

US
N

--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--

Internal Assessment Test 2 – August 2023

Sub:	Software Engineering							Sub Code:	22MCA23
Date :	29/08/2023	Duration:	90 min's	Max Marks:	50	Sem:	II	Branch	MCA

Note : Answer FIVE FULL Questions, choosing ONE full question from each Module

		MARKS	OBE	
			CO	RB T
PART I				
1	What is Object Oriented Analysis and Design? Describe the stages of Object oriented methodology, used in software development. OR	[10]	CO1	L1
2	With respect to object oriented methodology, describe the following terms with examples: (a)Class (b) Polymorphism (c) Generalization (d) Abstract Class	[10]	CO1	L1
PART II				
3	Define the term model and mention the need for modeling. Describe the three models which support for modeling system in different viewpoints. OR	[10]	CO3	L1
4	Differentiate between Association, Aggregation and Composition with suitable examples.	[10]	CO3	L2

PART III

5 Explain Association Class, Qualified Association and Reflexive Association with example.

OR

- 6 a. Explain different kinds of Multiple Inheritance with example and how to represent it in a class diagram.
b. Describe Propagation of Operation with suitable example

PART IV

- 7 a. Discuss how to apply constraints in Class Diagram
b. What is an Association End? What are the properties of end?

OR

- 8 Write short notes on:
a). Enumeration (b) Multiplicity (c) Reification (d) Visibility

PART V

9 Draw class diagram for Railway Reservation System.

OR

10 Draw Class diagram for Online Shopping System.

[10]	CO1	L2
[6+4]	CO3	L2
[4+6]	CO4	L2
[10]	CO4	L1
[10]	CO5	L3
[10]	CO5	L3

Solution

PART I

What is Object Oriented Analysis and Design? Describe the stages of Object oriented methodology, used in software development.

Ans. Object-oriented modeling and design is a way of thinking about problems using models organized around real world concepts. The fundamental construct is the object, which combines both data structure and behavior.

System conception: Software development begins with business analysis or users conceiving an application and formulating tentative requirements.

- **Analysis:** The analyst scrutinizes and rigorously restates the requirements from the system conception by constructing models. The analysis model is a concise, precise abstraction of what the desired system must do, not how it will be done.

The analysis model has two parts-

Domain Model- a description of real world objects reflected within the system.

Application Model- a description of parts of the application system itself that are visible to the user.

E.g. In case of stock broker application-

Domain objects may include- stock, bond, trade & commission.

Application objects might control the execution of trades and present the results.

- **System Design:** The development teams devise a high-level strategy- The System Architecture- for solving the application problem. The system designer should decide what performance characteristics to optimize, chose a strategy of attacking the problem, and make tentative resource allocations.

- **Class Design:** The class designer adds details to the analysis model in accordance with the system design strategy. His focus is the data structures and algorithms needed to implement each class.

- **Implementation:** Implementers translate the classes and relationships developed during class design into a particular programming language, database or hardware. During implementation, it is important to follow good software engineering practice.

OR

1 With respect to object oriented methodology, describe the following terms with examples:

(a) Class (b) Polymorphism (c) Generalization (d) Abstract Class

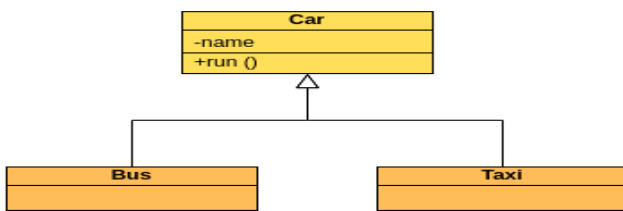
a. **Class:** A *class* is an abstraction that describes properties important to an application.

Each object is said to be an *instance* of its class.

An object has its own value for each attribute but shares the attribute names and operations with other instances of the class.

B **Polymorphism:** Polymorphism means that the same operation may behave differently for different classes. For E.g. move operation behaves differently for a pawn than for the queen in a chess game.

c. **Generalization: Inheritance** is also called **generalization** and is used to describe the relationship between parent and child classes. A parent class is also called a base class, and a subclass is also called a derived class. In the inheritance relationship, the subclass inherits all the functions of the parent class, and the parent class has all the attributes, methods, and subclasses. Subclasses contain additional information in addition to the same information as the parent class.



