CMR INSTITUTE OF TECHNOLOGY

USN | | | | | | | | | | |

Internal Assessment Test 2 – September 2024

Sub:	Software Engineering							Sub Code:	22MCA23
Date:	12/09/2023	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	RBT
	PART I			
1	What is OO development? List and explain various themes that are supported by OO technology.	[10]	CO1	L1
	OR			
2	With respect to object oriented methodology, describe the following terms with examples: (a) Class (b) Polymorphism (c) Meta Data (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.	[10]	CO3	L1
	OR			
4	Differentiate between Generalization, Aggregation and Composition with examples.	[10]	CO3	L2
	PART III			
5	Explain Reification, Qualified Association and Reflexive Association with example.	[10]	CO1	L2
	OR			
6	a. Explain N-ary association with example b. Describe Propagation of Operation with suitable example	[6+4]	CO3	L2
	PART IV			
7	a. Discuss how to apply constraints in Class Diagram b. What is an Association End? What are the properties of end?	[4+6]	CO4	L2
	OR			
8	Write short notes on: a). Enumeration (b) Multiplicity (c) Visibility	[10]	CO4	L1
	PART V			
9	Draw class diagram for Hospital Management System.	[10]	CO5	L3
	OR			
10	Draw Class diagram for Online Shopping System.	[10]	CO5	L3

Solution

PART I

1. What is OO development? List and explain various themes that are supported by OO technology.

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.

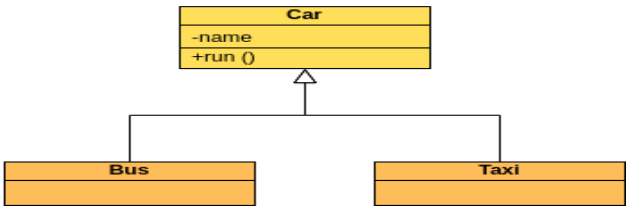
OR

2. 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

3. **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.

Different Types of Models:

The different types of modeling techniques are:

i) Class Model: It describes the structure of objects in a system – their identity, their relationships to other objects, their attributes and their operations. The goal of constructing the class model is to capture those concepts from the real world that are important to an application. Class diagram express the class model.

ii) State Model: It describes those aspects of objects concerned with time and the sequencing of operations – events that mark changes, state that define the context for events, and the organization of events and states. State diagram expresses the state model.

iii) Interaction Model: It describes interactions between objects – How individual objects collaborate to achieve the behavior of the system as a whole. Use case, sequence diagram and activity diagram documents the interaction model.

OR

4. **Differentiate between Generalization, Aggregation and Composition with examples.**

Ans. In Object-Oriented Analysis and Design (OOAD), **generalization**, **aggregation**, and **composition** are essential relationships that define how objects and classes interact and relate to one another. Understanding these relationships helps in designing a robust and maintainable system

Generalization:

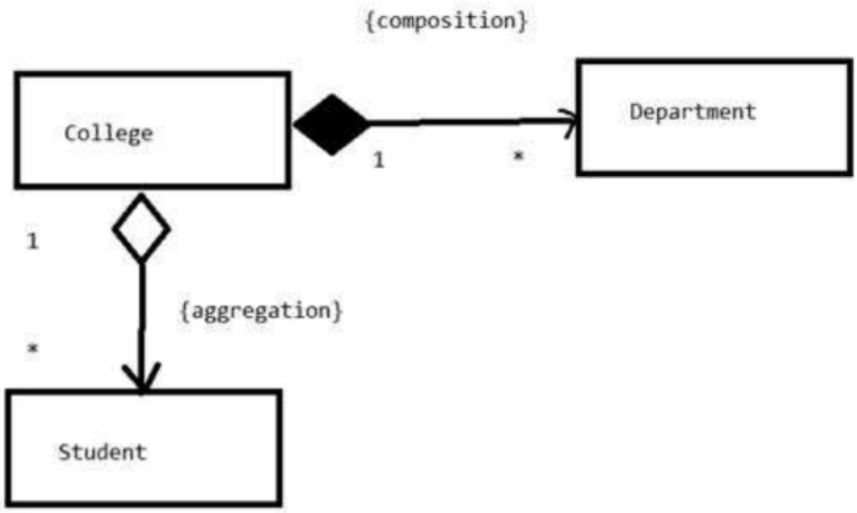
- Describes an "is-a" relationship.
- Inheritance hierarchy where subclasses inherit from a superclass.
- Subclass can exist independently of the superclass but in a general-to-specific context.

