Object Oriented Programming (OOPs)

1. What is Object Oriented Programming (OOPs)?
2. Why OOPs?
3. What are the Object Oriented Features supported by Java?
4. What is a Class?
5. What is an Object?
6. What is Encapsulation?
7. What is Abstraction?
8. What is Polymorphism?
9. What are the different types of Polymorphism?
10. What is Inheritance? What is its purpose?
11. Are there any limitations on Inheritance?
12. What different types of inheritance are there?
13. What are access specifiers? What is their significance in OOPs?
14. . What are the advantages and disadvantages of OOPs?
15. . What is the difference between Structured Programming and Object Oriented Programming?
16. What are some commonly used Object Oriented Programming Languages?
17. What is the difference between overloading and overriding?
18. What is an abstract class?
19. What is an interface?
20. How is an abstract class different from an interface?
21. How much memory does a class occupy?
22. Is it always necessary to create objects from class?
23. What is Constructor?
24. What are the various types of constructors in C++?
25. What is a destructor?
26. Can we overload the constructor in a class?
27. What is the virtual function?
28. What is pure virtual function?
29. What is a static method?
30. What is a final class?
31. What is composition?
32. What is method hiding?
33. What is a constructor chaining?
34. What is the diamond problem in multiple inheritance?
35. What is a shallow copy and a deep copy?
36. What is a virtual method?
37. How does Java achieve multiple inheritance?
38. What is the purpose of the final keyword in Java?
39. How is encapsulation related to data hiding?
40. What is the difference between composition and inheritance?
41. What is the role of a destructor in C++?
42. What is an Inline function?
43. What is a friend function?

44. What is function overloading?
45. What is operator overloading?
46. What is a ternary operator?
47. What is the use of finalize method?
48. What are the different types of arguments?
49. What are the different access specifiers used in Java?
50. What is the difference between composition and inheritance?
51. What is the purpose of an abstract class?
52. What are the differences between constructor and method of a class in Java?
53. What is the diamond problem in Java and how is it solved?
54. What is the difference between local and instance variables in Java?
55. What is a Marker interface in Java?
56. What is the super keyword?
57. What is ‘this’ pointer?

Object Oriented Programming (OOPs)

1. **What is Object Oriented Programming (OOPs)?**
   Object-Oriented Programming (OOP) is a programming paradigm that organizes code into objects, which are instances of classes.
   It focuses on encapsulating data and behavior together, promoting modularity and reusability.

2. **Why OOPs?**
   The main advantage of OOP is better manageable code that covers the following:
   1) The overall understanding of the software is increased as the distance between the language spoken by developers and that spoken by users.
   2) Object orientation eases maintenance by the use of encapsulation. One can easily change the underlying representation by keeping the methods the same.
   3) The OOPs paradigm is mainly useful for relatively big software.

3. **What are the Object Oriented Features supported by Java?**
   The main feature of the OOPs, also known as 4 pillars or basic principles of OOPs are as follows:
   - Encapsulation
   - Inheritance
   - Polymorphism
   - Abstraction

4. **What is a Class?**
   A class is a building block of Object Oriented Programs. It is a user defined data type that contains the data members and member functions that operate on the data members. It is like a blueprint or template of objects having common properties and methods.

5. **What is an Object?**
   An object is an instance of a class. Data members and methods of a class cannot be used directly. We need to create an object (or instance) of the class to use them. In simple terms, they are the actual world entities that have a state and behavior.

6. **What is Encapsulation?**
   Java allows encapsulation, which is the practice of hiding the implementation details of an object from other objects. This is achieved through the use of access modifiers.

   It is implemented as the processes mentioned below:
   - Data hiding: A language feature to restrict access to members of an object. For example, private and protected members in C++.
   - Bundling of data and methods together: Data and methods that operate on that data are bundled together.
     For example, the data members and member methods that operate on them are wrapped into a single unit known as a class.

7. **What is Abstraction?**

   Java allows abstraction, which is the process of hiding complex implementation details and providing a simplified interface for the user. This can be achieved through abstract classes and interfaces

8. **What is Polymorphism?**

   The word “Polymorphism” means having many forms.

   Java supports polymorphism, which allows objects of different classes to be treated as if they were objects of a common superclass. This can be achieved through method overriding and method overloading.