Abstract Class: An *abstract class* is a class that has no direct instances but whose descendant classes have direct instances. A *concrete class* is a class that is instantiable; that is, it can have direct instances. A concrete class may have abstract subclasses (but they, in turn, must have concrete descendants). Only concrete classes may be leaf classes in an inheritance tree.

PART II

2 Define the term model and mention the need for modeling. Describe the three models which support for modeling system in different viewpoints.

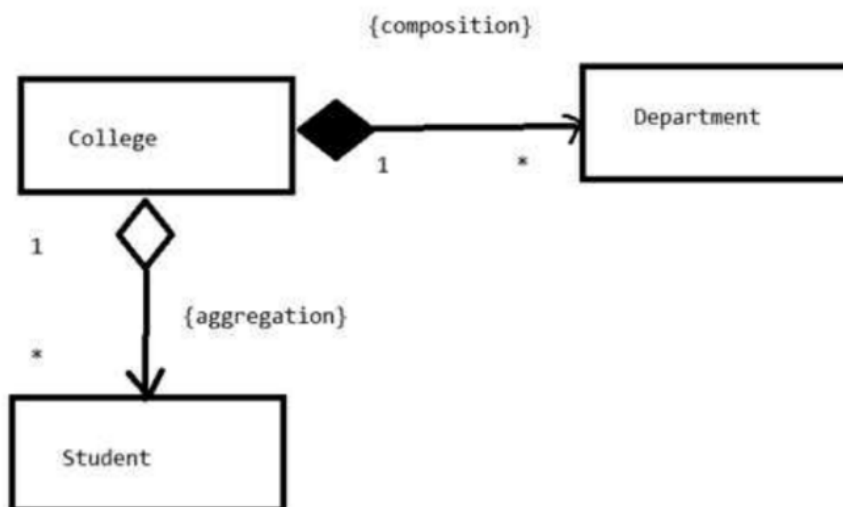
Ans. A *model* is an abstraction of something for the purpose of understanding it before building it. Models serve several purposes.

- **Testing a physical entity before building it.** The medieval masons did not know modern physics, but they built scale models of the Gothic cathedrals to test the forces on the structure. Engineers test scale models of airplanes, cars, and boats in wind tunnels and water tanks to improve their dynamics. Both physical models and computer models are usually cheaper than building a complete system and enable early correction of flaws.
- **Communication with customers.** Architects and product designers build models to show their customers. Mock-ups are demonstration products that imitate some or all of the external behavior of a system.
- **Visualization.** Storyboards of movies, television shows, and advertisements let writers see how their ideas flow. They can modify awkward transitions, dangling ends, and unnecessary segments before detailed writing begins. Artists' sketches let them block out their ideas and make changes before committing them to oil or stone.
- **Reduction of complexity.** Perhaps the main reason for modeling, which incorporates all the previous reasons, is to deal with systems that are too complex to understand directly. The human mind can cope with only a limited amount of information at one time. Models reduce complexity by separating out a small number of important things to deal with at a time.

OR

3 Differentiate between Association, Aggregation and Composition with suitable examples.

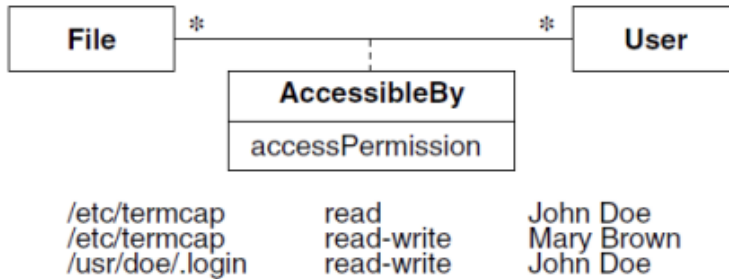
Ans. Aggregation is a stronger form of association. An association is a link connecting two classes. In UML, a link is placed between the “whole” and the “parts” classes with a diamond head attached to the “whole” class to indicate that this association is an aggregation. Composition is really a strong form of aggregation. Composition has only one owner. Composition cannot exist independent of their owner. Composition lives or dies with their owner. It is represented using a filled diamond head. The main differentiator between aggregation and composition is the lifecycle dependence between whole and part. In aggregation, the part may have an independent lifecycle, it can exist independently. When the whole is destroyed the part may continue to exist. Composition is a stronger form of aggregation. The lifecycle of the part is strongly dependent on the lifecycle of the whole. When the whole is destroyed, the part is destroyed too. Aggregation Example A car has many parts. A part can be removed from one car and installed into a different car. If we consider a salvage business, before a car is destroyed, they remove all saleable parts. Those parts will continue to exist after the car is destroyed. Composition Example For example, a building has rooms. A room can exist only as part of a building. The room cannot be removed from one building and attached to a different one. When the building ceases to exist so do all rooms that are part of it.