Aggregation:

- Describes a "has-a" relationship with independent lifecycle.
- Represents a whole-part relationship where the part can exist outside the whole.
- The destruction of the whole does not affect the existence of the part.

Composition:

- Describes a "contains-a" relationship with dependent lifecycle.
- Represents a stronger whole-part relationship where the part cannot exist outside the whole.
- The destruction of the whole results in the destruction of the parts



PART III

5. **Explain Reification, Qualified Association and Reflexive Association with example.**

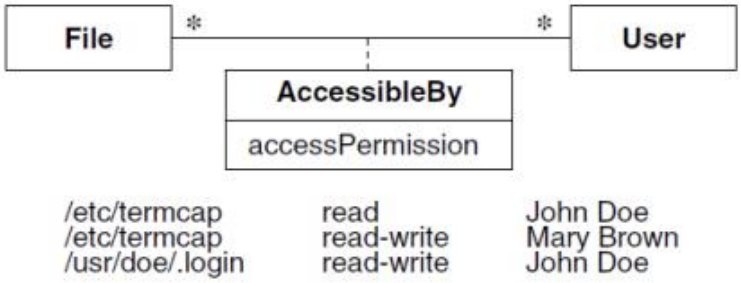
Ans. In class modeling, **reification** refers to the process of turning an abstract relationship or property between classes into a concrete class itself. This is useful when the relationship or property has attributes or behaviors of its own that need to be captured and managed.

Reification is used when an association between two entities has more information or significance than can be expressed directly through the association alone. By converting the relationship into a separate class, we can better model and represent additional details, constraints, and behaviors associated with that relationship.

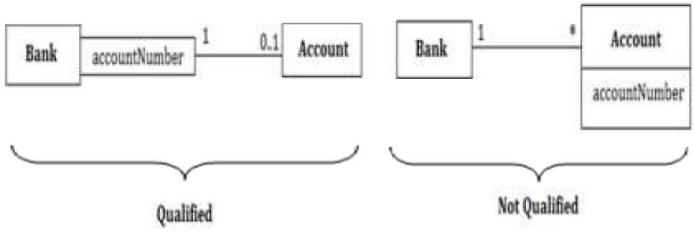
Example in Class Modeling

Scenario: File and User Relationship

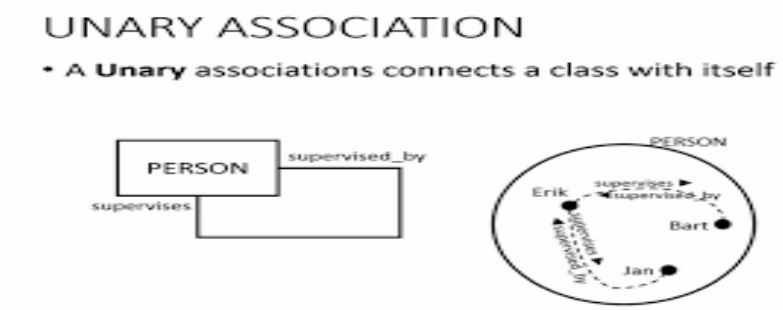
In a university system, we have two main entities: `Student` and `Course`. A student can enroll in many courses, and a course can have many students. This represents a many-to-many relationship.



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



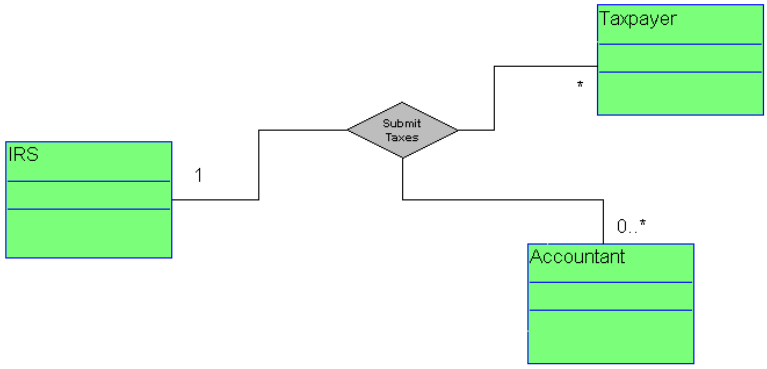
OR

6. **a** Explain N-ary association with example

Ans.

N-ary association in class modeling refers to a relationship between three or more classes. It is used to represent a situation where multiple entities are related in a meaningful way, and the relationship itself involves more than two classes. Unlike binary associations (which connect only two classes), n-ary associations involve multiple classes simultaneously.

Example of an N-ary Association



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.

