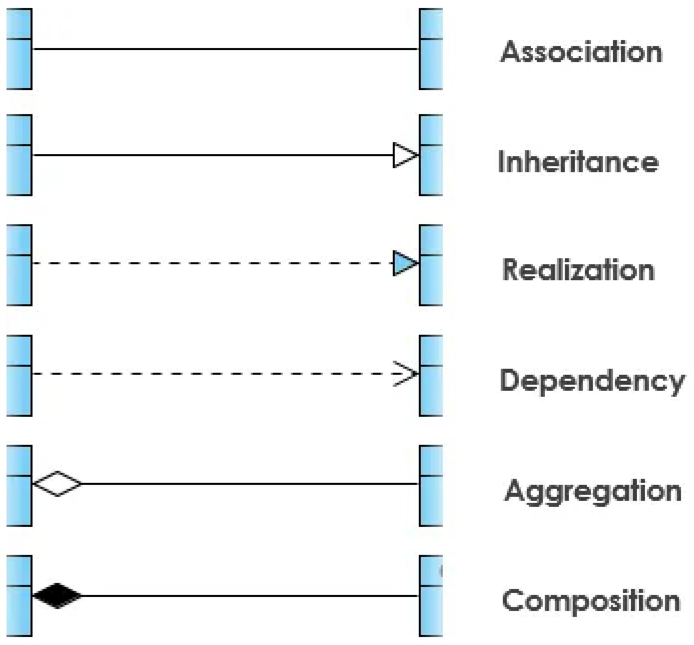
1908008- OBJECT ORIENTED ANALYSIS AND DESIGN
UNIT I ANSWER KEY
PART-A

1. Define Class diagram.

A class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among objects.

2. List the relationships used in class diagrams.



3. What is an attribute? Mention its types.

Attributes describe a value or a range of values that instances of the classifier can hold. You can specify an attribute's type, such as an integer or Boolean, and its initial value.

There are six such types of attributes:

1. Simple
2. Composite
3. Single-valued
4. Multi-valued
5. Derived attribute
6. Aggregated Attribute

4. What do you mean by sequence number in UML? Analyze Where and for what it is used?

In UML (Unified Modeling Language), a sequence number typically refers to the order in which messages or interactions occur within a sequence diagram. It helps in depicting the chronological flow of interactions between objects or components in a system

5. Express the meaning of Elaboration and What are the tasks performed in elaboration?

Elaboration in software development signifies a pivotal phase where requirements are meticulously analyzed, architectural frameworks are meticulously crafted, and risks are meticulously evaluated and managed.

Tasks in elaboration include:

1. Refining Requirements
2. Solidifying Architecture
3. Analyzing Risks
4. Prototyping
5. Planning Iterations
6. Developing Essential Use Cases
7. Evaluating Alternatives

6. Express why we call a domain model a “Visual Dictionary".

A domain model is often likened to a "Visual Dictionary" because, like a dictionary, it visually organizes and represents the essential elements and relationships within a specific domain. This visual representation facilitates comprehension and communication among stakeholders, promoting a shared understanding and effective collaboration throughout the software development process.

7. Define Domain Model. How to create a Domain model?

A domain model is a conceptual representation of the real-world domain within which a software system operates. It defines the various entities, their attributes, relationships, and behaviors that exist within the domain.

1.Identify Entities,
2.Define Attributes,
3.Determine Relationships,
4.Specify Behaviors,
5.Document the Model,
6.Iterate and Update

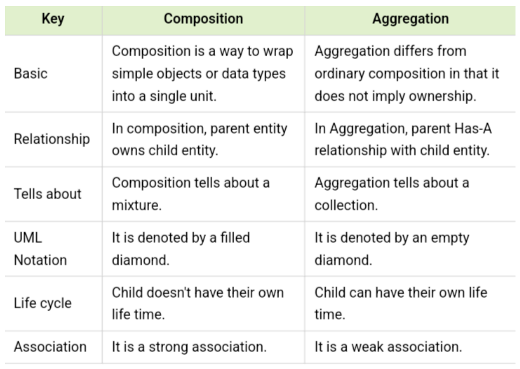
8.Define Conceptual class

A conceptual class is an idea, thing, or object. It is considered in terms of its symbol, intension, and extension. Symbol—words or images representing a conceptual class. Intension—the definition of a conceptual class. Extension—the set of examples to which the conceptual class applies.

9. Rank the 3 strategies to find conceptual class.

•Reuse or modify existing models.
•Use a category list.
•Identify noun and noun phrases.

10.Differentiate aggregation and composition.



Key	Composition	Aggregation
Basic	Composition is a way to wrap simple objects or data types into a single unit.	Aggregation differs from ordinary composition in that it does not imply ownership.
Relationship	In composition, parent entity owns child entity.	In Aggregation, parent Has-A relationship with child entity.
Tells about	Composition tells about a mixture.	Aggregation tells about a collection.
UML Notation	It is denoted by a filled diamond.	It is denoted by an empty diamond.
Life cycle	Child doesn't have their own life time.	Child can have their own life time.
Association	It is a strong association.	It is a weak association.

