

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(OPTIONAL) 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.

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

- CRC (Class, Responsibility, Collaboration) cards are a design tool used in object-oriented programming to capture and organize information about classes, their responsibilities, and their collaborations within a software system.
- 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.

Class Name	
Responsibilities	Collaborators

Fig: CRC card Format

Student	
Student number Name Address Phone number Enroll in a seminar Drop a seminar Request transcripts	Seminar

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.

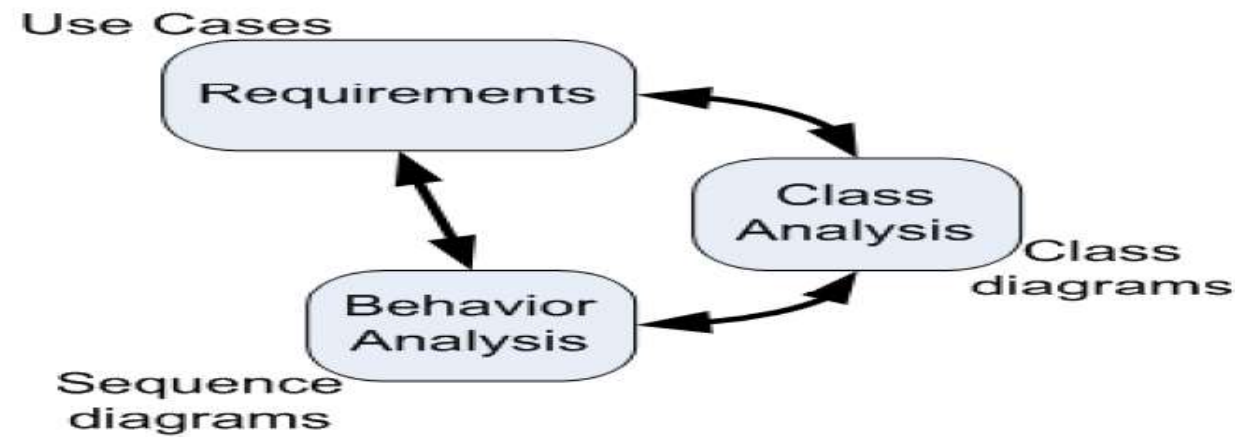
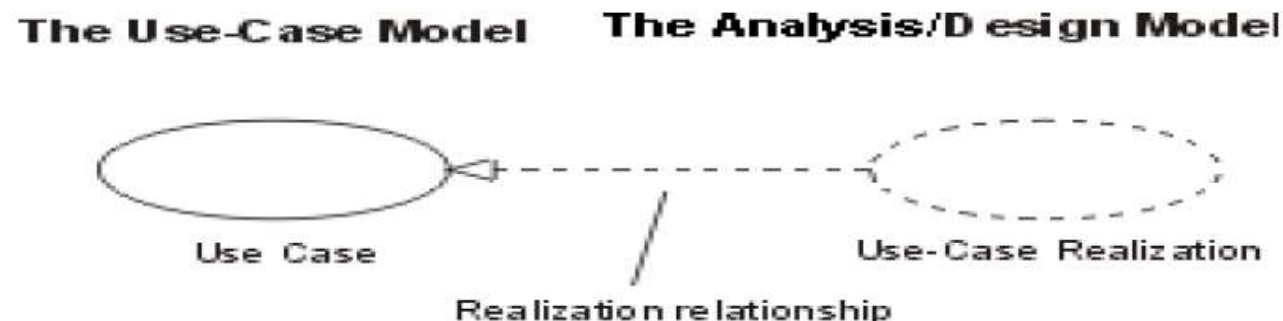


