SRI KRISHNA COLLEGE OF TECHNOLOGY
An Autonomous Institution
(Approved by AICTE and affiliated to Anna University)
Accredited by NAAC with "A" grade

Coimbatore, Tamil Nadu

23CS202 - Object Oriented Analysis and Design

QUESTION BANK

1. Differentiate Sequence and Collaboration diagrams

Sequence Diagrams	Collaboration Diagrams
The sequence diagram represents the UML, which is used to visualize the sequence of calls in a system that is used to perform a specific functionality.	The collaboration diagram also comes under the UML representation which is used to visualize the organization of the objects and their interaction.
The sequence diagram are used to represent the sequence of messages that are flowing from one object to another.	The collaboration diagram are used to represent the structural organization of the system and the messages that are sent and received.
The sequence diagram is used when time sequence is main focus.	The collaboration diagram is used when object organization is main focus.
The sequence diagrams are better suited of analysis activities.	The collaboration diagrams are better suited for depicting simpler interactions of the smaller number of objects.

2. Design an abstract class of a Heater with the members of location ,status and activities of turn on, off, read status

Heater
Attributes: String Location; Bool Status;

Functions:
 	
void turnoff();
void turnon();
void status();

3. How do the principles of abstraction and encapsulation in software development complement each other, and how does their integration with modularity result in a synergistic effect on the design and functionality of a software system?

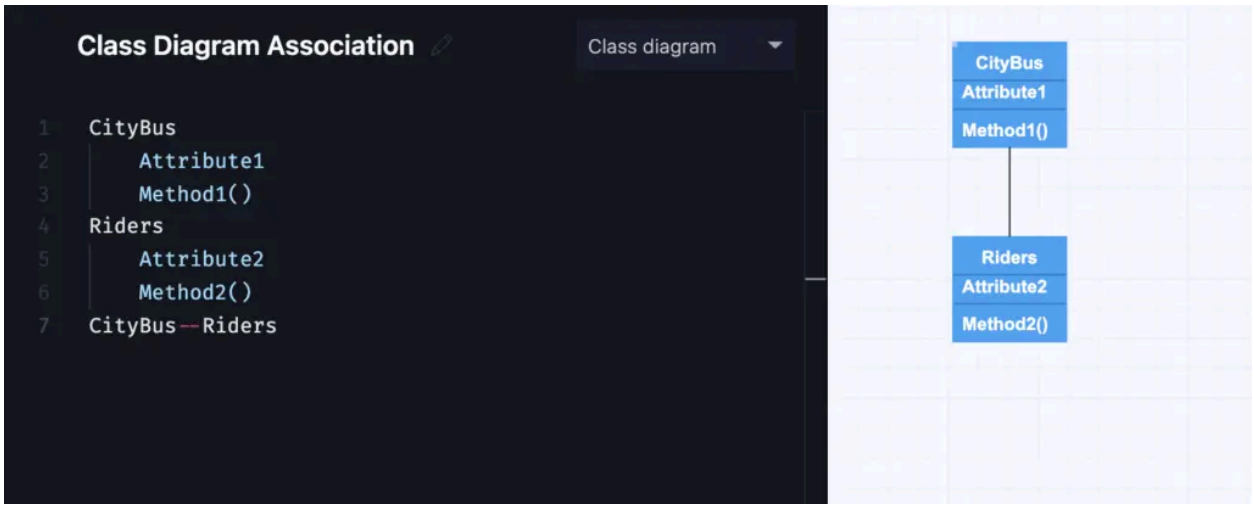
Abstraction and Encapsulation are complementary concepts. Through encapsulation only we are able to enclose the components of the object into a single unit and separate the private and public members. It is through abstraction that only the essential behaviors of the objects are made visible to the outside world. So we can say that encapsulation is the way to implement data abstraction. For example in class Student only the essential information like roll no name date_of_birth course etc. of the student will be visible. The secret information like calculation of grades, allotment of examiners etc. will be hidden.

