Object Oriented Programming In Java Questions And Answers

1. What is OOPs?

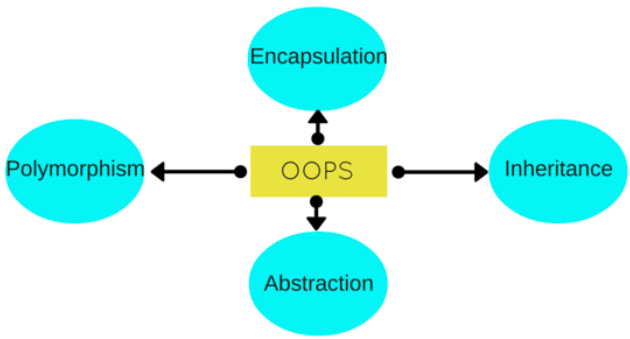
OOPs stands for Object-Oriented Programming system. It is a programming paradigm in which software design involves around data, or objects, rather than functions and logic. In OOPs, objects are data fields that have unique attributes and properties.

OOPs focuses on the objects that developers want to manipulate rather than the logic required to manipulate them. That's why large, complex and actively updated or maintained program is well suited for OOPs paradigm of programming.

2. What are the basic features of OOPs?

Following are the basic features of OOPs -

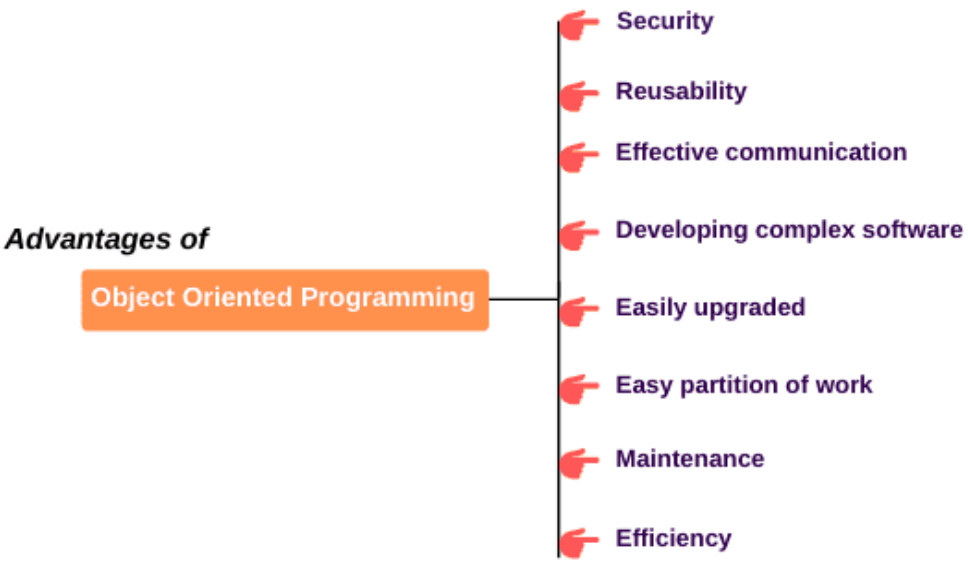
- Abstraction
- Encapsulation
- Inheritance
- Polymorphism



3. Why use OOPs?

OOPs provide the following advantages, that's why it is used.

- Code maintenance
- Reusability
- Security
- Better productivity
- Polymorphism flexibility
- Problem solving
- Easy troubleshooting
- Design benefits
- Data redundancy



4.What are the difference between OOP and SOP?

OOP	SOP
1. It stands for Object-Oriented Programming.	**It stands for Structural Oriented Programming.**
2. It is based on objects.	**It is based on functions.**
3. It follows Bottom-up programming approach.	**It follows Top-down programming approach.**
4. It is based on real world.	**It is based on unreal world.**
5. It provides data hiding so it is very secure.	**It doesn't provide data hiding so it is less secure.**
6. It provides reusability feature.	**It doesn't provide reusability feature.**
7. Eg. C++, Java, python etc.	**Eg. C, Pascal, Basic etc.**

5. What is a class?

A class is a collection of objects. Classes don’t consume any space in the memory.