Fig: Use case realization process



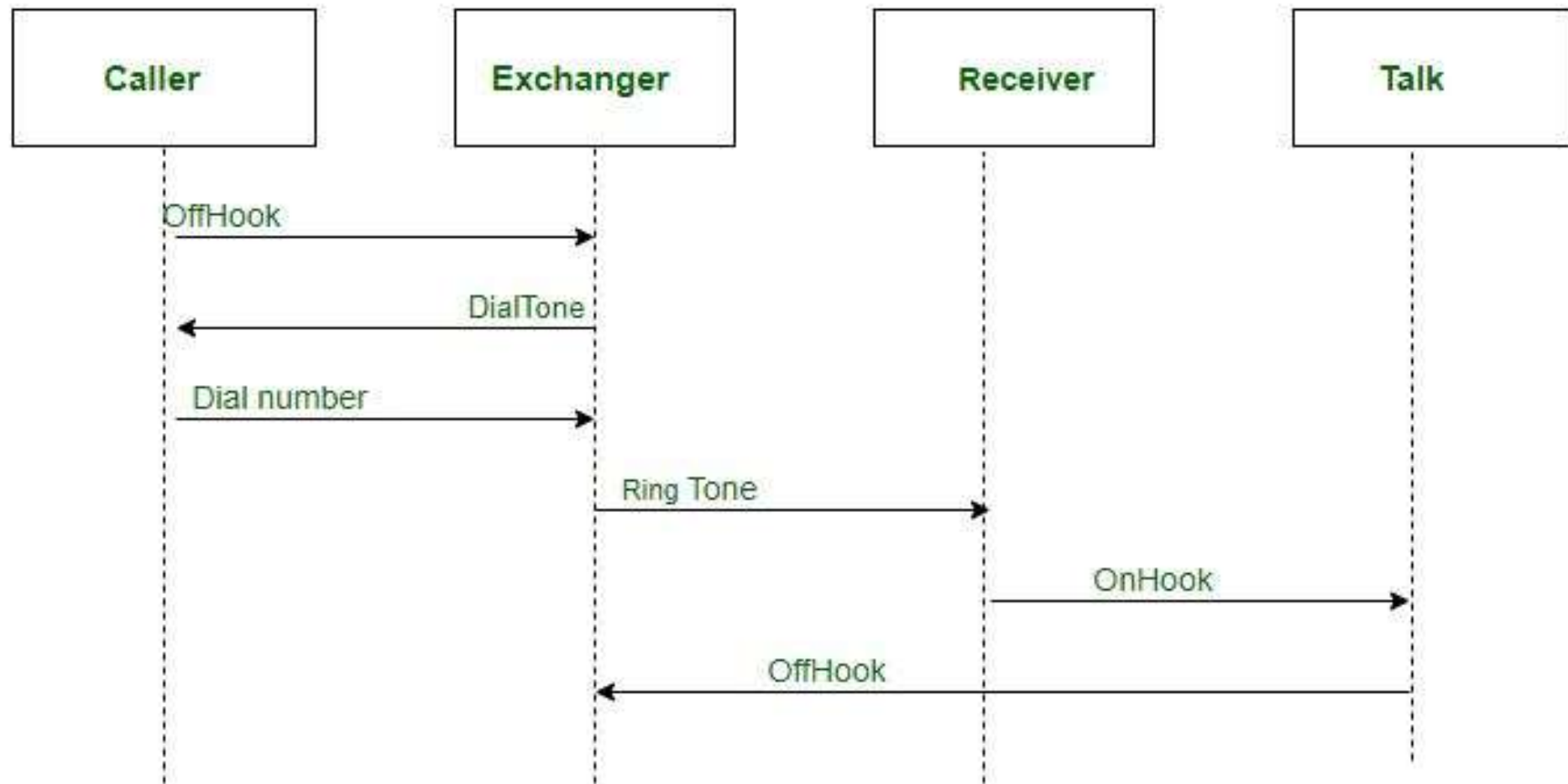
UML Interaction Diagram

- 