4. List any five inception artifacts.
   - Vision document. ...
   - Use case model. ...
   - Supplementary specification. ...
   - Business model. ...
   - Initial project plan. ...
   - Risk list. ...
   - Prototype.

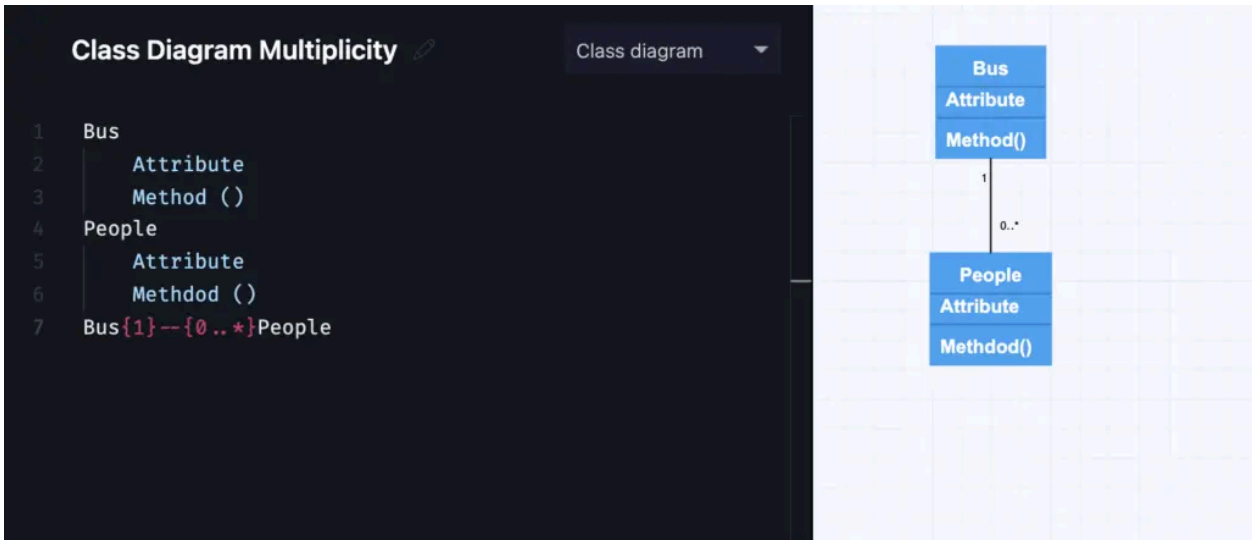
5. Differentiate association and cardinality in OOAD.

Association

This simply means that one model element is linked in some way to another model element. The association indicates the nature and rules that govern the relationship. The basic way to represent association is with a line between the elements.



An association relationship between elements can also have cardinality, for instance, one-to-one, one-to-many, many-to-one, or many-to-many, zero-to-many, and so on. This can also be shown in a label on the line.

