OOPS Interview Questions and Answers using Java

Basic OOPS Concepts

1. Why do we need to use OOPs?

OOPs needs to be used for:

- Making programming clearer and problem-solving more concise
- Reusing code with the help of inheritance
- Reducing redundancy
- Encapsulation
- Data hiding
- The division into subproblems
- Program flexibility using polymorphism

2. What is meant by the term OOPs?

OOPs stands for Object-Oriented Programming, a programming paradigm that utilizes objects to represent and manipulate data. OOPs aspects are based on the concept of objects, which have properties and methods, and the interactions between them.

3. What is a class?

A class defines a template for creating objects; the objects created from a class are known as instances of that class. A class also defines the interface for interacting with objects of that class, specifying which methods can be called and what parameters they take.

4. What is an object?

In object-oriented programming, an object is an instance of a class. It is a self-contained unit of code and data with its properties and methods. An object is a specific instance of a class and can be created at runtime.

5. What are the main features of OOPs?

Object-Oriented Programming (OOP)'s main features are:

- Encapsulation

- Inheritance
- Polymorphism
- Abstraction

6. What are some advantages of using OOPs?

Some advantages of object-oriented programming include:

- Improved code organization
- Reusability
- Maintainability
- Ability to model real-world concepts
- Encapsulation of data and behavior

7. Why are OOPs so popular?

OOPs is so popular because it allows developers to organize and structure their code to reflect real-world objects and their interactions, making it more intuitive and easier to understand.

8. What are some major Object Oriented Programming languages?

Major object-oriented programming languages include:

- Java
- C++
- Python
- C#
- Ruby

9. What are some other programming paradigms other than OOPs?

Other programming paradigms include:

- Functional Programming: This emphasizes using functions and immutable data to solve problems.
- Procedural Programming: This emphasizes breaking a program into small procedures or functions to improve readability and maintainability.
- Logic Programming: This emphasizes using logic and mathematical notation to represent and manipulate data.
- Event-driven Programming: This emphasizes handling events, and the control flow is based on the events.

10. What is meant by structured programming?

Structured programming is a programming paradigm that emphasizes breaking down a program into smaller, modular code units, such as functions and procedures, to improve readability and maintainability.

Inheritance

11. What is multiple inheritance?

Multiple inheritance is a feature in object-oriented programming where a class inherits from more than one parent class. This means the child class can access the properties and methods of all the parent classes.

12. Explain the concept of inheritance with a real-life example.

The parent class is a logical concept, such as a vehicle is a base class that defines the common properties shared by all vehicles. However, child classes are a more specific type of class such as truck, bus, car, etc. Inheritance allows subclasses to inherit common attributes of a vehicle and define specific attributes and methods to their own.

13. What is a subclass?

A subclass is a class derived from another, known as the superclass. The subclass inherits the properties and methods of the superclass and can also have its properties and methods.

14. Define a superclass?

A superclass is a class used as the base class for one or more derived classes. A superclass defines properties and methods shared by all of its derived classes.

15. What is the difference between a base class and a superclass?

The base class is the root class - the most generalized class. At the same time, the superclass is the immediate parent class from which the other class inherits.

16. What are the various types of inheritance?

In C++, there are several inheritance types including:

- Single inheritance
- Multiple inheritance
- Multi-level inheritance

- Hierarchical inheritance
- Hybrid inheritance

17. What is the function of a super keyword?

The super keyword is used to forward a constructor's call to a constructor in the superclass. It invokes the overridden method that allows access to these methods and the superclass's hidden members.

18. Are there any limitations of inheritance? If yes, then what?

Yes. The limitations of inheritance are:

- Increased execution effort and time
- Tight coupling of parent and child class
- Requires correct implementation
- Requires jumping between different classes

19. What is difference between Is-A and Has-A relationship?

Feature	Is-A Relationship	Has-A Relationship
Concept	Inheritance	Composition/Aggregation
Meaning	"This class is a specialized type of that class"	"This class has a reference to that class"
Example	Dog is an Animal	Car has an Engine
Code	`class Dog extends Animal`	`class Car { Engine engine; }`

Encapsulation & Abstraction

20. What is encapsulation?

Encapsulation is a mechanism for hiding the internal details of an object from the outside world. It is one of the fundamental principles of object-oriented programming, along with inheritance and polymorphism. Encapsulation is achieved by using access modifiers (such as "public" or "private") to restrict access to the members of a class.

