For a sender object to send message to a receiver object, the sender must be visible to the receiver.
- There are four ways to achieve visibility.

➢***Attribute Visibility***

- Attribute visibility from A to B exists when B is an attribute of A.
- It is relatively permanent i.e. it persists as long as A and B both exists.

➢***Parameter Visibility***

- Parameter visibility from A to B exists when B is passed as a parameter to a method of A.

- It is relatively temporary i.e. it exists within the scope of the method.
- It is generally transformed into attribute visibility within a method

➢***Local Visibility***

- Local visibility from A to B exists when B is declared as a local object within a method of A.
- It is temporary as it persists only within the scope of the method.
- It can be achieved by:

1. Create a new local instance and assign it to local variable.
2. Assign returning object from a method invocation to a local variable.

➢***Global Visibility***

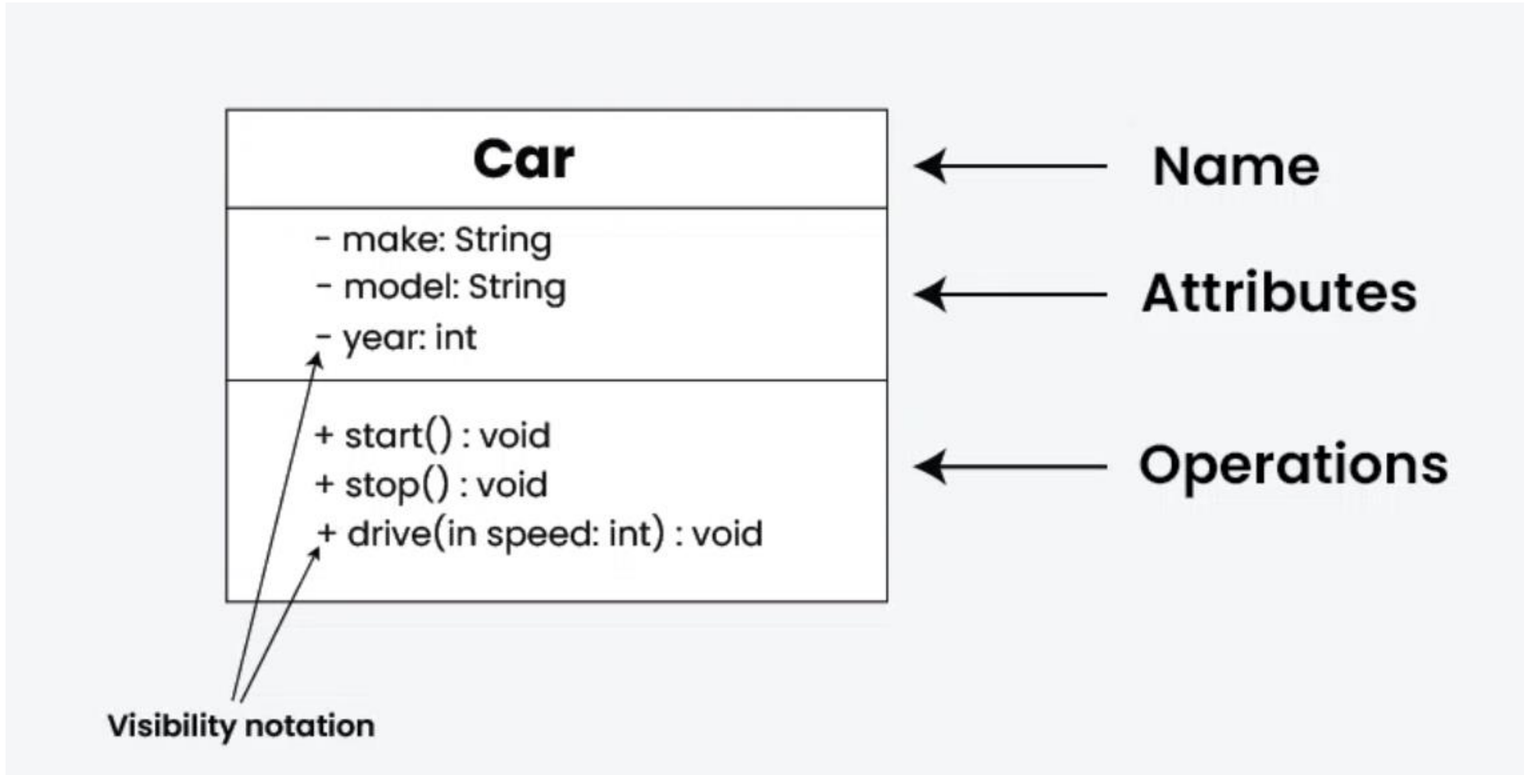
- Global visibility from A to B exists when B is global to A.It is permanent.
- It can be achieved by:

1. Assign an instance to a global variable.

❖ Class Diagrams

- Class diagram is used for static modeling that illustrates classes, interfaces and their relationships.
- It shows the structural view of the system.

