**OOPS Interview Questions**

OOPS Interview Questions

**What is method overriding?**

Method overriding occurs when sub class declares a method that has the same type arguments as a method declared by one of its superclass. The key benefit of overriding is the ability to define behavior that’s specific to a particular subclass type.  
Note:

* *The overriding method cannot have a more restrictive access modifier than the method being overridden (Ex: You can’t override a method marked public and make it protected).*
* *You cannot override a method marked static*
* *You cannot override a method marked final*

**What is method overloading?**

Method Overloading means to have two or more methods with same name in the same class with different arguments. The benefit of method overloading is that it allows you to implement methods that support the same semantic operation but differ by argument number or type.  
Note:

* *Overloaded methods MUST change the argument list*
* *Overloaded methods CAN change the access modifier*
* *Overloaded methods CAN change the return type*
* *A method can be overloaded in the same class or in a subclass*
* *Overloaded methods CAN declare new or broader checked exceptions*

**What is the difference between abstraction and encapsulation?**

* nAbstraction  
  focuses on the outside view of an object (i.e. the interface)  
  **Encapsulation**  
  (information hiding) prevents clients from seeing it’s inside view, where the behavior of the abstraction is implemented.
* **Abstraction** solves the problem in the design side while **Encapsulation** is the Implementation.
* **Encapsulation** is the deliverables of Abstraction. Encapsulation barely talks about grouping up your abstraction to suit the developer needs.

**What is super?**

Super is a keyword which is used to access the method or member variables from the superclass. If a method hides one of the member variables in its superclass, the method can refer to the hidden variable through the use of the super keyword. In the same way, if a method overrides one of the methods in its superclass, the method can invoke the overridden method through the use of the super keyword.  
Note:

* *You can only go back one level.*
* *In the constructor, if you use super(), it must be the very first code, and you cannot access any*this.xxx *variables or methods to compute its parameters.*

**What is runtime polymorphism or dynamic method dispatch?**

In Java, runtime polymorphism or dynamic method dispatch is a process in which a call to an overridden method is resolved at runtime rather than at compile-time. In this process, an overridden method is called through the reference variable of a superclass. The determination of the method to be called is based on the object being referred to by the reference variable.

**When should I use abstract classes and when should I use interfaces?**

**Use Interfaces when…**

* If various implementations only share method signatures then it is better to use Interfaces.
* you need some classes to use some methods which you don’t want to be included in the class, then you go for the interface, which makes it easy to just implement and make use of the methods defined in the interface.
* You see that something in your design will change frequently.

**Use Abstract Class when…**

* If various implementations are of the same kind and use common behavior or status then abstract class is better to use.
* Abstract classes are an excellent way to create planned inheritance hierarchies. They’re also a good choice for nonleaf classes in class hierarchies.
* When you want to provide a generalized form of abstraction and leave the implementation task with the inheriting subclass.

**Can there be an abstract class with no abstract methods in it?**

Yes, there can be an abstract class without abstract methods.

**What modifiers are allowed for methods in an Interface?**

Only public and abstract modifiers are allowed for methods in interfaces.

**Can we create an object for an interface?**

Yes, it is always necessary to create an object implementation for an interface. Interfaces cannot be instantiated in their own right, so you must write a class that implements the interface and fulfill all the methods defined in it.

**Can overloaded methods be override too?**

Yes, derived classes still can override the overloaded methods. Polymorphism can still happen. Compiler will not binding the method calls since it is overloaded, because it might be overridden now or in the future.

**When you declare a method as abstract, can other nonabstract methods access it?**

Yes, other nonabstract methods can access a method that you declare as abstract.

**Do interfaces have member variables?**

Interfaces may have member variables, but these are implicitly public, static, andfinal- in other words, interfaces can declare only constants, not instance variables that are available to all implementations and may be used as key references for method arguments for example.