It is a user defined data type that act as a template for creating objects of the identical type.

Once a class has been defined. A large number of objects can be created using the same class. Therefore, Class is considered as the blueprint for the object.

For eg. Fruit is a class and its objects are mango, apple etc. Furniture is a class and its objects are table, chair, desk etc.

Syntax :

```
class classname [extends inheritance]
{
      [field declararion]
      [method declaration]
}
```

6.What is an object?

An object is a real world entity which have properties and functionalities.

Object is also called an instance of class. Objects take some space in memory.

For eg. car, table, chair etc. are the example of objects.

Syntax of creation of object :

```
classname obj_reference = new classname();

                OR

Step I - classname obj_reference; // object declararion
Step II - obj_reference = new classname(); //object initialization
```

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

Class	Object
1. It is a collection of objects.	**It is an instance of a class.**
2. It doesn't take up space in memory.	**It takes space in memory.**
3. Class does not exist physically	**Object exist physically.**
4. Classes are declared just once	**Objects can be declared as and when required**

5. Eg. - Fruit, Vehicle, Laptop	**Eg. - Mango, Car, HP Laptop**

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

Class	Structure
1. Class is a collection of objects.	**Structure is a collection of variables of different data types under a single unit**
2. Class is used to combine data and methods together.	**Structure is used to grouping data.**
3. Class's objects are created on the heap memory.	**Structure's objects are created on the stack memory.**
4. A class can inherit another class.	**A structure can't inherit another structure.**
5. A class has all members private by default	**A structure has all members public by default**
6. Classes are ideal for larger or complex objects	**Structures are ideal for small and isolated model objects**

9. What is encapsulation?

Encapsulation is an striking feature of OOPs that provides data security. It is a mechanism of binding member data and member function together into a single place i.e. class.

The main advantage of encapsulation is that data is hidden and protected from access by outside non-member methods of a class. In other words, only member functions defined in a class will have access to the data.

In encapsulation, data(variables) are declared as private and methods are declared as public.

10. What are access specifiers?

Access specifiers are the most important part of object oriented programming paradigm. It allows us to restrict the scope or visibility of a package, class, constructor, methods, variables, or other data members.

By using access specifiers, we define how the members (attributes and methods) of a class can be accessed.

There are three types of most common access specifiers, which are following.

- Private
- Public
- Protected

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

Public Modifiers - When we declare any class, variable or method with public modifiers that means that class, variable or method is accessible throughout from within or outside the class, within or outside the package, etc.

It provides highest level of accessibility.

Private Modifiers - When we declare any class, variable or method with private modifiers that means that class, variable or method is not accessible from within or outside the class, within or outside the package, etc.

Private field or method can't be inherited to sub class. This provides lowest level of accessibility.

Protected Modifiers - When we declare any class, variable or method with protected modifiers that means that class, variable or method is accessible from classes in the same package, sub-classes in the same package, sub-classes in other packages but not accessible from classes in other packages.

We can summarize rules of public, private and protected access modifiers in following ways.

Access Modifiers	Accessible by classes in the same package	Accessible by classes in other packages	Accessible by sub-classes in the same package	Accessible by sub-classes in other packages
Private	**No**	**No**	**No**	**No**
Public	**Yes**	**Yes**	**Yes**	**Yes**
Protected	**Yes**	**No**	**Yes**	**Yes**

12. What is data abstraction?

Data abstraction is an important feature of OOPs that allows to hide unnecessary data from the user. This reduces program complexity efforts.

This means that it displays only the necessary information to the user and hides all the internal background details.

For example, when we order anything in a restaurant, we get our orders but we don't know all the background work that has been taken to process this order.

If we talk about data abstraction in programming language, the code implementation is hidden from the user

and only the necessary functionality is shown or provided to the user.

13. How to achieve data abstraction?

We can achieve data abstraction by using -

1. Abstract class
2. Interface

14. What is an abstract class?

Abstract class is that class which contains abstract method. Abstract methods are those methods which have only declaration not the implementation.

