**1. What is Object-oriented programming?**

This is the most basic OOPs interview question. Your answer should be:

Object-Oriented Programming refers to the programming paradigm defined using objects

instead of only functions and methods. The objects contain data, called fields or attributes,

and methods that provide the logic or supporting code. It provides capabilities such as inheritance,

polymorphism, encapsulation, abstraction.

**2. What are the main features of object-oriented programming?**

In this OOPs interview question, make sure you list the four main features:

Inheritance

Encapsulation

Polymorphism

Data Abstraction

**3. What are the advantages of Object-oriented programming?**

One of the most common oops interview question.

* Problems of any level of complexity can be supported by object-oriented programming.
* Highly complex problems can be handled by object-oriented programming
* It provides an efficient mechanism for code reuse using inheritance which reduces redundancy
* It provides a mechanism for hiding data
* It is based on a bottom-up approach
* It offers flexibility through polymorphism
* It improves maintainability of the code

**4. What is Structural programming?**

Structural programming refers to the traditional method of programming, which is based on functions.

The overall program logic is divided into functions to provide a logical structure.

It is based on a top-down approach. Structural programming is suitable for easy to moderately complex problems.

**5. What is a class?**

A class defines the template or the definition of an object.

It is used for creating objects at run time.

It provides the data structure, provides initial values for the attributes,

and methods that provide the logic for the intended behaviour of the object.

The class does not consume memory at runtime. A class refers to a logical entity.

E.g., a vehicle may be defined as a class.

**6. What do you mean by an object?**

An object refers to the run time instance created from the class during program execution. Objects can refer to real-world entities that have attributes or properties and methods to support the behaviour. Objects consume memory space when they are initialized.

**7. Is it always necessary to create objects from class?**

No, it is possible to call a base class method if it is defined as a static method.

**8. What is a constructor?**

A constructor method is used for initializing the objects. They are special types of methods and have the same name as the class.

**9. List the various types of constructors**

*Multiple types of constructors that are supported across multiple object-oriented programming languages are:*

* Default constructor
* Copy constructor
* Static constructor
* Private constructor
* Parameterized constructor

**10. What is a destructor?**

A destructor is a method used for freeing up the resources allocated to an object. This method is automatically invoked when an object is being destroyed.

**11. What is meant by a copy constructor?**

A copy constructor helps in cloning objects by replicating the values from one object into another object which belongs to the same class.

**12. Can you please highlight the difference between a class and a structure?**

A class means a user-defined template from which objects are created at runtime. A class is made up of methods that provide the logic for various behaviours supported by the objects.

A structure means a user-defined combination of attributes of various data types.

**13. Can you please explain the concept of inheritance with an example?**

Inheritance is a powerful feature of object-oriented programming which allows classes to inherit properties and methods from other classes. This helps improve code reuse.

For example, a base class represents a logical concept, such as a vehicle that may define only the common properties shared by all types of vehicles. However, child classes can inherit from this base class to define more specific types of classes such as a truck, a car, or a bus. In this case, the child classes will inherit the common attributes of the vehicle, and will be able to define attributes, method specific to its own.

**14. What are the limitations of inheritance?**

The inheritance requires more processing time for the programs as it has to navigate various classes during execution. Due to inheritance, the parent and child class are tightly coupled. When any changes are needed in the logic, it may require changes in both parent and child classes.

If the inheritance is not correctly implemented, it can lead to undesired results.

**15. What are the various types of inheritance?**

* Single
* Multiple
* Multi-level
* Hierarchical
* Hybrid

**16. What is the meaning of hierarchical inheritance?**

When multiple subclasses inherit a base class, it is called hierarchical inheritance.

**17. Distinguish between multiple and multi-level inheritances?**

In the case of the multiple inheritance, a class inherits more than one parent class. In contrast, multi-level inheritance means that class inherits from another class, which is a subclass of some other parent class.

**18. How do you define hybrid inheritance?**

The hybrid inheritance is defined as the usage of multiple and multilevel inheritance in a single class.

**19. What is a subclass?**

The child class which inherits from another class is referred to as the subclass.

**20. Define a superclass?**

A superclass implies a class from which other classes inherit. e.g., the vehicle will be referred to as superclass of classes car, bus, or truck if they all inherited from the same superclass.

**21. What is meant by an interface?**

An interface allows a declaration of methods without providing a definition.

You cannot create objects from the interface. When a class implements an interface, it needs to implement the methods provided by the interface.