**Can we instantiate an interface?**

You can’t instantiate an interface directly, but you can instantiate a class that implements an interface.

**Can we instantiate an abstract class?**

An abstract class can never be instantiated. Its sole purpose is to be extended (subclassed).

**What are the principle concepts of OOPS?**

There are four principle concepts upon which object oriented design and programming rest. They are:

* Abstraction
* Polymorphism
* Inheritance
* Encapsulation

**Explain the different forms of Polymorphism.**

There are two types of polymorphism one is

**Compile time polymorphism**

and the other is run time polymorphism. Compile time polymorphism is method overloading.

**Runtime time polymorphism**

is done using inheritance and interface.

Note: *From a practical programming viewpoint, polymorphism manifests itself in three distinct forms in Java:*

* *Method overriding through inheritance*
* *Method overloading*
* *Method overriding through the Java interface*

**Is it possible to override the main method?**

NO, because main is a static method. A static method can’t be overridden in Java.

**How to invoke a superclass version of an Overridden method?**

To invoke a superclass method that has been overridden in a subclass, you must either call the method directly through a superclass instance, or use the super prefix in the subclass itself. From the point of the view of the subclass, the super prefix provides an explicit reference to the superclass’ implementation of the method.

// From subclass

super.overriddenMethod();

**How do you prevent a method from being overridden?**

To prevent a specific method from being overridden in a subclass, use the final modifier on the method declaration, which means “this is the final implementation of this method”, the end of its inheritance hierarchy.

public final void exampleMethod() {  
// Method statements  
}

**How does Java implement polymorphism?**

(Inheritance, Overloading and Overriding are used to achieve Polymorphism in java).  
Polymorphism manifests itself in Java in the form of multiple methods having the same name.

* In some cases, multiple methods have the same name, but different formal argument lists (overloaded methods).
* In other cases, multiple methods have the same name, same return type, and same formal argument list (overridden methods).

**What is Abstraction?**

Abstraction refers to the act of representing essential features without including the background details or explanations.

**What is Encapsulation?**

Encapsulation is a technique used for hiding the properties and behaviors of an object and allowing outside access only as appropriate. It prevents other objects from directly altering or accessing the properties or methods of the encapsulated object.

**What is Dynamic Binding?**

Binding refers to the linking of a procedure call to the code to be executed in response to the call. Dynamic binding (also known as late binding) means that the code associated with a given procedure call is not known until the time of the call at run-time. It is associated with polymorphism and inheritance.

**What is Polymorphism?**

Polymorphism is briefly described as “one interface, many implementations.” Polymorphism is a characteristic of being able to assign a different meaning or usage to something in different contexts – specifically, to allow an entity such as a variable, a function, or an object to have more than one form.

**What is Inheritance?**

* Inheritance is the process by which objects of one class acquire the properties of objects of another class.
* Inheritance is done by using the keyword extends.
* The class that does the inheriting is called a subclass.
* A class that is inherited is called a superclass.
* The two most common reasons to use inheritance are:
  + To use polymorphism
  + To promote code reuse

**What is an Interface?**

An interface is a description of a set of methods that conforming implementing classes must have.

Note:

* *You can’t mark an interface as final.*
* *An Interface cannot extend anything but another interfaces.*
* *Interface variables must be static.*

**What is a marker interface?**

Marker interfaces are those which do not declare any required methods, but signify their compatibility with certain operations. The java.io.Serializable interface and Cloneable are typical marker interfaces. These do not contain any methods, but classes must implement this interface in order to be serialized and de-serialized.

**What is an abstract class?**

Abstract classes are classes that contain one or more abstract methods. An abstract method is a method that is declared, but contains no implementation.