An abstract class is declared with abstract keyword. An abstract class can also contain non-abstract methods.

15. What is an interface?

An interface is a collection of abstract method(methods without definition). We can't instantiate an interface.

Interface only contains the final fields and abstract methods not the regular methods. Interface also don't contain constructor.

The member of the class may or may not be private but the members of the interface can never be private.

The abstract method of the interface must be overridden in its child classes and the interface supports the concept of multiple inheritance which was not supported using the classes.

An interface can't be extended by class, it can only be implemented by class.

Syntax :

```
interface interfacename


{
    [final field declaration]
    [abstract method declaration]
}
```

16. Differentiate between data abstraction and encapsulation.

Data Abstraction	Encapsulation
1. In data abstraction, we hide unnecessary information from the user and only show the necessary information.	**In encapsulation, we combine data and functions together at a single unit i.e. class.**
2. Abstraction is implemented using abstract class and interfaces.	**Encapsulation is implemented using access modifiers such as private, public and protected.**
3. Data abstraction is implemented at the design or interface level	**Encapsulation is implemented at the implementation level.**

17. Can we create an instance of an abstract class?

No, instance of an abstract class can't be created.

18. What is inheritance?

Inheritance is a very powerful feature of OOPs. It is a mechanism of acquiring properties or behaviors of existing

class to a new class. In other words, A class inherits data fields or methods from another class.

For eg. Animal is a class and other classes like Dog, cat etc. can inherit the common properties of animal class. This helps to write redundant free code and also reduces code length.

19. Why we use inheritance?

Inheritance provides following benefits -

- Code reusability
- Less development and maintenance costs
- Reduces code redundancy and supports code extensibility
- Save time and effort to write codes

20. What is superclass?

Super class is that class which is inherited by other class. In other words, super class is a class from which sub class inherits the properties and behaviors. It is also called base class or parent class.

21. What is a subclass?

Sub class is that class which inherits other class. In other words, sub class is a class which inherits the properties and behaviors of super class. It is also called child class or derived class.

20. What are the types of inheritance?

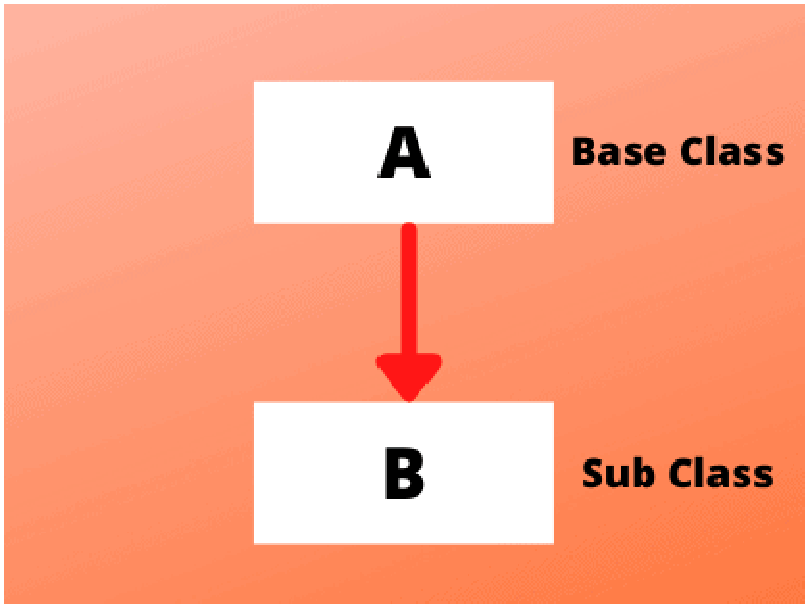
Inheritance have following types -

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

21. What is single inheritance?

When a class inherits properties and behavior of only one class then this type of inheritance is called single inheritance.