**22. What is polymorphism?**

Polymorphism is a significant feature of object-oriented programming. It means an ability to exist in multiple forms. A single interface can be implemented in multiple ways by providing various definitions.

**23. What is meant by static polymorphism?**

The static polymorphism or static binding allows us to link a function with objects during compilation. It can be implemented by method overloading of operator overloading.

**24. What is meant by dynamic polymorphism?**

A dynamic polymorphism or dynamic binding allows for a call to an overridden method at the run time.

**25. What is method overloading?**

One of the most common oops interview question. The method overloading is a very useful feature of object-oriented programming in which multiple methods can have the same method name; however, they have different arguments. The call to the method is resolved based on the arguments.

**26. What is the meaning of method overriding?**

Method overriding allows the child class to redefine methods of parent class by applying its implementations. However, the method name, arguments, and return types remain the same.

**27. Can you explain what operator overloading is?**

The term operator overloading means that depending on the arguments passed, the operators’ behaviour can be changed. However, it works only for user-defined types.

**28. How do you explain the difference between overloading and overriding?**

Overloading a method means that multiple methods share the same method name but have different arguments. However, in the case of the overriding, the child class can redefine the implementation of a method by retaining the same arguments. Another difference is that the overloading is resolved at compile-time while overriding is resolved at run time.

**29. What do you know about encapsulation?**

One of the most common OOPs interview question. Encapsulation is an important feature of object-oriented programming. It allows the binding of the data and the logic together in a single entity. It also allows the hiding of data.

**30. What is meant by data abstraction?**

The data abstraction refers to the ability of object-oriented programming that allows hiding the implementation details of logic yet allows for access to only important information.

**31. How can data abstraction be accomplished?**

Data abstraction can be accomplished through either an abstract class or an abstract method.

**32. What is meant by abstract class?**

Any OOPS Interview Question and Answers guide won’t complete without this question. An abstract class is made of abstract methods. The abstract methods are only declared, however, not implemented. When a subclass needs to use the methods, it needs to implement those methods.

**33. Can you please elaborate on ‘access specifiers’?**

*Support this* ***OOPs interview question’s*** *answer with examples:*

Access specifiers are special keywords that control the accessibility of methods or classes etc. They are also called access modifiers and are used to achieve the encapsulation. e.g., the keywords public, private, and protected are some examples of access specifiers.

**34. How do you create an instance of an abstract class?**

You cannot create an instance of an abstract class since it lacks implementation logic in its methods. You first need to create a subclass that implements all the methods before an object can be initialized.

**35. What is a virtual function?**

A virtual function is defined in the parent class and may have definitions implemented. A subclass can override these definitions.

**36. What is a pure virtual function?**

A pure virtual function is only declared in the parent class. It is also referred to as an abstract function. Pure virtual functions do not contain any definition in the base class. They must be redefined in the subclass for the implementation needed.

**37. Distinguish between data abstraction and encapsulation.**

Data abstraction is the ability to hide unwanted information.

The encapsulation refers to the ability to hide the data as well as the method together.

**38. What are the differences between interfaces and abstract classes?**

It is one of the general oops interview questions and answers guide. An abstract class can support both abstract and non-abstract methods. However, the interface allows only abstract methods.

In the case of an abstract class, both final and non-final variables are supported. However, the interface has variables that are, by default, defined as final.

The abstract class can have private, and public attributes, but interfaces have attributes as public by default.

**39. What is a final variable?**

A final variable means a variable whose value doesn’t change.

**40. What is meant by an exception?**

An exception is an event raised during a program execution caused by undesirable input or a condition that prevents further processing. An exception causes an interruption in the program’s normal execution and must be handled via exception handling logic to avoid the program’s termination.

**41. Define exception handling**

Exception handling refers to the mechanism used for handling the exceptions raised during program execution. It allows for the graceful handling of undesirable results.

**42. Is an error basically the same as an exception?**

An error means a problem that the program should not catch while the exception implies a condition that should be caught by the program.

**43. What is a try-catch block?**

A try-catch block is used for exception handling. The set of statements that may cause a potential error are enclosed in a try block. When an exception is raised, it is caught by the catch block. The logic to handle an exception is placed inside the catch block.

**44. What is a finally block?**

A ‘finally’ block is used for executing essential statements such as to free the memory, close files, or database connections, even if an exception occurs. The finally block always runs.

**45. What is the method ‘finalize’ used for?**

The finalize method is called to free the unused resources before the garbage collector gets initiated.