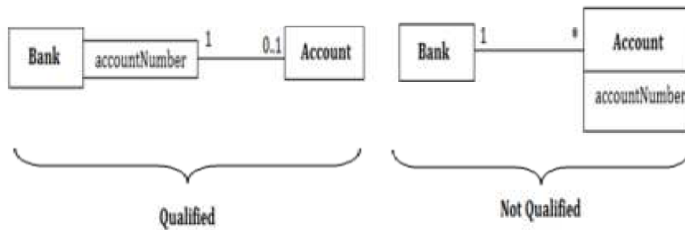
PART III

4 Explain Association Class, Qualified Association and Reflexive Association with example.

Ans. An association class is an association that is also a class. Like the links of an association, the instances of an association class derive identity from instances of the constituent classes. Like a class, an association class can have attributes and operations and participate in associations. You can find association classes by looking for adverbs in a problem statement or by abstracting known values. In Figure 3.17, accessPermission is an attribute of AccessibleBy. The sample data at the bottom of the figure shows the value for each link. The UML notation for an association class is a box (a class box) attached to the association by a dashed line.



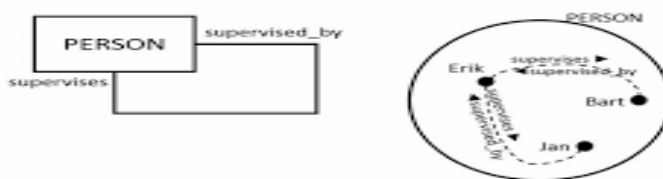
A **qualified association** is an association in which an attribute called the **qualified** disambiguates the objects for a “many” association end. It is possible to define qualifier for one-to-many & many-to-many association.



Qualified association: Qualification increases the precision of a model

UNARY ASSOCIATION

• A **Unary** associations connects a class with itself



OR

5 A. Explain different kinds of Multiple Inheritance with example and how to represent it in a class diagram.

Ans.

The most common form of multiple inheritance is from sets of disjoint classes. Each subclass inherits from one class in each set. In Figure 4.15 *FullTimeIndividualContributor* is both *FullTimeEmployee* and *IndividualContributor* and combines their features. *FullTimeEmployee* and *PartTimeEmployee* are disjoint; each employee must belong to exactly one of these. Similarly, *Manager* and *IndividualContributor* are also disjoint and each employee must be one or the other. The model does not show it, but we could define three additional combinations: *FullTimeManager*, *PartTimeIndividualContributor*, and *PartTimeManager*. The appropriate combinations depend on the needs of an application.

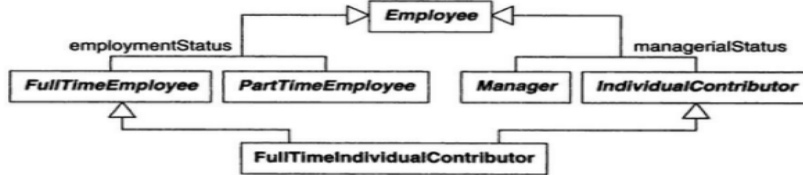


Figure 4.15 Multiple inheritance from disjoint classes. This is the most common form of multiple inheritance.

Multiple inheritance can also occur with overlapping classes. In Figure 4.16, *AmphibiousVehicle* is both *LandVehicle* and *WaterVehicle*. *LandVehicle* and *WaterVehicle* overlap, because some vehicles travel on both land and water. The UML uses a constraint (see Section 4.9) to indicate an overlapping generalization set; the notation is a dotted line cutting across the affected generalizations with keywords in braces. In this example, *overlapping* means that an individual vehicle may belong to more than one of the subclasses. *Incomplete* means that all possible subclasses of vehicle have not been explicitly named.