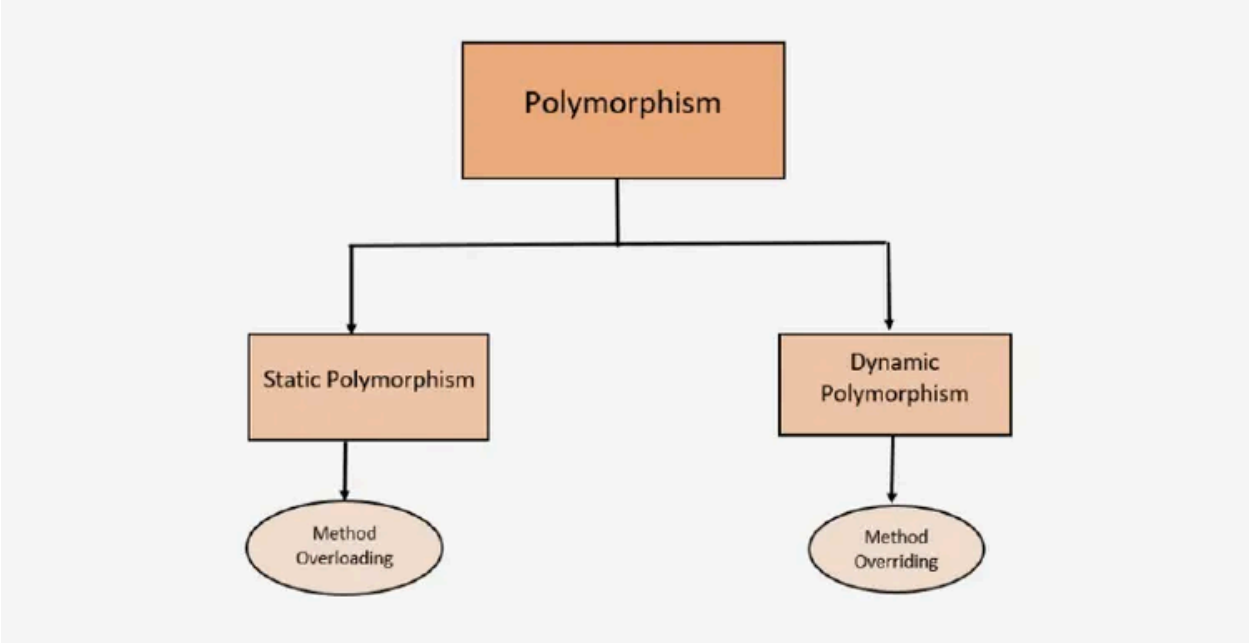


6. Indicate the necessity of polymorphism with an example.

Poly means “many” and morph means “form”. In the context of object- oriented systems, it means objects that can take on or assume many different forms. Polymorphism means that the

same operation may behave differently on different classes. Booch [1-3] defines polymorphism as the relationship of objects of many different classes by some common superclass

Polymorphism allows coders to write code that can work with objects of multiple classes in a generic way without knowing the specific class of each object.



There are two types of polymorphism in OOPs:

1. Static
2. Dynamic

1. Static Polymorphism

In object-oriented programming polymorphism languages, one can achieve static polymorphism by employing the overloading technique. The programmer would be able to use a variety of ways with this approach. Despite sharing names, they have different parameters. Overloading methods are an example of static polymorphism. Static polymorphism requires a few certain criteria to exist. As follows:

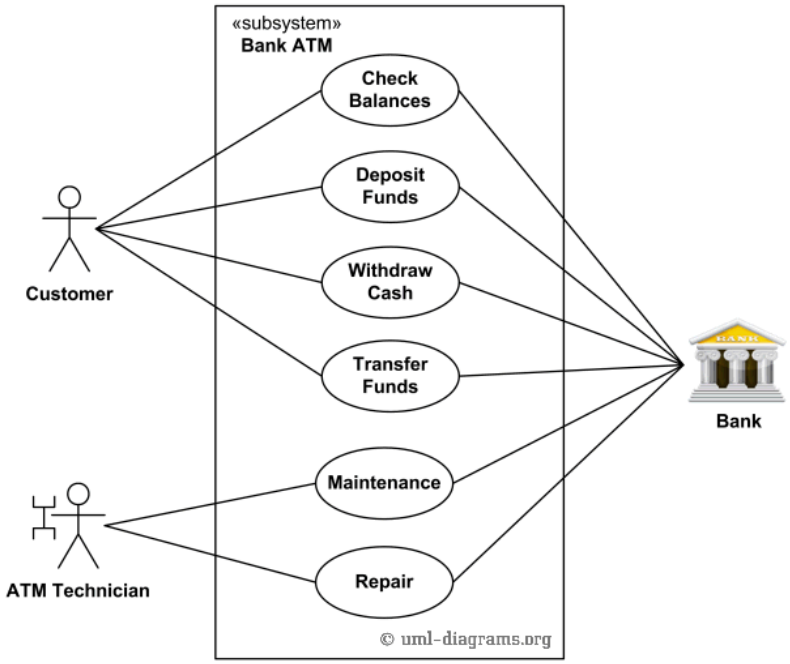
- The kinds of each parameter would need to be distinct.
- It's possible that the parameter's order needs to change.