21. Give an example of encapsulation.

Encapsulation is the process of hiding the internal details (data) of a class and only exposing necessary parts through methods. It helps in data hiding and protecting the integrity of the object. In Java, we achieve encapsulation by:

- Declaring fields (variables) as private.
- Providing public getter and setter methods to access and update the private fields.

22. What is abstraction?

Abstraction is a process of hiding the implementation details of a class or an object and showing only the necessary information to the users. It is a technique of displaying only the essential features of an object and hiding the background details, which helps to reduce complexity and increase efficiency.

23. How is encapsulation different from data abstraction?

Data abstraction refers to the ability to hide unwanted information. At the same time, encapsulation refers to hiding data as well as the method together.

24. One of the key OOPs interview questions could be to give a real-life example of data abstraction.

While driving a car, you know that on pressing the accelerator, the speed will increase. However, you do not know precisely how it happens. This is an example of data abstraction as the implementation details are concealed from the driver.

25. Define protected access modifier.

A protected access modifier is accessible by own class and accessible by derived class but not accessible by the world.

26. What are access specifiers, and what is their significance?

Access specifiers are keywords in object-oriented programming languages that determine the level of access to a class, method, or variable. The significance of access specifiers is to control the visibility and accessibility of class members, preventing unintended modification and promoting encapsulation.

Polymorphism

27. What is polymorphism?

Polymorphism is a concept in object-oriented programming that allows objects of different classes to be treated as objects of a common superclass. The basic idea behind polymorphism is that a single function or method can be written to operate on multiple things.

There are two types of polymorphism:

- Compile-time polymorphism (also known as static polymorphism) is achieved through function or operator overloading.
- Runtime polymorphism (also known as dynamic polymorphism) is achieved through function overriding.

28. What is the difference between overloading and overriding?

Aspect	Overloading	Overriding
Definition	Having multiple methods in the same class with the same name but different parameters (different method signatures).	Redefining a method in the child class that already exists in the parent class with the same signature.
Purpose	To perform similar operations with different types/number of inputs.	To change or enhance the behavior of a method from the parent class.
Where it occurs	Within the same class	In inheritance (parent-child classes)
Method signature	Must be different (different number or types of parameters).	Must be exactly the same (same method name, parameters, and return type).
Return type	Can be different (in methods with different parameters).	Must be same or covariant (subtype).
Access modifier	No restriction (can be anything).	Cannot have a more restrictive modifier than the parent method.

Compile-time/ Runtime	Checked at compile-time	Checked at runtime

29. What is compile time polymorphism?

When a polymorphic call is made, and the compiler knows which function is to be called; this is known as compile-time polymorphism. The features like function default arguments, overloading, and templates in C++ support compile-time polymorphism.

30. How does C++ support polymorphism?

C++ supports polymorphism through a feature called virtual functions. A virtual function is a member function declared virtual in the base class and can be overridden in derived classes. When a virtual function is called through a base class pointer or reference, the program will determine at runtime which version of the function to call based on the actual type of the object being pointed to or referenced.

31. What is meant by static polymorphism?

Static polymorphism, also known as compile-time polymorphism, is a form of polymorphism in which the type of an object is determined at compile-time. It is achieved through function overloading and operator overloading.

32. What is meant by dynamic polymorphism?

Dynamic polymorphism, also known as runtime polymorphism, is a form of polymorphism in which the type of an object is determined at runtime. It is achieved through function overriding.

33. Define virtual functions.

The functions that help achieve runtime polymorphism are a part of functions present in the parent class and overridden by a subclass.