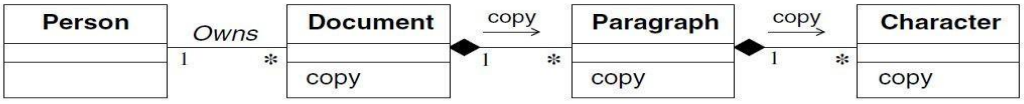


Figure 4.11 Propagation. You can propagate operations across aggregations and compositions.

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 keyword s 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

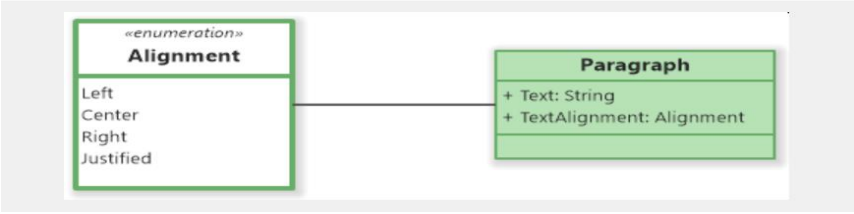
9. Write short notes on:

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

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

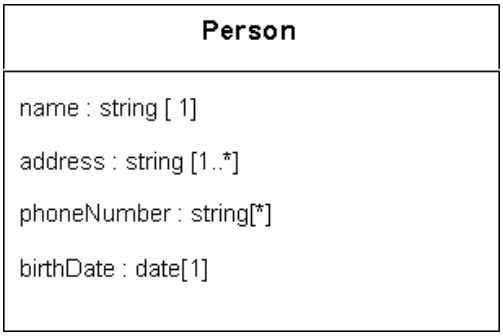
When constructing a model, one should carefully note enumerations, because they often occur & are important to users.

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.



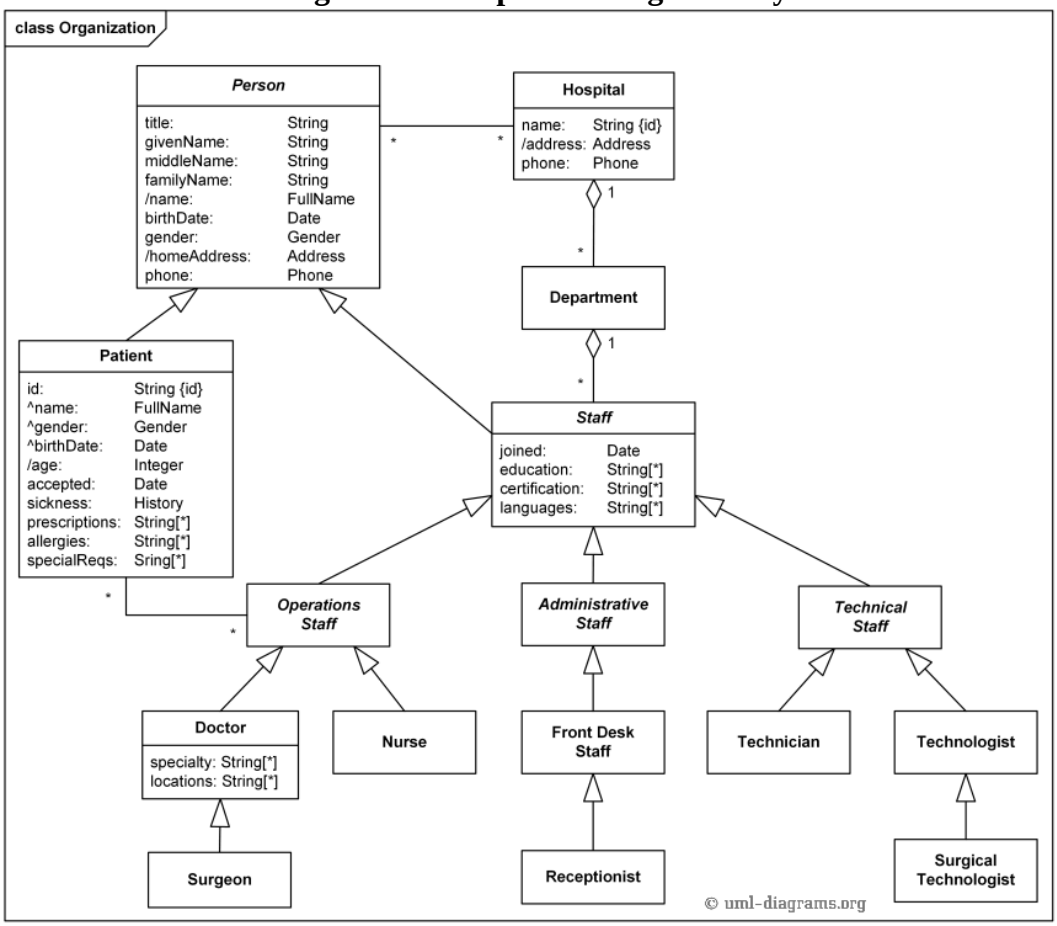
c. Visibility

- 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. **—+ public, —- private,**

—#‖protected, —~‖ package. Lack of a prefix reveals no information about visibility.