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.**

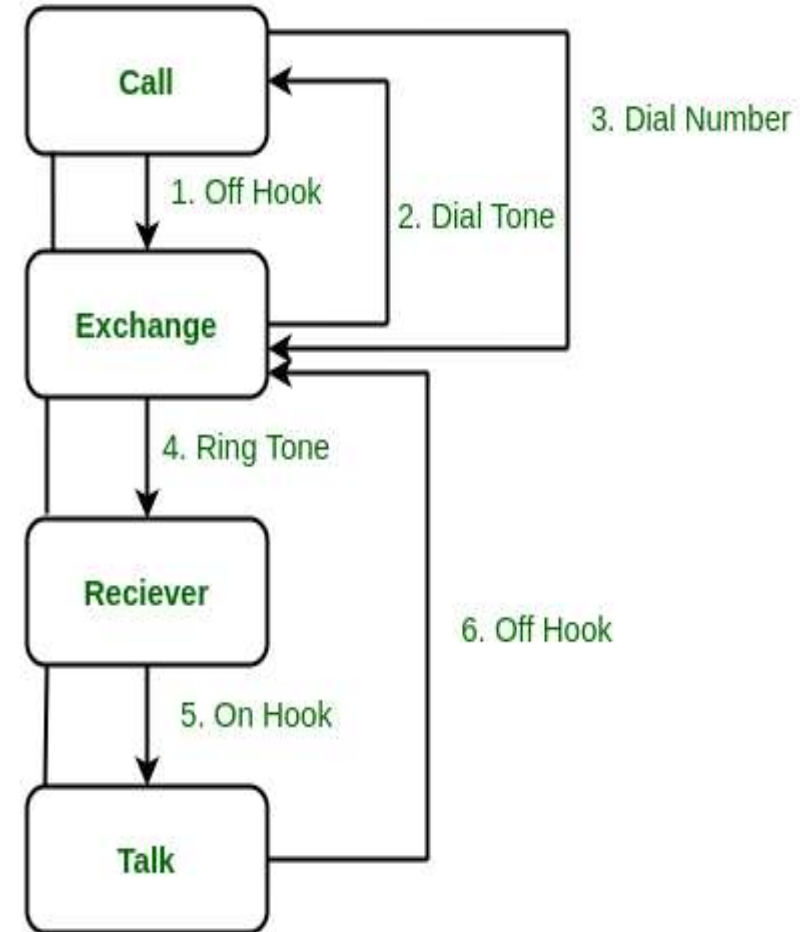
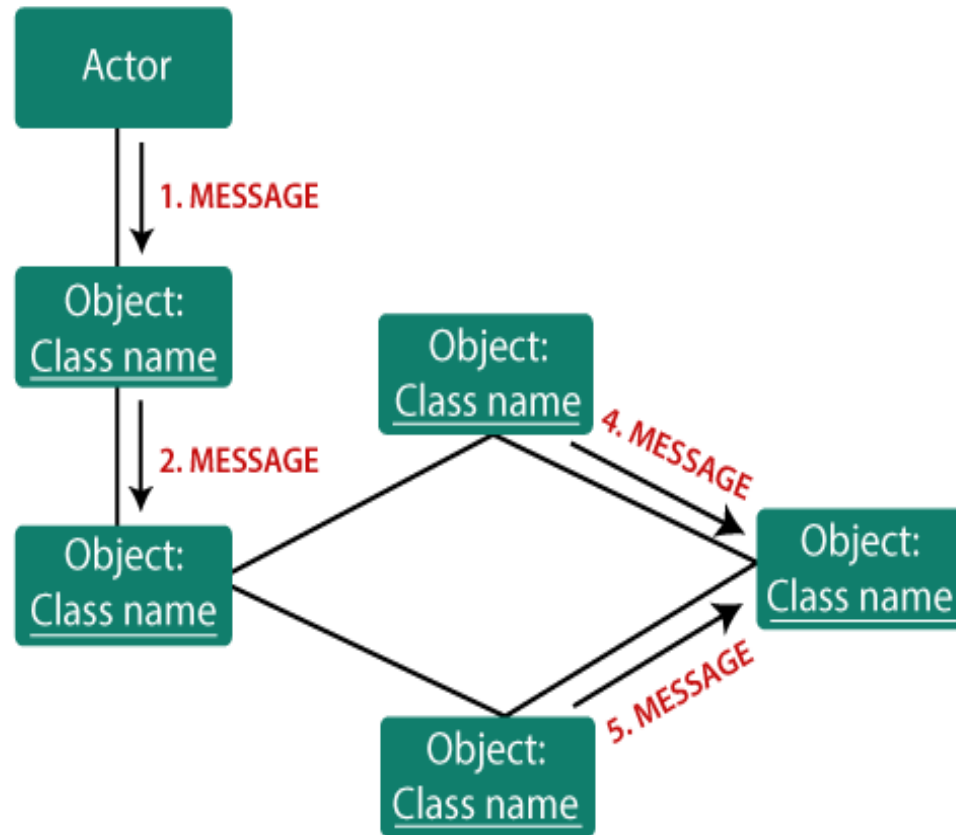