34. Name three operators that can't be overloaded.

- "::" Scope resolution operator
- ". *" Pointer to member operator
- "." dot or Member access operator

Interfaces & Abstract Classes

35. What is an interface?

An interface is a collection of pure virtual functions that define a contract for a class to implement. It is a way to achieve abstraction in C++.

36. What is an abstract class?

An abstract class is a class that cannot be instantiated, and it is usually used as a base class for other classes. An abstract class contains at least one pure virtual function.

37. What is an abstract function?

An abstract function is a function declared only in the base class. It is redefined in the subclass as it does not contain any definition in the base class.

38. How is an abstract class different from an interface?

An abstract class can contain concrete and pure virtual functions, while an interface can only contain pure virtual parts.

39. What is the access modifier for methods inside an interface?

All the methods inside an interface are public by default, and no other modifier can be specified.

Constructors & Destructors

40. What is a constructor?

A constructor is a special member function of a class that is automatically called when an object is created. It is used to initialize the object's state and allocate any resources it needs. Constructors have the same name as the class and do not have a return type.

41. What are the various types of constructors in C++?

In C++, several types of constructors include default constructor, parameterized constructor, copy constructor, and move constructors.

42. What is a copy constructor?

A copy constructor is a special constructor that is used to create a new object as a copy of an existing object. It takes a reference to an object of the same class as its argument and creates a new object with the same state as the original object.

43. What is a destructor?

A destructor is a special member function of a class that is automatically called when an object of the class goes out of scope or is deleted. It is used to release any resources that the object was holding and perform any other necessary cleanup. Destructors have the same name as the class with a tilde (~) prefix and do not have any return type.

Miscellaneous

44. What is the purpose of 'this' keyword?

To refer to the current object of a class, this keyword is used. It is used as a pointer that differentiates between the global object and the current object by referring to the current one.

45. How is a structure different from a class?

A structure is a user-defined collection of variables having different data types. However, it is not possible to instantiate a structure or inherit from it. Thus, it's not an OOPs concept.

46. How much memory does a class occupy?

The amount of memory a class occupies in C++ depends on the size of its data members and any base class it inherits from. The size of a class is the sum of the dimensions of all its data members. For example, a class with an int data member and a double data member would occupy 8 bytes on a system with 4-byte ints and 8-byte doubles.

47. Is it always necessary to create objects from class?

No, it is not always necessary to create objects from a class. A class can be used to define a blueprint for creating things, but it is not required to create an object from a class to use its methods or access its properties.

48. How can you call a base class method without creating an instance?

It is possible to call the base class without instantiation if it's a static method and some other subclass has inherited the base class.

49. List down the limitations of Object-Oriented programming.

- It requires intensive testing
- Not apt for minor problems
- It requires good planning

- It takes more time to solve problems
- Problems need to be thought in term of objects

50. What is an exception?

An exception is an abnormal event or error that occurs during the execution of a program, which can disrupt the normal flow of the program.

51. What is meant by exception handling?

Exception handling refers to catching and managing exceptions that occur during the execution of a program. It handles runtime errors and unexpected conditions, allowing the program to continue running instead of crashing.

52. What is meant by garbage collection in the OOPs world?

Garbage collection is a feature in object-oriented programming languages that automatically frees up a program's memory that is no longer being used.

53. Can we run a Java application without implementing the OOPs concept?

No, Java is an object-oriented programming language, and the core concepts of OOPs, such as classes, objects, and inheritance, are fundamental to the language.

54. Basic Difference Between Functions and Procedures

Aspect	Function (Method with Return Value)	Procedure (Method without Return Value)
Definition	A function is a method that performs a task and returns a value to the caller.	A procedure is a method that performs a task without returning any value (void method).
Return Value	Returns a value (using return keyword).	Does not return a value (return type is void).
Purpose	Used when you want to compute and return a result.	Used when you want to execute a process, like printing, updating, or performing steps.
Keyword Used	Can return int, String, double, etc.	Uses the keyword void.