   Polymorphism can be classified into two types based on the time when the call to the object or function is resolved. They are as follows:

   A. Compile Time Polymorphism

   B. Runtime Polymorphism

   A) Compile-Time Polymorphism Compile time polymorphism, also known as static polymorphism or early binding is the type of polymorphism where the binding of the call to its code is done at the compile time. Method overloading or operator overloading are examples of compile-time polymorphism.

   B) Runtime Polymorphism Also known as dynamic polymorphism or late binding, runtime polymorphism is the type of polymorphism where the actual implementation of the function is determined during the runtime or execution. Method overriding is an example of this method.

9. **What are the different types of Polymorphism?**

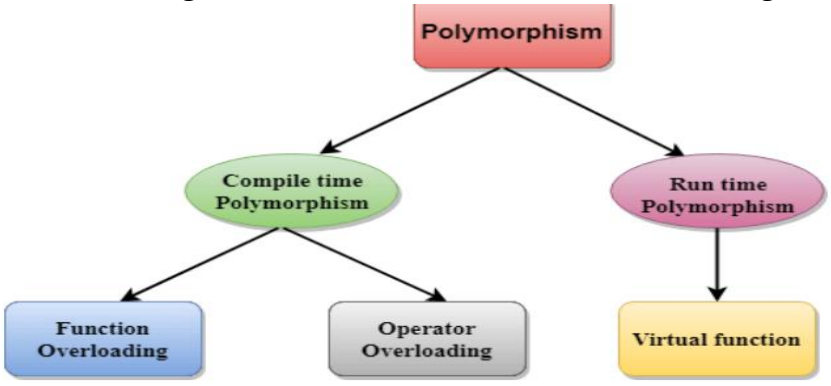
   Polymorphism can be classified into two types based on the time when the call to the object or function is resolved. They are as follows:

   A. Compile Time Polymorphism

   B. Runtime Polymorphism

   A) Compile-Time Polymorphism Compile time polymorphism, also known as static polymorphism or early binding is the type of polymorphism where the binding of the call to its code is done at the compile time. Method overloading or operator overloading are examples of compile-time polymorphism.

   B) Runtime Polymorphism Also known as dynamic polymorphism or late binding, runtime polymorphism is the type of polymorphism where the actual implementation of the function is determined during the runtime or execution. Method overriding is an example of this method.



10. What is Inheritance? What is its purpose?

The idea of inheritance is simple, a class is derived from another class and uses data and implementation of that other class. The class which is derived is called child or derived or subclass and the class from which the child class is derived is called parent or base or superclass. The main purpose of Inheritance is to increase code reusability. It is also used to achieve Runtime Polymorphism

11. Are there any limitations on Inheritance?

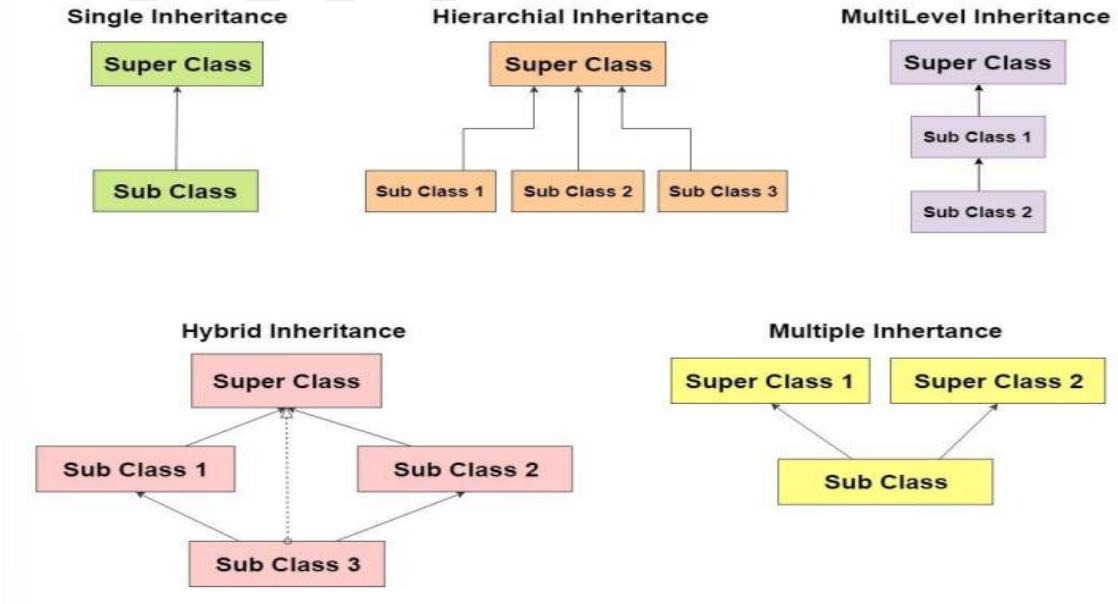
Yes, there are more challenges when you have more authority. Although inheritance is a very strong OOPs feature, it also has significant drawbacks.

