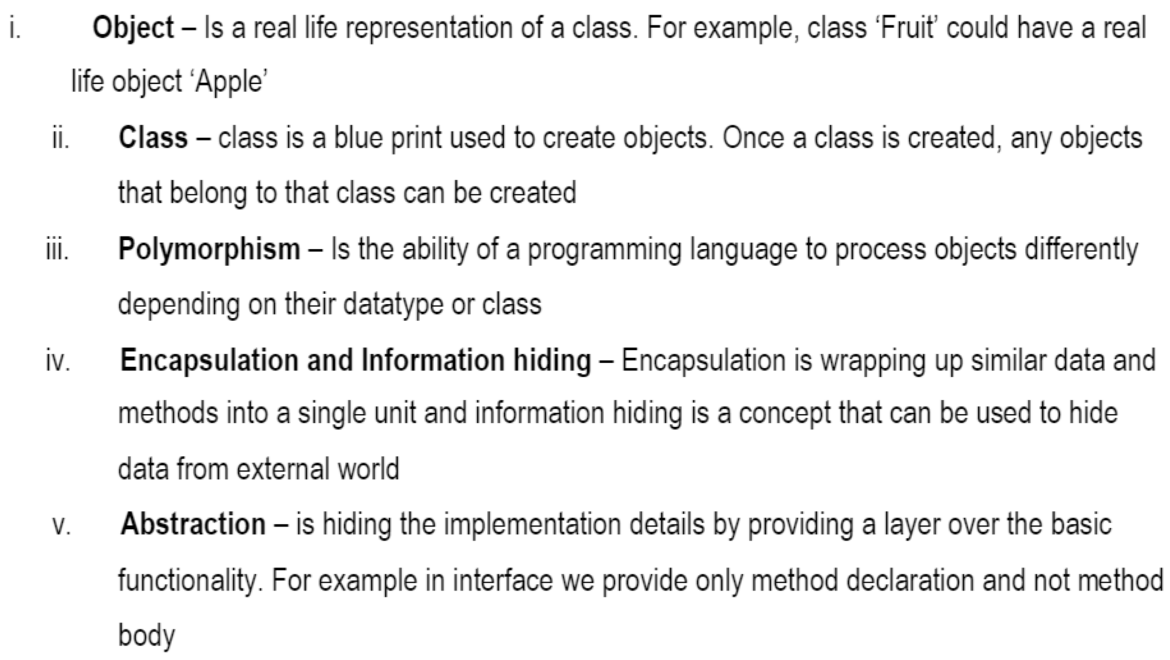
OOSD Question and Answers

Q1. Discuss the major differences between object-oriented development and structured approach.

Answer:

1. Structured programming design focuses on process while object-oriented design is focused on data.
2. Structured programming is less secure as there is no way of hiding data while OOP is more secure since it has data hiding capabilities.
3. Structured programming provides less reusability while OOP provides more code reusability.

Q2. Using diagrams, explain the following concepts as applied in Object Oriented Analysis and Design.



i. **Object –** Is a real life representation of a class. For example, class 'Fruit' could have a real life object 'Apple'

ii. **Class –** class is a blue print used to create objects. Once a class is created, any objects that belong to that class can be created

iii. **Polymorphism –** Is the ability of a programming language to process objects differently depending on their datatype or class

iv. **Encapsulation and Information hiding –** Encapsulation is wrapping up similar data and methods into a single unit and information hiding is a concept that can be used to hide data from external world

v. **Abstraction –** is hiding the implementation details by providing a layer over the basic functionality. For example in interface we provide only method declaration and not method body

Q3. Explain use case modelling with an example?

Use cases are descriptions of the functionality of the system from the users' perspective. Use case diagrams are used to show the functionality that the system will provide and to show which users will communicate with the system in some way to use that functionality.

The following figure shows an example of a use case diagram.

Purpose of Use case diagrams

The use case model is called as the requirements model; which also include a problem domain object model and user interface descriptions in this requirements model.