- One method would need to have a different set of parameters than the other method.

Dynamic Polymorphism

Another name for this procedure is runtime polymorphism. Under this procedure, a single call to an overridden method is resolved while the application is running. The best example of runtime polymorphism is method overriding, where the compiler does not handle the call.

7. Draw the Use case Diagram for ATM Application.



8. How does functional semantics differ from time and space semantics of an operation?

Functional semantics describe what an operation does, such as a sort method arranging a list in ascending order. Time and space semantics describe the efficiency, like the sort method's time complexity (e.g., O(n log n)) and space complexity (e.g., O(n) for additional memory usage).

9.Compare the concepts of Object Oriented analysis, Object Oriented Design and Object Oriented Programming with suitable example.

OOA: we find and describe business objects or concepts in the problem domain
OOD: we define how these software objects collaborate to meet the requirements. Attributes and methods.
OOP: Implementation: we implement the design objects in, say, Java, C++, C#, etc.

Object-Oriented Analysis (OOA) vs Object-Oriented Design (OOD)

Object-Oriented Analysis (OOA)	**Object-Oriented Design (OOD)**
Concerned with understanding the problem and its requirements.	Focuses on transforming requirements into a detailed design.
Involves gathering and defining what the software should do.	Involves how the software will do what is needed.
Emphasizes capturing real-world concepts into a software model.	Emphasizes translating the captured model into a technical solution.
Primarily uses techniques like use cases, class diagrams, and behavior diagrams to understand the problem domain.	Involves architecture, detailed class design, object interactions, and design patterns.

Identifies entities, their relationships, and their interactions in the problem domain.	Refined identified entities into detailed classes, their attributes, methods, and interactions.
Focuses on stakeholder's needs, domain understanding, and requirements gathering.	Focuses on implementation details, coding strategies, and testing plans.
Provides the foundation for creating a model of the problem domain.	Provides the foundation for the implementation and coding of the solution.

9. Identify the important attributes to consider when designing complex systems

- Hierarchic Structure
- Relative Primitives
- Separation of Concerns
- Common Patterns
- Stable Intermediate Forms

10. How can we differentiate between actions that define a class and those that don't?

Actions that define a class are those central to its core responsibilities and directly interact with its attributes, like deposit() and withdraw() in a BankAccount class. Actions that don't define a class are peripheral and not essential to its primary functionality, like printStatement() or sendNotification().

11. “Abstraction and encapsulation are complementary concepts; abstraction, encapsulation and modularity are synergistic concepts” Justify this statement

Ans: Abstraction and Encapsulation are complementary concepts. Through encapsulation only we are able to enclose the components of the object into a single unit and separate the private and public members. It is through abstraction that only the essential behaviors of the objects are made visible to the outside world.

12. Distinguish between strong/weak typing and dynamic/ static typing

Strong/week
Weakly-typed languages make conversions between unrelated types *implicitly*; whereas, **strongly-typed** languages don’t allow implicit conversions between unrelated types.

Ex: Python is a strongly-typed language:

```
var = 21;           #type assigned as int at runtime.
var = var + "dot";   #type-error, string and int cannot be concatenated.
print(var);
```