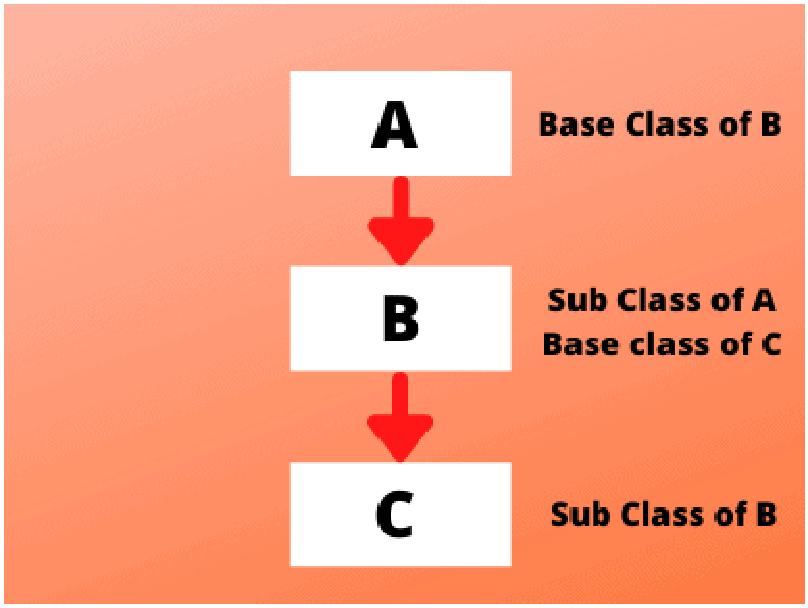
In other words, in single inheritance there is only one base class and only one sub class.



22. What is multilevel inheritance?

When a class inherits other class and then that inherited class further inherited by other class, this type of inheritance is called multilevel inheritance.

In multilevel inheritance, sub class contains the properties of the base class as well as properties of base class of its own base class.

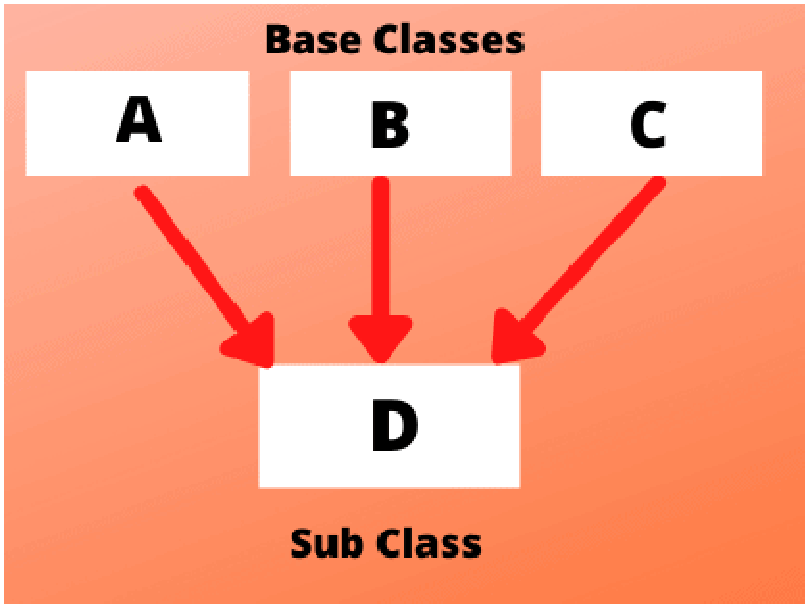


23. What is multiple inheritance?

When a class inherits the properties and behaviors of more than one class then this type of inheritance is called multiple inheritance.

In multiple inheritance, there is only one sub class but more than one super class.