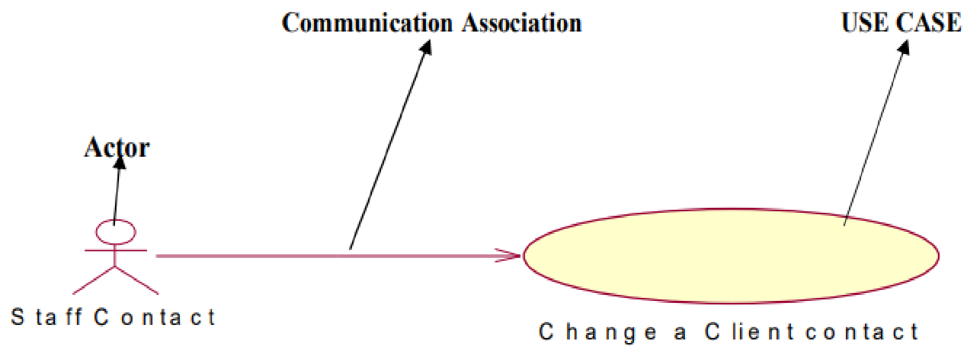
Use cases specify the functionality that the system will offer from the users' perspective. They are used to document the scope of the system and the developer's understanding of what it is that the users require.

Use cases are supported by behavior specifications. These specify the behaviour of each use case either using UML diagrams, such as collaboration diagrams or sequence diagrams or in text form as use case descriptions.

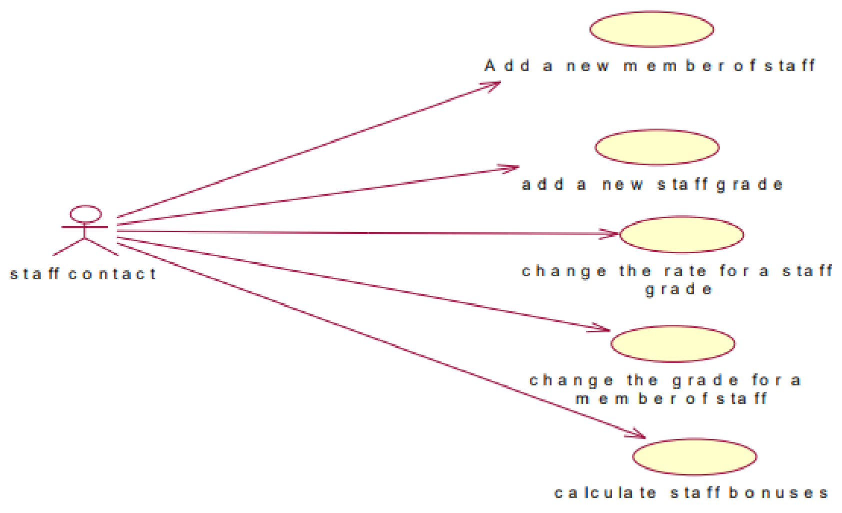
Textual use case descriptions provide a description of the interaction between the users of the system called actors, and the high level functions within the system called use cases.

Notation of USE CASE Diagrams

Use case diagrams show three aspects of the system: actors, use cases and relationships of the system or sub-system boundary. The following Figure shows the elements of the notation.



Example


Add a new member of staff
add a new staff grade
change the rate for a staff grade
change the grade for a member of staff
calculate staff bonuses
staff contact

Q4. Explain about object oriented analysis of design model?

Object-oriented analysis (OOA) and design (OOD) are techniques which have evolved since 1988 to support the creation of object-oriented software targeted at languages like Smalltalk, C++ and Eiffel.

A contrast is drawn between the older structured systems analysis and design and the newer OOA&D. The aims and goals of SSA&D are different from the aims and goals of OOA&D.

A large number of OO methods for analysis and design have been put forward.

These can roughly be categorised as:

1. Relationship and attribute centred approaches
2. Behaviour centred approaches
3. High industrial profile approaches
4. Synthesized approaches

No one approach has all the best techniques. Very few approaches cover all aspects in equal depth.

Purposes of Analysis and Design

Analysis looks at a system in terms of problem-domain concepts and seeks to elicit natural interactions and discover natural constraints. The aim is for the Software Engineer to raise his understanding of the problem domain, communicate with the Client (the domain expert) and sometimes reveal inconsistencies and incompleteness in the Client's awareness of the problem domain.

Design has the task of converting the analysis model into concepts and abstractions present in the programming style of the target language. Such concepts can include procedures, modules, objects, or processes and vary from language to language. The design model may have to integrate with existing subsystems or aim to use components from an existing software library.