Class notation



Relationships between classes

➢Association

- It enables objects to communicate with each other.
- It describes a connection between classes.
- A link is the physical or conceptual connection between object instances.
- The association relationship of a class with itself is known as recursive association.
- It is represented by drawing a straight line between concerned classes.
- Arrow -head can be used to indicate reading direction.
- The multiplicity is noted on each side.

➢Aggregation

- It is the association in which the involved classes represent a whole-part relationship.
- It takes the responsibility of leadership.
- When an instance of one object contains instances of some other objects, then aggregation exists between composite object and component object.
- It is represented by a diamond symbol at the end of a relationship.
- It can never be recursive and symmetric.

➢Composition

- It is the strict form of aggregation, in which the parts are existence-dependent on whole.
- It is represented as a filled diamond drawn at composite end.
- The part instance can only be the part of one composite at a time.

- Creation and deletion of parts is managed by composite.

➢**Inheritance (Generalization and Specialization)**

- It describes 'is a kind of' relationships between classes.
- Object of derived class inherits the properties of base class.
- It is defined statically.

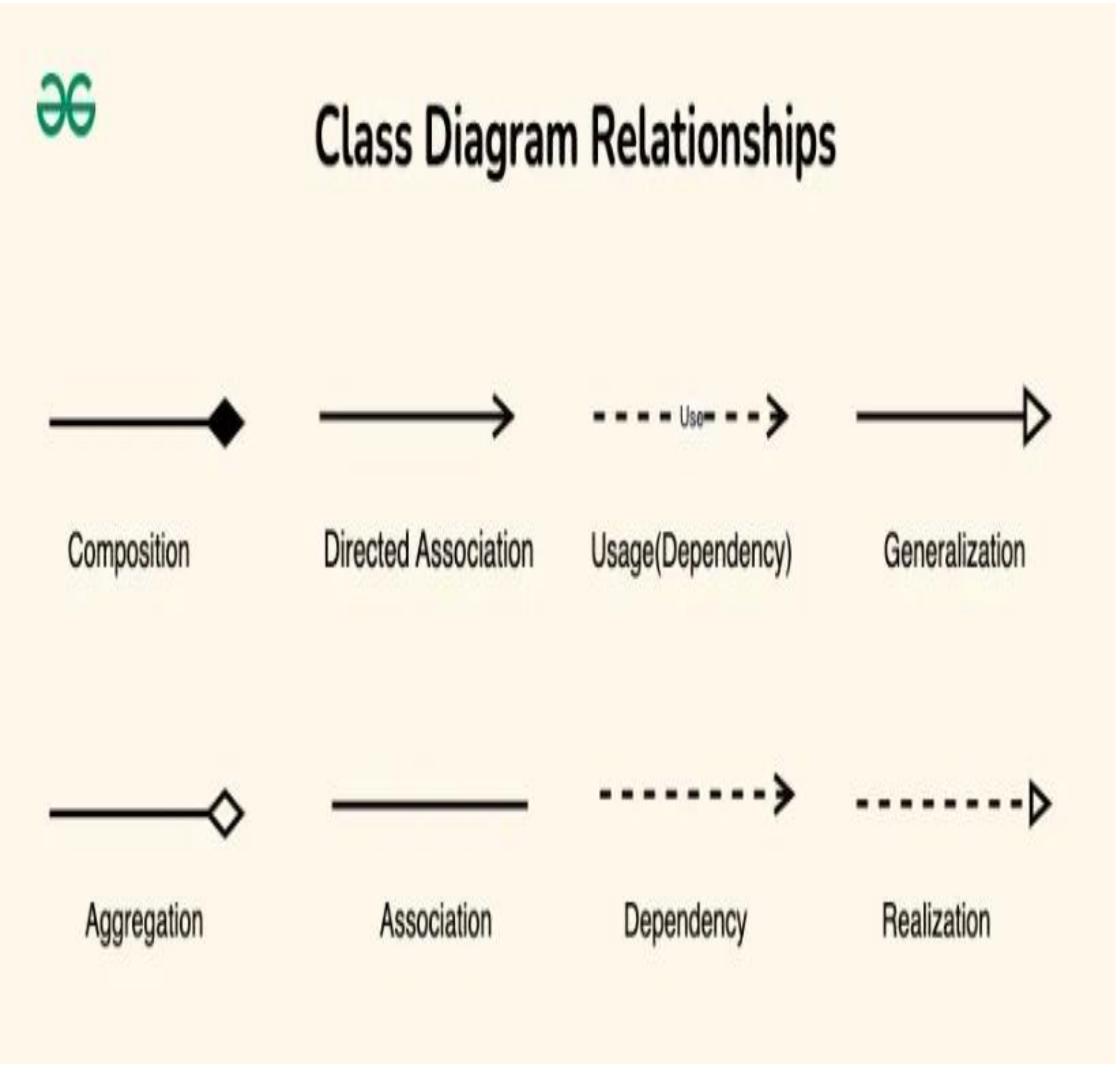
➢**Dependency**

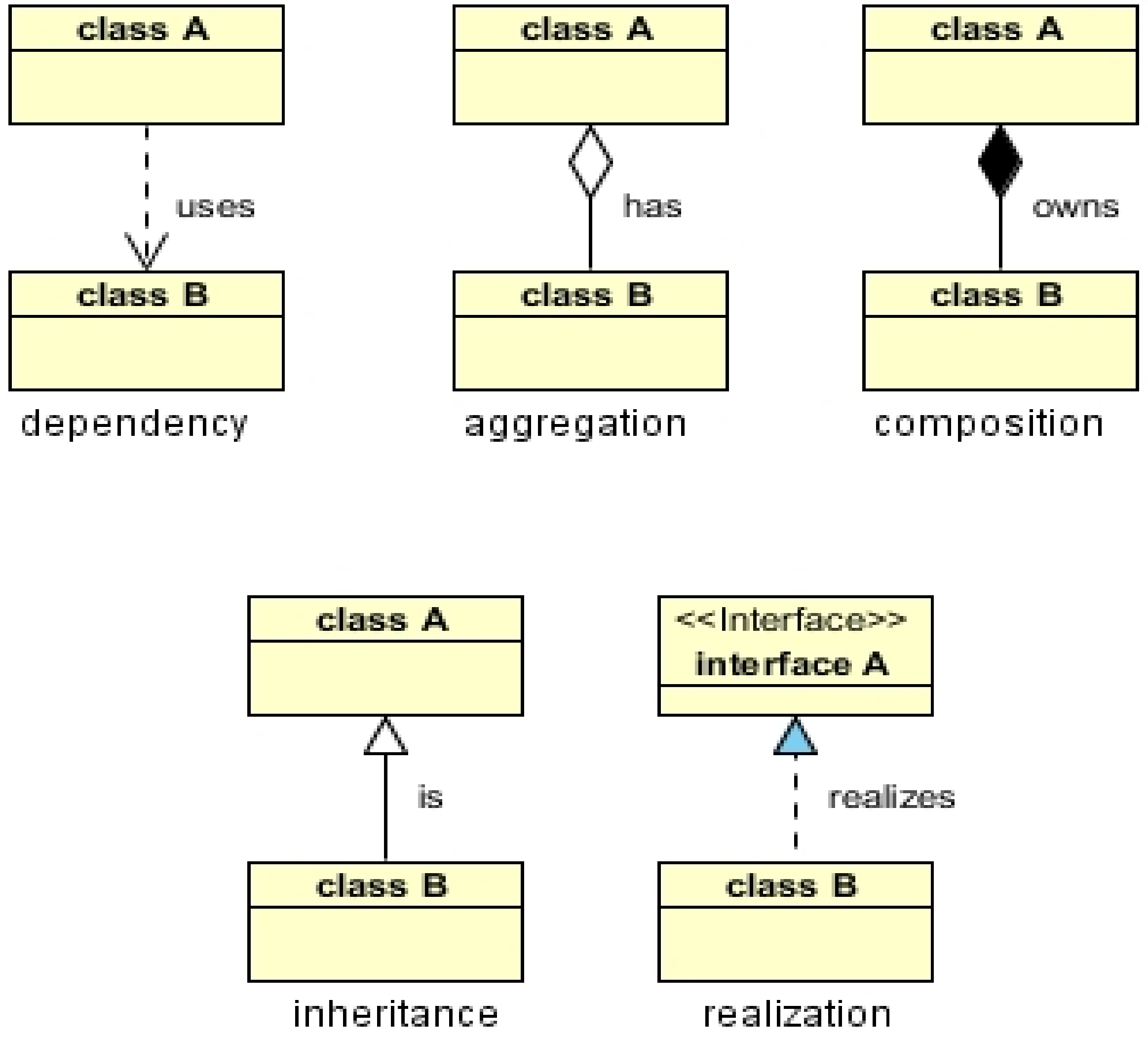
- It indicates that one element has knowledge of another element.
- It states that a change in specification of one thing may affect another thing using it, but not necessarily the reverse.
- It depicts non-attribute visibility between classes.

