Interview Questions with Answers

***Note**: These questions are based on my past experiences and research*

Chapter 1. Object Oriented Programming

Basic Level:

1. What is Object Oriented Programming?

Object-Oriented Programming(OOPs) is a type of programming that is based on objects rather than just functions and procedures. Individual objects are grouped into classes. OOPs implement real-world entities like inheritance, polymorphism, hiding, etc into programming. It also allows binding data and code together.

2. Why use OOPs? OR Advantages of OOPs

- OOPs allows clarity in programming thereby allowing simplicity in solving complex problems
- Code can be reused through inheritance thereby reducing redundancy
- Data and code are bound together by encapsulation
- OOPs allows data hiding, therefore, private data is kept confidential
- Problems can be divided into different parts making it simple to solve
- The concept of polymorphism gives flexibility to the program by allowing the entities to have multiple forms

3. What are the main features/pillars of OOPs?

- Inheritance
- Encapsulation
- Polymorphism
- Data Abstraction

4. What are the main features of OOPs?

- Inheritance
- Encapsulation
- Polymorphism
- Data Abstraction

5. What is a class?

A class is a prototype that consists of objects in different states and with different behaviors. It has a number of methods that are common to the objects present within that class.

6. What is an object?

An object is a real-world entity which is the basic unit of OOPs for example chair, cat, dog, etc. Different objects have different states or attributes, and behaviors.

Note: Class is a concept and to map this concept in real life is an object.

7. What is the difference between a class and a structure?

Class: User-defined blueprint from which objects are created. It consists of methods or sets of instructions that are to be performed on the objects.

Structure: A structure is basically a user-defined collection of variables which are of different data types.

8. Can you call the base class method without creating an instance?

Yes, you can call the base class without instantiating it if:

- It is a static method
- The base class is inherited by some other subclass

Object	Class
A real-world entity which is an instance of a class	A class is basically a template or a blueprint within which objects can be created
An object acts like a variable of the class	Binds methods and data together into a single unit
An object is a physical entity	A class is a logical entity
Objects take memory space when they are created	A class does not take memory space when created
Objects can be declared as and when required	Classes are declared just once

9. What is inheritance?

Inheritance is a feature of OOPs which allows classes to inherit common properties from other classes. For example, if there is a class such as ‘vehicle’, other classes like ‘car’, ‘bike’, etc can inherit common properties from the vehicle class. This property helps you get rid of redundant code thereby reducing the overall size of the code.

9. What are the different types of inheritance?

- Single inheritance
- Multiple inheritance
- Multilevel inheritance
- Hierarchical inheritance
- Hybrid inheritance

10. What is the difference between multiple and multilevel inheritance?

Multiple Inheritance	Multilevel Inheritance
Multiple inheritance comes into picture when a class inherits more than one base class	Multilevel inheritance means a class inherits from another class which itself is a subclass of some other base class
Example: A class defining a child inherits from two base classes Mother and Father	Example: A class describing a sports car will inherit from a base class Car which inturn inherits another class Vehicle

.11. What is hierarchical inheritance?

Hierarchical inheritance refers to inheritance where one base class has more than one subclass. For example, the vehicle class can have 'car', 'bike', etc as its subclasses.

12. What are the limitations of inheritance?

- Increases the time and effort required to execute a program as it requires jumping back and forth between different classes
- The parent class and the child class get tightly coupled
- Any modifications to the program would require changes both in the parent as well as the child class
- Needs careful implementation else would lead to incorrect results.

13. What is a superclass?

A superclass or base class is a class that acts as a parent to some other class or classes. For example, the Vehicle class is a superclass of class Car.

14. What is a subclass?

A class that inherits from another class is called the subclass. For example, the class Car is a subclass or a derived of Vehicle class.

15. What is polymorphism?

Polymorphism refers to the ability to exist in multiple forms. Multiple definitions can be given to a single interface. For example, if you have a class named Vehicle, it can have a method named

speed but you cannot define it because different vehicles have different speeds. This method will be defined in the subclasses with different definitions for different vehicles.

16. What is static polymorphism?

Static polymorphism (static binding) is a kind of polymorphism that occurs at compile time. An example of compile-time polymorphism is method overloading.

17. What is dynamic polymorphism?

Runtime polymorphism or dynamic polymorphism (dynamic binding) is a type of polymorphism which is resolved during runtime. An example of runtime polymorphism is method overriding.