Analysis and design are often assumed to be completely independent from the chosen programming language. However, this is not the case! To see this, we need to step back and consider analysis, design and programming as human conceptual activities.

Q6. What is object oriented analysis and design?

During object-oriented analysis there is an emphasis on finding and describing the objects or concepts in the problem domain. For example, in the case of the flight information system, some of the concepts include Plane, Flight, and Pilot. During object-oriented design (or simply, object design) there is an emphasis on defining software objects and how they collaborate to fulfil the requirements. For example, a Plane software object may have a tailNumber attribute and a getFlightHistory method

Q7. Explain sequence diagram with example?

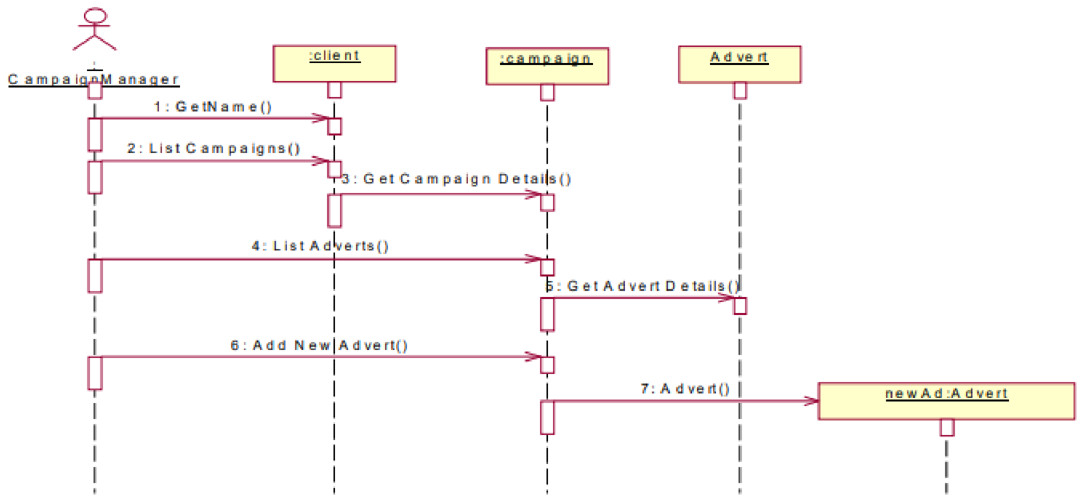
A sequence diagram shows an interaction between objects arranged in a time sequence. Sequence diagrams can be drawn at different levels of detail and to meet different purposes at several stages in the development life cycle. The commonest application of a sequence diagram is to represent the detailed object interaction that occurs for one use case or for one operation.

When a sequence diagram is used to model the dynamic behaviour of a use case it can be seen as a detailed specification of the use case.

Basic concepts and notation:

In Sequence diagram the vertical dimension represents time and all objects involved in the interaction are spread horizontally across the diagram.

Time normally proceeds down the page. However, a sequence diagram may be drawn with a horizontal time axis if required, and in this case, time proceeds from left to right across the page. Each object is represented by a vertical dashed line, called a lifeline, with an object symbol at the top. A message is shown by a solid horizontal arrow from one lifeline to another and is labelled with the message name. Each message name may optionally be preceded by a sequence number that represents the sequence in which the messages are sent, but this is not usually necessary on a sequence diagram since the message sequence is already conveyed by their relative positions along time axis.



Q9. Explain about object interaction and collaboration ?

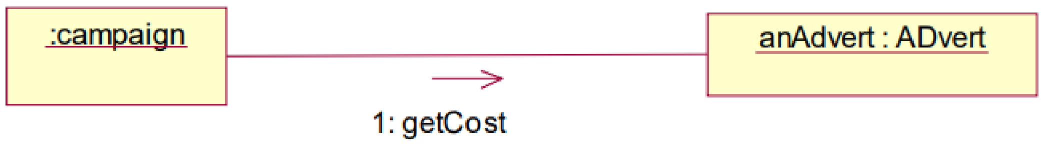
When an object sends a message to another object, an operation is invoked in the receiving object.