- As it must pass through several classes to be implemented, inheritance takes longer to process.
- The base class and the child class, which are both engaged in inheritance, are also closely related to one another (called tightly coupled). Therefore, if changes need to be made, they may need to be made in both classes at the same time.
- Implementing inheritance might be difficult as well. Therefore, if not implemented correctly, this could result in unforeseen mistakes or inaccurate outputs.

12. What different types of inheritance are there?

Inheritance can be classified into 5 types which are as follows:

1) Single Inheritance: Child class derived directly from the base class
2) Multiple Inheritance: Child class derived from multiple base classes.
3) Multilevel Inheritance: Child class derived from the class which is also derived from another base class.
4) Hierarchical Inheritance: Multiple child classes derived from a single base class.
5) Hybrid Inheritance: Inheritance consisting of multiple inheritance types of the above specified.



13. What are access specifiers? What is their significance in OOPs?

Access specifiers are special types of keywords that are used to specify or control the accessibility of entities like classes, methods, and so on. Private, Public, and Protected are examples of access specifiers or access modifiers.

The key components of OOPs, encapsulation and data hiding, are largely achieved because of these access specifiers.

14. What are the advantages and disadvantages of OOPs?

Advantages of OOPs	Disadvantages of OOPs
OOPs provides enhanced code reusability.	The programmer should be well-skilled a should have excellent thinking in terms of objects as everything is treated as an object OOPs.
The code is easier to maintain and update.	Proper planning is required because OOPs is little bit tricky
It provides better data security by restricting da access and avoiding unnecessary exposure.	OOPs concept is not suitable for all kinds problems
Fast to implement and easy to	he length of the programs is much larger

15. What is the difference between Structured Programming and Object Oriented Programming?

Object-Oriented Programming	Structural Programming
Programming that is object oriented is built objects having a state and behavior.	program ' s logical structure is provided structural programming, which divides prograr into their corresponding functions
It follows a bottom-to-top approach.	It follows a Top-to-Down approach.
Restricts the open flow of data to authorized pa only providing better data security.	No restriction to the flow of data. Anyone c access the data.
In this, methods are written	n this, the method works dynamically,

16. What are some commonly used Object Oriented Programming Languages?

OOPs paradigm is one of the most popular programming paradigms. It is widely used in many popular programming languages such as:

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

17. What is the difference between overloading and overriding?

Method overloading is the ability to define multiple methods in a class with the same name but different parameter lists. The methods are differentiated based on the number or types of parameters they accept.

Method overriding is the process by which a subclass provides a specific implementation for a method that is already defined in its superclass. The overridden method in the subclass has the same name, return type, and parameters.

18. What is an abstract class?

An abstract class is a class that cannot be instantiated on its own. It may contain abstract methods (methods without a body) that must be implemented by its concrete subclasses. Abstract classes provide a common interface

19. What is an interface?

An interface is a contract that defines a set of methods that a class must implement. It allows multiple classes to adhere to the same interface, promoting a form of multiple inheritance. Interfaces only declare method signatures, not implementations.