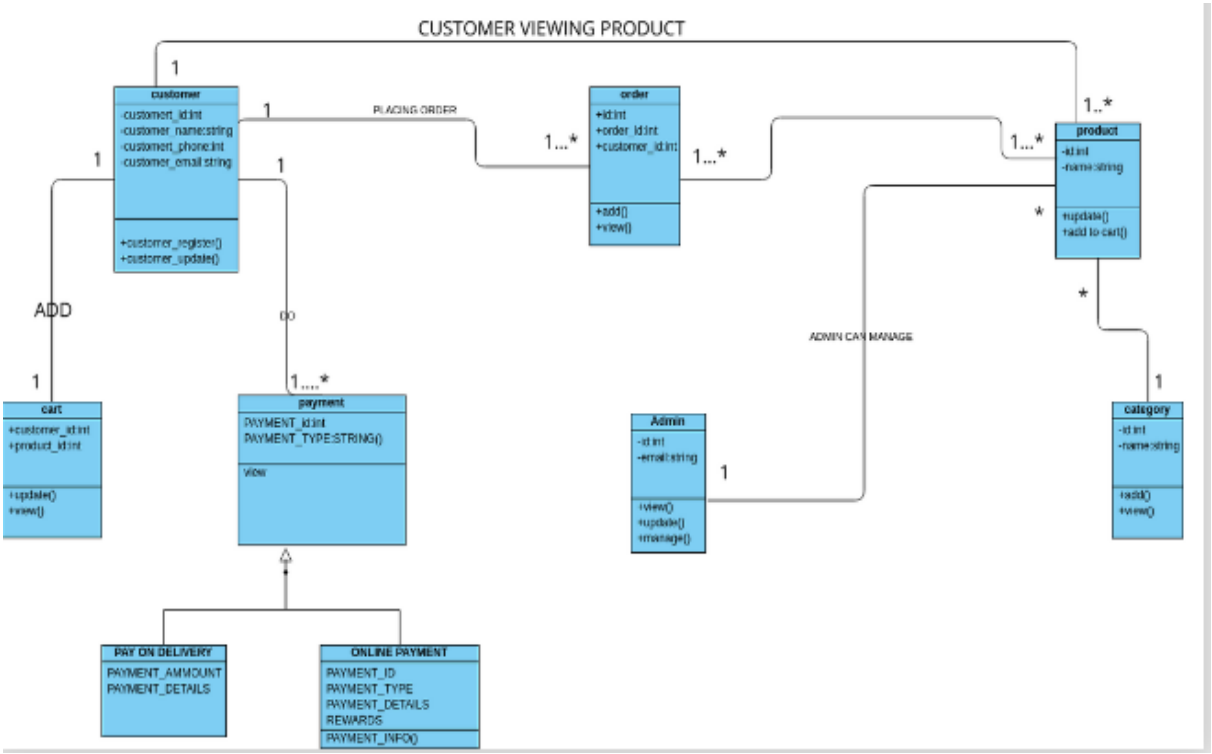
PART V

10. **Draw class diagram for Hospital Management System**.



OR

11. Draw Class diagram for Online Shopping System.


CUSTOMER VIEWING PRODUCT
customer
-customer_id:int
-customer_name:string
-customer_phone:int
-customer_email:string
+customer_register()
+customer_update()
PLACING ORDER
order
+id:int
+order_id:int
+customer_id:int
+add()
+view()
product
-id:int
-name:string
+update()
+add to cart()
ADD
cart
+customer_id:int
+product_id:int
+update()
+view()
payment
PAYMENT_id:int
PAYMENT_TYPE:STRING()
view
ADMIN CAN MANAGE
Admin
-id:int
-email:string
+view()
+update()
+manage()
category
-id:int
-name:string
+add()
+view()
PAY ON DELIVERY
PAYMENT_AMMOUNT
PAYMENT_DETAILS
ONLINE PAYMENT
PAYMENT_ID
PAYMENT_TYPE
PAYMENT_DETAILS
REWARDS
PAYMENT_INFO()