For example, in the Agate case study there is a requirement to be able to determine the current cost of the advertisements for an advertising campaign. This responsibility is assigned to the Campaign class. For a particular campaign this might be achieved if the Campaign object sends a message to each of its Advert objects asking them for their current cost.

In a programming language, sending the message getCost () to an Advert object, might use the following syntax.

advertCost = anAdvert.getCost()

The cost of each advert returned by the operation getCost () is totalled up in the attribute actualCost in the sending object, Campaign, so in order to calculate the sum of the costs for all adverts in a campaign the above statement must be executed repeatedly. For this purpose we are using message passing mechanism for object interaction. This Message passing can be represented on an object diagram, as shown below.



It can be difficult to determine what messages should be sent by each object. In this case, the getCost () operation should be located in the Advert class. This operation requires data that is stored in the advertCost attribute, and this has been placed in Advert. We can also see that an operation that calculates the cost of a Campaign must be able to find out the cost of each Advert involved.

But this is a simple collaboration and the allocation of these operations is largely dictated by the presence of particular attributes in the classes. More complex requirements may involve the performance of complex tasks, such that an object receiving one message must itself send messages that initiate further collaboration with other objects.

The objective of OOSD to distribute system functionality appropriately among its classes. This does not mean that all classes have exactly equal levels of responsibility but rather that each class should have appropriate responsibilities. Where responsibilities are evenly distributed, each class tends not to be complex and easy to develop , test and maintain.

An appropriate distribution of responsibility among classes has the important side effect of producing a system that is more resilient to changes in its requirements. When the users' requirements for a system change it is reasonable to expect that the application will need some modification, but ideally the change in the application should be of no greater magnitude than the change in the requirements.

Q12. What is meant by object? How are these created?

An object is a single unit having both data and the processes that operate on that data. For example, in object oriented programming language like C++, the data and functions are bundled together as a self contained unit called an object. An object is an entity which has some properties and behavior associated with it. Objects are the basic run time entities in an object oriented system. programming problems are analyzed in terms of objects. The main purpose of using objects are following.

- They correspond to the real life entities.
- They provide interactions with the real world.
- They provide practical approach for the implementation of the solution.

All the objects have a state, behavior and identity.

- **State of an object** - The state or attributes are the built in characteristics or properties of an object. For example, a T.V has the size, colour, model etc.
- **Behaviour of the object** - The behavior or operations of an object are its predefined functions. For example, a T.V. can show picture , change channels, tune for a channel etc. in object oriented programming terminology the behavior is implemented through methods.

- **Object identity** - Each object is uniquely identifiable. For example, the fridge can not become the T.V.

Object representation

An object is represented as shown below:

An object consists of the following.

- Data members - Data members are the variables. They establish the state or attributes for the object.
- Member functions - Member functions represent the code to manipulate the data. The behaviour of the object is determined by the member functions.

Q13. What do you mean by object modeling technique?

The object modeling techniques is an methodology of object oriented analysis, design and implementation that focuses on creating a model of objects from the real world and then to use this model to develop object–oriented software. object modeling technique, OMT was developed by James Rambaugh. Now-a-days, OMT is one of the most popular object oriented development techniques. It is primarily used by system and software developers to support full life cycle development while targeting object oriented implementations.

OMT has proven itself easy to understand, to draw and to use. It is very successful in many application domains: telecommunication, transportation, compilers etc. The popular object modeling technique are used in many real world problems. The object-oriented paradigm using the OMT spans the entire development cycle, so there is no need to transform one type of model to another.

Phase of OMT

The OMT methodology covers the full software development life cycle. The methodology has the following phase.

1. **Analysis** - Analysis is the first phase of OMT methodology. The aim of analysis phase is to build a model of the real world situation to show its important properties and domain. This phase is concerned with preparation of precise and correct modelling of the real world. The analysis phase starts with defining a problem statement which includes a set of goals. This problem statement is then expanded into three models; an object model, a dynamic model and a functional model. The object model shows the static data structure or skeleton of the real world system and divides the whole application into objects. In others words, this model represents the artifacts of the system. The dynamic model represents the interaction between artifacts above designed represented as events, states and transitions. The functional model represents the methods of the system from the data flow perspective. The analysis phase generates object model diagrams, state diagrams, event flow diagrams and data flow diagrams.
2. **System design** - The system design phase comes after the analysis phase. System design phase determines the overall system architecture using subsystems, concurrent tasks and data storage. During system design, the high level structure of the system is designed. The decisions made during system design are:

- The system is organized in to sub-systems which are then allocated to processes and tasks, taking into account concurrency and collaboration.
- Persistent data storage is established along with a strategy to manage shared or global information.
- Boundary situations are checked to help guide trade off priorities.
- **Object design** - The object design phase comes after the system design phase is over. Here the implementation plan is developed. Object design is concerned with fully classifying the existing and remaining classes, associations, attributes and operations necessary for implementing a solution to the problem. In object design:
    1. Operations and data structures are fully defined along with any internal objects needed for implementation.
    2. Class level associations are determined.
    3. Issues of inheritance, aggregation, association and default values are checked.
- **Implementation** - Implementation pahse of the OMT is a matter of translating the design in to a programming language constructs. It is important to have good software engineering practice so that the design phase is smoothly translated in to the implementation phase. Thus while selecting programming language all constructs should be kept in mind for following noteworthy points.
    1. To increase flexibility.
    2. To make amendments easily.
    3. For the design traceability.
    4. To increase efficiency.

Q14. How do object model, dynamic model and functional model differ from each other?

Follwing are the differences between object model , dynamic model and functional model:

Object Model

1. It represents the static structure of the application.
2. It specifies to whom it happens to.
3. Operations in an object model corresponds to events in dynamic model and functions in functional model.
4. It is represented using class diagrams.

Dynamic Model

1. It represents the essential behavior of the application.
2. It specifies when it happens.
3. Dynamic model describes the control structure of the objects. It defines decisions which are dependents of object values and which can cause action to change object values and invoke their functions.
4. It is represented using state diagrams.

Functional Model

1. It represents what the application does and not how it does.
2. It specifies what happens.
3. It describes functions to be invoked by operations in object model and actions in dynamic models.
4. It is represented using data flow diagrams.

Q15. What is JSD? Explain this design with its advantages and disadvantages.

JSD, Jackson Structure Design is a methodology to specify and design systems in which time factor is significant and system may be described using sequence of events. Developed by Michael A. Jackson , this design method considers the fact that the design of the system is an extension of the programme design. The purpose of this design method is to create a maintainable software. The method addresses all stages of the software development life cycle. It has three phases:

1. **Modeling phase** - A JSD model starts with real world consideration. This phase is a part of analysis process. The aspects of the real world relevant to the system being developed are modeled in this phase.
2. **Specification phase** - This phase focuses on the specification. In this phase JSD determines what is to be done? The previous phase i.e. the modeling phase provides the basic for the system specifications to achieve the required functionality.
3. **implementation phase** - In this phase , JSD determines how to achieve required functionality. Operational specifications are executed so that it expresses desired system behavior in terms of some abstract machine.

Ways of working

- The basic principle of operation of JSD is that development must start with describing and modeling the real world rather than specifying the function performed by the system.
- The second principle states that an adequate model of a time ordered world must itself be time ordered. The aim is to map progress in the real world on progress in the system that models it.
- The third principle stetes way of implementing the system based on transformation of specification in to efficient set of processes. These processes should be designed in a manner to make them run on available software and hardware.

Steps for JSD software development

Originally presented by Jackson in 1983 the method consisted of six steps which are following.

1. Entity /action step
2. Entity structure step
3. Initial model step
4. function step
5. system timing step

6. system implementing step

Later , some steps were combined to create a method with only three steps

1. **Modeling stage (Analysis)**
    - Action step
    - Structure step
2. **Network stage (design)**
    - Initial model step
    - Function step
    - System timing step
3. **Implementation stage (Realisation)**
    - System implementation step

Advantages of JSD

1. specially designed to handle real time problem.
2. JSD considers concurrent processing and timing.
3. JSD modeling focuses on time.
4. It provide functionality in the real world.
5. Excellent methodology for micro code application.

Disadvantages of JSD

1. JSD is a poor approach for high level analysis and data base design.
2. More complex due to pseudo code representation.
3. complex and difficult to understand.
4. less graphically oriented than SA/ SD and OMT.

Q16. What is the difference between SA/SD and OMT?