➢**Realization**

- It indicates the implementation of functionality defined in one class by another class.

Class Diagram Relation





THANK YOU

MCQ: OOD

1. What does a simple name in UML Class and objects consist of?
a) Letters
b) Digits
c) Punctuation Characters
d) All of the mentioned

2. What Does a Composite name consist of in a UML Class and object diagram?
a) Delimiter
b) Simple names
c) Digits
d) All of the mentioned

3. A Class consists of which of these abstractions?
a) Set of the objects
b) Operations
c) Attributes
d) All of the mentioned

4. A class is divided into which of these compartments?
a) Name Compartment
b) Attribute Compartment
c) Operation Compartment
d) All of the mentioned

5. What should be mentioned as attributes for conceptual modelling?
a) Initial Values
b) Names
c) All of the mentioned
d) None of the mentioned

6. What among the following statement is true?
a) Associations may also correspond to the relation between instances of three or more classes
b) Association lines may be unlabeled, or they may show association name
c) All of the mentioned
d) None of the mentioned

7. What is **multiplicity** for an association?
a) The multiplicity at the target class end of an association **is the number of instances** that can be associated with a **single instance o**f source class
b) The multiplicity at the target class end of an association is **the number of instances** that can be associated with a **number instance of** source class
c) All of the mentioned
d) None of the mentioned

8. Which among these are the rules to be considered to form Class diagrams?
a) Class symbols must have at **least a name compartment**
b) Compartment can be in random order
c) Attributes and operations can be listed at any suitable place
d) None of the mentioned

9. Which of the following statement is true?
a) A transition is a change from one state to another
b) Transitions may be spontaneous, but usually some event triggers them.
c) An event is a noteworthy occurrence at a particular time; events have no duration.
d) All of the mentioned

10. Which of the following determines the state diagram?
a) The UML notation for specifying finite automata is the state diagram
b) In state diagrams, states are represented by rounded rectangles
c) All of the mentioned
d) None of the mentioned

11. Mid -level generation design techniques are classified into which of the following?
a) Creational Techniques
b) Transitional Techniques
c) All of the mentioned
d) None of the mentioned

12. Why does designers look for candidate classes?
a) To model entities in charge of or involved in program tasks
b) To model things in the world that interact directly with the program
c) To model structures and collections of objects
d) All of the mentioned

13. Which of the following is referred for the conceptual modelling?
a) Change actors to interface classes
b) Add actor domain classes
c) Convert or add controllers and coordinators
d) All of the mentioned

14. What is the Interaction diagram?
a) Interaction diagrams are the UML notations for dynamic modeling of **collaborations**
b) Interaction diagrams are a **central focus of engineering design.**
c) All of the mentioned.
d) None of the mentioned

15. What are the different interaction diagram notations does UML have?
a) A sequence diagram
b) A communication diagram

c) An interaction overview diagram
d) All of the mentioned

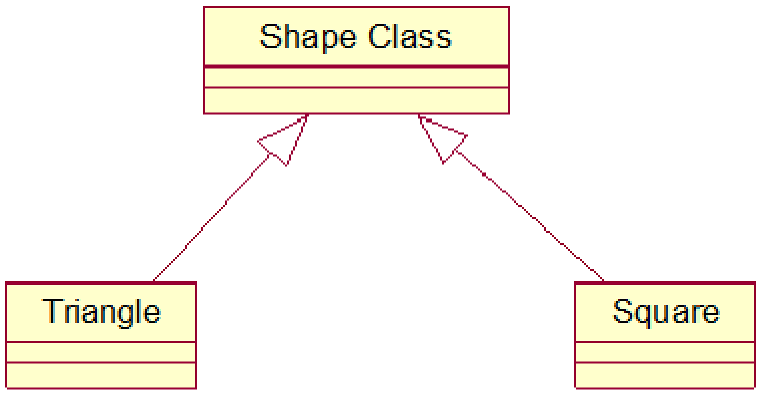
16. What is a sequence diagram?
a) A diagram that shows interacting individuals along the top of the diagram and messages passed among them **arranged in temporal order** down the page
b) A diagram that shows messages superimposed on a diagram depicting collaborating individuals and the links among them
c) A diagram that shows the change of an individual's state over time
d) All of the mentioned