Note:

* *Abstract classes may not be instantiated, and require subclasses to provide implementations for the abstract methods.*
* *You can’t mark a class as both abstract and final.*
* *If even a single method is abstract, the whole class must be declared abstract.*

**What are the differences between Interface and Abstract class?**

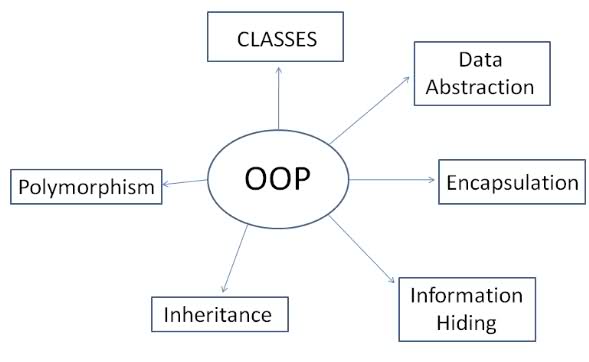
|  |  |
| --- | --- |
| **Abstract Class** | **Interfaces** |
| An abstract class can provide complete, default code and/or just the details that have to be overridden. | An interface cannot provide any code at all,just the signature. |
| An abstract class can have instance variables. | An Interface cannot have instance variables. |
| An abstract class can have any visibility: public, private, protected. | An Interface visibility must be public (or) none. |
| In case of abstract class, a class may extend only one abstract class. | A Class may implement several interfaces. |
| An abstract class can have non-abstract methods. | All methods of an Interface are abstract. |
| Abstract classes are fast. | Interfaces are slow as it requires extra indirection to find corresponding method in the actual class. |
| If we add a new method to an abstract class then we have the option of providing default implementation and therefore all the existing code might work properly. | If we add a new method to an Interface then we have to track down all the implementations of the interface and define implementation for the new method. |
| An abstract class can contain constructors. | An Interface cannot contain constructors. |

**What are the differences between method overloading and method overriding?**

|  |  |  |
| --- | --- | --- |
|  | **Overloaded Method** | **Overridden Method** |
| **Arguments** | Must change | Must not change |
| **Return type** | Can change | Can’t change except for covariant returns |
| **Exceptions** | Can change | Can reduce or eliminate. Must not throw new or broader checked exceptions |
| **Access** | Can change | Must not make more restrictive (can be less restrictive) |
| **Invocation** | Reference type determines which overloaded version is selected. Happens at compile time. | Object type determines which method is selected. Happens at runtime. |

/////////////////////////////////////////////////////////////////////////////////////////////////////////////

**Top 50 OOP Interview Questions**

[](http://4.bp.blogspot.com/-k83B90muUnQ/UcFi1aQmuUI/AAAAAAAAAJg/QuAlCkuNZFI/s1600/oopc.jpg)

**1.       What is OOPS?**

OOPS is abbreviated as Object Oriented Programming system in which programs are considered as a collection of objects. Each object is nothing but an instance of a class.

**2.       Write basic concepts of OOPS?**

Following are the concepts of OOPS and are as follows:.

1. Abstraction.
2. Encapsulation.
3. Inheritance.
4. Polymorphism.

**3.       What is a class?**

A class is simply a representation of a type of object. It is the blueprint/ plan/ template that describe the details of an object.

**4.       What is an object?**

Object is termed as an instance of a class, and it has its own state, behavior and identity.

**5.       What is Encapsulation?**

Encapsulation is an attribute of an object, and it contains all data which is hidden. That hidden data can be restricted to the members of that class.

Levels are Public, Protected, Private, Internal and Protected Internal.

**6.       What is Polymorphism?**

Polymorphism is nothing but assigning behavior or value in a subclass to something that was already declared in the main class. Simply, polymorphism takes more than one form.

**7.       What is Inheritance?**

Inheritance is a concept where one class shares the structure and behavior defined in another class. If inheritance applied on one class is called Single Inheritance, and if it depends on multiple classes, then it is called multiple Inheritance.

**8.       What are manipulators?**

Manipulators are the functions which can be used in conjunction with the insertion (<<) and extraction (>>) operators on an object. Examples are endl and setw.