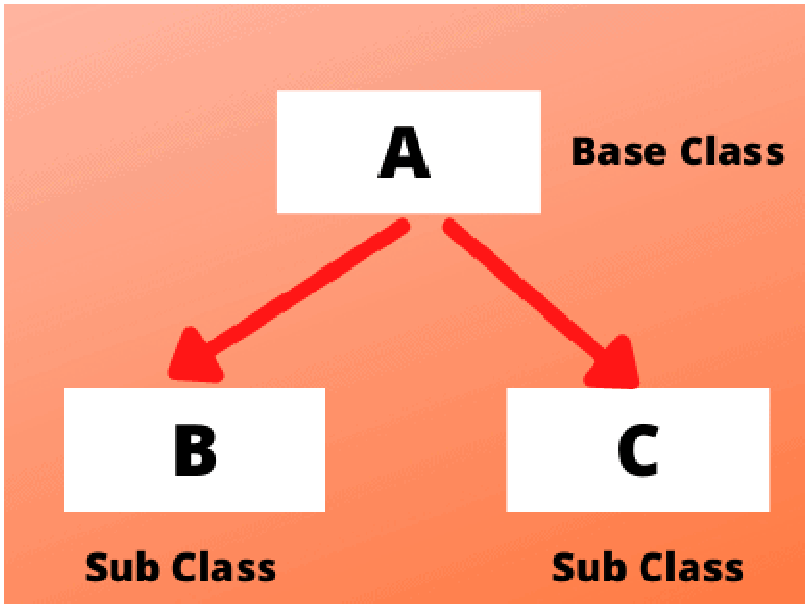
Java doesn't support multiple inheritance.



24. What is hierarchical inheritance?

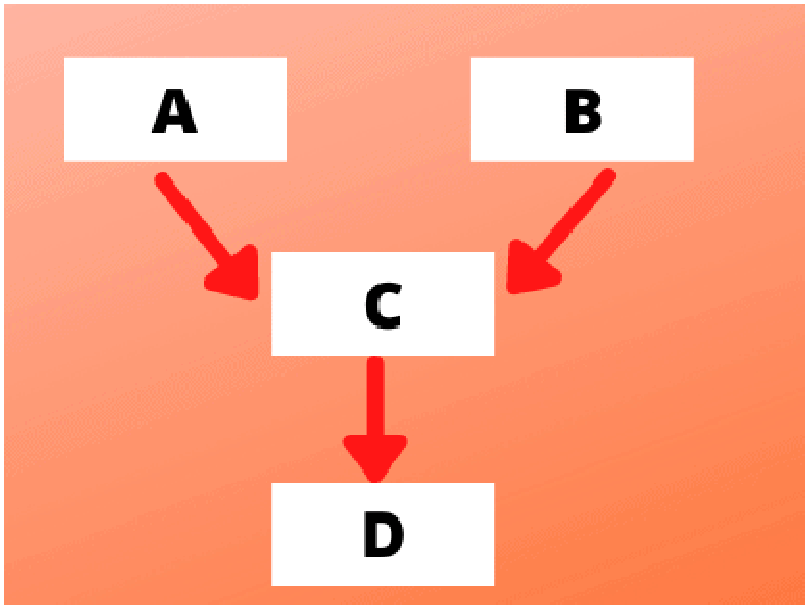
When more than one class inherits the properties or behavior of only one class then this type of inheritance is called hierarchical inheritance.

In hierarchical inheritance, there are only one parent class but many child class.



25. What is hybrid inheritance?

Hybrid inheritance is a combination of more than one type of inheritance.



26. Why Java doesn't support multiple inheritance?

Java doesn't support multiple inheritance because of following reasons -

Ambiguity Around The Diamond Problem

Multiple inheritance does complicate the design and creates problem during casting, constructor chaining etc.

So to simplify the language and reduce code complexity, Java doesn't support multiple inheritance.

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

Multiple Inheritance	Multilevel Inheritance
1. In multiple inheritance, one sub class inherits the properties and behaviors of more than one super class.	**In multilevel inheritance, a class inherits other class and then that inherited class further inherited by other class.**
2. It has two class levels namely, base class and derived	**It has three class levels namely, base class, intermediate class**

class.	**and derived class.**
3. It inherits from the base class.	**It inherits from the derived class.**

28. What are the limitations of inheritance?

- **Decreases Execution Speed -** Inheritance execution takes time and effort so it decreases the execution speed of code.
- **No Independence -** Inheritance increases the coupling between base class and derived class. A change in base class will affect all the child classes.
- **Uncertainty of results -** Improper use of inheritance may lead to wrong solutions.
- **Memory wastage -** Often, data members in the base class are left unused which may lead to memory wastage

29. What is polymorphism?

polymorphism is one of the major pillars of OOPs. It is a multiple form of a single entity.