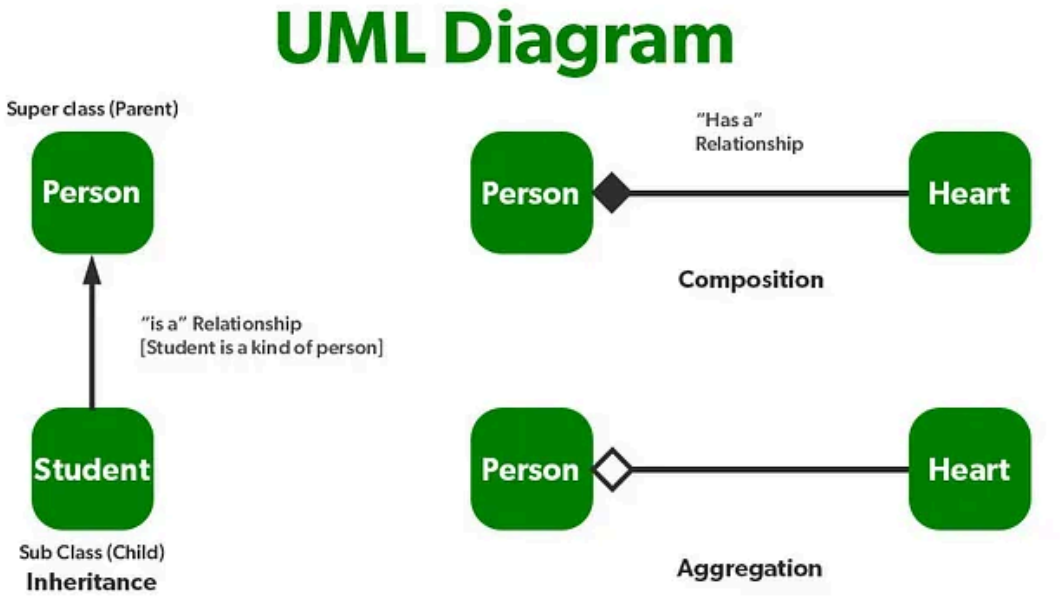
JavaScript is a weakly-typed language:

```
value = 21;
value = value + "dot";
console.log(value);
/*
This code will run without any error. As Javascript
is a weakly-typed language, it allows implicit conversion
between unrelated types.
*/
```

13. Define Association, Aggregation, Composition

- Association(using relationship): The objects created and destroyed independently and represented as one-to-one, one-to-many, or many-to-many (also known as cardinality).
- Aggregation(HAS-A relationship): parent can exist without child and a child ***can*** exist independently of the parent and this involve a one-to-one, one-to-many, or many-to-many relationship between the participating objects.

- composition(“death” relationship, PART-OF): parent destroyed, child dies
- Inheritance : is-a relationship.



14. Which of the OO concept relate to the following:

a. Every square is a rectangle and every rectangle is a quadrilateral

b. A programming language supports the struct or a record data type

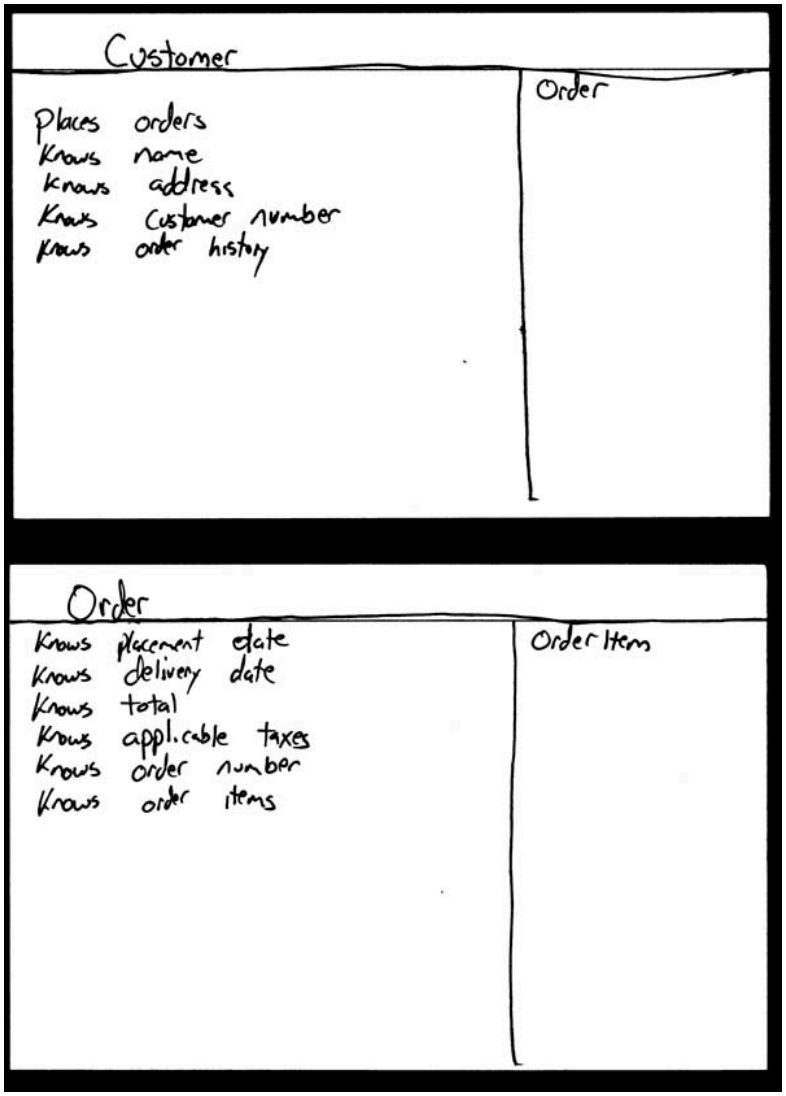
c. A program calls a routine called sort and passes it to the address of an input integer array, the address of an output integer array, and the number of elements.