**9.       Define a constructor?**

Constructor is a method used to initialize the state of an object, and it gets invoked at the time of object creation. Rules for constructor are:.

* Constructor Name should be same as class name.
* Constructor must have no return type.

**10.   Define Destructor?**

Destructor is a method which is automatically called when the object is made of scope or destroyed. Destructor name is also same as class name but with the tilde symbol before the name.

**11.   What is Inline function?**

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.

**12.   What is a virtual function?**

Virtual function is a member function of class and its functionality can be overridden in its derived class. This function can be implemented by using a keyword called virtual, and it can be given during function declaration.

Virtual function can be achieved in C++, and it can be achieved in C Language by using function pointers or pointers to function.

**13.   What is friend function?**

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.

Friend can be declared anywhere in the class declaration, and it cannot be affected by access control keywords like private, public or protected.

**14.   What is function overloading?**

Function overloading is defined as a normal function, but it has the ability to perform different tasks. It allows creation of several methods with the same name which differ from each other by type of input and output of the function.

Example

void add(int& a, int& b);

void add(double& a, double& b);

void add(struct bob& a, struct bob& b);

**15.   What is operator overloading?**

Operator overloading is a function where different operators are applied and depends on the arguments. Operator,-,\* can be used to pass through the function, and it has their own precedence to execute.

Example:

class complex {  
double real, imag;

public:  
complex(double r, double i) :  
real(r), imag(i) {}

complex operator+(complex a, complex b);  
complex operator\*(complex a, complex b);  
complex& operator=(complex a, complex b);  
}

a=1.2, b=6

**16.   What is an abstract class?**

An abstract class is a class which cannot be instantiated. Creation of an object is not possible with abstract class, but it can be inherited. An abstract class can contain only Abstract method.

**17.   What is a ternary operator?**

Ternary operator is said to be an operator which takes three arguments. Arguments and results are of different data types, and it is depends on the function. Ternary operator is also called as conditional operator.

**18.   What is the use of finalize method?**

Finalize method helps to perform cleanup operations on the resources which are not currently used. Finalize method is protected, and it is accessible only through this class or by a derived class.

**19.   What are 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, 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 it 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.

**20.   What is super keyword?**

Super keyword is used to invoke overridden method which overrides one of its superclass methods. This keyword allows to access overridden methods and also to access hidden members of the superclass.

It also forwards a call from a constructor to a constructor in the superclass.

**21.   What is method overriding?**

Method overriding is a feature that allows sub class to provide implementation of a method that is already defined in the main class. This will overrides the implementation in the superclass by providing the same method name, same parameter and same return type.

**22.   What is an interface?**

An interface is a collection of abstract method. If the class implements an inheritance, and then thereby inherits all the abstract methods of an interface.

**23.   What is exception handling?**

Exception is an event that occurs during the execution of a program. Exceptions can be of any type – Run time exception, Error exceptions. Those exceptions are handled properly through exception handling mechanism like try, catch and throw keywords.

**24.   What are tokens?**

Token is recognized by a compiler and it cannot be broken down into component elements. Keywords, identifiers, constants, string literals and operators are examples of tokens.

Even punctuation characters are also considered as tokens – Brackets, Commas, Braces and Parentheses.

**25.   Difference between overloading and overriding?**

Overloading is static binding whereas Overriding is dynamic binding. Overloading is nothing but the same method with different arguments, and it may or may not return the same value in the same class itself.

Overriding is the same method names with same arguments and return types associates with the class and its child class.

**26.   Difference between class and an object?**

An object is an instance of a class. Objects hold any information, but classes don’t have any information. Definition of properties and functions can be done at class and can be used by the object.

Class can have sub-classes, and an object doesn’t have sub-objects.