Sr. No.	**SA/SD**	**OMT**
1	It manages a system around procedures.	It manages system around real world objects.
2	More importance is on functional model and less on object model.	More importance is on object model and less on functional model.
3	Dominance order is functional, dynamic and object.	Dominance order is object, dynamic and functional.
4	It is a historical approach.	It is much advanced approach.

5	It clearly marks the system boundaries across which software procedures should communicate with real world. It is difficult to extend boundries.	System boundries can be easily extending by adding new objects, relationships.
6	Decomposition of process to sub-process is not standardized. Different people provide different decomposition.	Decompsition is based on objects so different people provide similar decomposition.
7	Reusability of components across projects is less as compared to in OMT.	Reusablity of components across projects is more as compared to SA/AD.
8	Not easily modifiable and extensible.	Easily modifiable and extensible.
9	Used when functions are more important than data.	Used when data is more important than functions.
10	Difficult to merge programming code organized about functions while database is organized around data.	It better integrates data with programming code as database is organized around data.

Q17. Write a short note on SD

A Structured Design, SD is a methodology to break down a problem in to smaller problems and then solving the smaller problems. Sub-problems are then arranged in to a hierarchy to forms a sequence of procedures. Structured design is the art of designing the components of a system and the their interrelationship in the best possible way.

Importance of structured design

- Good designing to make good programs.
- Code is easier to understand.
- Helps to make programs modular.
- Easier to troubleshoot.
- It helps making programming more systematic and less ambiguous.

Basic steps of structured design

- Identify major components of task /problem /goal /system.
- Decompose the system in to sub systems. These sub systems represent procedures.
- Group related components.
- Repeat the process as and when needed on individual components.
- Organize components in a hierarchy and a consider the data flow .

Q18. Explain the various models available in object oriented languages in brief. Also explain relationship among different models.

Answer:

There are three types of models in object oriented languages.

1. Object model

2. Dynamic model
3. Functional model

Object model

The object model identifies the classes in the system and their relationship, as well as their attributes and operations. It represents the static structure of the system. The object model is represented graphically by a class diagram.

Step for object modeling

Following steps are performed in constructing an object model.

1. Read carefully, the problem statement.
2. Locate the object classes by underlining nouns.
3. Remove unnecessary and incorrect classes.
4. Prepare a data dictionary.
5. Locate associations between object classes.
6. Remove unnecessary and incorrect attributes.
7. Use inheritance to share common structure.
8. Traverse access paths to identify deficiency.
9. Remove unnecessary and incorrect associations.
10. Locate attributes of the object classes.

Dynamic model

The dynamic model indicates the dynamics of the objects and their changes in state. The dynamic model captures the functional behavior of the system by exploring the behavior of the objects over time and the flow of control and events among the objects.

Steps for dynamic modeling

Following steps are performed in constructing dynamic model.

1. Locate use cases and prepare scenarios of typical interaction sequence.
2. Locate events between objects and prepare an event trace diagram for each scenario.
3. Develop an event flow diagram for the system.
4. Develop state diagrams for classes with important dynamic behavior.
5. Check for consistency and completeness of events shared among the state diagrams.

Functional model

The functional model is a data flow diagram of the system and describes what the system does, not how it is done. A DFD is a network representation of the system to show the functional relationships of the values that are computed by a system. Data flow diagrams consist of processes , data flows , actors and data stores.

1. A process transforms input data values in to output data values. A process is presented as an ellipse, with the name of the process inside the ellipse.
2. A data flow shows the flow of data through a network of processes.
3. An actor, drawn as a rectangle, is an object.
4. A data store is a respository for the temporary storing of data . A data store is represented as a pair of parallel lines containing the name of a data store. Data stores may also be objects.

Steps for functional model

Following steps are performed in constructing a functional model.

1. Identify input and output values.
2. Use data flow diagrams as needed to show functional dependencies.
3. Describe what each function does.
4. Identify constraints.
5. Specify optimization criteria.