17. What does a message mean?
a) It Passes all communications from one object to another and are represented by message arrows in sequence diagrams
b) The message goes from the sending object's lifeline to the receiving object's lifeline
c) It is a rectangle containing an identifier with a dashed line extending below the rectangle
d) All of the mentioned

18. What are the interaction fragments?
a) A fragment which is a rectangular frame with a pentagonal operation compartment in the upper left-hand corner
b) A fragment which has a marked part of an interaction specification
c) The regions resulting from these divisions will not hold the interaction fragment operations
d) All of the mentioned

19. Which of the following UML diagrams has a static view?
a) Collaboration
b) Use case
c) State chart
d) Activity

20. What type of relationship is represented by Shape class and Square?



a) Realization
b) Generalization

c) Aggregation
d) Dependency

21. Which diagram in UML shows a complete or partial view of the structure of a modeled system at a specific time?
a) Sequence Diagram
b) Collaboration Diagram
c) Class Diagram
d) Object Diagram

22. Which of the following diagram is time oriented?
a) Collaboration
b) Sequence
c) Activity
d) None of the mentioned

23. Classes and interfaces are a part of
a) Structural things
b) Behavioral things
c) Grouping things
d) Annotational things

24. Which of the following term is best defined by the statement: "a structural relationship that specifies that objects of one thing are connected to objects of another"?
a) Association
b) Aggregation
c) Realization
d) Generalization

25.who considers diagrams as a type of Class diagram, component diagram, object diagram, and deployment diagram?

a) structural
b) behavioral
c) non-behavioral
d) nonstructural

26.__________ represented by In UML diagrams, relationship between component parts and object.

a) ordination
b) aggregation
c) segregation
d) increment

27.Which of the following is correct list of classifications of design patterns.
a) Creational, Structural and Behavioral patterns

b) Executional, Structural and Behavioral patterns
c) Creational, Executional and Behavioral patterns
d)None of the above

28. Which of the following is a design pattern?
a) Behavioral
b) Structural
c) Abstract Factory
d) All of the mentioned

29. You want to minimize development costs by reusing methods? Which design pattern would you choose?
a) Adapter Pattern
b) Singleton Pattern
c) Delegation pattern
d) Immutable Pattern

30. The recurring aspects of designs are called design
a) patterns
b) documents
c) structures
d) methods

31. Which pattern prevents one from creating more than one instance of a variable?
a) Factory Method
b) Singleton
c) Observer
d) None of the mentioned

32. You want to avoid multiple inheritance. Which design pattern would you choose?
a) Abstraction-Occurrence Pattern
b) Player-Role Pattern
c) General Hierarchy Pattern
d) Singleton Pattern

33. Which design pattern defines one-to-many dependency among objects?
a) Singleton pattern
b) Facade Pattern
c) Observer pattern
d) Factory method pattern

34. Why are Patterns important?
a) They capture expert design knowledge.
b) They make captured design accessible to both novices and other experts
c) All of the mentioned
d) None of the mentioned

35. Which of the following Choices and standardizes patterns for a problem domain promotes software reuse and, hence, quality and productivity?
a) Promoting Communication
b) Streamlining Documentation
c) Increasing Development Efficiency
d) Supporting Software Reuse

36. What is a pattern?
a) It is a model proposed for imitation
b) It solves a software design problem
c) All of the mentioned
d) None of the mentioned

37. Which of the following UML diagrams has a static view?
a) Collaboration
b) Use case
c) State chart
d) Activity

1.(d) 11.(c) 21.(d) 31.(b)

2.(d) 12.(d) 22.(b) 32.(b)

3.(d) 13.(d) 23.(a) 33.(c)

4.(d) 14.(c) 24.(a) 34.(c)

5.(c) 15.(d) 25.(a) 35.(d)

6.(c) 16.(a) 26.(b) 36.(c)

7.(a) 17.(a) 27.(a)

8.(a) 18.(d) 28.(d)

9.(d) 19.(b) 29.(c)

10.(c)20.(b) 30.(a)