It is process of performing a single task in different ways. In polymorphism, one method have multiple forms based

on the type of parameters, order of parameters, and number of parameters.

In simple words, Polymorphism is the ability of an object to take many forms.

30. What are the types of polymorphism?

There are two types of polymorphism -

- Compile time polymorphism
- Run time polymorphism

31. What is compile time polymorphism?

The polymorphism that takes place during compile time is called compile time polymorphism. It is also called static polymorphism or early binding.

In this type of polymorphism, a class contain multiple methods having same name but different signatures.

Example of CTP - Method Overloading, Operator Overloading.

32. What is runtime polymorphism?

The polymorphism that takes place during run time is called compile time polymorphism. It is also called dynamic polymorphism or late binding.

Runtime polymorphism refers to the process when a call to an overridden process is resolved at the run time.

In this type of polymorphism, the sub class and base class both contain methods having same name which can have different functionalities.

33. What is method overloading?

When a class contains multiple methods having same name but different signature, this is called method overloading.

In method overloading, multiple methods of same names performs different tasks within the same class.

It depends upon the number and type of argument in the argument list and doesn't depend upon the return type of the method.

This is an example of compile time polymorphism.

34. What is method overriding?

When base class contain a method and sub class also contain same name method of its parent class, this is called method overriding.

In other words, base class and subclass both have same name methods as well as signatures too.

Method overriding is an example of run time polymorphism.

Method overriding is used to provide the specific implementation of a method which is already provided by its superclass.

35. What is operator overloading?

Operator overloading is a mechanism in which the operator is overloaded to provide the special meaning to the user-defined data type.

It is an example of compile time polymorphism.

36. What is static function?

Static functions are those functions that can be called without creating an object of the class. That means, Static methods do not use any instance variables of any object of the class they are defined in.

Static methods can not be overridden. They are stored in heap space of the memory.

37. What are virtual functions?

Virtual function is a function or method used to override the behavior of the function in an inherited class with the same signature to achieve the polymorphism. Virtual function defined in the base class and overridden in the inherited class.

The Virtual function cannot be private, as the private functions cannot be overridden. It is used to achieve runtime polymorphism.

38. What are pure virtual functions?

A pure virtual function is that function which have no definition. That means a virtual function that doesn't need implementation is called pure virtual function.

A pure virtual function have not definitions but we must override that function in the derived class, otherwise the derived class will also become abstract class.

39. What is Constructor?

Constructor is a special type of member function which is used to initialize an object. It is similar as functions but it's name should be same as its class name and must have no explicit return type.

It is called when an object of the class is created. At the time of calling constructor, memory for the object is allocated in the memory.

We use constructor to assign values to the class variables at the time of object creation.

40. What are the types of Constructor?

Constructor have following types -

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

41. What is default constructor?

A constructor with 0 parameters is known as default constructor.

42. What is parameterized constructor?

The constructor method having the argument list or parameter list is called as parameterized constructor as it initializes the fields with the values of the parameters.

43. What is copy constructor?

A copy constructor is that constructor which use existing object to create a new object. It copy variables from another object of the same class to create a new object.

44. What is static constructor?

A static constructor is automatically called when the first instance is generated, or any static member is referenced. The static constructor is explicitly declared by using a static keyword. However, the static constructor is not supported in Java.

45. What is private constructor?

Java enables us to declare a constructor as private. We can declare a constructor private by using the private access specifier. Note that if a constructor is declared private, we cannot create an object of the class. Instead, we can use this private constructor in Singleton Design Pattern.

46. What is destructor?

Destructor is a type of member function which is used to destroy an object. It is called automatically when the object goes out of scope or is explicitly destroyed by a call to delete.

It destroy the objects when they are no longer in use. A destructor has the same name as the class, preceded by a tilde (~).

47. What is Constructor Overloading?

The constructor method of the class may or may not have the argument list, and it has no return type specified and we also know that method overloading depends only on the argument list and not on the return type. Thus, we can say that the constructor method can be overloaded.