Relationship among models

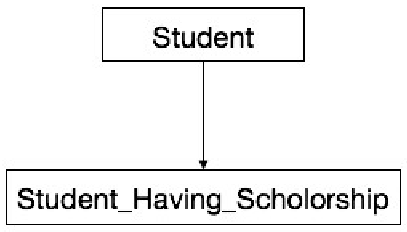
The three models – object model , dynamic model and functional model are closely related to one another. To implement a system, all the three models are required to be modeled. After integration of these models , a complete view of the system can be achieved. The working of all three models explains how they are interrelated which is as follows:

1. Each model describes an aspect of the system but contains references to the other models.
2. The object model describes the data structure on which the dynamic and functional models operate on.
3. The operations in the object model correspond to events in the dynamic model and functions in the functional model.
4. The dynamic model describes the controlling structure of objects. It shows decisions which depend on object values and which cause actions that change object values and invoke functions.
5. The functional model describes functions invoked by operations in the object model and actions in the dynamic model.
6. Functions operate on data values specified by the object model. The functional model also shows constraints on object values.

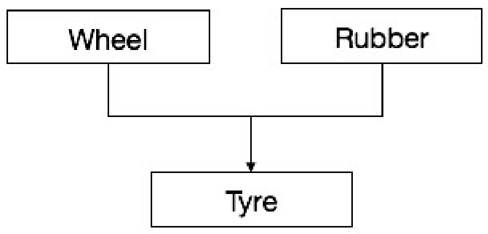
Q19. What is inheritance? Discuss types of inheritance with example.

Inheritance

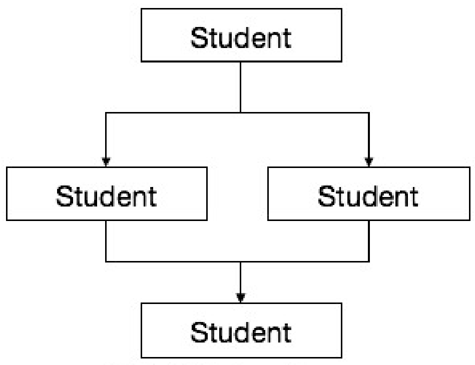
Inheritance is the mechanism to allow a class A to inherit properties of class b i.e A inherits from B. Objects of class A thus have accesses to attribute and methods of class B without the need to redefine them. If class A inherits from class B , then B is called super class of A and A is called subclass of B. Supper classes are also called base classes or parents classes. Subclasses may be called derived classes or child classes.



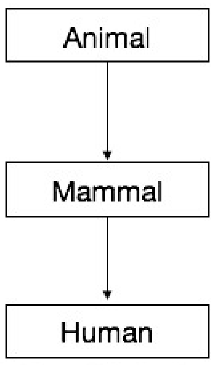
Single Inheritance



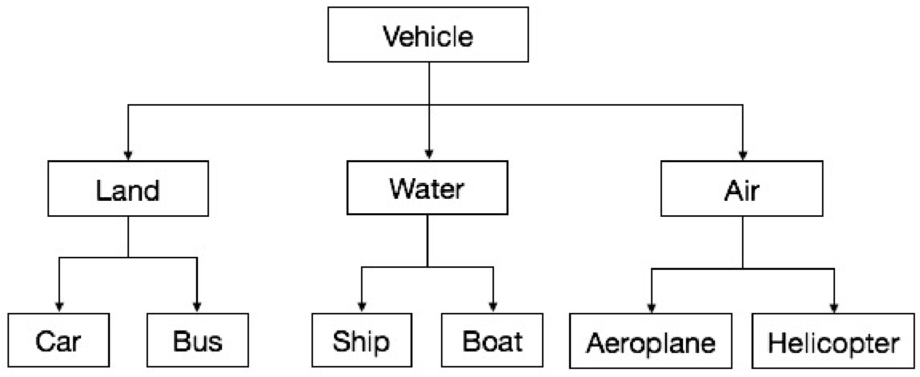
Multiple Inheritance



Hybrid Inheritance



Multi-level Inheritance



Hierarchical Inheritance

Importance of inheritance

Inheritance is one of the most powerful features of object oriented programming. Most important advantages of inheritance are:

1. **Reusability** - Inheritance allows deriving new classes from existing classes without modifying it. This helps in reusability of information in the child class as well as adding extra functionality in to it.