18. What is function/method overloading?

Method overloading is a feature of OOPs which makes it possible to give the same name to more than one method within a class if the arguments passed differ.

19. What is function/method overriding?

Method overriding is a feature of OOPs by which the child class or the subclass can redefine methods present in the base class or parent class. Here, the method that is overridden has the same name as well as the signature meaning the arguments passed and the return type.

20. What is operator overloading?

Operator overloading refers to implementing operators using user-defined types based on the arguments passed along with it.

21. Differentiate between overloading and overriding.

Overloading	Overriding
Two or more methods having the same name but different parameters or signature	Child class redefining methods present in the base class with the same parameters/ signature
Resolved during compile-time	Resolved during runtime

22. What is encapsulation?

Encapsulation refers to binding the data and the code that works on that together in a single unit. For example, a class. Encapsulation also allows data-hiding as the data specified in one class is hidden from other classes.

23. What is data abstraction?

Data abstraction is a very important feature of OOPs that allows displaying only the important information and hiding the implementation details. For example, while riding a bike, you know that if you raise the accelerator, the speed will increase, but you don't know how it actually happens. This is data abstraction as the implementation details are hidden from the rider.

24. How to achieve data abstraction?

Data abstraction can be achieved through:

- Abstract class
- Abstract method

25. What is an abstract class?

An abstract class is a class that consists of abstract methods. These methods are basically declared but not defined. If these methods are to be used in some subclass, they need to be exclusively defined in the subclass.

26. Can you create an instance of an abstract class?

No. Instances of an abstract class cannot be created because it does not have a complete implementation. However, instances of subclass inheriting the abstract class can be created.

27. What is an interface?

It is a concept of OOPs that allows you to declare methods without defining them. Interfaces, unlike classes, are not blueprints because they do not contain detailed instructions or actions to be performed. Any class that implements an interface defines the methods of the interface.

28. Differentiate between data abstraction and encapsulation.

Data abstraction	Encapsulation
Solves the problem at the design level	Solves the problem at the implementation level
Allows showing important aspects while hiding implementation details	Binds code and data together into a single unit and hides it from the world

29. What are virtual functions?

Virtual functions are functions that are present in the parent class and are overridden by the subclass. These functions are used to achieve runtime polymorphism.

30. What are pure virtual functions?

Pure virtual functions or abstract functions are functions that are only declared in the base class. This means that they do not contain any definition in the base class and need to be redefined in the subclass.

31. What is a constructor?

A constructor is a special type of method that has the same name as the class and is used to initialize objects of that class.

32. What is a destructor?

A destructor is a method that is automatically invoked when an object is destroyed. The destructor also recovers the heap space that was allocated to the destroyed object, closes the files and database connections of the object, etc.

33. Types of constructors

Types of constructors differ from language to language. However, all the possible constructors are:

- Default constructor
- Parameterized constructor
- Copy constructor
- Static constructor
- Private constructor

34. What is a copy constructor?

A copy constructor creates objects by copying variables from another object of the same class. The main aim of a copy constructor is to create a new object from an existing one.

35. What is an exception?

An exception is a kind of notification that interrupts the normal execution of a program. Exceptions provide a pattern to the error and transfer the error to the exception handler to resolve it. The state of the program is saved as soon as an exception is raised.

36. What is exception handling?

Exception handling in Object-Oriented Programming is a very important concept that is used to manage errors. An exception handler allows errors to be thrown and caught and implements a centralized mechanism to resolve them.

37. What is the difference between an error and an exception?

Error	Exception
Errors are problems that should not be encountered by applications	Conditions that an application might try to catch

38. What is a try/ catch block?

A try/ catch block is used to handle exceptions. The try block defines a set of statements that may lead to an error. The catch block basically catches the exception.

39. What are the limitations of OOPs?

- Usually not suitable for small problems
- Requires intensive testing
- Takes more time to solve the problem
- Requires proper planning
- The programmer should think of solving a problem in terms of objects.

40.. What are ‘access specifiers’?

Access specifiers or access modifiers are keywords that determine the accessibility of methods, classes, etc in OOPs. These access specifiers allow the implementation of encapsulation. The most common access specifiers are public, private and protected. However, there are a few more which are specific to the programming languages.