❖ **Collaboration Diagram**

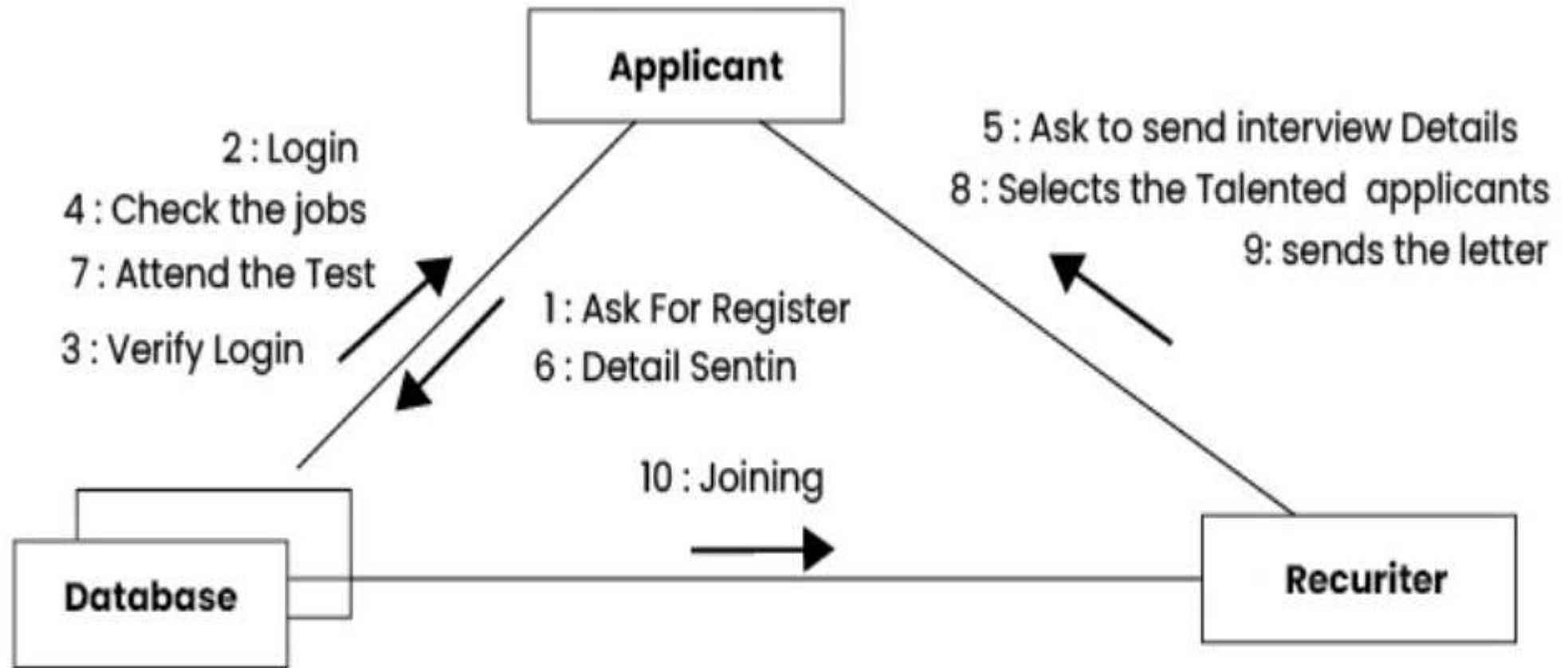
- Is a behavioral diagram which is also referred to as a communication diagram, It illustrates how objects or components interact with each other to achieve specific tasks or scenarios within a system.
- Collaboration diagram shows both structural and behavioral aspects.
- Structural aspect includes 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

Components of a 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.**

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

- **Scope:** Design patterns are about solving specific problems, while frameworks provide a broad structure for developing applications.
- **Implementation:** Patterns are implemented at the design level, while frameworks offer libraries and tools that developers use to build software.
- **Flexibility:** Design patterns can be adapted and implemented in various ways, while frameworks often have specific rules and structures that must be followed.

❖ **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.

- In object-oriented design, visibility refers to the accessibility of classes, methods, and attributes in relation to other parts of a program. It's primarily controlled by access modifiers, which dictate how and where these elements can be accessed. The main access modifiers are:

1.Public: Members are accessible from anywhere in the program.