**46. What is a Garbage Collection, and how does it work?**

The garbage collection is the ability of the programming language to perform automatic memory management. It automatically frees up the memory by removing the objects that are no longer required.

**47. Should you always use Object-oriented programming? Are there any limitations of Object-oriented programming?**

This is one of the advanced oops interview question. Though object-oriented programming offers many advantages, it has some disadvantages too. First of all, it has a steep learning curve compared to procedural programming. It may take a while to get used to thinking and program in terms of objects for many people. Secondly, it may take more experience to design a program in terms of objects. Using OOPs concepts for smaller programming tasks may not be efficient.

**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

**Can the overridden methods in case of inheritance have different return type ?**

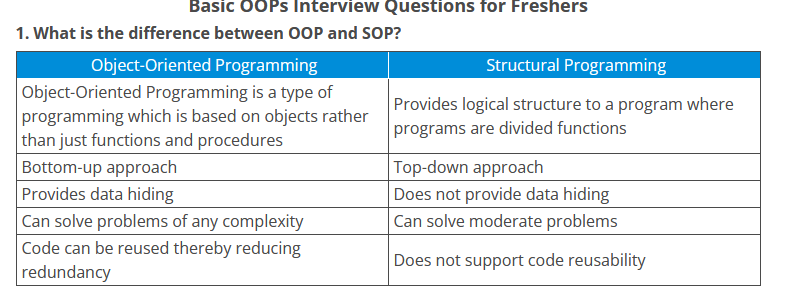
Answer : By default we tend to say NO , overridden method in sub class should have same signature as that of in base class. But this is half correct ,

Correct answer is : Overridden method can have different return type only in case if they are returning the sub-type.

Example :

1. public class base {
3. base show(){
5. System.out.println("base class");
7. return new base();
9. }
10. }
12. public class sub extends base{
14. @Override
15. sub show(){
17. System.out.println("sub class");
19. return new sub();
21. }
22. }
24. public class Test{
26. public static void main(String []args) {
28. sub obj = new sub();
30. obj.show();
31. }
32. }

In above example show() method return instance of that class , still its an overridden method



**“*How would you describe the fundamental difference between Encapsulation and Abstraction to a classroom of 7th grade students?*”**

Đây là một thách thức nhiều mặt mà hầu hết các nhà phát triển đều thất bại vì họ không trả lời câu hỏi như nó đã được đặt ra. Hầu hết cố gắng vẽ một bức tranh dễ hiểu để xác định từng khái niệm bằng hình ảnh đơn giản, nhưng họ không trả lời được câu hỏi “Mô tả sự khác biệt giữa hai khái niệm”.

Một điều khác khiến điều này trở thành một thách thức là họ không thể đưa ra câu trả lời mà bạn mong đợi trong một lớp Khoa học Máy tính. Các ứng viên cần phải xây dựng một câu trả lời phù hợp với một đứa trẻ 12 tuổi.

Trừu tượng là thể hiện sự đơn giản bên ngoài và Encapsulation là để che giấu sự phức tạp bên trong.

Câu hỏi nhằm đặt họ vào gót chân của họ và xem họ có thể lấy lại bình tĩnh và bắt đầu câu hỏi nhanh như thế nào. Một trong những điều khiến câu hỏi này trở thành một câu hỏi đầy thách thức là các khái niệm hoàn toàn trái ngược nhau, nhưng việc triển khai chúng có chung một chủ đề: Bạn phải ẩn một thứ gì đó với người dùng để triển khai từng phương pháp.

Giả sử chúng tôi muốn người dùng tương tác với một đối tượng có tên là Tivi bằng cách sử dụng một trong các phương pháp này.

Sử dụng Encapsulation, bạn sẽ ẩn tất cả hệ thống dây điện và mạch thành phần, bởi vì không ai cần hiểu tất cả kỹ thuật đó hoạt động như thế nào để xem TV.

Sử dụng Trừu tượng, bạn sẽ ẩn mọi thứ trừ màn hình TV và giao diện trình nhấp chuột để bạn có thể xem hình ảnh và sử dụng điều khiển.

Basic:

* Difference between *overloading* and *overriding* ?
* What are the different *types of constructors* ?
* Difference between *virtual* and *pure virtual* functions ?
* What are the different ways for a method to be overloaded?

Average:

* Explain *Static binding and Dynamic binding*.
* Difference between *assignment operator* and *copy constructor* ?
* Define an *abstract class*.

Advance:

* Define a *Singleton* class. What are its advantages ? How is it different form a static class ?
* Define a factory design pattern. What are its usage ?