11. Estimate the purpose of association relationship.
The purpose of an association relationship in object-oriented modeling is to depict connections and interactions between classes or objects within a system. Associations represent the relationships between instances of classes, showing how they are related or interact with each other.

12. Give the meaning of abstract conceptual class
An abstract conceptual class in object-oriented analysis and design (OOAD) is a class that defines a common set of attributes and behaviors shared by a group of related classes within a problem domain. However, unlike concrete classes, abstract conceptual classes cannot be instantiated directly

13.Identify the usage of Description class.
The Description class in Object-Oriented Analysis and Design (OOAD) serves to provide detailed textual or graphical descriptions of various elements within the system being designed. These descriptions help in documenting important attributes, behaviors, relationships, or other characteristics of classes, objects, use cases, or other components in the system. Essentially, it aids in communicating vital information about the system to stakeholders during the design process, ensuring clarity and understanding of the system's structure and functionality.

14.Organize the guideline to partition a class into subclass.
Here are some guidelines to help you effectively partition a class into subclasses:

- Identify Common Characteristics,
- Apply the Single Responsibility Principle (SRP),
- Use Inheritance Hierarchies,
- Encapsulate Variation,
- Follow the Open/Closed Principle (OCP),
- Consider Polymorphism,
- Factor Out Reusable Behavior,
- Use Composition,

15.Illustrate When to use class diagram
Class diagrams in Object-Oriented Analysis and Design (OOAD) are used to depict the static structure of a system, including classes, attributes, methods, and their relationships. They aid in visualizing system architecture, identifying design flaws, and facilitating communication among stakeholders. Additionally, they help in object-oriented modeling, communication with stakeholders, and refactoring/maintenance of the system.

16.When to define new data type classes?
Defining new data type classes is appropriate when encapsulating complex data structures, domain-specific data, validation rules, or custom behaviors within your application. This approach promotes abstraction, encapsulation, reusability, and modularity, enhancing the maintainability and robustness of software systems.

17.Interpret the meaning of Generalization.
Generalization in object-oriented modeling refers to the relationship between classes where one class (the subclass or child class) inherits attributes, behaviors, and relationships from another class (the superclass or parent class). This relationship represents an "is-a" relationship, indicating that the subclass is a specialized version of the superclass.

18.Compare qualified association and reflexive association.

Qualified association involves adding qualifiers to an association to uniquely identify instances based on certain criteria. It's useful when additional information is needed to navigate through the association, ensuring precise identification of related instances.

On the other hand, reflexive association signifies a relationship between instances of the same class, indicating self-referential connections within the class. This allows modeling of hierarchical or self-referential structures within the system.

19.Experiment with an example how to name an association in UML with its guidelines.

Let's consider another example with classes "Author" and "Book." The association represents the relationship between authors and the books they have written.

Following the guidelines:

1. **Use Nouns**: Select descriptive nouns that represent the relationship between the classes.
2. **Be Clear and Specific**: Ensure the name clearly indicates the connection between authors and books.
3. **Avoid Ambiguity**: Choose a name that doesn't lead to confusion about the relationship.
4. **Be Concise**: Keep the name succinct while conveying the essential information.

Based on these guidelines, we can name the association between "Author" and "Book" as "Writes":

[Author] --- Writes --- [Book]

This name succinctly communicates that authors write books, adhering to the guidelines for naming associations in UML.

20.Distinguish sequence diagram and Use case diagram.

Use Case diagram

Use case diagrams show business use cases, actors, and the relationships between them. The relationships between actors and business use cases state that an actor can use a certain functionality of the business system. You will not find any information about how or in what chronological sequence these services are rendered

Sequence diagram

A sequence diagram can map a scenarios described by a use case in step by step detail to define how objects collaborate to achieve your application's goals.

21.How the domain model is illustrated?

A domain model is depicted using class diagrams in UML, where classes represent entities, attributes define their properties, and behaviors outline their actions. Associations between classes illustrate relationships and interactions within the domain, encapsulating how entities are related. This visual representation aids in understanding the structure and dynamics of the problem domain, facilitating effective communication among stakeholders during the software development process.

22.What is a Domain Model?

A domain model is a conceptual representation of the problem domain within which a software system operates. It defines the various entities, their attributes, relationships, and behaviors that exist within the domain.

23.What are the key ideas for Planning the Next Iteration?

Planning the next iteration involves evaluating what worked well and what needs improvement from the previous cycle. It's important to set achievable goals and prioritize tasks based on their importance and complexity. Additionally, estimating the effort required for each task and assigning responsibilities ensures that the team can effectively manage their workload. Finally, communicating the plan and any potential risks or dependencies with stakeholders helps to align expectations and ensure everyone is on the same page.

24.Define Association.

In object-oriented modeling, an association represents a relationship between two or more classes, indicating that instances of one class are somehow connected to instances of another class. Associations capture the connections or interactions between objects within a software system, allowing for the modeling of complex relationships and behaviors.