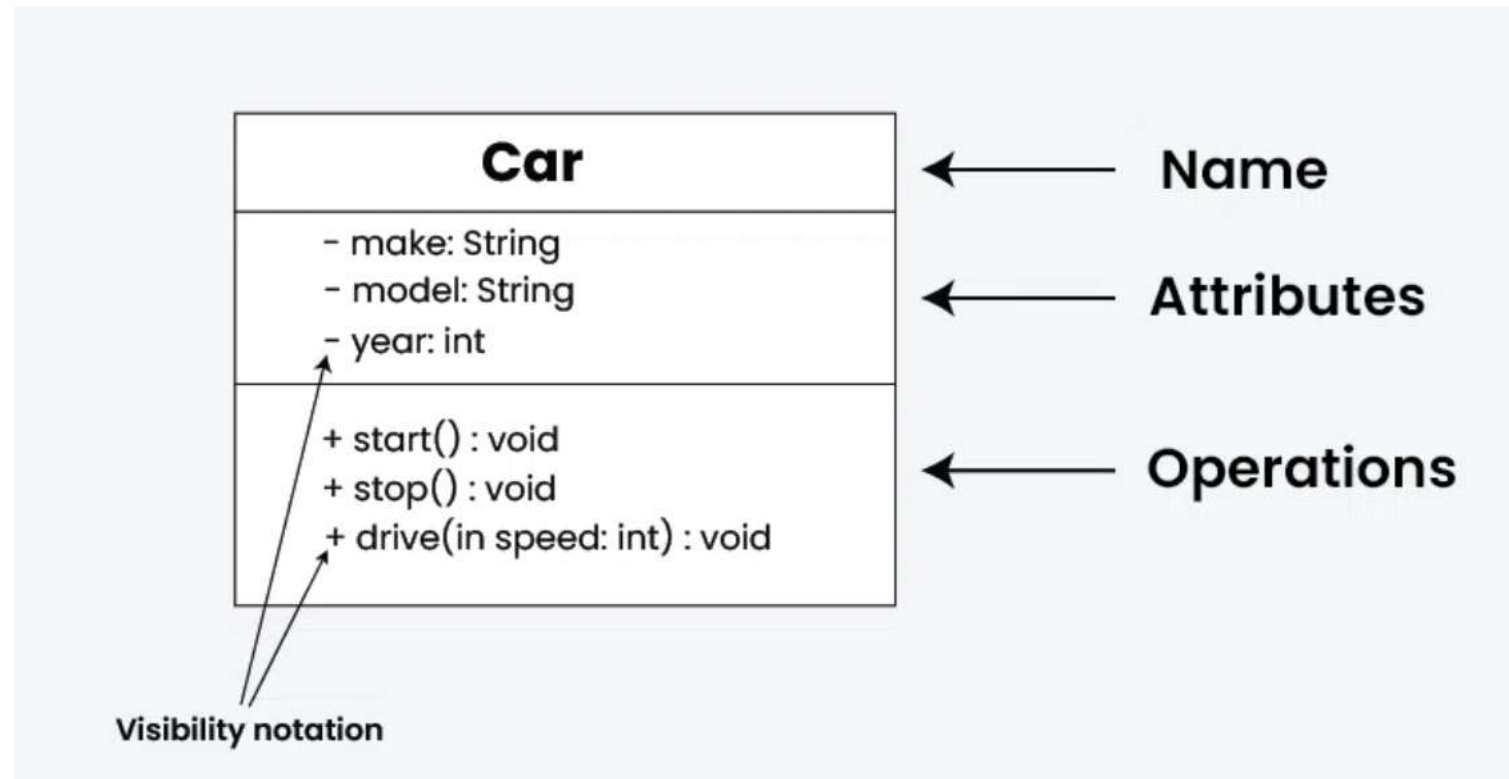
2.Private: Members are accessible only within the defining class, promoting encapsulation and hiding implementation details.

3.Protected: Members are accessible within the defining class and its subclasses, allowing for controlled inheritance.