**27.   What is an abstraction?**

Abstraction is a good feature of OOPS, and it shows only the necessary details to the client of an object. Means, it shows only necessary details for an object, not the inner details of an object. Example – When you want to switch on television, it not necessary to show all the functions of TV. Whatever is required to switch on TV will be showed by using abstract class.

**28.   What are access modifiers?**

Access modifiers determine the scope of the method or variables that can be accessed from other various objects or classes. There are 5 types of access modifiers, and they are as follows:

* Private.
* Protected.
* Public.
* Friend.
* Protected Friend.

**29.   What is sealed modifiers?**

Sealed modifiers are the access modifiers where it cannot be inherited by the methods. Sealed modifiers can also be applied to properties, events and methods. This modifier cannot be applied to static members.

**30.   How can we call the base method without creating an instance?**

Yes, it is possible to call the base method without creating an instance. And that method should be,.

Static method.

Doing inheritance from that class.-Use Base Keyword from derived class.

**31.   What is the difference between new and override?**

The new modifier instructs the compiler to use the new implementation instead of the base class function. Whereas, Override modifier helps to override the base class function.

**32.   What are the various types of constructors?**

There are three various types of constructors, and they are as follows:

-          Default Constructor – With no parameters.

-          Parametric Constructor – With Parameters. Create a new instance of a class and also passing arguments simultaneously.

-          Copy Constructor – Which creates a new object as a copy of an existing object.

**33.   What is early and late binding?**

Early binding refers to assignment of values to variables during design time whereas late binding refers to assignment of values to variables during run time.

**34.   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. Basically, it refers to the current object.

**35.   What is the difference between structure and a class?**

Structure default access type is public, but class access type is private. A structure is used for grouping data whereas class can be used for grouping data and methods. Structures are exclusively used for data and it doesn’t require strict validation, but classes are used to encapsulates and inherit data which requires strict validation.

**36.   What is the default access modifier in a class?**

The default access modifier of a class is Private by default.

**37.   What is pure virtual function?**

A pure virtual function is a function which can be overridden in the derived class but cannot be defined. A virtual function can be declared as Pure by using the operator =0.

Example -.

Virtual void function1 () // Virtual, Not pure

Virtual void function2 () = 0 //Pure virtual

**38.   What are all the operators that cannot be overloaded?**

Following are the operators that cannot be overloaded -.

1. Scope Resolution (:: )
2. Member Selection (.)
3. Member selection through a pointer to function (.\*)

**39.   What is dynamic or run time polymorphism?**

Dynamic or Run time polymorphism is also known as method overriding in which call to an overridden function is resolved during run time, not at the compile time. It means having two or more methods with the same name, same signature but with different implementation.

**40.   Do we require parameter for constructors?**

No, we do not require parameter for constructors.

**41.   What is a copy constructor?**

This is a special constructor for creating a new object as a copy of an existing object. There will be always only on copy constructor that can be either defined by the user or the system.

**42.   What does the keyword virtual represented in the method definition?**

It means, we can override the method.

**43.   Whether static method can use non static members?**

False.

**44.   What are base class, sub class and super class?**

Base class is the most generalized class, and it is said to be a root class.

Sub class is a class that inherits from one or more base classes.

Super class is the parent class from which another class inherits.

**45.   What is static and dynamic binding?**

Binding is nothing but the association of a name with the class. Static binding is a binding in which name can be associated with the class during compilation time, and it is also called as early Binding.

Dynamic binding is a binding in which name can be associated with the class during execution time, and it is also called as Late Binding.

**46.   How many instances can be created for an abstract class?**

Zero instances will be created for an abstract class.

**47.   Which keyword can be used for overloading?**

Operator keyword is used for overloading.

**48.   What is the default access specifier in a class definition?**

Private access specifier is used in a class definition.

**49.   Which OOPS concept is used as reuse mechanism?**

Inheritance is the OOPS concept that can be used as reuse mechanism.

**50.   Which OOPS concept exposes only necessary information to the calling functions?**

Data Hiding / Abstraction