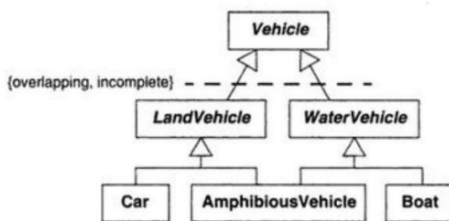


Figure 4.16 Multiple inheritance from overlapping classes. This form of multiple inheritance occurs less often than with disjoint classes.

b. Describe Propagation of Operation with suitable example.

Ans. *Propagation* (also called *triggering*) is the automatic application of an operation to a network of objects when the operation is applied to some starting object. For example, moving an aggregate moves its parts; the move operation propagates to the parts. Propagation of operations to parts is often a good indicator of aggregation.

You can indicate propagation on class models with a small arrow indicating the direction and operation name next to the affected association.

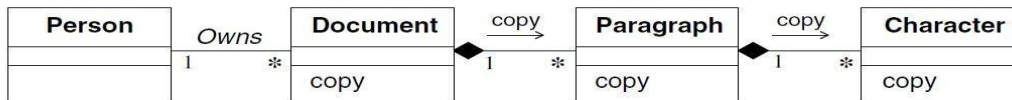


Figure 4.11 Propagation. You can propagate operations across aggregations and compositions.

PART IV

a. a. Discuss how to apply constraints in Class Diagram

- Ans. Constraint is a condition involving model elements, such as objects, classes, attributes, links, associations, and generalization sets.
- Class models capture many Constraints through their very structure. For example, the semantics of generalization imply certain structural constraints.
- The UML defines the following keywords for generalization.
 - **Disjoint:** The subclasses are mutually exclusive. Each object belongs to exactly one of the subclasses.
 - **Overlapping:** The subclasses can share some objects. An object may belong to more than one subclass.
 - **Complete:** The generalization lists all the possible subclasses.
 - **Incomplete:** The generalization may be missing some subclasses.
- Multiplicity is a constraint on the cardinality of a set. Multiplicity for an association restricts the number of objects related to a given object.
- Multiplicity for an attribute specifies the number of values that are possible for each instantiation of an attribute.

b. What is an Association End? What are the properties of end?

an association end is an end of an association. A binary association has two ends, a ternary association has three ends, and so forth. Following are few properties of associations:

- Association end name- An association end may have a name. The names within the proper context.
- Multiplicity- You can specify multiplicity for each association end.
- Ordering- the objects have an explicit order.
 - Bags and sequences. The objects for a “many” association end can also be a bag or sequence.
 - Qualification. One or more qualifier attributes can disambiguate the objects for a “many” association end.

Association ends have some additional properties.

- Aggregation. The association end may be an aggregate or constituent part.
 - Changeability. This property specifies the update status of an association end. The possibilities are changeable (can be updated) and readonly (can only be initialized).
 - Navigability. Conceptually, an association may be traversed in either direction
 - Visibility. Similar to attributes and operations, association ends may be public, protected, private, or package

OR

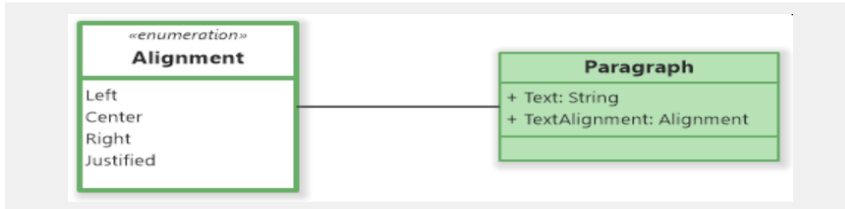
9. Write short notes on:

a). Enumeration (b) Multiplicity (c) Reification (d) Visibility

a). An **enumeration** is a data type that has a finite set of values.