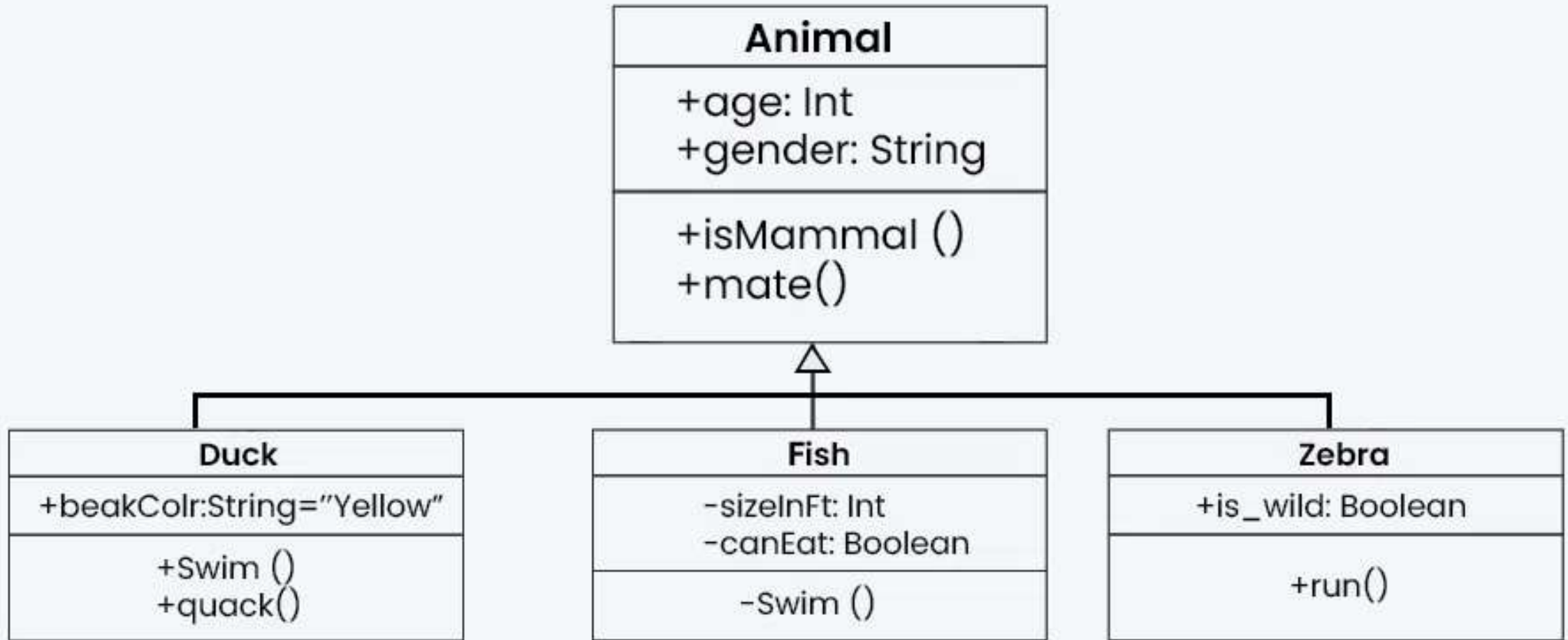
❖ Class Diagrams

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

Class notation



❖ Class Diagrams



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.