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

Abstract Class	Interface
When an abstract class is inherited, however, t subclass is not required to supply the definition of t abstract method until and unless the subclass actua uses it.	When an interface is implemented, t subclass is required to specify all of t interface's methods as well as th implementation.
A class that is abstract can have both abstract and non-abstract methods.	An interface can only have abstra methods.
An abstract class can have final, non- final, static a non-static variables.	The interface has only static and fi variables.
Abstract class doesn ' t support multiple inheritanc	An interface supports multiple inheritance

21. How much memory does a class occupy?

Classes do not use memory. They merely serve as a template from which items are made. Now, objects actually initialize the class members and methods when they are created, using memory in the process

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

No. If the base class includes non-static methods, an object must be constructed. But no objects need to be generated if the class includes static methods. In this instance, you can use the class name to directly call those static methods

23. What is Constructor?

A constructor is a special method that is automatically called when an object is created. It initializes the object's attributes and prepares the object for use. Constructors often have the same name as the class.

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

The most common classification of constructors includes:

1) Default Constructor
2) Parameterized Constructor
3) Copy Constructor

1. Default Constructor

• The default constructor is a constructor that doesn't take any arguments. It is a non-parameterized constructor that is automatically defined by the compiler when no explicit constructor definition is provided.

• It initializes the data members to their default values.

2. Parameterized Constructor

• It is a user-defined constructor having arguments or parameters.

3. Copy Constructor

• A copy constructor is a member function that initializes an object using another object of the same class.

• In Python, we do not have built-in copy constructors like Java and C++ but we can make a workaround using different methods.

25. What is a destructor?

- A destructor is a method that is automatically called when the object is made of scope or destroyed.
- In C++, the destructor name is also the same as the class name but with the (~) tilde symbol as the prefix.
- In Python, the destructor is named __del__.
-

26. Can we overload the constructor in a class?

No. A destructor cannot be overloaded in a class. The can only be one destructor present in a class.

27. What is the virtual function?

A virtual function is a function that is used to override a method of the parent class in the derived class. It is used to provide abstraction in a class. In C++, a virtual function is declared using the virtual keyword, In Java, every public, non-static, and non-final method is a virtual function. Python methods are always virtual.

28. What is pure virtual function?

A pure virtual function, also known as an abstract function is a member function that doesn't contain any statements. This function is defined in the derived class if needed.

29. What is a static method?

A static method belongs to the class itself, not to instances of the class. It can be called using the class name and is often used for utility functions or operations that don't require instance-specific data.

30. What is a final class?

A final class is a class that cannot be subclassed. It prevents other classes from extending it and inheriting its behavior.

31. What is composition?

Composition is a design principle where a class contains objects of other classes as part of its attributes.

It allows creating complex structures by combining simpler classes

32. What is method hiding?

Method hiding is a concept in object-oriented programming where a subclass defines a method with the same name as a method in its superclass, but the subclass method doesn't override the superclass method.

Instead, it creates a new, independent method with the same name. This can lead to confusion and unexpected behavior, so it's generally recommended to use method overriding instead.

33. What is a constructor chaining?

Constructor chaining is the process of calling one constructor from another within the same class or between a superclass and a subclass. It allows for code reuse and efficient initialization.

34. What is the diamond problem in multiple inheritance?

The diamond problem occurs in languages that support multiple inheritance, where a class inherits from two classes that have a common base class.

This can lead to ambiguity in method resolution. Some languages provide mechanisms to handle this, like virtual inheritance.

35. What is a shallow copy and a deep copy?

A shallow copy copies the references of objects contained within an object, while a deep copy creates new instances of the objects contained.

A deep copy results in a completely independent copy of the original object and its contained objects.

36. What is a virtual method?

A virtual method is a method declared in a base class that can be overridden by its subclasses. The actual implementation invoked is determined by the runtime type of the object, supporting polymorphism.

37. How does Java achieve multiple inheritance?

Java achieves multiple inheritance through interfaces. A class can implement multiple interfaces, allowing it to inherit multiple sets of method declarations.

38. **What is the purpose of the final keyword in Java?**

The final keyword can be applied to classes, methods, and variables. A final class cannot be subclassed, a final method cannot be overridden, and a final variable cannot be reassigned after its initial assignment.

39. How is encapsulation related to data hiding?

Encapsulation involves bundling data and methods together.

Data hiding is the practice of making the internal data of an object inaccessible to the outside world, except through well-defined methods. Encapsulation helps achieve data hiding

40. What is the difference between composition and inheritance?

Inheritance creates a relationship between a subclass and a superclass, allowing the subclass to inherit properties and methods.

