**# OOMD Question Bank……..**

1. **What is Aggregation?**  
   Aggregation is a type of association that represents a whole-part relationship between two classes in object-oriented programming. It is a "has-a" relationship where one class (the whole) is composed of one or more instances of another class (the part). However, the part can exist independently of the whole.

Example: A Library has multiple Books, but the Books can exist without the Library.

1. **Define Aggregation.**  
   Aggregation is a specialized form of association where a class contains a reference to one or more instances of another class, representing a part-whole relationship. The lifetime of the part is not dependent on the lifetime of the whole.
2. **Benefits of Aggregation:**
   * Reusability: Promotes code reuse by allowing complex classes to be built from simpler ones.
   * Maintainability: Makes the system easier to maintain as individual parts can be updated independently.
   * Flexibility: Allows changing the composition without modifying the main class.
   * Improved Abstraction: Helps in modeling real-world relationships more accurately.
3. **Real-life Examples of Aggregation:**
   * A University has multiple Departments (Departments can exist without the University).
   * A Car has an Engine (Engine can be used in different cars).
   * A Team has Players (Players can move to different teams).
4. **Challenges in Aggregation:**
   * Increased Complexity: Managing relationships between multiple parts can be complex.
   * Memory Management: Large aggregated structures can consume significant memory.
   * Testing Difficulties: Testing individual components without full context can be challenging.
5. **Difference Between Aggregation and Association:**
   * **Association:** A general relationship where classes are related but not necessarily dependent (e.g., a Teacher teaches a Student).
   * **Aggregation:** A special form of association where a part-whole relationship exists, but the parts can exist independently (e.g., a Library has Books).
6. **What is Generalization? Define Generalization.**  
   Generalization is a relationship where a subclass inherits the properties and behaviors of a superclass. It represents an "is-a" relationship, supporting inheritance and polymorphism.

Example: A Car is a Vehicle, and a Truck is also a Vehicle, both inheriting common properties from the Vehicle superclass.

1. **Benefits of Generalization:**
   * Code Reusability: Common features are defined once in a superclass and reused in subclasses.
   * Reduced Code Redundancy: Prevents duplication of common code.
   * Polymorphism: Allows subclasses to define their own unique behaviors while sharing a common interface.
2. **Difference Between Generalization and Specialization:**
   * **Generalization:** Extracting common features from multiple classes to form a superclass (e.g., Vehicle → Car, Bike).
   * **Specialization:** Creating subclasses with additional attributes or behaviors (e.g., Car → Electric Car, Sports Car).
3. **Define Class Diagram:**  
   A class diagram is a type of UML (Unified Modeling Language) diagram that shows the structure of a system by displaying its classes, attributes, methods, and the relationships between classes. It is used to model the static view of an application.
4. **Define Object Diagram, Object, and Instance:**

* **Object Diagram:** Represents the instances of classes and their relationships at a specific moment in time. It provides a snapshot of the system state.
* **Object:** An instance of a class that contains specific values for the attributes defined by the class.
* **Instance:** A concrete occurrence of a class, representing a real-world entity in the system.

1. **Types of Object Diagrams:**

* Instance Diagrams: Focused on specific instances at a point in time.
* Structural Diagrams: Represent static structure and relationships.
* Composite Diagrams: Combine multiple object relationships.

1. **How to Draw Class Diagrams and Object Diagrams in the Real World:**

* Identify the classes and their attributes.
* Define the relationships (aggregation, association, generalization).
* Use appropriate UML notations.
* Focus on clarity and simplicity for better understanding.