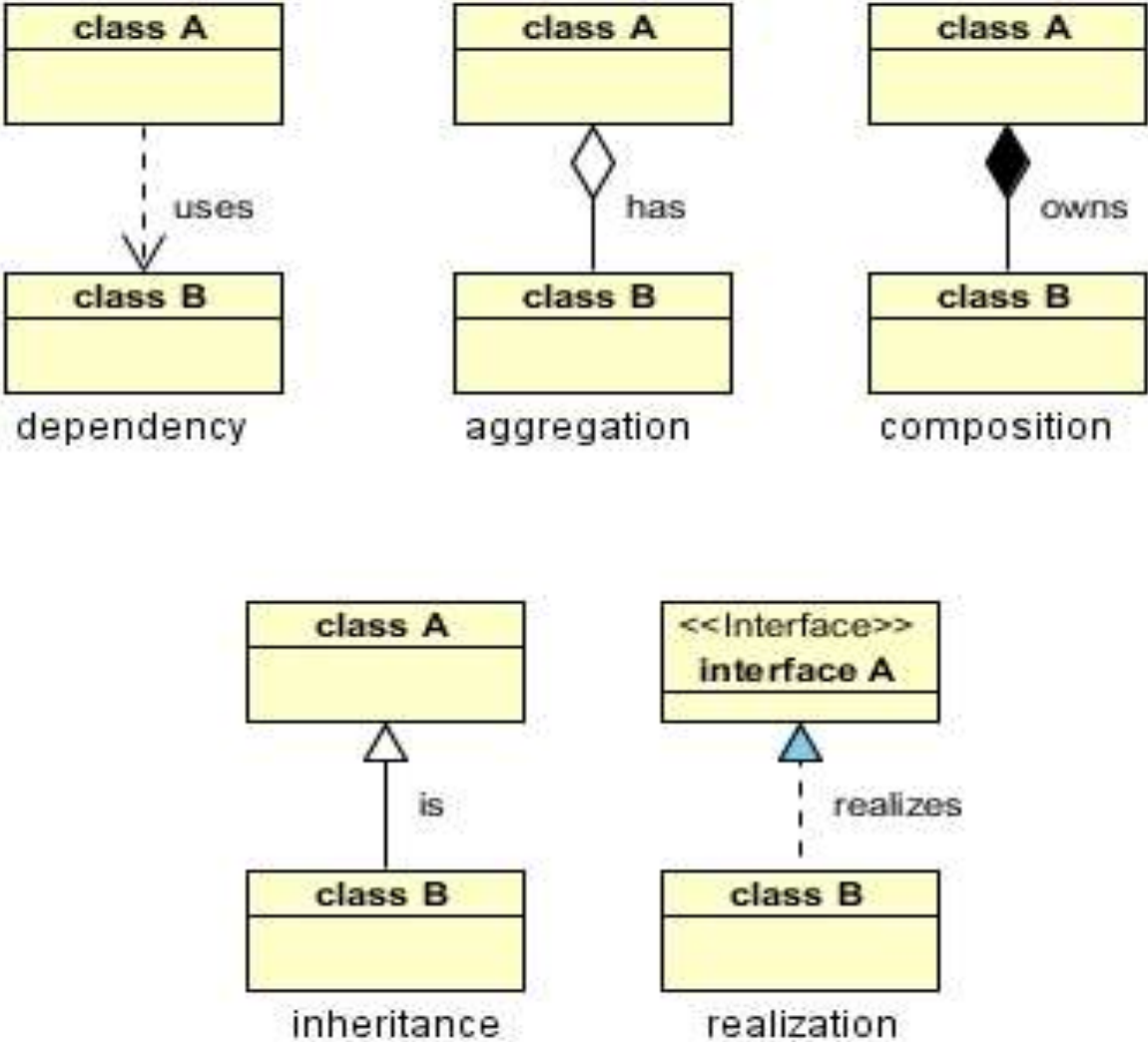
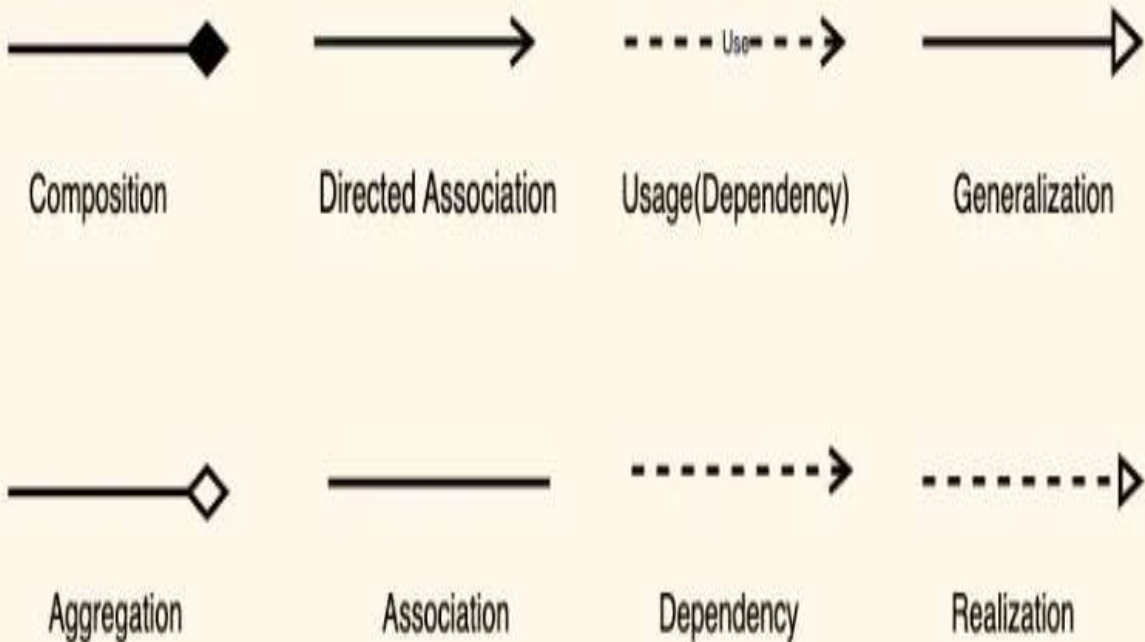
➤ **Realization**

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

Class Diagram Relation



Class Diagram Relationships



THANK YOU