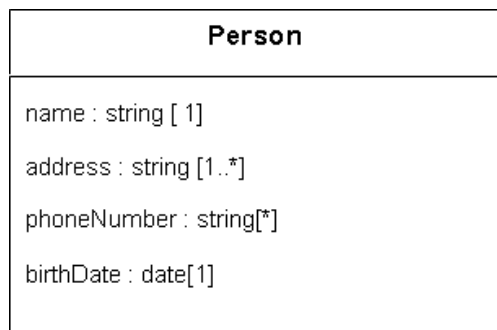
When constructing a model, one should carefully note enumerations, because they often occur & are important to users.

Enumerations are also significant for an implementation; you may display the possible values with a pick list & you must restrict data to the legitimate values.



b. Multiplicity

- Multiplicity is a collection on the cardinality of a set, also applied to attributes (database application).
- Multiplicity of an attribute specifies the number of possible values for each instantiation of an attribute. i.e., whether an attribute is mandatory ([1]) or an optional value ([0..1] or * i.e., null value for database attributes) .
- Multiplicity also indicates whether an attribute is single valued or can be a collection.



c. Reification

- Reification is the promotion of something that is not an object into an object.
- Reification is a helpful technique for Meta applications because it lets you shift the level of abstraction.
- On occasion it is useful to promote attributes, methods, constraints, and control information into objects so you can describe and manipulate them as data.

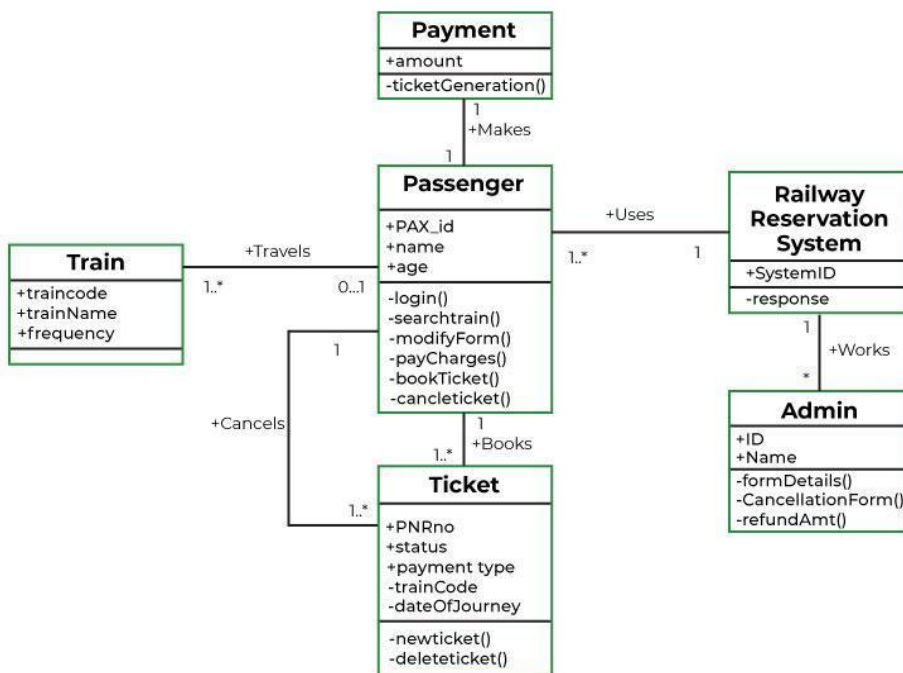
As an example of reification, consider a database manager. A developer could write code for each application so that it can read and write from files. Instead, for many applications, it is better idea to reify the notion of data services and use a database manager.

d. Visibility

- Visibility refers to the ability of a method to reference a feature from another class and has the possible values of *public*, *protected*, *private*, and *package*.
- Any method can access **public** features.
- Only methods of the containing class and its descendants via inheritance can access **protected** features.
- Only methods of the containing class can access **private** features.
- Methods of classes defined in the same package as the target class can access **package** features
- The UML denotes visibility with a prefix. **+||** \in **public**, **-||** \in **private**, **-#||** \in **protected**, **-~||** \in **package**. Lack of a prefix reveals no information about visibility.

PART V

10. Draw class diagram for Railway Reservation System.



OR

11. Draw Class diagram for Online Shopping System.

CUSTOMER VIEWING PRODUCT