2. **Saves times and efforts** - The concept of reusability achieved by inheritance saves a lot of programmer time and effort as the main code written can be reused in various situations as needed.
3. **Closeness with the real world** - Inheritance allows the capability to express the inheritance relationship and ensures closeness with the real world models.
4. **Easy modification** - Inheritance allows modification to be done in an easier manner.
5. **Transitive Nature of inheritance** - If a class A inherits properties of another class B then all subclasses of A will automatically inherits the properties of B. This concept is called transitive nature of inheritance.

Syntax of defining a sub class from super class

```
class subclass: visibility_mode  superclass_name
{
...
}
```

Here class is keyword. superclass_name is the neme of the derived class which is a valid identifier. Visibility modes are of three types public, private and protected.

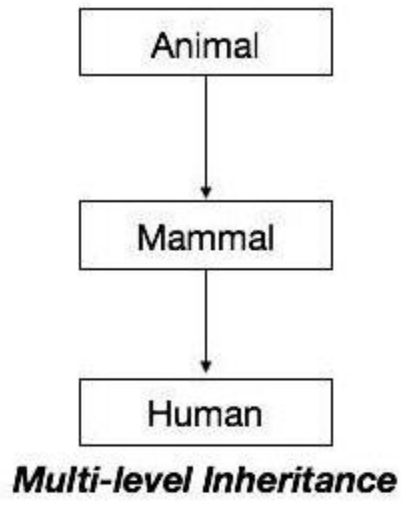
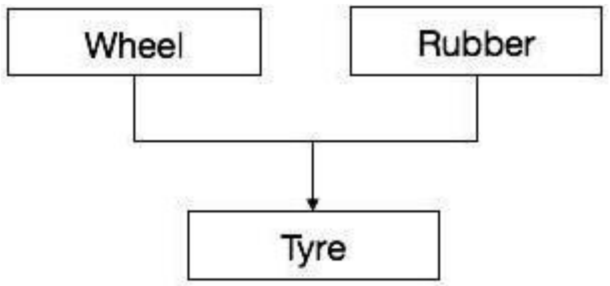
Types of Inheritance

Various types of inheritance are supported by C++. These are:

1. **Single Inheritance** - When a subclass inherits properties from one base class, it is called single inheritance. In single inheritance , there is only one base class and one derived class.
2. **Multiple Inheritance** - When a subclass inherits properties from multiple base classes, it is known as multiple inheritance. In multiple Inheritance , there is only one derived class and several base classes.
3. **Hierarchical Inheritance** - When two or more subclasses inherit properties from a single base class, It is known as Hierarchical inheritance . In hierarchical inheritance , the feature of one class may be inherited by more then one class.
4. **Multilevel Inheritance** - In multilevel Inheritance , A class is derived from an already derived class. The transitive nature of inheritance is reflected by this form of inheritance.
5. **Hybrid Inheritance** - When two or more type of inheritance are required to code a program, such type of inheritance is called hybrid inheritance. Figure shown below is a combination of multilevel and multiple inheritance.

Q20. What is the difference between Multilevel and Muliple Inheritance?

Sr. No.	**Multilevel Inheritance**	**Multiple Inheritance**

1	In multilevel inheritance, features of only one base class are allowed in derived class.	In multile inheritance, features of several classes can be clubbed in a derived class.
2	It consists of many levels of inheritance.	It consists of only two levels of inheritance.
3	A class can be derived from already derived class as shown below.  **Multi-level Inheritance**	A class can be derived from multiple classes as shown below.  **Multiple Inheritance**

4	Syntax is `class x {}`
	`class y: visibility_control x {}`
	`class z: visibility_control y {}`
		Syntax is `class x {}`
		`class y {}`
		`class z: visibility_control x, visibility_control y {}`

Q21. What is the difference between publicly derived inheritance and privately derived inheritance?

Sr. No.	**Publicly derived inheritance**	**Privately derived inheritance**
1	The public derivation of inheritance means that the derived class can access public and protected members of the base class.	The private derivation of inheritance means that the derived class can access public and protected members of the base class privately.
2	The public derivation does not change the access specifiers for inherited	The inherited members become private in the derived class and they can

	members in the derived class. They can be inherited further.	not be inherited further by any other derived class derived from this derived class.
3	The protected member of base class will act as a protected member in derived class.	The protected members of a base class will act as a private member in derived class.
4	The public member of base class will remain public member in derived class.	The public members of a base class will act as a private member in derived class.