d. In (c) above, the caller can additionally specify the sort method to be bubble sort or quick sort

Ans :

a. Inheritance
b. User defined datatype
c. Function
d. Polymorphism

15. Design a CRC card for order processing of an online shopping



16. Consider a smart home system capable of controlling various functionality such as switching on light, adjusting the AC temperature inside a room. The system employs different sensors to understand and adjust to room conditions and react accordingly. In particular, the system has deployed photo resisters to detect the light condition of the room to switch on the lights and a thermostat to detect the temperature changes to control the AC

Identify the IS A relationship from the below options
Thermostat , Sensor
Photo resister, Sensor
Light, Sensor
AC, Sensor

Ans:
Thermostat , Sensor
Photo resister, Sensor

Identify the HAS A relationship from the below options

AC, Controller
Light, Thermostat
Light, Switch
AC, Photo resistor

Ans: AC, Controller
Light, Switch

17 . Sketch the Classes Responsibilities and Collaborators involved in the class **Animal**

. 18. Evaluate and Name the UML diagrams used for the following:

a) Modelling Requirements

Ans : Usecase Diagram

b) Modelling Workflows

Activity Diagram

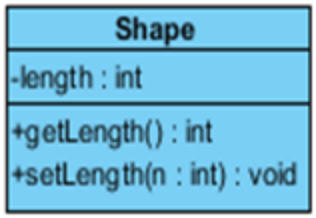
c) Modelling behaviour of an object.

Ans :Timing diagrams are very similar to sequence diagrams. They represent the behavior of objects in a given time frame

d) Interaction between groups of objects.

Ans: Collaboration Diagram and Sequence Diagram

19.Generate the Java code for the given UML class notation.




```
class shape
{
int length;

int getLength( )
{
….
….
}
void getLength(int )
{
….
….
}
}
```

17. Compare and contrast the Unified Process with other software development methodologies, such as Agile or Waterfall, in terms of their approach to manage the project requirements and adapt the change in customer needs

18. Highlight the criteria to assess the quality of a class and its corresponding objects in object-oriented programming. Provide examples of metrics or indicators you would consider and discuss how they contribute in determining the effectiveness and efficiency of the design.

19. Draw an abstraction diagram for this scenario of object modelling. Also explain the major, minor elements of object model for the problem of calculating an insurance premium, which is a function of age, health, and gender. The method of calculating the premium is confidential to the company

20. Design the use case diagram and discover the users and actors of Library system

21. Discuss the rules for identifying the classes and objects relevant to a particular problem. Also discuss the identification of key abstractions and mechanisms

22. Design the use case diagram for the coffee Vending machine dispenses coffee to customers.

23. Design a set of use cases for an online shopping application, Illustrate how you would apply them to design the system architecture.