Object-Oriented Programming is one of the main concepts in the programming world, therefore, every interview that you attend requires knowledge of OOPs. This article compiles the most frequently asked OOPs Interview Questions for freshers which will help you ace your interviews. So go ahead and get prepared.

Let's take a quick look at all the topics of discussion:

OOPs Interview Questions

Basic OOPs Interview Questions for Freshers

1. What is the difference between OOP and SOP?

Object-Oriented Programming	Structural Programming
Object-Oriented Programming is a type of programming which is based	Provides logical structure to a program where programs are divided functions

on objects rather than just functions and procedures	
Bottom-up approach	Top-down approach
Provides data hiding	Does not provide data hiding
Can solve problems of any complexity	Can solve moderate problems
Code can be reused thereby reducing redundancy	Does not support code reusability

2. 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 implements real-world entities like inheritance, polymorphism, hiding, etc into programming. It also allows binding data and code together.

3. Why use 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**

4. What are the main features of OOPs?

- **Inheritance**
- **Encapsulation**
- **Polymorphism**
- **Data Abstraction**

To know more about OOPs in JAVA, Python, and C++ you can go through the following blogs:

- **JAVA OOPs Concepts**

- Python OOPs Concepts
- C++ OOPs Concepts

Classes and Objects OOPs Interview Questions and Answers

5. 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.

6. 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 the objects present within that class.

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 set 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**

9. What is the difference between a class and an object?

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

To know more about objects and classes in JAVA, Python, and C++ you can go through the following blogs:

- Objects in Java
- Class in Java
- Objects and classes in Python
- Objects in C++

OOPs Interview Questions – Inheritance

10. What is inheritance?

Inheritance is a feature of OOPs which allows classes 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.

11. What are the different types of inheritance?

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

12. 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**

13. What is hybrid inheritance?

Hybrid inheritance is a combination of multiple and multi-level inheritance.

14. What is hierarchical inheritance?

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

15. 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**

To know more about inheritance in Java and Python, read the below articles:

- **Inheritance in Java**
- **Inheritance in Python**

16. 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.

17. 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.

OOPs Interview Questions – Polymorphism

18. 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 speed. This method will be defined in the subclasses with different definitions for different vehicles.

19. 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.

20. 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.

21. What is method overloading?

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

22. What is 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.

23. What is operator overloading?

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

24. 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

To know more about polymorphism in Java and Python, read the below articles:

- Polymorphism in Java
- Polymorphism in Python

OOPs Interview Questions – Encapsulation

25. 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.

26. 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.

27. What is the difference between public, private and protected access modifiers?

Name	Accessibility from own class	Accessibility from derived class	Accessibility from world
Public	Yes	Yes	Yes
Private	Yes	No	No
Protected	Yes	Yes	No

To know more about encapsulation read along:

- Encapsulation in Java
- Encapsulation in C++
- Encapsulation in Python

Data abstraction

28. 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.

29. How to achieve data abstraction?

Data abstraction can be achieved through:

- **Abstract class**
- **Abstract method**

30. 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.

31. 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.

32. 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.

33. 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

To know more about data abstraction, below articles might help you:

- Abstraction in Java
- Abstraction in Python

Methods and Functions OOPs interview questions

34. 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.

35. 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.

36. 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.

37. 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.

38. 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

39. 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.

40. What is the use of 'finalize'?

Finalize as an object method used to free up unmanaged resources and cleanup before Garbage Collection(GC). It performs memory management tasks.

41. What is Garbage Collection(GC)?

GC is an implementation of automatic memory management. The Garbage collector frees up space occupied by objects that are no longer in existence.

42. Differentiate between a class and a method.

Class	Method
A class is basically a template that binds the code and data together into a single unit. Classes consist of methods, variables, etc	Callable set of instructions also called a procedure or function that are to be performed on the given data

43. Differentiate between an abstract class and an interface?

Basis for comparison	Abstract Class	Interface
Methods	Can have abstract as well as other methods	Only abstract methods
Final Variables	May contain final and non-final variables	Variables declared are final by default
Accessibility of Data Members	Can be private, public, etc	Public by default
Implementation	Can provide the implementation of an interface	Cannot provide the implementation of an abstract class

44. What is a final variable?

A variable whose value does not change. It always refers to the same object by the property of non-transversity.

OOPs Interview Questions – Exception Handling

45. 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.

46. 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.

47. 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

48. 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.

49. What is a finally block?

A finally block consists of code that is used to execute important code such as closing a connection, etc. This block executes when the try block exits. It also makes sure that finally block executes even in case some unexpected exception is encountered.

OOPs Interview Questions – Limitations of OOPs

50. 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**