Composition involves creating an object within another object to achieve complex behavior without the tight coupling of inheritance

41. What is the role of a destructor in C++?

In C++ , a destructor is a special method with the same name as the class, preceded by a tilde (~). It is called when an object is about to be destroyed, allowing for cleanup tasks such as releasing resources or memory.

42. What is an Inline function?

An inline function is a technique used by the compilers and instructs to insert complete body of the function wherever that function is used in the program source code.

43. What is a friend function?

A friend function is a friend of a class that Is allowed to access to Public, private, or protected data in that same class. If the function is defined outside the class cannot access such information.

44. What is function overloading?

Function overloading is a regular function, but it is assigned with multiple parameters. It allows the creation of several methods with the same name which differ from each other by the type of input and output of the function.

45. What is operator overloading?

Operator Overloading is s method by which we can redefine or customize the behavior of certain operators in Programming. Therefore, Operator Overloading is a way to implement Polymorphism by providing different implementations to a single operators based on the data types on which they are operated.

Operator Overloading is a way to redefine the behavior of existing operators (like +, -, *, /) for user-defined types. We can specify how operators should behave when they are applied to user-defined data types or objects, providing a way to implement operations that are relevant to those objects.

46. What is a ternary operator?

The ternary operator is a conditional operator that takes three operands: a condition, a value to be returned if the condition is true, and a value to be returned if the condition is false. It evaluates the condition and returns one of the two specified values based on whether the condition is true or false.

47. What is the use of finalize method?

finalize is a method that is called by the garbage collector when an object is no longer in use.

48. What are the different types of arguments?

A parameter is a variable used during the declaration of the function or subroutine, and arguments are passed to the function body, and it should match with the parameter defined. There are two types of Arguments.

- Call by Value – Value passed will get modified only inside the function, and it returns the same value whatever it is passed into the function.
- Call by Reference– Value passed will get modified in both inside and outside the functions and it returns the same or different value.

49. What are the different access specifiers used in Java?

Java has 4 access specifiers.

- Public Can be accessed by any class or method
- Protected Can be accessed by the class of the same package, or by the sub-class of this class, or within the same class
- Default Are accessible only within the package, is the default option for all classes, methods and variables.
- Private Can be accessed only within the class

50. What is the difference between composition and inheritance?

Composition is a "has-a" relationship, where a class contains an object of another class as a member variable. Inheritance is an "is-a" relationship, where a subclass extends a superclass to inherit its attributes and methods.

51. What is the purpose of an abstract class?

An abstract class is a class that cannot be instantiated and is used as a base class for other classes to inherit from. It can contain abstract methods, which are declared but not implemented in the abstract class and must be implemented in the subclasses.

52. What are the differences between constructor and method of a class in Java?

Constructor is used for initialising the object state whereas method is used for exposing the object's behaviour. Constructors have no return type but Methods should have a return type. Even if it does not return anything, the return type is void. If the constructor is not defined, then a default constructor is provided by the java compiler. The constructor name should be equal to the class name. A constructor cannot be marked as final because whenever a class is inherited, the constructors are not inherited. A method can be defined as final but it cannot be overridden in its subclasses.

53. What is the diamond problem in Java and how is it solved?

The diamond problem is an issue that can arise in programming languages that support multiple inheritance, where a class inherits from two or more classes that have a common ancestor. This can cause ambiguity in the method resolution order, leading to unpredictable behaviour. In Java, multiple inheritance is not supported directly, but it can be simulated using interfaces. A class can implement one or more interfaces, effectively inheriting their properties and methods

54. What is the difference between local and instance variables in Java?

Instance variables are accessible by all the methods in the class. They are declared outside the methods and inside the class. These variables describe the properties of an object and remain

bound to it. Local variables are those variables present within a block, function, or constructor and can be accessed only inside them. The utilisation of the variable is restricted to the block scope

55. What is a Marker interface in Java?

Marker interfaces or tagging interfaces are those which have no methods and constants defined in them. They help the compiler and JVM get run time-related object information.

56. What is the super keyword?

The super keyword is used to refer to the parent class or superclass. It can be used to call methods and constructors from the superclass.

57. What is 'this' pointer?

THIS pointer refers to the current object of a class. THIS keyword is used as a pointer which differentiates between the current object with the global object. It refers to the current object.